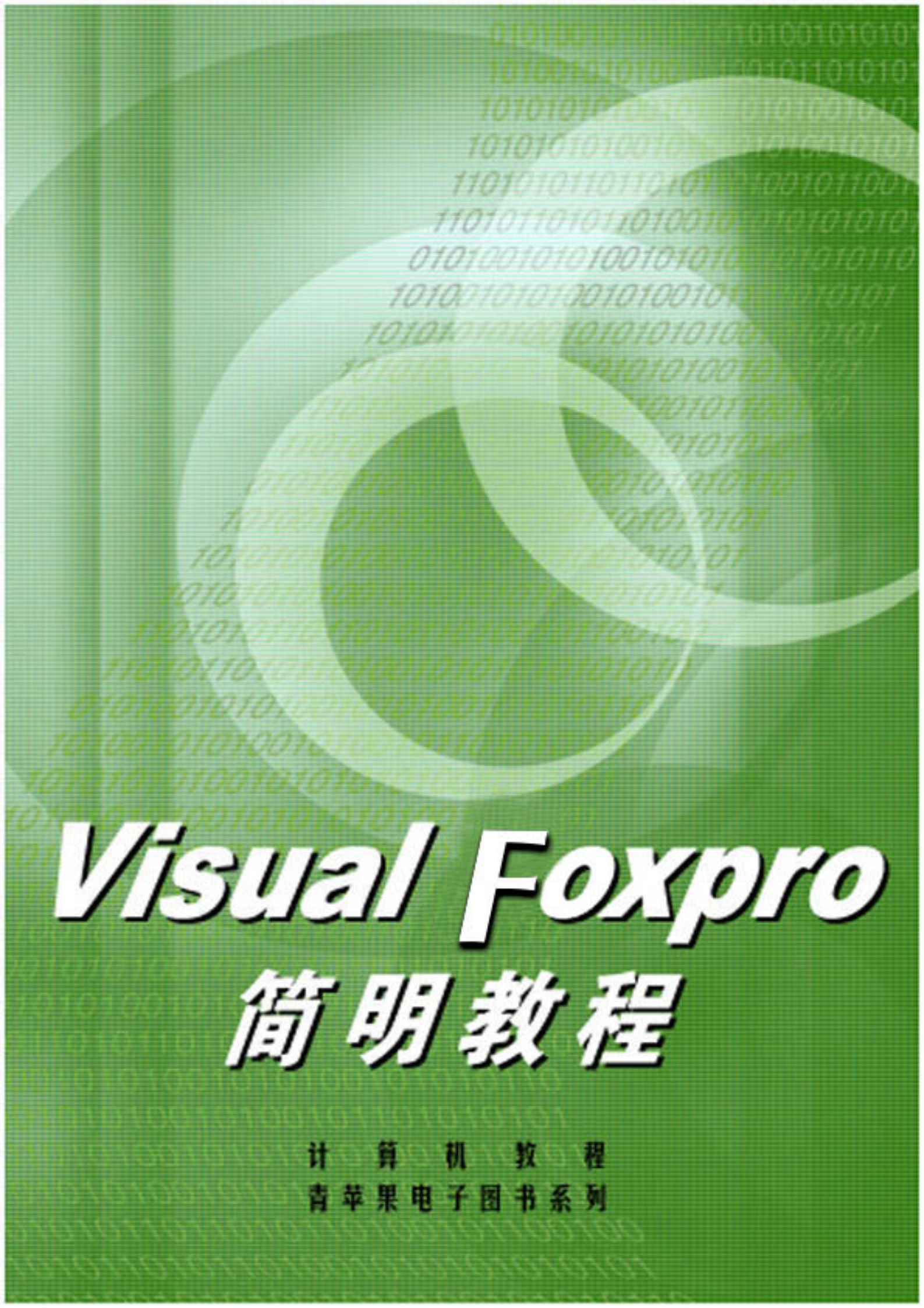

Visual Foxpro
简明教程
计 算 机 教 程
青苹果电子图书系列

Visual FoxPro 简明教程

王　中　　编著
张　晋　　审校

北京

丛 书 前 言

计算机软件技术日新月异，编程软件种类繁多且各具特色。众多的计算机爱好者都希望能成为电脑编程的行家里手，如何才能向“卡乃基殿堂”迈出正确的第一步？

为满足各级 Windows 程序开发人员、大专院校相关专业的师生和业余编程爱好者学习和使用各种流行编程软件的需求，编者在搜集了不同层次读者意见的基础上，经过仔细探讨，精心策划了本套丛书，目的在于引导读者学习二十一世纪真正的编程工具！

本丛书名为“流行编程软件简明教程丛书”，主要包括：

《C++ Builder 简明教程》

《Delphi 简明教程》

《Visual C++ 简明教程》

《Visual Basic 简明教程》

《Visual FoxPro 简明教程》

《Quick BASIC 简明教程》

《C 语言简明教程》

本套丛书针对当前最流行的开发工具，通过浅显易懂、活泼生动的语言，采用逐步引导的方式将读者带入电脑编程的殿堂，使读者轻松掌握编程的基础知识，为进一步深造打下坚实基础。生动友好的例子帮助读者了解、掌握编程技巧，同时还大大激发了读者的兴趣。

本套丛书既不是面面俱到的“功能大全”，也不是单一查询函数的“用户手册”，而是独具特色的操作和编程指导书。丛书注重对软件的主要功能和新增特性进行介绍，结合实例和项目，讲解软件的主要功能、主要原理。

我们希望，本套丛书对于读者能“抛砖引玉、触类旁通”，指引读者踏上通往编程高手之路。

总的来说，本套丛书有以下几个最优越的特点：

- ✧ “精”——精选软件，精选作者，精选图书，是值得用户收藏的精品
- ✧ “准”——紧跟软件版本更新，出版时间准，图书出版市场定位准
- ✧ “简”——教程风格简单朴素，要让读者学习起来感觉简单实用
- ✧ “明”——丛书内容十分明晰，让读者读完之后能够真正明白

我们相信：

如果你是一位初学编程的读者，本套丛书将带领你入门！

如果你是一位对编程有些了解的业余爱好者，本套丛书为你指明前进的方向！

如果你是一位软件编程专业人员或计算机行家里手，本套丛书将为你的程序锦上添花！

编 者

前　言

Visual FoxPro 具有功能强大、界面友好、简单易学的特点，它使用了面向对象编程技术，采用了可视化的概念，首次提出支持客户机/服务器体系结构，并且彻底更新了“数据库”的概念。其最新版本 Visual FoxPro 6.0 中又加强了 Internet 的功能，以适应 Internet 时代的应用及开发。

本书从熟悉 Visual FoxPro 的开发环境以及基本操作入手，结合了大量的数据库使用、开发实例，深入浅出，系统地介绍了 Visual FoxPro 的数据库操作、开发应用程序等操作。是初学者入门并得以提高的很好的指导书，也可作为中级用户开发过程的参考书。

本书主要向读者介绍了如下内容：

第一章：向读者介绍 Visual FoxPro 6.0 的系统环境；

第二章～第五章：向读者介绍 Visual FoxPro 自由表，数据库，数据库的索引、查询和视图以及创建数据报表，这是 Visual FoxPro 最常用的操作；

第六章～第八章：在这一部分介绍了 Visual FoxPro 高级操作——开发应用程序，详细介绍了基本控件的使用，添加菜单、工具栏，使用对象的链接与嵌入；

第九章：向读者介绍了客户/服务器解决方案；

在附录中提供了关于 Visual FoxPro 编程语言的基础知识，有助于读者深入学习 Visual FoxPro。

本书的作者长期应用 Visual FoxPro 进行程序设计和数据库软件开发，对 Visual FoxPro 的应用积累了大量的经验，这在本书中也有所体现。

本书是集体智慧的结晶，王中、梁久裕、谢风、张平、皇家明、石晓芸等同志参与了本书的编写工作，王中同志进行了最终的统稿工作。

欢迎读者使用本书，并多提宝贵意见。

内 容 简 介

本书从熟悉 Visual FoxPro 的开发环境入手，结合了大量的数据库使用、开发实例，介绍了数据库使用的基本操作：数据库表、查询、视图、报表，同时向读者系统地介绍了 Visual FoxPro 的开发应用程序的操作。本身是初学者入门并得以提高的很好的指导书，也可作为中级用户开发过程的参考书。

目　　录

第一章 Visual FoxPro 6.0概述

Visual FoxPro 目前的最高版本为 6.0，该版本继承了以往版本的优点，同时又比以往版本具有更多更先进的功能，特别是在 Internet 方面的功能，以适应当今世界网络技术的发展。

Visual FoxPro 6.0 的界面是一个标准的 Windows 界面，它由标题栏、菜单栏、工具条、命令窗口和状态栏几部分组成，如图 1-1 所示。

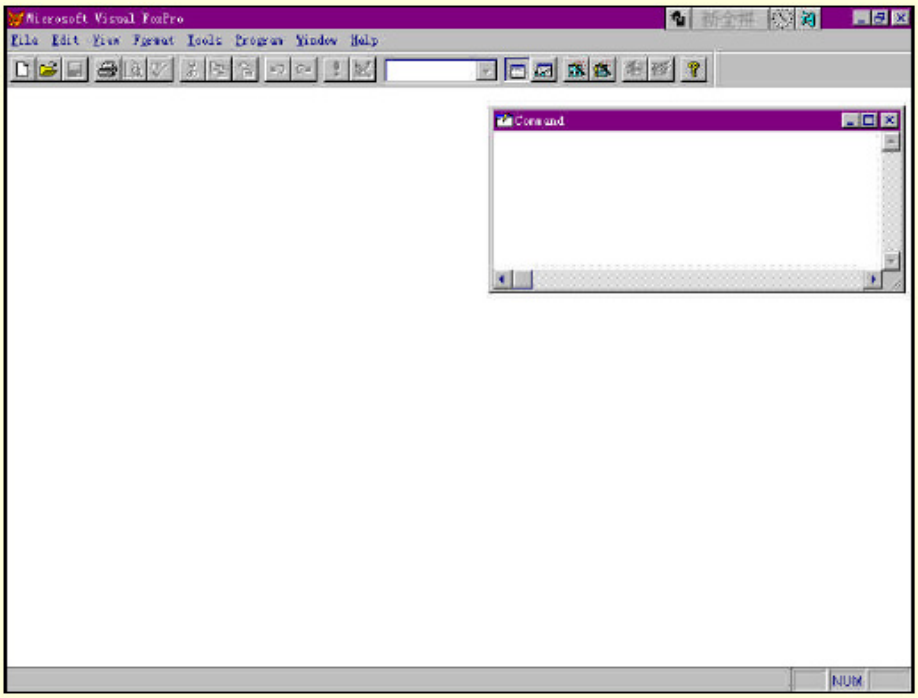


图 1-1 Visual FoxPro 6.0 的界面

单击窗口左上角的小狐狸图标，会下拉下来一个标准 Windows 菜单，如图 1-2 所示，该菜单在所有的 Windows 应用程序的左上角都有，通过选择它的子菜单可以实现窗口最大化、最小化、关闭、移动、还原等等几项功能。

图 1-2 Visual FoxPro 6.0 的菜单

窗口右上角的三个小按钮分别是最小化、还原、最大化按钮，分别实现窗口的最小化、最大化和还原功能。

窗口最下面的横条是状态栏。它显示当前工作的状态，Caps Lock 键和 Num Lock 键是否按下等情况。用户可随时通过状态栏获取当前工作的状态。

Visual FoxPro 6.0 的菜单是动态的，随着操作的不同而显示不同的菜单，这也体现了 Visual FoxPro 6.0 在界面环境上的优秀的设计思想。

Visual FoxPro 6.0 的工具栏也是随着操作不同的对象而不同，同时它的工具条也是可以定制的。

命令窗口浮动在 Visual FoxPro 6.0 的窗口之上，通过它可向系统发出操作命令。通过选择菜单 Window（窗口）下的 Command Window（命令窗口）可使其关闭或显示。

1.1 最常用的窗口——命令窗口

在 Visual FoxPro 6.0 中，执行命令的方法有多种，可以通过它的菜单来执行，也可通过 Command（命令）窗口直接执行命令。Command（命令）窗口是系统定义的窗口，当用户打开 Visual FoxPro 6.0 时，它就会出现在主屏幕上，如图 1-3 所示。Visual FoxPro 6.0 中的所有可执行的命令、函数等都可以通过命令窗口输入命令来执行。

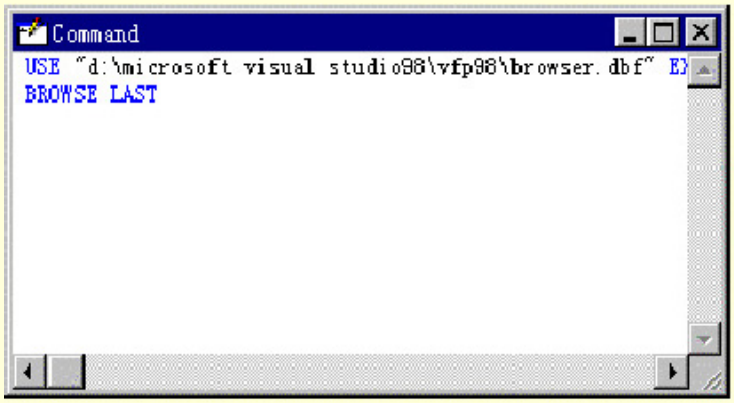


图 1-3 Command（命令）窗口

Visual FoxPro 6.0 之所以保留它的命令窗口而不完全采用菜单、对话框等来执行命令，主要是考虑以下原因：

（1）一般情况下，输入一个命令要比采用菜单和对话框来执行命令更简单快捷。例如浏览一个表，如果用命令窗口输入，只要输入 USE “表名”，按回车，再输入 BROWSE 回车即可。而用菜单则要选择“File”主菜单下的“Open”，然后选择要打开的表，按“确定”后，再选择“View”菜单下的“Brows”才能看到要浏览的表。所以用命令要比菜单有时要快捷。

（2）Visual FoxPro 6.0 系统中要大量的菜单和命令，不可能把所有的命令与函数都列在菜单和对话框中，即使把这些命令与函数都列在菜单和对话框中，也一定会使菜单和对话框特别冗繁，不便使用。

（3）通过输入 Visual FoxPro 6.0 的各种命令和函数，用户可以深入掌握 Visual FoxPro 6.0 的命令和函数。而对这些命令和函数越熟悉，对 Visual FoxPro 6.0 掌握的越深入，因为这些命令和函数是编程的基础。Visual FoxPro 6.0 程序的绝大多数命令都可以在命令窗口中键入执行，只是在命令窗口中每次只能执行一条命令，而在程序中把所有的命令一起全部执行。

（4）命令窗口是一种帮助调试程序的工具。虽然在 Visual FoxPro 6.0 中提供了调试工具“跟踪”和“调试”，但是它们不能代替命令窗口的作用。在“调试”窗口只能对一些函数、变量求值，不能输入命令。在“跟踪“窗口中也不能直接输入命令，只能对文件进行跟踪。若要查看一条命令执行的结果时，就要必须用到命令窗口了。在命令窗口输入一条命令，马上就会看到执行的结果，以便于及时发现程序的错误。

1.1.1 使用命令窗口

可以直接在命令窗口中输入 Visual FoxPro 6.0 的命令。在命令窗口中可以按如下方法编辑和操作命令：

（1）在按 ENTER 键执行命令之前，可按 ESC 键删除文本。

（2）将光标移到以前命令行的任意位置按 ENTER 键重新执行此命令。

（3）选择要重新执行的代码块，然后按 ENTER 键。

（4）若要分割很长的命令，可以在所需位置的空格后接分号，然后按 ENTER 键。

（5）可在命令窗口内或向其他编辑窗口中移动文本，选择需要的文本，并将其拖动到需要的位置。

（6）可在命令窗口内或向其他编辑窗口中复制文本，而不用使用“编辑”菜单的命令。选择需要的文本，按住 CTRL 键，将其拖动到需要的位置。

1.1.2 设置命令窗口格式

从 Format（格式）菜单中选择合适的命令，可以改变命令窗口中的字体、行间距和缩进方式。

1.1.3 弹出菜单

在命令窗口中单击鼠标右键，可弹出一个快捷菜单，如图 1-4 所示。

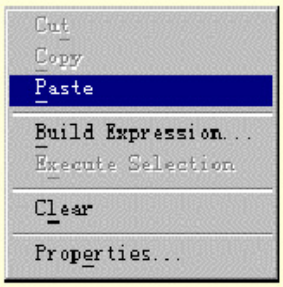


图 1-4 命令窗口快捷菜单

此菜单用来对命令进行编辑和命令文本的格式的调整。该菜单各选项如下：

- Cut、Copy、Paste：剪切、复制、粘贴。在命令窗口中移动或删除字符，或向命令窗口中移动或删除字符。
- Build Expression：生成表达式。显示“表达式生成器”窗口，在该窗口中用户可以使用命令、原义字符串、字段或其他表达式定义一个表达式。单击“确定”后，所生成的表达式就粘贴到“命令”窗口中。
- Execute Selection：运行所选区域。将命令窗口中选定的文本作为新命令执行。
- Clear：清除。从命令窗口中移去以前执行命令的列表。
- Properties：属性。显示“编辑属性”窗口，在该窗口中，可以改变命令窗口中的编辑行为、制表符宽度、字体和语法着色选项。

1.1.4 在命令窗口中处理错误

在命令窗口中输入命令时，有时难免会输入一些错误的命令，比如敲错了命令没输完就执行，这时 Visual FoxPro 6.0 就会给出一个出错信息。这个信息很简短，但很明确。Visual FoxPro 6.0 会试图提示到底是命令的什么地方错了。比如将 MODIFY COMMAND 命令敲成了 MODIFY COMMAMDE，那么 Visual FoxPro 6.0 就会给出提示 Command contains unrecognized phrase/keyword（命令中含有不能识别的短语或关键字）以提示命令输错，如图 1-5 所示。

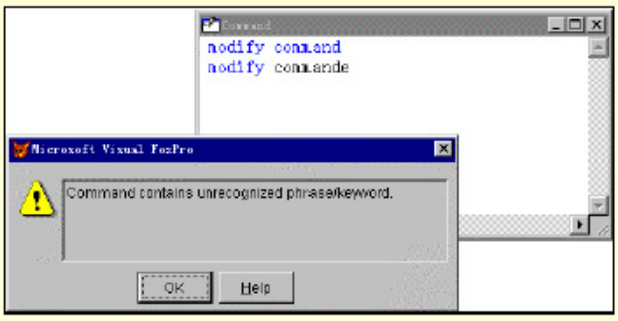


图 1-5 错误信息提示

如果一条命令没有输完整，比如为一个表建索引，如果仅输入了 INDEX ON 就执行，则 Visual FoxPro 6.0 会给出“缺少表达式”提示，告诉用户虽然有这个命令，但是命令中必要的成分未输全。

在以前版本的 Foxpro 中，错误信息显示在一个单独的警告窗口中，该窗口除显示错误信息外没有其他功能，且按任意键或单击鼠标即可消去该警告窗口。而在 Visual FoxPro 6.0 中，错误提示窗口改成了对话框的形式，增加了两个按钮：“确定”和“帮助”按钮，单击“确定”按钮即关闭错误信息对话框，单击“帮助”按钮可寻求在线帮助，以找到问题所在。

出现错误信息对话框后，按下 ESC 键或单击 OK 按钮均可消除此对话框。一旦出现错误，就要根据相应的提示检查是否命令输错了，或者对照函数手册查看是否命令没有输入完整。在中文环境下使用时，尤其要注意半个汉字问题，有时一条命令怎么检查都发现不了错误，但就是不能执行，这时就该考虑是否在命令中输入了半个不可见的字符，如半个汉字。此时再将命令输入一遍也许就正常了。

在 Visual FoxPro 6.0 中，命令与函数仅识别前四个字符，如 MODIFY COMMAND 命令敲成 MODI COMMAND 也是正确的。为减少出错，建议输入命令时只输入前四个字母。一定要注意一旦输入超过四个字符就一定要输全，否则也是错误。

1.2 常用菜单介绍

Visual FoxPro 6.0 和其他 Windows 应用程序一样，具有典型的 Windows 菜单系统并服从 Windows 的菜单约定：

（1）在 Visual FoxPro 6.0 的主菜单下有许多子菜单，如果这些子菜单的右边有一个黑色的三角时，说明此菜单还会弹出下一级子菜单。

（2）如果这些子菜单的右边是省略号，说明选择此菜单会弹出一个对话框，通过对话框可进行交互操作。

（3）如果菜单右边有组合键，说明该操作可通过快捷键实现。如打开一个表或数据库等，可以按下 Ctrl＋O 弹出 Open（打开）对话框，可选择要打开的文件。

（4）如果菜单系统中某些子菜单呈灰色，说明该子菜单在当前操作下不可用，只有当操作进行到需要该命令或功能时，该子菜单才可用。例如：如果刚刚进入系统，没有进行任何操作，则编辑菜单下的 Redo、Undo、Copy、Cut、Paste 等等子菜单都不可用。

1.2.1 八项主菜单

刚刚打开 Visual FoxPro 6.0，整个菜单系统只显示 File（文件）、Edit（编辑）、View（显示）、Format（格式）、Tool（工具）、Program（程序）、Window（窗口）、Help（帮助）八项主菜单，同时，这些菜单下还有许多子菜单可用。所有可用的菜单提示了当前可以进行的操作。下面是这几项菜单的内容及用法。

1．File（文件）菜单

该菜单如图 1-6 所示。

图 1-6 文件菜单

主要包括如下选项：

（1）New（新建）：打开新建对话框，创建新文件。文件类型为 Project（项目）、Database（数据库）、Table（表）、Query（查询）、Connect（连接）、View（视图）、Remote view（远程视图）、Form（表单）、Report（报表）、Label（标签）、Program（程序）、Class （类）、Text file（文本文件）、Menu（菜单）。

（2）Open（打开）：打开上面提到的几种文件。
（3）Close（关闭）：关闭当前窗口。
（4）Save（保存）：保存当前修改的文件。
（5）Save As（另存为）：用新文件名保存当前的文件。
（6）Save As HTML（另存为 HTML 文件）：把当前的文件存为 HTML 文件。
（7）Revert（还原）：将当前文件还原为最后保存的版本。
（8）Import（导入）：导入 Visual FoxPro 文件或其他应用程序的文件。
（9）Export（导出）：将 Visual FoxPro 6.0 文件以其他应用程序的文件格式输出。
（10）Page Setup（页面设置）：设置页面布局和打印机。
（11）Page Preview（打印预览）：在打印前显示整个页面。
（12）Print（打印）：打印报表、标签、文本文件、命令窗口或剪贴板的内容。
（13）Exit（退出）：退出 Visual FoxPro 6.0。

2．Edit（编辑）菜单

Edit 菜单如图 1-7 所示。

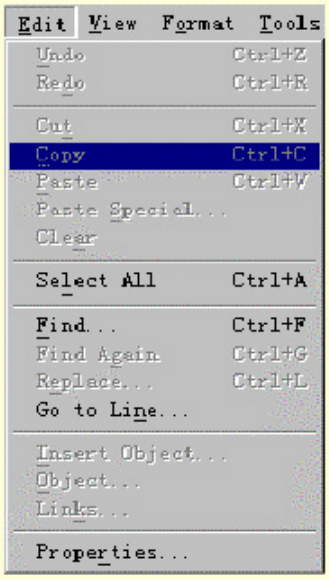


图 1-7 编辑菜单

它主要有以下几项内容：
（1）Undo（撤销）：撤销上一次命令或操作。
（2）Redo（重复）：重复上一次命令或操作。
（3）Cut（剪切）：移去选定内容，并将其放到剪贴板上。
（4）Copy（复制）：将选定的内容复制到剪贴板上。
（5）Paste（粘贴）：将剪贴板上的内容粘贴到当前位置。
（6）Paste Special（选择性粘贴）：链接或嵌入剪贴板上的 OLE 对象。
（7）Clear（清除）：删除窗口。
（8）Select All（全部选定）：选择当前窗口中的所有的对象及文本。
（9）Find （查找）：搜索指定的文本。
（10）Find Again（再次查找）：重复上一次查找。
（11）Replace（替换）：用其他文本代替当前文本。
（12）Go to Line（定位行）：在指定行放置插入点。
（13）Insert Object（插入对象）：向通用字段中链接或嵌入一个对象。
（14）Object（对象）：编辑选定对象。
（15）Links（链接）：修改或断开一个链接。
（16）Properties（属性）：设置编辑属性。

该菜单中的命令对文本及对象进行编辑，其“Cut”、“Copy”、“Paste”命令是 Windows 应用程序中最常用的命令，其快捷键 Ctrl＋X、Ctrl＋C、Ctrl＋V 应当记住，以利于更快的编辑。

3．View（显示）菜单

View 菜单是一个动态菜单，它随着当前操作的不同具有不同的子菜单。

当刚进入 Visual FoxPro 6.0 的环境还未进行任何操作时，它只有 Toolbars（工具栏）一个选项。如图 1-8 所示。

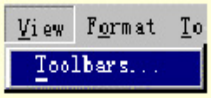


图 1-8　无操作时的 View 菜单

选择 Toolbars 时，系统显示 Toolbars 对话框，定制系统的工具栏。在当前正对一个表进行操作时，显示菜单的内容主要是对表的操作命令，如图 1-9 所示。

图 1-9　对表进行操作时的 View 菜单

（1）Brows（浏览）：在浏览窗口中显示当前表。

（2）Edit（编辑）：在浏览窗口的编辑方式下显示当前表。

（3）Append Mode（追加方式）：启用浏览窗口的追加方式自动添加记录。

（4）Table Designer（表设计器）：显示表设计器修改表的结构,如：添加、删除字段、修改字段。

（5）Grid Lines（网格线）：在当前表中显示或隐藏网格线。当此选项前面划了对号时为选中，表中有网格线；当此选项前面未划对号时为没选中，表中没有网格线。

（6）Toolbars（工具栏）：显示 Toolbars 对话框，定制系统的工具栏。

当对数据库中的表进行操作时，显示菜单会多一项子菜单 Database Designer（数据库设计器），如图 1-10 所示。选择此菜单，系统便转入数据库设计器中，这时可对数据库进行添加、删除表的操作或创建视图等操作。

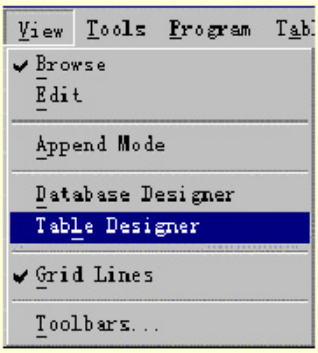


图 1-10　对数据库操作时的 View 菜单

4．Format（格式）菜单

Format 菜单中的命令用来控制窗口中的文本和其他对象的显示效果。如图 1-11 所示。

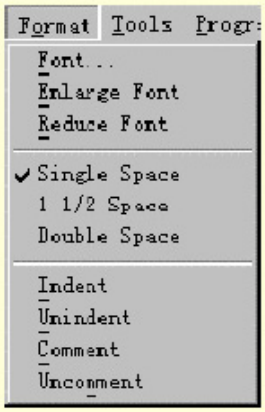


图 1-11 格式菜单

（1）Font（字体）：该命令启动标准 Windows 字体对话框，对字体的类型、字形及大小进行调整。
（2）Enlarge Font（放大字体）：将字体放大到更大的尺寸。
（3）Reduce Font（缩小字体）：将字体缩小到更小的尺寸。
（4）Single Space（单倍行距）：显示文本时文本行之间无空白行。
（5）1 1/2 Space（1.5 倍行距）：以 1.5 倍行距显示文本。
（6）Double Space（两倍行距）：以 2 倍行距显示文本。
（7）Indent（缩进）：将选定的行缩进一个 Tab 键宽度。
（8）Unindent（撤销缩进）：一次删除一个先前插入的缩进。
（9）Comment（注释）：注释所选的行。
（10）Uncomment（撤销注释）：撤销对所选行的注释。

5．Tools（工具）菜单

Tools 菜单是命令比较多的一个菜单，如图 1-12 所示。

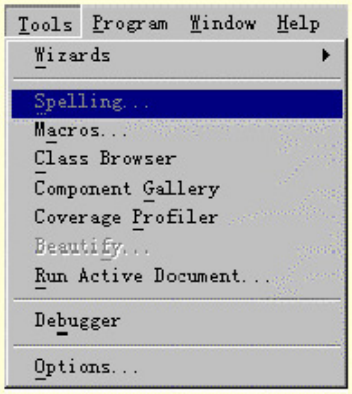


图 1-12 工具菜单

（1）Wizards（向导）：它不但包括了新建对话框中的所有向导，还包括了 MailMerge（邮件合并）、PivotTable（数据透视表）、Import（导入）、Documenting（文档）、Setup（安装）、Upsizing（升迁）、Application（应用程序）、Web Publishing（Web 页发布）、All（全部）。图 1-13 所示为向导子菜单。

图 1-13　向导子菜单

（2）Spelling（拼写检查）：拼写检查错误。

（3）Mzcros（宏）：创建、删除或修改键盘宏。

（4）Class Browser（类浏览器）：运行类浏览器。

（5）Component Gallery（组件管理库）：打开组件管理库。

（6）Coverage Profiler.（代码范围分析器应用程序）：运行代码范围分析器应用程序。

（7）Beautify（修饰）：对程序进行修饰。

（8）Run Active Document（运行 Active Document）：显示 Run Active Document 对话框，选择 Active Document 运行。

（9）Debugger（调试器）：显示调试器，调试程序。

（10）Options（选项）：显示选项卡，更改 Visual FoxPro 6.0 的选项。在本章的 2.6 节将对此选项卡作详细阐述。

6．Program（程序）菜单

Program 菜单如图 1-14 所示，用户通过该菜单对程序进行运行、编译等操作。

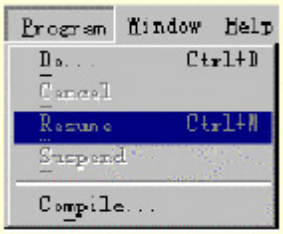


图 1-14　程序菜单

该菜单主要包括以下几个子菜单：

（1）Do（运行）：用户通过选择该命令运行一个程序、应用程序、表单、报表、查询或菜单。

（2）Cancel（取消）：停止运行当前程序。

（3）Resume（继续执行）：继续执行当前挂起的程序。

（4）Suspend（挂起）：挂起当前正在运行的程序。

（5）Compile（编译）：编译当前程序或选定程序。

7．Window（窗口）菜单

通过该窗口菜单，用户可以方便的调整、安排多个窗口。菜单选项见图 1-15 窗口菜单。该菜单具体选项用法如下：

（1）Arrange All（全部重排）：用非重叠方式重排窗口。

（2）Hide （隐藏）：隐藏活动窗口。

（3）Cycle（循环）：在所打开的窗口间循环切换。

（4）Command Window（命令窗口）：显示或隐藏命令窗口。

（5）Data Session （数据工作器）：显示数据工作期窗口。

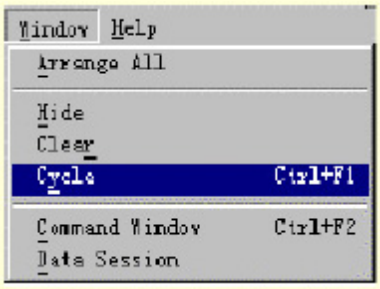


图 1-15 窗口菜单

8. Help（帮助）菜单

Help 菜单提供了对各种问题的帮助。见图 1-16 帮助菜单。如果用户在安装时没有安装帮助，则该菜单下的选项都是不可用的。

（1）Microsoft Visual FoxPro Help Topics（Microsoft Visual FoxPro 帮助主题）：用户可以通过输入关键字来查找帮助主题。

（2）Contents（文档）：打开 Visual FoxPro 6.0 的联机文档。该文档对 Visual FoxPro 6.0 的全部功能与用法以及所有的函数与对象等等都作了全面的讲述，而且可以目录的形式进行查阅。

（3）Index（索引）：以索引的形式显示帮助主题。

（4）Search （查找）：查找帮助主题。

（5）Technical Support（技术支持）：提供获得另外的技术帮助的信息。

（6）Microsoft on the Web ：启动用户的 Web 浏览器进入微软的 Web 节点。

（7）About Microsoft Visual FoxPro 6.0 ：显示 Visual FoxPro 6.0 的版本和版权信息。

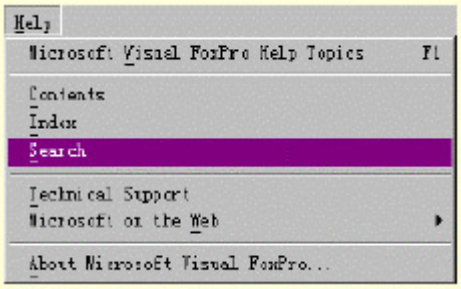


图 1-16 帮助菜单

1.2.2 动态变化的菜单

通常情况下，Visual FoxPro 6.0 的菜单仅显示上面介绍的几个菜单及其子菜单。当系统运行过程中用到某些功能时，系统会动态的增加或修改一些菜单项。这里介绍动态菜单和前面介绍的初始菜单的目的是让读者通过 Visual FoxPro 6.0 的菜单系统而了解 Visual FoxPro 6.0 的功能与用法。通过一个好的应用程序的菜单往往能显示出本应用程序的功能与用法。

下面介绍 Visual FoxPro 6.0 可动态显示的一些菜单。

1. Table（表）菜单

当打开一个表或新建了一个表后，菜单系统中会添加一个 Table（表）菜单，如图 1-17 所示。同时 Format（格式）菜单消失了。View 菜单中也增加了一些子菜单，如前面所述。

图 1-17 表菜单

表菜单包括以下选项：

（1）Properties（属性）：打开 Work Area Properties （工作区属性）对话框，设置表的一些属性。

（2）Fonts （字体）：打开标准 Windows 字体对话框，设置表中显示的记录的字体。

（3）Append New Record （追加新记录）：向表中添加新的记录。

（4）Toggle Deletion Mark（切换删除标记）：切换当前记录的删除标记，若当前记录有删除标记则设为无；若无删除标记则添加删除标记。

（5）Go to Record（转到记录）：它具有一个子菜单，如图 1-18 所示，分别可以转到 Top（第一个）、Bottom（最后一个）、Next（下一个）、Previous（前一个）、Record #（记录号）、Locate（定位）各子菜单项。其中 Record #和 Locate 项还会分别打开记录号和定位记录对话框。

图 1-18 转到记录子菜单

（6）Append Records（追加记录）：打开 Append From（追加记录）对话框，选择要追加的记录的来源。如图 1-19 所示。

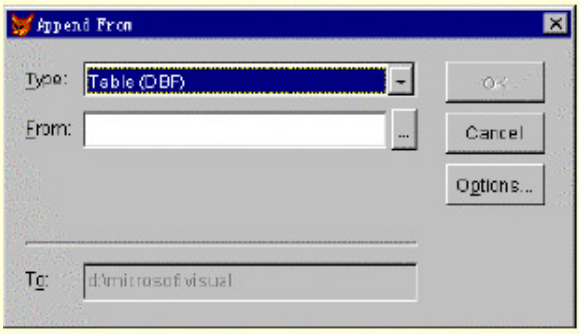


图 1-19 追加记录

（7）Delete Records（删除记录）：打开 Delete（删除）对话框，如图 1-20 所示，为满足条件的记录添加删除标记。

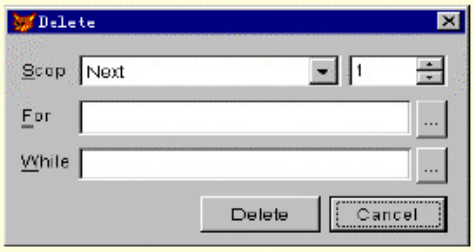


图 1-20 删除记录

（8）Recall Records（恢复记录）：打开 Recall（恢复）记录对话框，撤销满足条件的记录的删除标记。如图 1-21 所示。

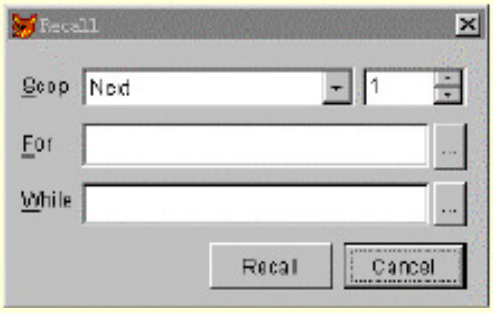


图 1-21 恢复记录

（9）Remove Deleted Record（彻底删除记录）：将已有删除记录的记录从表中完全删除。

（10）Replace Fields（替换字段）：显示 Replace Fields（替换字段）对话框，如图 1-22 所示，将满足条件的记录的字段替调。

（11）Size Field （调整字段大小）：在浏览窗口中调整字段显示的大小，此时鼠标变为双向箭头。此菜单项在浏览窗口的编辑方式下不可用。

（12）Move Field （移动字段）：移动字段在表中显示的位置，此时鼠标也变为双向箭头。

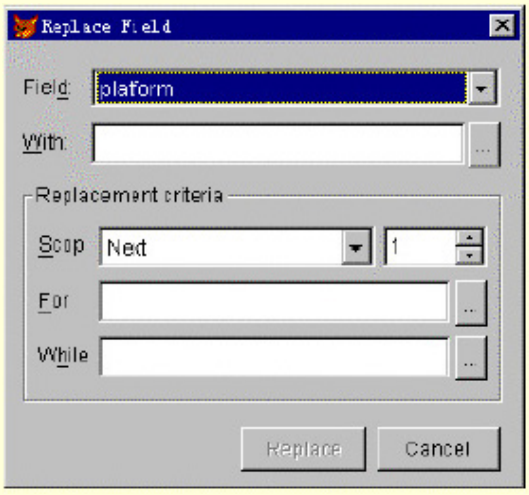


图 1-22 替换字段

2．Database（数据库）菜单

当打开或正在创建一个数据库时，菜单栏中会添加一项 Database（数据库）菜单如图 1-23 所示，以提供对数据库操作的命令与操作。该菜单提供了向数据库中添加表、添加视图以及编辑参照完整性等操作。以下是对该菜单的说明：

（1）New Table（新表）：在数据库中创建一个新表。

（2）Add Table（添加表）：向数据库中添加一个已经存在的表。

（3）New Remote View（新远程视图）：为数据库创建一个远程视图。

（4）New Local View（新本地视图）：为数据库创建一个本地视图。

（5）Modify（修改）：只有当选中数据库中的一个表时该命令可用。该命令打开 Table Designer（表设计器），

对当前表的结构、显示格式、字段验证规则等等项进行修改。

（6）Browse（浏览）：以浏览方式查看当前表的内容。

（7）Remove（移去）：从数据库中去掉当前选定的表。

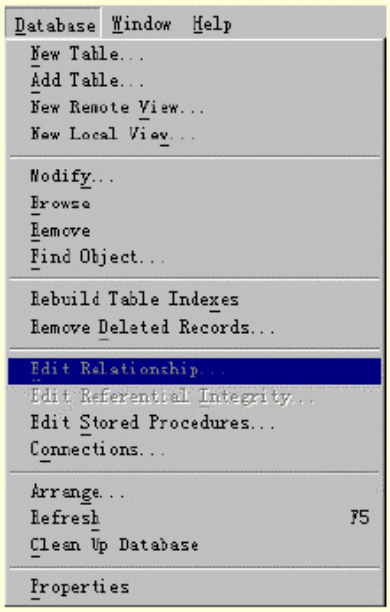


图 1-23 数据库菜单

（8）Find Object（查找对象）：显示 Find Table and View 对话框，此对话框列出此数据库中全部的表和视图，如图 1-24 所示，可通过选择一个表或视图直接定位到数据库的表或视图中。此操作对包含有大量表和视图的数据库特别有用。

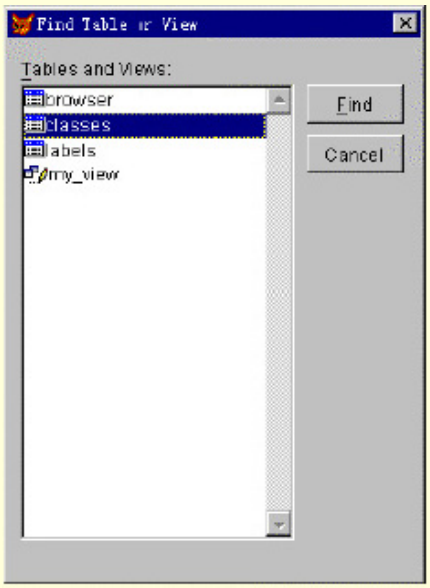


图 1-24 查找对象对话框

（9）Rebuild Table indexes（重建表索引）：对当前的表建立索引。

（10）Remove Deleted Records（彻底删除记录）：删除当前表中有删除标记的记录。同时提示是否真的要删除。

（11）Edit Relationship（编辑关系）：当选中表与表之间的关系连线时可用。该命令显示 Edit Relationship（编辑关系）对话框如图 1-25 所示，编辑数据库中表与表之间的关系。

（12）Edit Referential Integrity （编辑参照完整性）：显示 Referential Integrity Builder（参照完整性生成器）对话框，如图 1-26 所示。编辑更新、删除、插入规则。

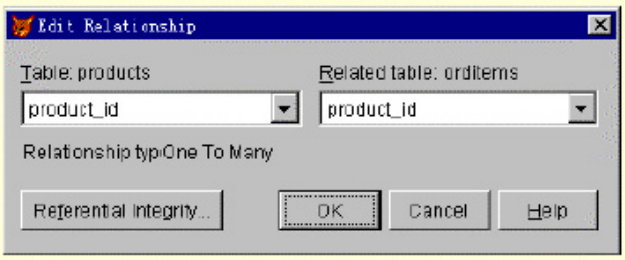


图 1-25 编辑关系对话框

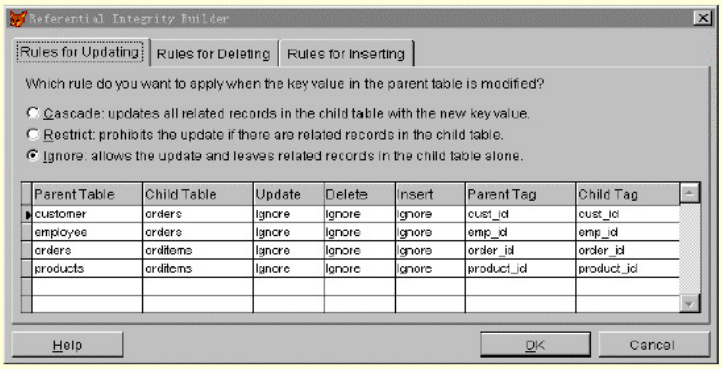


图 1-26 参照完整性生成器

（13）Edit Stored Procedure（编辑存储过程）：在编辑存储过程输入过程代码，存储过程是保存在数据库中的独立程序，属于数据库管理的对象。存储过程可供数据库中的有关对象调用。也可设置 Trigger（触发器）。如图 1-27 所示是在插入过程中的一个触发器。

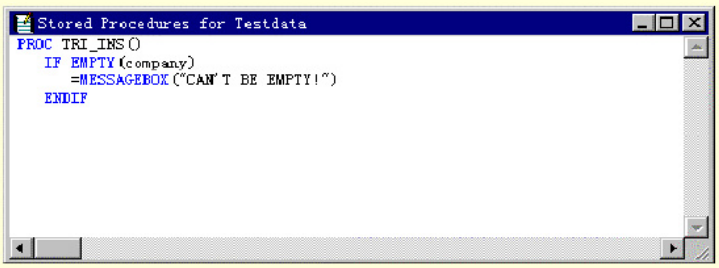


图 1-27 编辑存储过程对话框

（14）Connections（连接）：显示 Connections（连接）对话框，在对话框中可以选择修改与一个远程数据源的现有连接，或者创建一个新的连接，如图 1-28 所示。

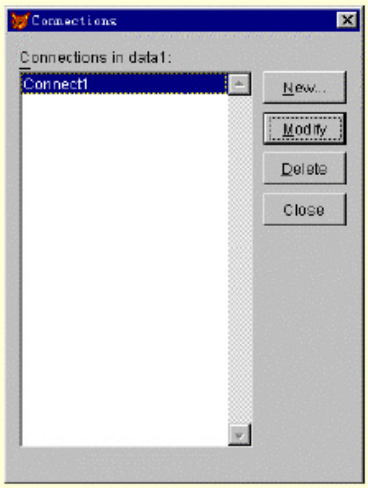


图 1-28 连接对话框

（15）Arrange（重排）：显示 Arrange Tables and Views（重排表和视图）对话框，如图 1-29 所示。在对话

框中可以选择选项来排序、对齐或者调整显示在数据库设计器中对象（包括表和视图）的大小。

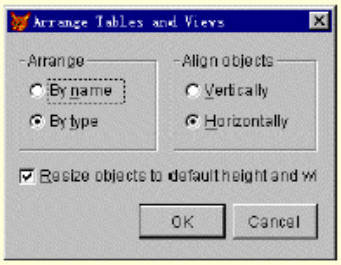


图 1-29　重排表和视图对话框

（16）Refresh（刷新）：使用数据库的当前磁盘图像更新数据库设计器的显示。

（17）Clean Up Database（清理数据库）：运行 PACK 命令删除带有删除标记的行，以减小数据库的大小。也可以使用 VALIDATE DATABASE 命令清除数据库与表之间的链接。

（18）Properties（属性）：显示 Database Properties（数据库属性）对话框，在对话框中可以选择规划时要显示的对象，或者向数据库添加注释，如图 1-30 所示。

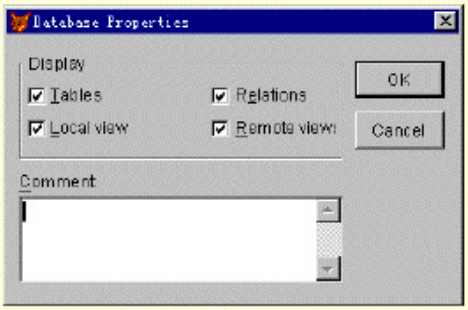


图 1-30　数据库属性对话框

3. Project（项目）菜单

Project（项目）菜单包含创建和修改项目的菜单项。此菜单只有在当前对一个项目进行操作时可用。如图 1-31 所示。

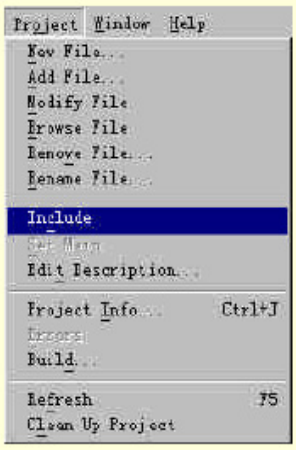


图 1-31　项目菜单

项目菜单项的子菜单的功能及用法如下：

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

（2）Add File（添加文件）：显示 Open（打开）对话框，从中可以向项目添加一个现有文件。该命令与在“项目管理器”中选择 Add（添加）按钮效果相同。当在“项目管理器”中选定一个要添加的文件类型时，该命令可用。Open 对话框中显示的文件以及文件扩展名反映出用户的选择。

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

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

（5）Remove File（移去文件）：显示如图 1-32 所示的对话框，可通过该对话框确定是从项目中移去所选定的文件，还是移去并从硬盘上删除该文件。Remove（移去）命令把文件从项目中删除，但不从硬盘上删除。Delete（删除）命令把文件从项目及硬盘中同时删除。Remove File（移去文件）命令与“项目管理器”中的 Remove 按钮效果相同。

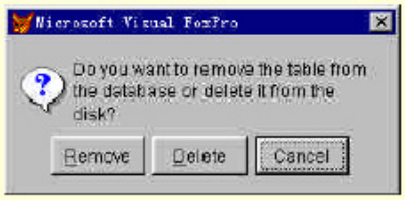


图 1-32　移去文件对话框

（6）Rename File（重命名文件）：显示 Rename File（重命名文件）对话框，从中可以重命名选定的文件。当在“项目管理器”中选定某个文件时，该命令可用，如图 1-33 所示。

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

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

（9）Edit Description（编辑说明）：显示 Description（说明）对话框，如图 1-34 所示。可在框中修改文本，该文本作为文件说明出现在“项目管理器”窗口的底部。

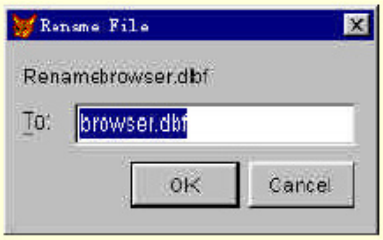


图 1-33　重命名文件对话框

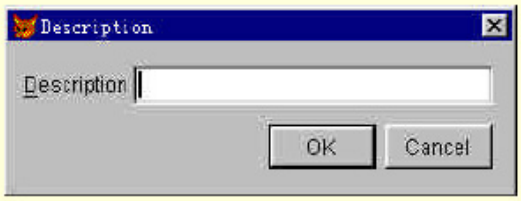


图 1-34　Description（说明）对话框

（10）Project Info（项目信息）：显示 Project Information（项目信息）对话框，如图 1-35 所示。从中可以查看和编辑有关项目及其文件的信息。

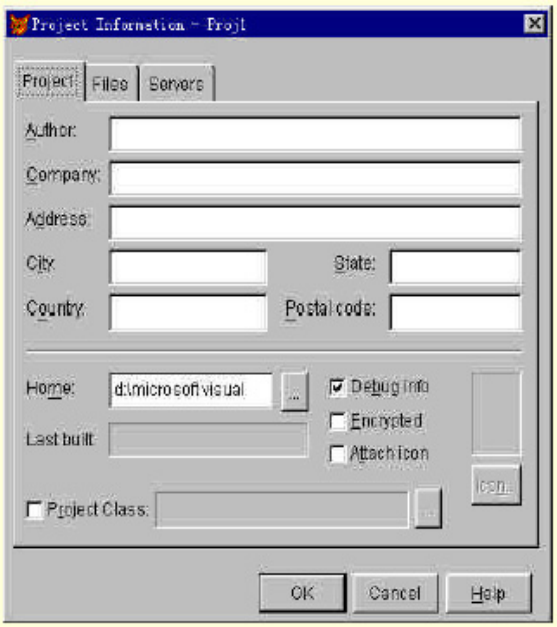


图 1-35 项目信息对话框

（11）Error（错误）：显示编译应用程序期间生成的错误日志文件。

（12）Build（连编）：显示 Build Options（连编选项）对话框，如图 1-36 所示，从而可以创建自定义应用程序并更新已有的项目。

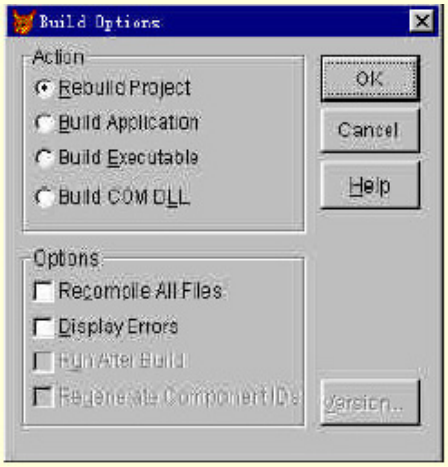


图 1-36 连编选项对话框

（13）Refresh（刷新）：刷新项目。

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

4．Query（查询）菜单

Query（查询）菜单包含创建、修改和运行查询的命令。当打开“查询设计器”或“视图设计器”时，此菜单有效，如图 1-37 所示。

图 1-37 查询菜单

（1）Add Table（添加表）：显示 Add Table or View（添加表或视图）对话框，如图 1-38 所示，从而可以向设计器窗口中添加表或视图。

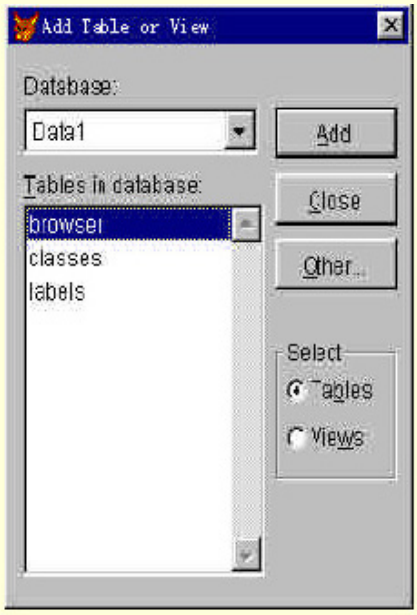


图 1-38 添加表或视图对话框

（2）Remove Table（移去表）：从设计器窗口的上窗格中移去所选定的表。只有在“查询设计器”或“视图设计器”中至少选择了一个表时，才可以使用此命令。

（3）Remove Join Condition（移去联接条件）：移去表之间所选中的联接线。只有当选定了一个表之间的联接线时，才可以使用此命令。

（4）Output Fields（输出字段）：将 Fields（字段）选项卡位于“查询设计器”或“视图设计器”窗口的最上面。在 Output Fields（输出字段）选项卡中允许指定查询结果中显示的字段。

（5）Join（联接）：将 Join（联接）选项卡位于“查询设计器”或“视图设计器”窗口的最上面。

（6）Filter（筛选）：将 Filter（筛选）选项卡位于“查询设计器”或“视图设计器”窗口的最上面。在 Filter 选项卡中允许指定查询结果中所出现的记录。

（7）Order By（排序依据）：将 Order By（排序依据）选项卡位于“查询设计器”或“视图设计器”窗口的最上面。在 Order By（排序依据）选项卡中允许指定字段，作为查询中排序记录的依据。

（8）Group By（分组依据）：使 Group By（分组依据）选项卡位于“查询设计器”或“视图设计器”窗口的最上面。在 Group By 选项卡中允许指定在查询结果中进行记录分组的条件。

（9）Miscellaneous（杂项）：使 Miscellaneous 选项卡位于“查询设计器”或“视图设计器”窗口的最上面

（10）Query Destination（查询去向）：显示 Query Destination（查询去向）对话框，该对话框允许把查询结果发送到七个不同的输出目的地，如图 1-39 所示。

下面是对这七个查询去向的说明：

- Browse 浏览　　在 Browse（浏览）窗口中显示查询结果。
- Cursor 临时表　　将查询结果保存于临时表中。
- Table 表　　将查询结果作为表文件保存起来。
- Graph 图形　　使查询结果可用于 Microsoft Graph，图形是包含在 Visual FoxPro 中的一个独立的 OLE 应用程序。
- Screen 屏幕　　在活动输出窗口中显示查询结果。
- Report 报表　　向报表文件发送查询结果。
- Label 标签　　向标签文件发送查询结果。

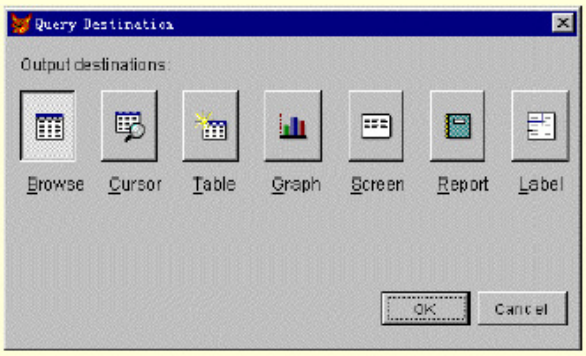


图 1-39　查询去向对话框

（11）View SQL（查看 SQL）：在只读文本文件中，显示创建查询或视图的 SQL 语句。

（12）Comments（备注）：显示 Comments（备注）对话框，在对话框中用户可以输入注意事项或注释，以标识查询或视图以及它们的目的。

（13）Run Query（运行查询）：执行所建立的 SQL SELECT 语句，并向所选定的输出目的地发送结果。

5．Report（报表）菜单

Report（报表）菜单菜单包含用于创建和修改报表的命令。当设计一个报表时显示。如图 1-40 所示。

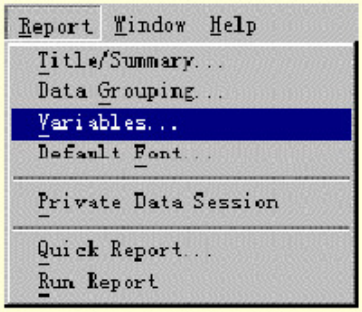


图 1-40　报表菜单

（1）Title/Summary（标题/总结）：显示 Title/Summary（标题/总结）对话框，可以指定是否将“标题”和（或）“总结”带区包括在报表中。

（2）Data Grouping（数据分组）：显示 Data Grouping（数据分组）对话框，可以创建数据组并指定其属性。

（3）Variables（变量）：显示 Report Variables（报表变量）对话框，可以创建报表中的变量。

（4）Default Font（默认字体）：显示标准 Windows“字体”对话框，可以指定报表和标签中标签和字段控件的永久字体、字体样式和字体大小。此设置随报表一起存储，这样，每次修改报表时，默认字体都是相同的。

（5）Private Data Session（私有数据工作期）：在一个私有工作期中打开报表使用的表，这样它们将不受其

他报表、表单或程序的影响。单击此命令将它打开或关闭。

（6）Quick Report（快速报表）：自动将选定字段放入一个空的“报表设计器”窗口中。该命令提示选择一个表，并显示 Quick Report（快速报表）对话框，可以选择字段及字段布局。

（7）Run Report（运行报表）：显示“打印”对话框，可以将报表传送给打印机。运行报表不会改变表、索引或备注字段中的数据。

在创建一个报表时，菜单栏还出现了 Format（格式）菜单。它和使用 Command（窗口）包含的命令就有所不同。该菜单实现了对报表内容的格式，如字体、对齐方式、颜色、置前还是置后的调整。

6．Form（表单）菜单

当正创建或修改一个表单时，菜单栏会出现 View（查看）、Format（格式）、Form（表单）三个菜单。其中，View 菜单用来定制用户的工作方式，如显示属性框、工具条等等；Format 菜单用来显示网格线、调整对齐方式等等。而 Form 菜单则是实现对表单的操作，Form 菜单包含创建和修改表单及表单集的命令。如图 1-41 所示。

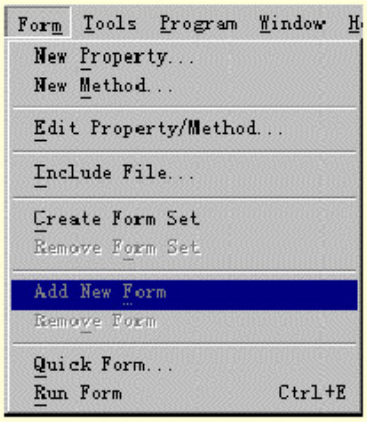


图 1-41 表单菜单

以下是对该菜单的说明：

（1）New Property（新建属性）：显示 New Property（新建属性）对话框，从中可以向表单或表单集添加新属性。

（2）New Method（新方法程序）：显示 New Method（新方法程序）对话框，从中可以向表单或表单集添加新方法程序。

（3）Edit Property/Method（编辑属性/方法程序）：显示 Edit Property/Method（编辑属性/方法程序）对话框，从中可以编辑现有的属性或方法程序。该属性或方法程序局限于独立的表单或表单集，也就是仅可用于该表单或表单集。

（4）Include File（包含文件）：显示 Include File（包含文件）对话框，从中可以指定一个包含预定义编译常量的头文件。当编译表单代码时，会使用该文件。

（5）Create Form Set（创建表单集）：创建一个新表单集，该表单集为一个或多个表单的父容器。

（6）Remove Form Set（移除表单集）：删除一个现有的表单集。仅当创建了一个表单集，并且其中只有一个表单时，该命令可用。

（7）Add New Form（添加新表单）：如果用户已创建了表单集，就可以向其中添加新表单。此命令仅在用户处理表单集而不是处理表单时可用。

（8）Remove Form（移除表单）：一旦创建表单集之后，可删除其中的选定表单。仅当表单集中有两个以上的表单时，该命令可用。

（9）Quick Form（快速表单）：显示 Form Builder（表单生成器）对话框，如图 1-42 所示。它可以自动创建一个简单的表单，用户可以添加自己的控件来定制它。用户可以在 Field Selection（选取字段）页选定数据环境，如选定某个表的某些字段。同时也可在 Style（样式）页中选定表单的样式。

（10）Run Form（执行表单）：创建了一个表单后，就可以使用这个命令运行表单。运行之前，系统会提示保存该表单。

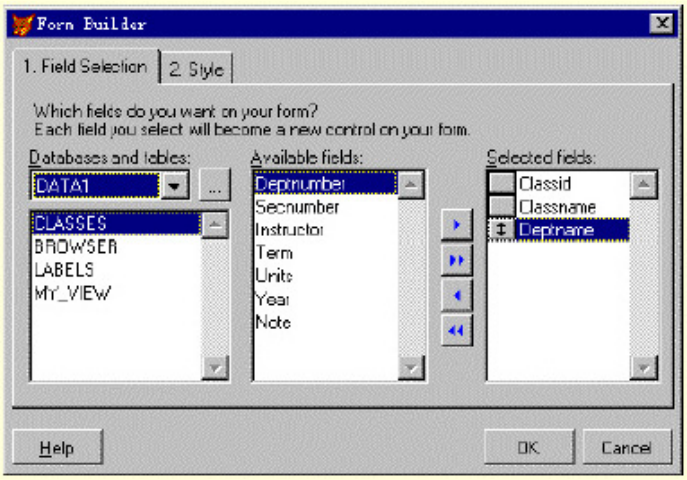


图 1-42 表单生成器对话框

7．Menu（菜单）菜单

当创建一个菜单时，Menu 菜单显示，如图 1-43 所示。Menu 菜单包含用于创建和修改菜单系统的命令。

（1）Quick Menu（快速菜单）：把 Visual FoxPro 6.0 的主菜单系统加载到“菜单设计器”，将其作为创建菜单系统的基础。

（2）Insert Item（插入菜单项）：在“菜单设计器”窗口中插入新的一行。

（3）Insert bar（插入栏）：显示“插入系统菜单条”对话框，它允许用户向“菜单设计器”窗口中添加菜单项。

（4）Delete Item（删除菜单项）：从“菜单设计器”窗口中删除选定行。

（5）Generate（生成）：显示 Generate Menu（生成菜单）对话框，从中指定生成的菜单程序名称，而后生成该程序。

（6）Preview（预览）：在没有编译菜单程序代码情况下，当用户设计菜单系统时，显示该菜单系统。

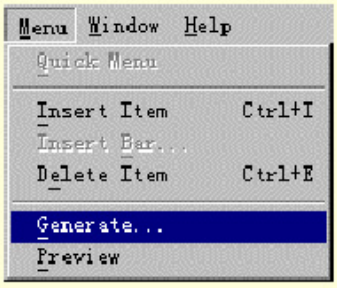


图 1-43 “菜单”菜单

8．Class（类）菜单

Class（类）菜单包含了一些命令，这些命令允许创建和编辑属性和方法程序，指定包含的文件，以及控制一般类的属性，如图 1-44 所示。

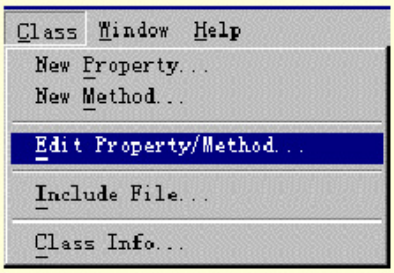


图 1-44 类菜单

（1）New Property（新建属性）：创建新属性。显示 New Property（新建属性）对话框。

（2）New Method（新方法程序）：创建新方法程序。显示 New Method（新方法程序）对话框。

（3）Edit Property/Method（编辑属性/方法程序）：打开 Edit Property/Method（编辑属性/方法程序）对话框，在对话框中可以编辑一个现有的属性或方法程序。在添加新的属性或方法程序时启用。

（4）Include File（包含文件）：可以指定一个包含预处理器命令的文件。显示 Include File（包含文件）对话框。

（5）Class Info（类信息）：显示 Class Info（类信息）对话框，对话框中显示了类成员的所有属性和方法程序，包括它们在设计时的外观，以及它们是否被保护。

1.3 工具栏的功能

Visual FoxPro 6.0 有很大的工具栏，最常用的工具栏列出在系统菜单栏的下面。其他的工具栏用户可自己定制。同时这些工具栏也是动态的，它随着用户的操作显示或关闭工具栏。这些工具栏提供了快捷地执行命令的一种方式。

1.3.1 自己定制工具栏

打开 View 菜单中的 Toolbars 项，或用鼠标在任一工具栏区域内单击右键，会弹出 Toolbars 对话框，如图 1-45 所示。打开 Toolbars 对话框后，就可以控制系统当前显示的工具栏了。

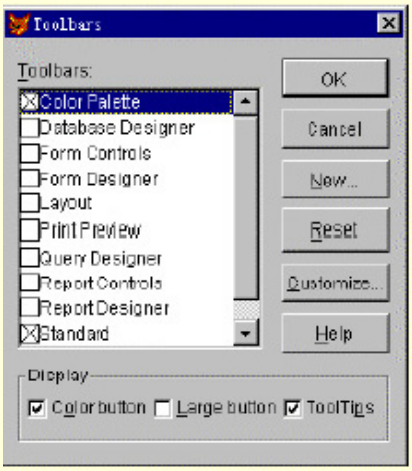


图 1-45 工具条对话框

对于打开的工具栏，可以用鼠标移到其任意空白区或标题区并按下左键拖动改变其位置，也可通过把鼠标放在其边框上（此时鼠标为双箭头形状）按下左键拖动改变其形状。

单击工具栏右上角的“×”可关闭工具栏。

将鼠标放在工具栏上稍作停留时，系统会提示鼠标所在位置的按钮功能。

（1）Toolbars（工具栏）：列出紧跟着自定义工具栏的内置工具栏。选定的工具栏在活动输出窗口中平铺或显示。在工具栏对话框中，如果工具栏左边的方框有“×”，表示选中，按下 OK 后会显示该工具条。

（2）New（新建）：显示“新工具栏”对话框，在其中为新建的工具栏输入名称。

（3）Reset（重置）：将选定工具栏返回到它的内置状态。如果选定了自定义工具栏，这个按钮将变成“删除”。不能重置自定义工具栏。

（4）Customize（定制）：显示“定制工具栏”对话框，从中可以添加或删除工具栏按钮。

（5）Color（彩色按钮）：在内置工具栏和自定义工具栏中显示彩色按钮。如果使用的是黑白监视器，应清除此复选框。

（6）Large button（大按钮）：显示比标准尺寸大的工具栏按钮。如果监视器具有高分辨率，用户可能希望显示大按钮。

（7）ToolTips（工具提示）：当鼠标指针在工具栏任何一个按钮上方时，此命令显示该按钮的名称。

1.3.2 所有工具栏的用法

下面是 Visual FoxPro 6.0 的所有的工具栏的名称与用法。

1．常用工具栏

常用工具栏包括最常用动作的按钮，如图 1-46 所示。

图 1-46 常用工具条

表 1-1 是对此常用工具条的各个按钮的说明。

表 1-1 常用工具条按钮说明

按 钮	功 能	说 明
	新建	使用设计器或向导创建新文件。
	打开	打开一个已有的文件或创建一个新文件。
	保存	保存对活动文件所做的修改。
	打印一个副本	打印文本文件、报表、标签，以及“命令”窗口或剪贴板中的内容。
	打印预览	显示工作的结果但不打印。可以预先浏览要打印的文档的效果。
	拼写检查	进行拼写检查。编辑文本或备注字段时可用。
	剪切	把选定的文本、控件或任何其他可选定的对象移动到剪贴板。
	复制	复制选定的文本、控件或任何其他可选的对象到剪贴板。
	粘贴	把剪切或复制的文本、控件或任何其他可选定的对象放置在插入点位置。
	撤消	撤消上一个操作。
	重做	重作上一个撤消的操作。
	运行	执行“运行查询”、“运行表单”、“执行 <程序>”或“运行报表”命令。
	修改表单	对表单进行修改。
	数据库	指定当前的数据库。
	“命令”窗口	打开命令窗口，显示执行的命令，并提供输入命令的空间。
	“查看”窗口	打开表、建立关系以及设置工作区属性等。
	表单向导	运行 Visual FoxPro 6.0 表单向导。其中包括“表单向导”和“一对多表单向导”。
	报表向导	运行报表向导。其中包括“报表向导”、“分组/总计报表向导”和“一对多报表向导”。
	自动表单向导	不使用向导创建一个表单。
	自动报表向导	不使用向导创建一个报表。
	帮助	显示联机帮助

2．Color Palette（调色板）工具栏

Color Palette（调色板）工具栏用于为表单或报表中的控件指定颜色，如图 1-47 所示。表 1-2 列出了调色板工具栏的说明。

表 1-2 调色板工具栏说明

按 钮	功 能	说 明
	前景色	设置控件的默认前景色。
	背景色	设置控件的默认背景色。
	其他颜色	显示标准 Windows 颜色对话框，可定制用户自己的颜色。

3. Database Desiger（数据库设计器）工具栏

当对数据库进行操作时该工具栏显示，如图 1-48 所示。

图 1-47 调色板工具栏

图 1-48 数据库设计器工具栏

表 1-3 列出了数据库设计器工具栏的说明。

表 1-3 数据库设计器工具栏说明

按 钮	功 能	说 明
	新建表	使用向导或设计器创建新表。
	添加表	把已有的表添加到数据库中。
	移去表	把选定的表从数据库移走或从磁盘删除。
	新的远程视图	使用向导或设计器创建远程视图。
	新的本地视图	使用向导或设计器创建本地视图。
	修改表	在“表设计器”或“查询设计器”中打开选定的表或查询。
	浏览表	在“浏览”窗口中显示选定的表或视图并进行编辑。
	编辑存储过程	在“编辑”窗口中显示一个 Visual FoxPro 6.0 存储过程。
	连接	显示“连接”对话框，以便访问可用的连接；或通过“连接设计器”添加新的连接。

4. Form Control（表单控件）工具栏

当用户要设计表单时，只要打开表单设计器，Form Control（表单控件）工具栏就会自动打开。该工具栏用来在表单中创建控件。如图 1-49 所示。

图 1-49 表单控件工具栏

表 1-4 列出了表单控件工具栏的说明。

表 1-4 表单控件工具栏说明

按 钮	功 能	说 明
	选定对象	移动和改变控件的大小。在创建了一个控件之后，“选择对象”按钮被自动选定，除非按下了“按钮锁定”按钮。
	查看类	使用户可以选择显示一个已注册的类库。选择一个类后，工具栏只显示选定类库中类的按钮。
	标签	创建一个标签控件，用于保存不希望用户改动的文本，如复选框上面或图形下面的标题。
	文本框	创建一个文本框控件，用于保存单行文本，用户可以在其中输入或更改文本。
	编辑框	创建一个编辑框控件，用于保存多行文本，可以在其中输入或更改文本。
	命令按钮	创建一个命令按钮控件，用于执行命令。
	命令组	创建一个命令组控件，用于把相关的命令编成组。
	选项组	创建一个选项组控件，用于显示多个选项，并且只能从中选择一项。
	复选框	创建一个复选框控件，可以选择开关状态，或显示多个选项，可从中选择多于一项。
	组合框	创建一个组合框控件，用于创建一个下拉式组合框或下拉式列表框，用户可以从列表项中选择一项或人工输入一个值。
	列表框	创建一个列表框控件，用于显示可以选择的列表项。当列表项很多，不能同时显示时，列表可以滚动。
	微调控件	创建一个微调控件，用于接受给定范围之内的数值输入。
	表格	创建一个表格控件，用在电子表格样式的表格中显示数据。
	图像	在表单上显示图像。
	计时器	创建计时器控件，可以在指定时间或按照设定间隔运行进程。此控件在运行时不可见。
	页框	显示控件的多个页面。
	OLE 容器控件	向应用程序中添加 OLE 对象。
	OLE 绑定型控件	与 OLE 容器控件一样，可用于向应用程序中添加 OLE 对象。与 OLE 容器控件不同的是，OLE 绑定型控件绑定在一个通用字段上。
	线条	设计时用于在表单上画各种类型的线条。
	形状	设计时用于在表单上画各种类型的形状。可以画矩形、圆角矩形、正方形、圆角正方形，椭圆或圆。
	s 分隔符	在工具栏的控件间加上空格。

	生成器锁定	为任何添加到表单上的控件打开一个生成器。
	按钮锁定	使用户可添加同种类型的多个控件，而不需多次按此控件的按钮。
	容器	将容器控件置于当前的表单上。

5. Form Designer（表单设计器）工具栏

Form Designer（表单设计器）工具栏在使用表单设计器时会自动显示。如图 1-50 所示。表 1-5 列出了表单设计器工具栏的说明。

表 1-5 表单设计器工具栏说明

按 钮	功 能	说 明
	设置 Tab 键次序	在设计模式和 Tab 键次序方式之间切换，Tab 键次序方式设置对象的 Tab 键次序方式。当表单含有一个或多个对象时可用。
	数据环境	显示数据环境设计器。
	属性窗口	显示一个反映当前对象属性设置值的窗口。
	代码窗口	显示当前对象的代码窗口，以便查看和编辑代码。
	表单控件工具栏	显示或隐藏表单控件工具栏。
	调色板工具栏	显示或隐藏调色板工具栏。
	布局工具栏	显示或隐藏布局工具栏。
	表单生成器	运行表单生成器，它提供一种简单、交互的方法把字段作为控件添加到表单上，并可定义表单的样式和布局。
	自动格式	运行自动格式生成器，它提供一种简单、交互的方法为选定控件应用格式化样式。要使用此按钮先选定一个或多个控件。

6. Layout（布局）工具栏

使用布局工具栏可以在报表或表单上对齐和调整控件的位置。此工具栏在创建报表或表单时显示，如图 1-51 所示。

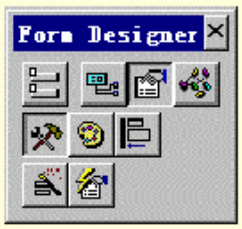


图 1-50 表单设计器工具栏

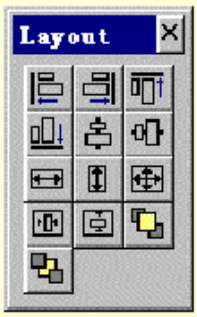


图 1-51 布局工具栏

表 1-6 列出了布局工具栏的说明。

表 1-6 布局工具栏说明

按 钮	功 能	说 明
	左边对齐	按最左边界对齐选定控件。当选定多个控件时可用。
	右边对齐	按最右边界对齐选定控件。当选定多个控件时可用。
	顶边对齐	按最上边界对齐选定控件。当选定多个控件时可用。
	底边对齐	按最下边界对齐选定控件。当选定多个控件时可用。

	垂直居中对齐	按照垂直轴线对齐选定控件中心。选定多个控件时可用。
	水平居中对齐	按照水平轴线对齐选定控件中心。选定多个控件时可用。
	相同宽度	把选定控件的宽度调整到与最宽控件的宽度相同。
	相同高度	把选定控件的高度调整到与最高控件的高度相同。
	相同大小	把选定控件的尺寸调整到最大控件的尺寸。
	水平居中	按照通过表单中心的垂直轴线对齐选定控件的中心。
	垂直居中	按照通过表单中心的水平轴线对齐选定控件的中心。
	置前	把选定控件放置到所有其他控件的前面。
	置后	把选定控件放置到所有其他控件的后面。

7．Print Preview（打印预览）工具栏

当创建一个报表时，该工具栏可用，用来更改报表预览的页面，如图 1-52 **所示。**

图 1-52　**打印预览工具栏**

表 1-7 **列出了打印预览工具栏的说明。**

表 1-7　**打印预览工具栏说明**

按　钮	功　能	说　明
	第一页	显示第一页。
	前一页	显示要打印报表的前一页。
	转到页	显示“转到页”对话框。
	下一页	显示要打印报表的后一页。
	最后一页	显示要打印报表的最后一页。
缩放	缩放	在预览窗口中按文本实际大小的 100%、75%、50%、25% 或 10% 缩放显示。
	关闭预览	关闭打印预览窗口。

8．Query Designer（查询设计器）工具栏

Query Designer（查询设计器）工具栏在使用查询设计器时显示，如图 1-53 所示。

图 1-53　**查询设计器工具栏**

表 1-8 **列出了查询设计器工具栏的说明。**

表 1-8 查询设计器工具栏说明

按 钮	功 能	说 明
	添加表	显示添加表或视图对话框，从而能够向查询添加一个表或视图。
	移去表	从设计器窗口的上窗格中移去选定的表。
	添加联接	在查询中的两个表之间创建联接条件。
SQL	显示/隐藏 SQL 窗口	显示或隐藏建立当前查询的 SQL 语句。
	最大化/最小化上部窗格	放大或缩小查询设计器的上窗格。
	查询去向	显示查询去向对话框，可以把查询结果发送到八个不同的输出地点。

9. Report Controls（报表控件）工具栏

Report Controls（报表控件）工具栏在设计报表或标签时显示，可利用此工具栏在报表或标签上创建控件，以用来显示数据源（如表或数据库等）的字段或这字段的组合，还可以显示来自数据源的图像或者播放来自数据源的声音等，如图 1-54 所示。

图 1-54 报表控件工具栏

表 1-9 列出了报表控件工具栏的说明。

表 1-9 报表控件工具栏说明

按 钮	功 能	说 明
	选定对象	移动或更改控件的大小。在创建了一个控件后，会自动选定“选定对象”按钮，除非按下了“按钮锁定”按钮。
A	标签	创建一个标签控件，用于保存不希望用户改动的文本，如复选框上面或图形下面的标题。
abl	字段	创建一个字段控件，用于显示表字段、内存变量或其他表达式的内容。
	线条	设计时用于在表单上画各种线条样式。
	矩形	用于在表单上画矩形。
	圆角矩形	用于在表单上画椭圆和圆角矩形。
OLE	图片/ OLE 绑定型控件	用于在表单上显示图片或通用数据字段的内容。
	按钮锁定	允许添加多个同类控件，而不需多次按此控件的按钮。

10. Report Designer（报表设计器）工具栏

Report Designer（报表设计器）工具栏在打开报表设计器创建报表是会自动显示。如图 1-55 所示。

图 1-55 报表设计器工具栏

表 1-10 列出了报表设计器工具栏的说明。

表 1-10 报表设计器工具栏说明

按 钮	功 能	说 明
	数据分组	显示数据分组对话框，从中可以创建数据组并指定其属性。
	数据环境	显示数据环境窗口。
	报表控件工具栏	显示或隐藏报表控件工具栏。
	调色板工具栏	显示或隐藏调色板工具栏。
	布局工具栏	显示或隐藏布局工具栏。

11. View Designer（视图设计器）工具栏

当在 View Designer（视图设计器）中设计或创建视图时，该工具栏自动显示，如图 1-56 所示。

图 1-56 视图设计器工具栏

表 1-11 列出了视图设计器工具栏的说明。

表 1-11 视图设计器工具栏说明

按 钮	功 能	说 明
	添加表	显示添加表或视图对话框，从而可以向设计器窗口添加一个表或视图。
	移去表	从设计器窗口的上窗格中移去选定的表。
	添加联接	在视图中的两个表之间创建联接条件。
SQL	显示/隐藏 SQL 窗口	显示或隐藏建立当前视图的 SQL 语句。
	最大化/最小化上部窗口	放大或缩小视图设计器的上窗格。

1.4 用选项卡设置系统环境

Visual FoxPro 6.0 提供了很多设置，可以通过这些设置来改变 Visual FoxPro 6.0 的环境，例如主窗口标题、默认目录、临时文件的存放位置和其他的选项。系统选项对话框中几乎列出了 Visual FoxPro 6.0 的所有系统设置项。选择 Tools（工具）菜单下的 Options（选项）将激活该对话框。当使用这个对话框时，除非按了“设置

为默认值”按钮，否则所有对设置的改变都是临时的，一旦退出 Visual FoxPro 6.0，这些改变就丢掉了。Visual FoxPro 6.0 将这些设置信息存放在登录数据库中。

该对话框共包括十二个选项卡，下面就详细介绍这些选项卡中各个选项的作用和功能。

1.4.1 调试选项卡

该选项卡用于指定调试器窗口的显示和跟踪选项，例如，调试器使用何种字体和颜色，是否显示行数等，如图 1-57 所示。其各项的意义如下。

1．Environment（环境）

该区用于定义调试环境，它包括如下选项：

（1）Debug frame：此时所有调试窗口均出现在 Visual FoxPro 6.0 主窗口以外，它们带有自己的菜单和工具条，这使得应用程序能较少受应用程序运行环境的影响。如果选择了该选项，可通过选择工具菜单中的“调试程序”选项来打开调试窗口。

（2）FoxPro frame：此时调试窗口将出现在 Visual FoxPro 6.0 主窗口中。但是，如果此时有打开的调试程序，则不能改变该选项。

2．Specify Window（指定窗口）选项区

该选项区用于指定所设置的窗口，它有如下选项。

（1）Call Stack（调用堆栈），选择该选项后，将在调试选项卡窗口的中间位置显示有关窗口的下列选项。

① 显示堆栈调用顺序，它在调用堆栈窗口程序列表附近显示一个数，通过高亮该数来指示当前执行程序的情况。

② 显示当前行的指示器，它决定是否在调用堆栈窗口时显示行指示器“→”。

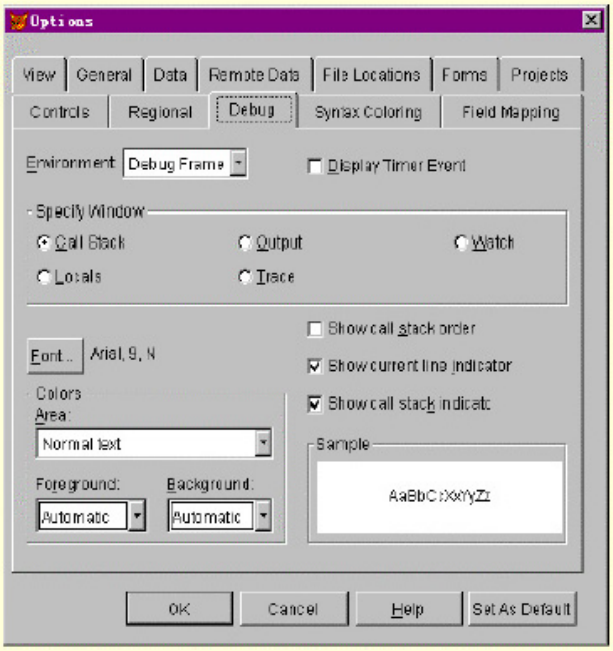


图 1-57 调试选项卡

③ 显示堆栈调用指示器，它决定是否在调用堆栈窗口时显示右向三角号，以指示在跟踪窗口显示的过程。如果当前行和调用堆栈指示器相同，Visual FoxPro 6.0 仅显示当前行指示器。

（2）Locals（局部变量），该窗口无附加选项。

（3）Output（输出），选择该选项后，系统将显示记录调试输出区，它包括如下选项。

① 记录调试输出，选择该项后，系统将复制调试窗口中的值到一文本文件。如果选择该项，则必须指定一具体的文件（其扩展名为.LOG）。

② 追加，将调试输出内容追加至指定文件已有内容之后。

③ 改写，将调试输出内容取代指定文件已有内容。

（4）Trace（跟踪），它显示下列几个选项。

① 显示行号，选择该项后，系统将在跟踪窗口代码行的左侧显示行号。
② 跟踪断点之间的部分，以慢速执行断点之间代码。
③ 在两代码行之间暂停，指定两代码行之间延迟的秒数。

（5）Watch（监视），该窗口无附加选项。

3. 所有调试窗口的通用选项

这些选项主要包括字体、区域选择以及选定区域的前景和背景颜色设置。

1.4.2 Syntax Coloring（语法着色）选项卡

Syntax Coloring（语法着色）选项卡用于指定命令窗口和所有编辑窗口中 Visual FoxPro 6.0 程序元素的颜色和字体，如图 1-58 所示。其中 Area（区域）选项用于选择要设置颜色的程序元素类型，如注释、关键字、数字等，Font style（字体）、Foreground（前景）和 Background（背景）选项用于设置选定区域的显示。

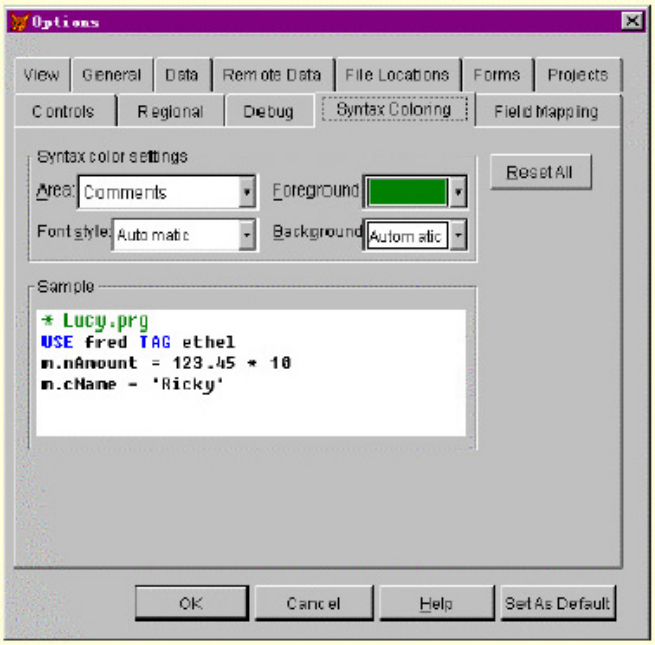


图 1-58 语法着色选项卡

1.4.3 Field Mapping（字段映象）选项卡

Field Mapping（字段映象）选项卡用于指定当从数据环境设计器、数据库设计器和项目管理器托动字段至表单时，将创建的控件类型。当选择修改按钮时，系统将显示字段类型映象对话框，可通过该对话框修改指定字段的映象类型，如图 1-59 所示。其他四个按钮的意义已十分明确，不再多作解释。

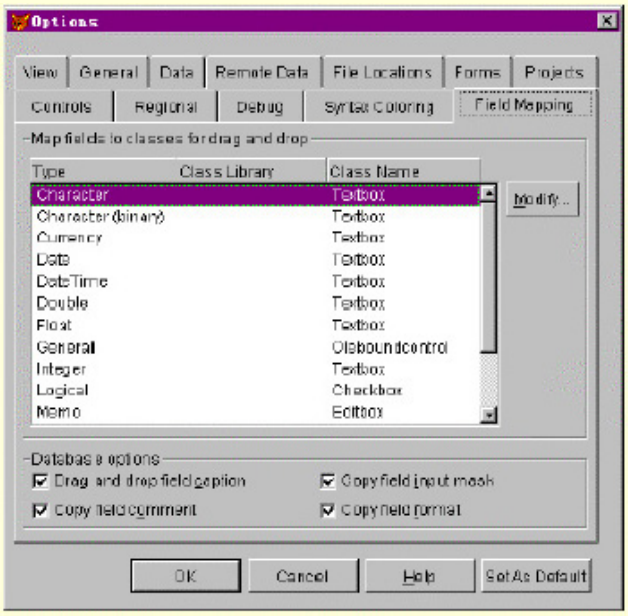


图 1-59　字段映象选项卡

1.4.4　View（视图）选项卡

View（视图）选项卡用于控制 Visual FoxPro 6.0 的显示特性，如图 1-60 所示。

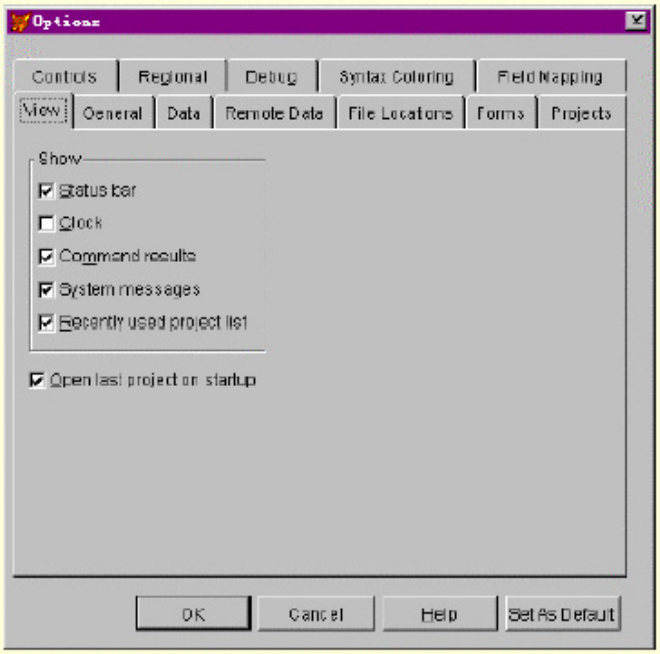


图 1-60　视图选项卡

1.4.5　General（常规）选项卡

General（常规）选项卡用于设置声音及其他一些选项，如图 1-61 所示。General 选项卡选项功能如下。

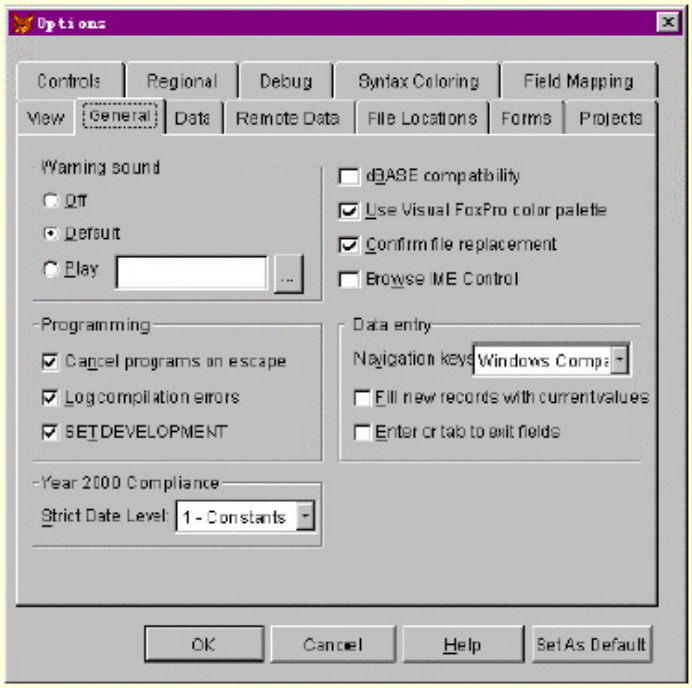


图 1-61 常规选项卡

1．Warning sound（警告声音）选项组

（1）关闭，该项用于关掉 Visual FoxPro 6.0 在发生错误时的声音提示；

（2）默认：将 Visual FoxPro 6.0 的错误提示声音作为系统的默认设置；

（3）播放：设置错误提示信息为一个 WAV 文件。既可在文本框中直接输入.WAV 文件的名字，也可单击文本框右侧的按钮打开对话框，然后从中选择一个.WAV 文件。

2．一般选项组

（1）与 dBASE 兼容：该项用于设定 Visual FoxPro 6.0 与 FoxBase＋ 和其他 Xbase 语言的兼容性。该项对应于命令 SET COMPATIBLE。

（2）使用 Visual FoxPro 6.0 调色板 指定使用默认的 Visual FoxPro 6.0 调色板，它对应于 SET PALETTE 命令。

（3）文件替换时加以确认：该项用于控制当新文件与已有文件重名时是否给出一个提示，以便确认这个文件的覆盖操作。

（4）浏览 IME 控件：当在浏览窗口激活一文本框时，该选项用于指定是否显示一输入方法编辑框（IME，Input Method Edithor）。

3．Programming（编程）选项组

（1）按 ESC 键取消程序运行 设置是否允许在允许程序时用 ESC 键中止程序。它对应于命令 SET ESCAPE。

（2）记录编译错误：允许或禁止登录编译错误。如果选中该项，则 Visual FoxPro 6.0 将记录被编译后的目标文件的生成日期。打开这一项可确保 Visual FoxPro 6.0 所执行的是该程序的最新版本。

4．Data entry（数据输入）选项组

（1）定位键：该项用于确定光标移动的方式是采用 Windows 兼容方式还是 MS－DOS 兼容方式。

（2）用当前值填充新记录：该项用来允许或禁止使用当前定义的值填充新记录。

（3）输入或跳转到已有字段：确定是否用 Enter 或 Tab 键离开一字段。

5．2000 Year Compliance

对字段进行限制，以解决 2000 年问题。

1.4.6 Data（数据）选项卡

Data（数据）选项卡用于处理数据和表，如图 1-62 所示。Data 选项卡选项的功能如下。

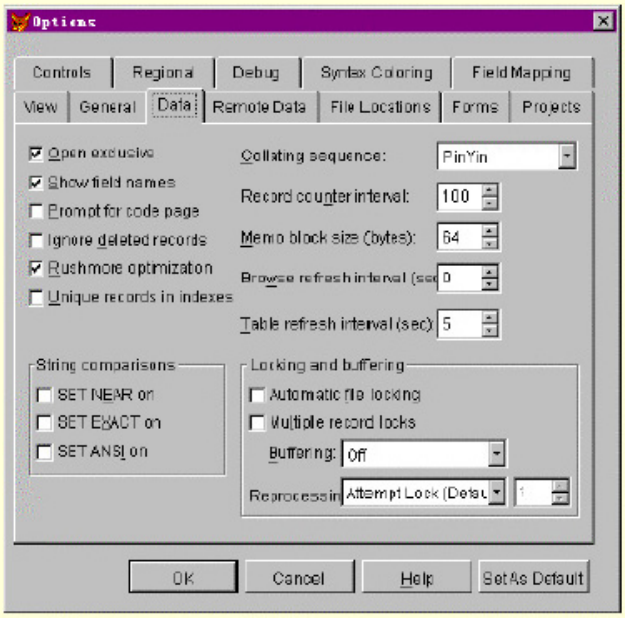


图 1-62 数据选项卡

1．数据选项组

（1）以独占方式打开：当选中该项时，所有表都以独占方式打开。它对应于 SET EXCLUSEVE 命令。

（2）显示字段名：设定在使用 AVERAGE，CALCULATE，DISPLAY LIST 及 SUM 命令输出时是否使用字段名作为每一列的题头。它对应于命令 SET HEADING。

（3）提示代码页：当选中该项时，每当打开一个没有附加代码表的页，Visual FoxPro 6.0 都会提示要求加入一个代码表。该选项对应于命令 SET CPDIALOG。

（4）忽略已删除记录：使用这一项来告诉 Visual FoxPro 5.0 是否处理被标记为删除的记录。该选项对应命令 SET DELETED。

（5）Rushmore 优化：该选项用来允许或禁止使用 Rushmore 优化技术。它对应于 SET OPTIMIZE 命令。

（6）在索引中不出现重复记录：该项用来确定是否在索引文件中全部保留具有相同索引键值的记录。它对应命令 SET UNIQUE。

（7）排序列：该选项确定在进行索引及排序操作时对字符型字段的排列顺序。如果使用语言不同，索引及排序结果就不同。它对应命令 SET COLLATE。

（8）记录计数器间隔：用这一项设定系统报告某命令已处理记录数的间隔。其范围从 1 条记录到 32767 条记录。默认设置为 100，即记录计数器在某命令每处理一百条记录时在状态条上报告一次。如果将这个间隔设为 1，则每处理一条记录都将在状态条上报告一次。它对应命令 SET ODOMETER。

（9）备注块大小：在这里指定用于存储备注字段的磁盘块的大小，以字节为单位。它对应命令 SET BLOCKSIZE。

（10）浏览窗口刷新时间间隔：设定隔多长时间（秒）刷新一次处于活动状态的浏览窗口。它对应命令 SET REFRESH。

（11）表刷新时间间隔：设定隔多长时间（秒）刷新一次处于活动状态的表。这一选项对应于在网络上共享文件时是非常有用的。它对应命令 SET REFRESH。

2．String comparisons（字符串）比较选项组

（1）SET NEAR on：当这一项被选中时，如果一个查找未找到匹配记录，那么 Visual FoxPro 6.0 就将记录指针定位到一个最接近的记录上。它对应命令 SET NEAR。

（2）SET EXACT on：设定当比较不同长度的串时，两串的每个字符是否都必须相同，尾部空格被忽略。它对应命令 SET EXACT。

（3）SET ANSI：设定当在 FoxPro 6.0 的 SQL 命令中用等号（.）比较两个不同长度的字符串时，在较短的字符串后面加空格以使长度与较长的字符串一致。它对应命令 SET ANSI。

3．Locking and buffering（锁定和缓冲）选项组

（1）文件自动锁定：它允许或禁止自动文件加锁。它对应命令 SET LOCK。

（2）多个记录锁定：允许或禁止锁定多条记录。它对应命令 SET MULTILOCKS。

（3）缓冲：设定在多用户环境下数据的更新与维护是否处于保护状态。该选项只有在选中了多个记录锁定后才可用。它对应 CURSETPROP〔 〕函数。

（4）重新处理：该选项用来设置当 Visual FoxPro 6.0 在锁定文件或记录失败时，再重复试几次，都不成功后放弃。它对应命令 SET REPROCESS。

1.4.7 Remote Data（远程数据）选项卡

Remote Data（远程数据）选项卡仅适用于处理远程（ODBC）数据，如图 1-63 所示。Remote Data 选项卡选项功能如下：

1．Remote view defaults（远程视图默认值）选项组

（1）共享连接：设定对新的视图是否使用当前连接。

（2）取备注字段：当选中这一项时，如果在视图的输出中有一个备注字段被激活，则备注字段内容将从数据取回。

（3）SQL 更新条件：该选项用于当试图更新一个视图中正在使用的数据时，系统根据这个标准来确定是否在用户打算更新这些数据之前已经被其他用户修改过了。如果远程表中这个选中的条件被改变的话，则更新操作失败。

（4）SQL 更新方法：控制如何在服务器上改变关键字段中的信息，这些选项决定了当记录中的关键字段被更新时，发送到服务器或源表的更新语句使用什么样的 SQL 命令。

（5）每次取的记录数：用于控制一次从远地数据源取回多少条记录。它右侧的全部复选框表示是否可以处理所有的记录。

（6）要取的最大记录：用该项指定在一个视图中可使用记录数的限制，它右侧的全部复选框表示是否可以一次取回所有的记录。

（7）使用备注字段的长度>＝：用于确定何时在视图的输出中将长字符型字段转换为备注型字段。

（8）成批更新的记录数：指定单条命令可更新的记录数。

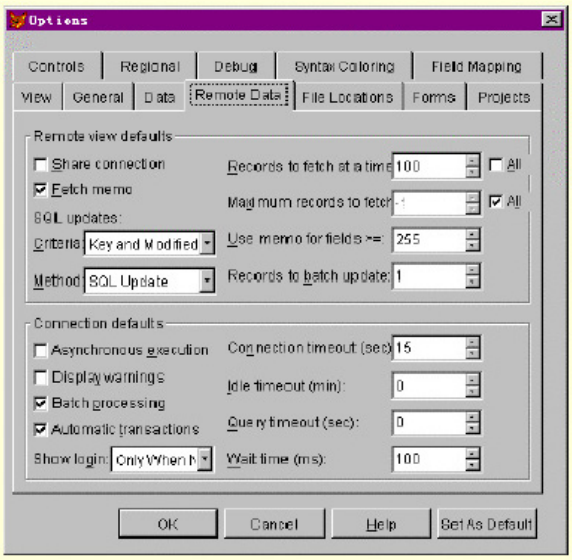


图 1-63 远程数据选项卡

2．Connections defaults（连接默认值）选项组

（1）异步执行：允许或禁止异步处理。

（2）显示警告信息：允许或禁止显示警告信息。

（3）批处理：允许或禁止批处理。如果选中的话，则 Visual FoxPro 6.0 只有在所有的单个操作都完成后才给 SQLEXEC（）的调用返回一个值。

（4）自动事务处理：事务处理是指可完成某种功能或动作的一系列处理步骤，它可使一系列的动作被作为一个工作单元对待。这一选项用来决定连接如何管理远地表的事务处理，选中时，相当于将 SQLSETPROP（）函数的 Transactions 属性值设为 1，未选中相当于将 Transactions 属性的值设为 2。

（5）显示登录信息：当在连接或视图定义中来指定登录信息时显示登录提示。

（6）连接超时设定（秒）：在这里给出一个时间限制（以秒为单位），在这个时间段内进行与远程服务器的连接。如果超出这个时间范围仍不能建立连接的话，Visual FoxPro 6.0 就给出一个错误信息。

（7）空闲超时设定（分钟）：在这里给出一个时间限制（以分为单位），如果在指定的时间内没有向服务器发出任何请求，则 Visual FoxPro 6.0 将终止连接。但是如果在连接被超时终止后向服务器发出了一个请求，那么 Visual FoxPro 6.0 会试图自动重新建立连接。

（8）查询超时设定（秒）：在这里给出一个时间限制（以秒为单位），用于确定服务器响应查询的时间。如果服务器处理查询的时间比这个限制长，Visual FoxPro 6.0 就会给出一个错误信息。

（9）等待时间（毫秒）：以毫秒为单位给出一个时间，每隔这么长时间系统查一下是否请求已经完成了。

1.4.8 File Locations（文件位置）选项卡

File Locations（文件位置）选项卡设置 Visual FoxPro 6.0 用到的不同文件的路径。既可直接在文本框中输入路径及文件名，也可单击每个文本框右侧的按钮调出对话框进行选择，如图 1-64 所示。

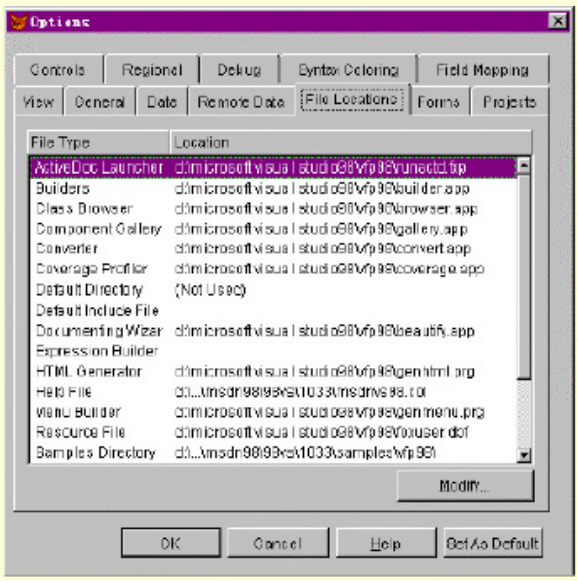


图 1-64 文件位置选项卡

该选项卡各选项功能如下：

（1）默认目录：在这里设置 Visual FoxPro 6.0 的默认工作目录。它对应于 SET DEFAULT 命令。

（2）搜索路径：这个设置告诉 Visual FoxPro 6.0 到哪些目录中去寻找在默认工作目录下找不到的文件。各路径间必须以逗号或分号隔开，使用这一选项右侧的按钮可以在加入多个路径时自动在各路径间插入分号。该设置对应于 SET PATH 命令。

（3）临时文件：设置 Visual FoxPro 6.0 存放临时文件的目录。

（4）帮助文件：指定 Visual FoxPro 6.0 显示帮助文件的方式，左侧的复选框用于打开或关闭帮助文件。如果帮助文件的扩展名是.HLP，则 Visual FoxPro 6.0 将使用 Windows 风格的帮助系统。若帮助文件的扩展名是.DBF，则 Visual FoxPro 6.0 将使用数据库风格的帮助系统。如果不加扩展名，Visual FoxPro 6.0 默认为.HLP，即 Windows 风格的帮助系统。这一设置对应于 SET HELP 命令。

（5）资源文件：为 Visual FoxPro 6.0 指定一个存储各种资源设置的文件，左边的复选框用于打开或关闭资源文件。该设置对应于 SET Resource 命令。如果没有选择一个资源文件，那么在选项对话框中的某些选项将不可用。

（6）转换器：指定一个用作转换应用程序的文件。它对应于_CONVERTER 系统内存变量。

（7）菜单生成器：指定用作菜单生成器的程序的文件名。它对应于_GENMENU 系统内存变量。

（8）拼写检查器：指定用作拼写检查器的文件名。它对应于_SPELL CHK 系统内存变量。

（9）生成器：指定用作生成应用程序的文件。它对应于_BUILDER 系统内存变量。

（10）向导：指定用作向导的文件。它对应于_WIZARD 系统内存变量。

1.4.9 Forms（表单）选项卡

Forms（表单）选项卡用于给表单生成器设定各种选项，如图 1-65 所示。

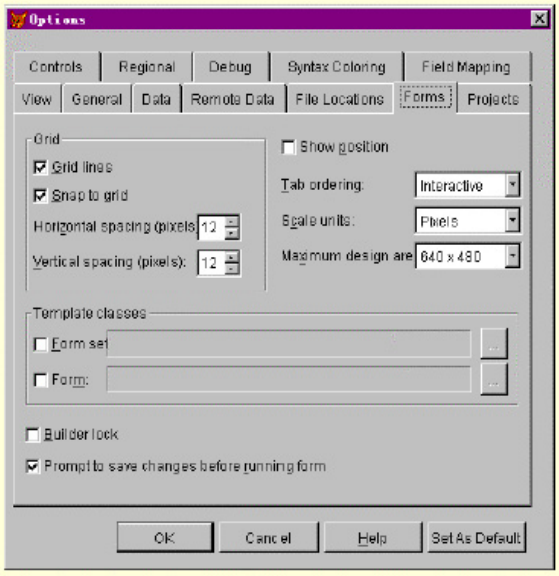


图 1-65 表单选项卡

该选项卡中的各选项功能如下。

（1）网格线：选中该项将在表单生成器主窗口上显示网格线，用以帮助各对象在表单上的定位、对齐。

（2）对齐网格线：它是一开关选项，用于设置是否自动将对象对齐网格线放置。

（3）水平间距：用于设置网格线水平宽度，其单位由度量单位区设置。

（4）垂直间距：用于设置网格线垂直宽度，其单位为象素。

（5）显示位置：该选项打开将在 Visual FoxPro 6.0 主表单的状态条的右侧显示当前选中对象的位置及尺寸信息，也可在查看菜单中设置显示位置。

（6）Tab 键次序：Tab 次序是指运行表单时按 Tab 键后，焦点在各对象间的转移顺序。这个选项用于指定改变对象的 TAB 次序的方式：使用“ SHIFT＋单击 ”的交互式鼠标方式或者使用“ 对象次序 ”对话框来设置 TAB 键次序。

（7）度量单位：为表单生成器和类生成器设定缺省的度量单位，可以指定以像素为基本度量单位或以字符为基本度量单位。

（8）最大设计区：设定在设计时表单的最大尺寸，对于这种窗口不能使其变得比指定的尺寸更大，在程序设计中一个好的习惯是尽量将表单大小设置成为用户可用的最小尺寸，这样对于所有用户来讲所设计的表单都是完全可见的（因为用户的监视器大小，Windows 的显示模式不尽相同，应用程序窗口设定太小会使用户见不到完整屏幕）。

（9）模板类选项组：可在这里指定类库和类来作为新表单和表单集的基类。如果什么都不指定，Visual FoxPro 6.0 将使用缺省基类，在这里指定类可以使用户设计的表单保持一致的风格。

① 表单集：在已注册类库下垃表中包含了所有的已经注册了的类库。可在选项对话框的控件表注册类库，类名下拉表中包括了所有在被选中的已注册类库中的表单集类。

② 表单：在已注册类库下垃表中包含了所有的已经注册了的类库，类名下拉表中包括了所有在被选中的已注册类库中的表单类。

（10）生成器锁定：生成器锁定的意思是当生成器被锁定后，每放一个控件到表单都将自动调出该种控件所对应的向导来修改这个控件的各种属性、表现等。

（11）提示在运行之前保存表单：在运行表单之前询问用户是否保存表单。

1.4.10 Projects（项目）选项卡

Projects（项目）选项卡用于为项目管理器设置选项，如图 1-66 所示。

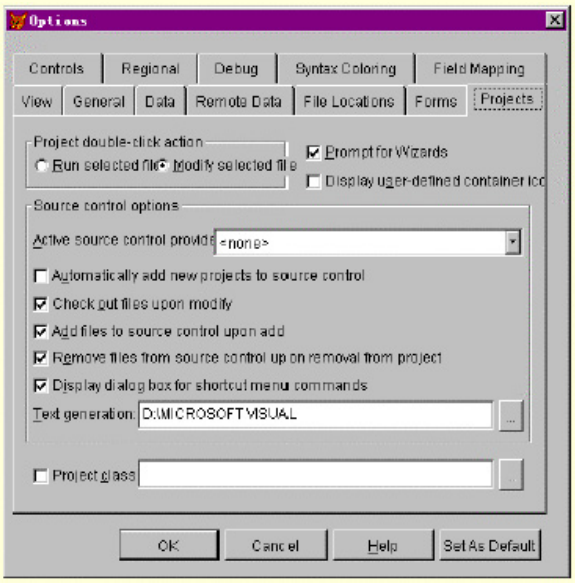


图 1-66 项目选项卡

该选项卡的各选项功能如下：

1．项目双击操作选项组

在这里设置当在项目管理器中双击一文件时是运行它还是修改它。

（1）运行选定文件：选中该按钮时，如果用鼠标双击一文件（在项目管理器中），则将运行该文件。

（2）修改选定文件：选中这一项后，如果用鼠标双击一文件（在项目管理器中），则打开该文件对其进行编辑。

2．向导指示

当选中这一项时，Visual FoxPro 6.0 将自动提示用户是否使用程序向导。

1.4.11 Controls（控件）选项卡

Controls（控件）选项卡用于注册可视类库，以便在设计表单时指定模板类，如图 1-67 所示。

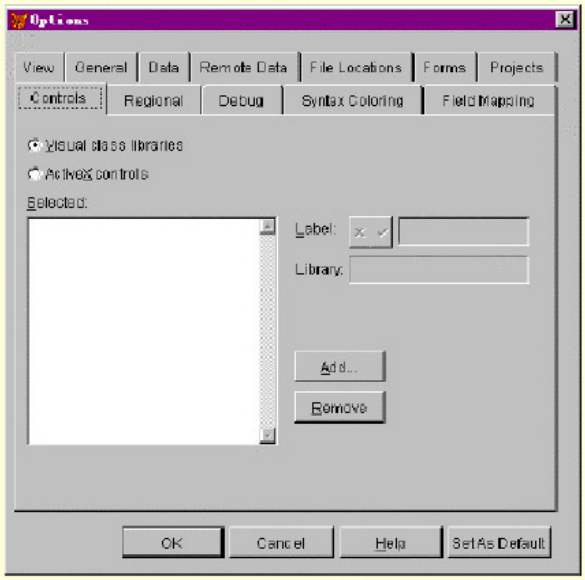


图 1-67 控件选项卡

该选项卡也有其他用途，比如通过表单控制工具条上的查看按钮可以指定哪些控件可以出现在表单控件工具条上。这样做是为了更方便地向表单中添加用户控件。Controls 选项卡功能如下。

1．Visual class libraries（可视类库）

这一项与“OLE 控制”选项是互斥选项，选中可视类库将该项“选定”列表框中的内容，使其只显示类库名称。

2．ActiveX controls（Active X 控件）

选中该按钮时，系统将打开显示选择区（位于窗口右下方）并更新选定列表框使其只显示 Active X 控件。

3．显示选择区

该区包括下列两个按钮。

（1）可插入对象：选中该项将在选定列表框中显示哪些可插入对象；

（2）控件：选中该项将在选定列表框中显示哪些属于可插入对象的 OLE 控件；

4．选定列表框

在这里指定哪些类库将被加入到表单控制工具条上。

5．Label（标签）

当选择了可视类库按钮时，将在这里显示当前类库的标签名。

6．Library（库）

当选择了可视类库单选钮时，在这里显示当前可视类库的路径。

6．Add（添加）按钮

当选中了可视类库单按钮时，使用这个按钮可增加类库到选定列表框中。

7．Remove（移去）按钮

当选中了可视类库按钮时，使用这个按钮可从选定列表框中移去一个类库。使一个类库出现在表单控制工具条上查看类按钮的菜单上的步骤如下：

（1）在对话框的顶部选择可视类库。

（2）单击添加按钮来寻找并选中一个可视类库，被选中的类库将出现在选定列表框中。

（3）单击确定按钮保存这个设置。

（4）随便打开一个表单文件，从表单控件工具条中选择查看类按钮，这时可以看到所增加的类库已出现在这个菜单中了。

1.4.12 Regional（区域）选项卡

Regional（区域）选项卡可设置日期、时间及数字的格式，可以临时或永久性地修改 Visual FoxPro 6.0 的系统设置。该选项卡的设置效果将反映在日期、时间及货币的输出上，如图 1-68 所示。

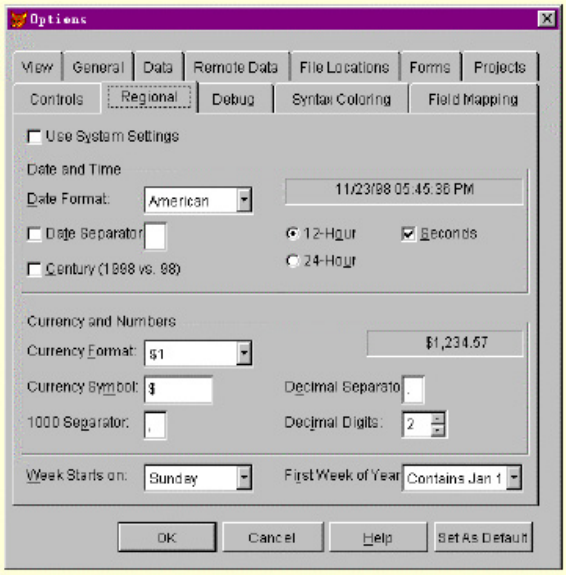


图 1-68 区域选项卡

它包括下列选项：

1．Use system setting（使用系统设置）

如果选中该项，则所有本表选项的设置都将从系统读出；显示设置依据在控制表中的设置。在这种模式下，大部分选项是只读的。如果不选这一项，那么就可以用自己指定的设置覆盖掉系统设置。

2．Data and time（日期和时间）选项组

在这一组中所作的修改都可直接在这一组的右上角的文本框中马上看到。例如，如果用户改变了时间中秒的设置，那么马上就可在右上角的文本框中看到结果。

（1）日期格式：在这里指定日期格式。尽管可选日期格式很多，但基本上只有三种形式：月/日/年、年/月/日、日/月/年。其他仅是分隔符不同。

（2）日期分隔符：该选项用于指定一个不同的日期分隔符，例如短划线，默认时日期分隔符为右斜线“/”。

（3）年：选中时，表示年的数字有 4 位长（如 1995），不选时表示年的数字只有 2 位（如 95）。

（4）12 小时制：将时间显示设为 12 小时制，用“am”和“pm”区分上下午。

（5）24 小时制：将时间显示设为 24 小时制。

（6）秒：选中时将显示秒数。

3．Currency and Numbers 货币和数字选项组

在这一组中所作的修改也都可直接在这一组的右上角的文本框中马上看到，例如，如果用户修改了小数分隔符，那么马上就会在右上角的文本框中看到结果。

（1）货币格式：设置货币符号是出现在数字串的前面还是后面。

（2）货币符号：改变货币符号的设置。

（3）千位分隔符：如果指定了一个分隔符，Visual FoxPro 6.0 系统将在大数字的小数分隔符向左每三位数字插入一该字符。这样做的目的是为了更清晰地阅读大数字，常用的千位分隔符是逗号。

（4）小数分隔符：指定一字符用作小数位开始的标记。

（5）小数位数：指定在小数分隔符右边的数字位数。

4．Week Starts on（星期开始于）

指定一周是从哪一天开始。

5．First Week of Year（一年的第一周）

指定哪儿是一年日历的开始。

1.5 本章小结

本章主要是让读者熟悉 Visual FoxPro 6.0 的系统环境，为后面的操作打下基础。只有熟悉了操作环境，才能游刃有余地进行各种操作，避免各种不必要的障碍和麻烦。本章主要介绍了系统的以下几个方面：

- Visual FoxPro 的初始界面
- 最常用的菜单和动态菜单
- 工具栏的定制和使用
- 使用命令窗口
- 利用选项卡设置系统环境

本章所介绍的内容将在以后的操作中时刻用到，读者可以在使用中加以熟练。

第二章　自由表

表是数据库的重要组成部分，对掌握对表的操作也是学习对数据库操作的基础。本章主要讲述对自由表的操作，使用户掌握对表的字段、记录的深入操作以及对多个表的操作，关于对数据库表的操作，将在其他章节作详细讲述。

2.1　自由表的数据类型

表是用来存放数据的，各种数据都属于某一字段，每一字段都应该有其数据类型，以存放一定类型的数据。Fox 提供的数据类型是一种弱数据类型，不少数据类型之间可以互相转换，Visual FoxPro 6.0 共提供了以下十三种数据类型可以用于字段中：

1．字符型（Character）

字符是大多数表格中常用的数据类型。字符型字段可以储存 1 到 256 个字符，由可打印的字符，如字母、数字、空格和标点组成。某些特定字符，如回车键，不能出现在普通的字符型字段中。必须将需要更多字符的字段定义为备注型字段。

2．货币型（Currency）

为了储存美元金额，Fox 使用了一种称为货币（Currency）的特殊字符。这种数据类型的最大值可以超过＄922 万亿，它缺省保留 4 位小数，在表单中需要 8 字节的存储空间。

3．日期型（Date）和日期时间型（Date Time）

Date 和 Date Time 这两种字段类型都储存日期，它们都需要 8 个字节以 YYYYMMDD 的格式储存日期。Data Time 字段用另外 6 个字节以 HHMMSS 的格式存储时间，其中 HH 由一个 24 小时的时钟进行记录。如果将日期型字段转化为日期时间型字段，时间将缺省为 12：00：00AM。

应该注意的是，日期范围为 01/01/100～12/31/9999，时间范围为 12：00：00A.M～11：59：59P.M。

4．双精度型（Double）

Double 字段是浮点数据类型字段，它以 8 个字节的压缩格式存储最多 18 个数字。实际上不管使用多少数字，字节个数总是 8。用户能做的只是决定小数点后的位数。

应该注意的是，双精度型数据取值的范围是：－4.94065648541247E－324～1.797693 13486232E＋308

5．整型（Integer）

整型数值是不带小数点的数值，整型数据在表中以二进制存储，占用内存少，只占 4 个字节。

整型数值的数据取值范围是：－2147483647～2147483646。

6．浮点型（Float）和数值型（Numeric）

Float 和 Numeric 都支持最多 19 位小数的 20 位数字，但每个数字都需要 1 字节的存储空间。Foxpro 同等的处理这两种数据类型，所以它们的精度是相同的。

与 Double 字段不同，Float 和 Numeric 字段允许指定所需字节的数目，因为 Foxpro 用单独的字节存储每个数字的 ASCII 码。

如果字段的宽度值小于正在存储的值，Visual FoxPro 就会强迫以*号存储。

应该注意的是，浮点型数据类型的取值范围为：－9999999999E－19～0.9999999999E＋20。

7．常规型（General）

General 型最常见的用途是用来储存图形。General 字段是专门的 Memo 字段。Foxpro 把 Genral 型字段存储在表格的其他 Memo 字段使用的同一个.FPT 文件中，但不能用同一种方法使用它。它主要用于存储引用来链接 OLE 对象。

8．逻辑型（Logical）

逻辑型字段以.T.或.F.的格式存储二进制信息。逻辑型数据只存储具有两种状态的信息，如男或女、已婚或未婚。

9．备注型（Memo）

备注型字段存储超过 256 个字符的大型字符串。备注型字段实际上是一个 4 字节的引用，它指向一个实际的备注内容，而实际的内容被保存在一个单独的备注文件里，该文件的名称与所在表的名称相同，扩展名为.DBT，备注字段的长度只受磁盘空间大小的限制。

10．二进制字符型和二进制备注型

二进制字符型和二进制备注型字段是将数据以二进制格式存储，所存储的数据不受代码页改变的影响。

以上数据类型可以用于字段中，其中双精度型、浮点型、通用型（常规型）、整型、备注型、二进制字符型和二进制备注型只能用于字段中，其余的可以用于变量、数组和字段中。

2.2 表向导的使用

表向导中提供了一些现成的字段，可以通过选择表向导中的一些字段，同时设置其字段类型来快速地建立一个自由表或数据库表。

应用表向导创建自由表的步骤如下：

（1）选择 File 菜单下的 New 命令，系统会弹出如图 2-1 所示的 New 对话框。在 New 对话框中可以选择 Visual FoxPro 6.0 可以创建的文件类型，例如表、数据库、查询、报表或者程序等等。

（2）选择 Table，并按下 Wizard 按钮，系统便弹出表向导第 1 步的对话框，如图 2-2 所示。在此对话框中可以进行一些选择。

在 Sample Tables（样表）中可以选择与所要建立的表类似的表，因为这些表中可能包含需要的字段。

Add（加入）按钮可以加入另外的一些现成表。

图 2-1 新建对话框

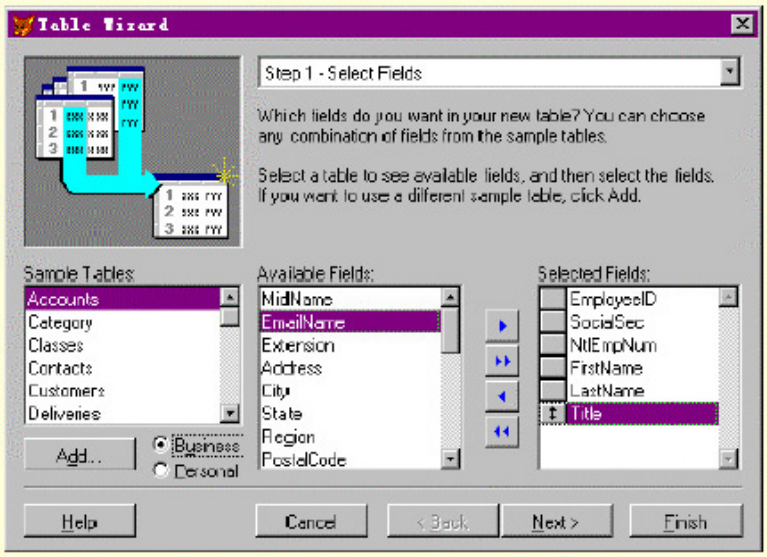


图 2-2　表向导第 1 步

Available Fields（可用字段）列出了所选择的表的所有的字段，可以从中选出需要的字段。

向导第一步对话框中的四个按钮的用途依次如下：

- 将某一选定字段从可用字段列表框中移入 Selected Fields（选定字段）列表框。
- 将可用字段列表框中的所有字段全部移入选定字段列表框。
- 将某一选定字段从选定字段列表框中移回可用字段列表框。
- 将全部字段从选定字段列表框中移回可用字段列表框。

（3）选择好需要的字段后按下 Next（下一步）后会进入如图 2-3 所示的对话框。在此可以选择是建立自由表还是数据库表，如果是数据库表则选择数据库表所在的数据库。按下 Next 后会进入如图 2-4 所示的表向导第 2 步对话框。

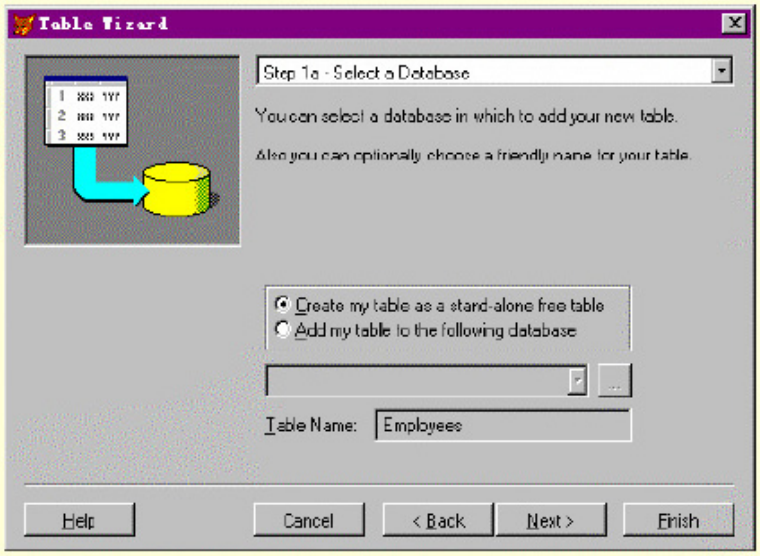


图 2-3　表向导第 1a 步

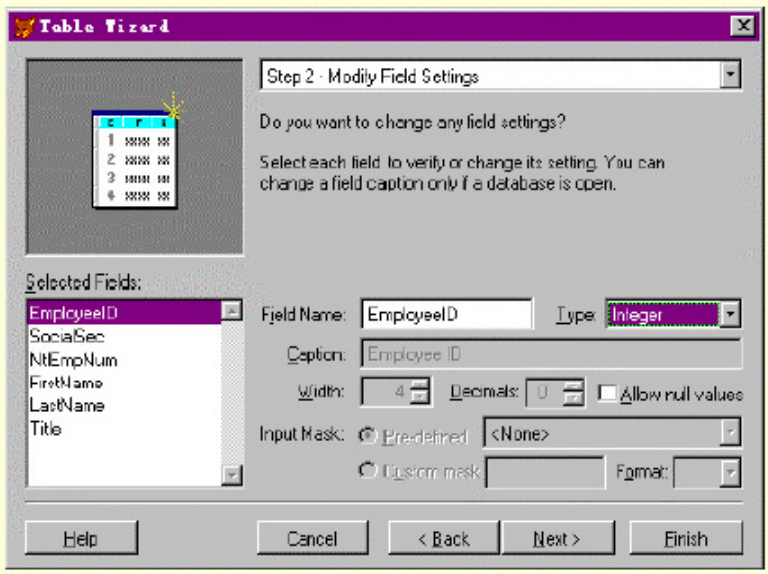


图 2-4 表向导第二步对话框

可以在此对话框中改变正在创建的表字段的设置。例如，可以改变一个字符字段的最大宽度。也可以指定字段是否包含 Null 值，或者更改字段标题或类型。

如果在数据库中创建表，也可以为每个字段类型选择预定义数据输入掩码，或者创建自定义输入掩码。Format 窗口显示所选择的掩码和格式。

（4）为每一个字段设置数据类型以后，按下 Next，会进入如图 2-5 所示的第 3 步对话框。在此对话框中可以设置表的索引，单击字段名前面的小方框，使其前面出现对号，则该字段即为索引字段。

选择在新表中成为主索引关键字的字段。同时可以指定其他字段为候选索引关键字。

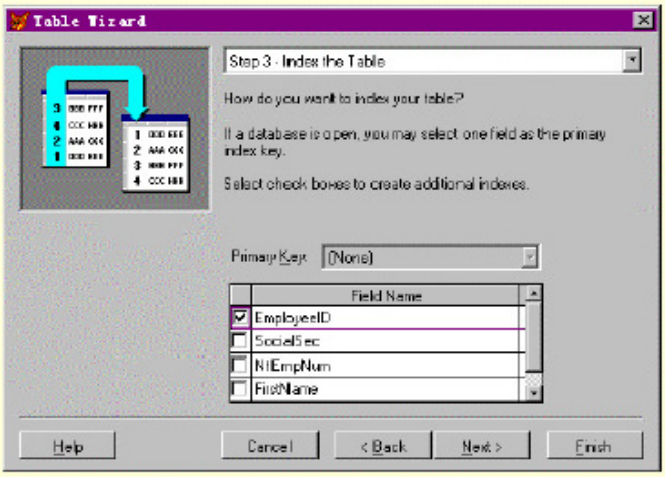


图 2-5 表向导第 3 步对话框

（5）设置好索引以后，按下 Next 按钮，系统即显示如图 2-6 所示的完成对话框。

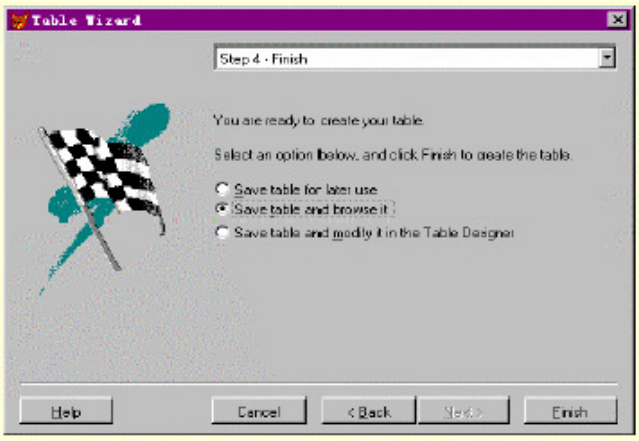


图 2-6 表向导第 4 步完成对话框

在此对话框中可以选择：

① 保存表，以后再用。

② 保存表并浏览。

③ 保存表并在表设计器中修改。

（6）选择“保存表并浏览”按 Finish（完成），会出现 Save As 对话框，输入表名并确定，建立的表就保存起来了。至此，利用表向导建立一个表的步骤就完成了。

2.3 表设计器的使用

在创建一个表之前，应该先确定表的结构和内容，这样，才能确定创建什么样的表。同时，也应该确定每个字段（Field）的数据类型，这样才能具备创建一个表的基本条件。在本节中利用表 2-1 所示的表的内容作为示例讲述对表的各种操作。

表 2-1 创建的自由表的内容

学 号	姓 名	年 龄	性 别	总成绩
0001	王刚	20	男	567
0002	李宁	21	男	587
0003	谢小飞	19	男	498
0004	刘虹	21	女	521
0005	杨兰	20	女	600
0006	徐浩	22	男	584
0007	王云	19	女	456

各个字段的数据类型见表 2-2。

表 2-2 表中字段的数据类型

学 号	数值型
姓名	字符型
年龄	数值型
性别	逻辑型
总成绩	浮点型
备注	备注型

确定了表的结构和内容以后，就可以在表设计器中建立一个新表了。建立新表的步骤如下：

（1）选择 File 菜单下 New 或按快捷键 Ctrl＋N ，也可单击工具条上的 New 按钮，则显示 New（新建）对话框。

（2）选择 Table 一项，并按下 New File 按钮。

（3）系统紧跟着会出现“ Save As ”对话框，选择一个路径并保存为“ educate ”文件名。

（4）这时会出现如图 2-7 所示的 Table Designer（表设计器）对话框。

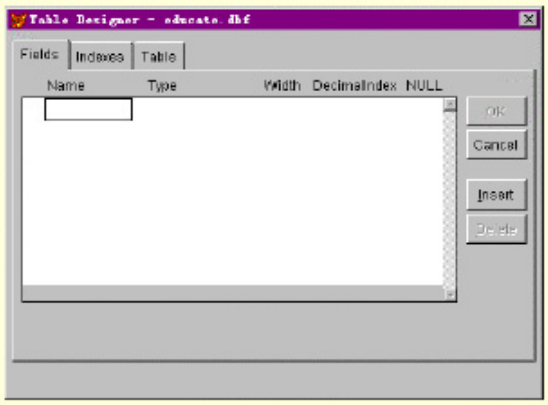


图 2-7　表设计器对话框

（5）在表设计器中输入如图 2-8 所示字段名并选择数据类型。具体操作如下：

① 选择 Table Designer 的 Fields 选项卡，在 Name 区域键入字段的名称。

② 在 Type 字段中，选择列表中的某一字段类型。

③ 在 Width 列中，设置以字符为单位的列宽。

④ 如果字段类型是 Numeric 或 Float 类型，设置 Decimal 框中的小数点位数。

⑤ 如果希望为字段添加索引，请在 Index 列中选择一种排序方式。

⑥ 如果想让字段能接受 null 值，选中“ NULL ”。

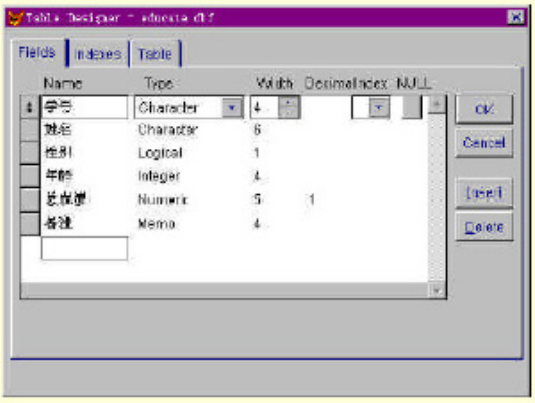


图 2-8　在表设计器中确定字段

（6）然后系统会显示如图 2-9 所示的对话框，可以通过选择 Yes 或 No 来选择是立即开始输入记录，还是在以后准备好所有记录后再打开表进行输入。

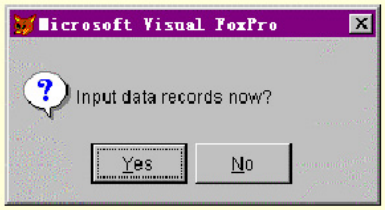


图 2-9　是否立即输入记录

（7）选择 Yes，系统会显示如下窗口，如图 2-10 所示，在此窗口下可以输入记录，完成表的记录的输入。

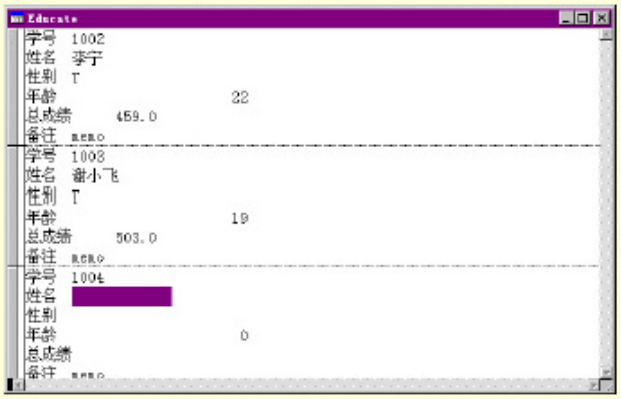


图 2-10 输入表的记录

此窗口被称为编辑窗口，在此窗口中，各个记录上下排列，记录中的各个字段也是上下排列，便于输入；另一种情况是浏览表中的记录时，如果想在这一个窗口中显示，可以选择 View 菜单下 Edit。

（8）输入完成后，一个完整的自由表就建成了。如果在 View 主菜单下选择 Browse，这时候就可以在浏览窗口中浏览全部的数据了，如图 2-11 所示。

图 2-11 在浏览窗口中查看记录

Visual FoxPro 在用户输入记录的过程中自动保存记录，所以就不必再保存了。

2.4 浏览表

2.4.1 无条件浏览表的内容

（1）单击 File 主菜单下的 Open 子菜单，或者单击工具条上的 Open 按钮，选择上面建立的表 educate。

（2）在 Visual FoxPro 主屏幕下，这时候什么也没有看到，点击 View 菜单下 Browse 菜单（这时候 View 菜单的标题已经加上打开的自由表的路径及文件名了），刚才选择的表便在浏览窗口显示出来了，如图 2-11 所示。注意：这时候，主菜单中多了一个“表”菜单。如果选择“显示”菜单下“编辑”菜单，表便在编辑窗口中显示，如图 2-12 所示。

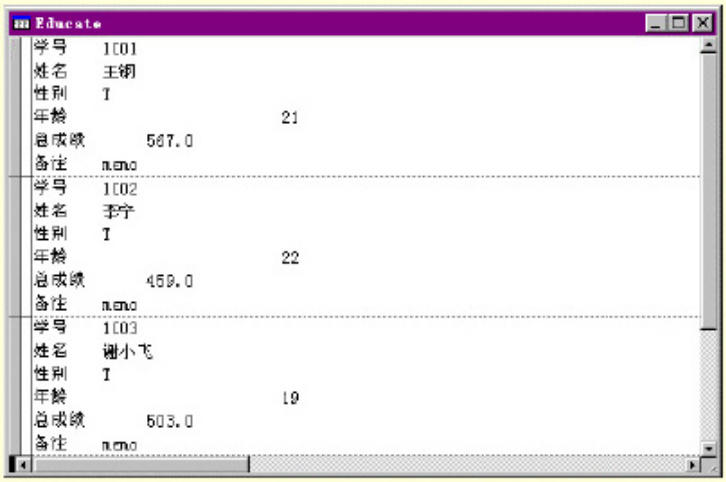


图 2-12 记录在编辑窗口中显示

如果记录很多或表格不足以容下所有的记录时，可以通过滚动条来浏览全部的记录。

在浏览方式下，可以通过鼠标拖动调整各列的顺序，改变列的宽度。选择 View 菜单下 Grid lines 菜单，可以显示或隐藏表格线，如图 2-13 所示。

把鼠标放到浏览窗口的左下角并拖动，可以把窗口分为两部分如图 2-14 所示，在默认情况下，两个窗口是互相链接的，即一个表的变化也会反映到另一个表中。去掉 Table 主菜单下 Link Partitions（链接分区）的选中状态，可以使它们成为两个功能相对独立的窗口。

图 2-13 隐藏网格线

图 2-14 在两个窗口下浏览记录

2.4.2 查看具体记录

当表的记录很多时，通过滚动条来查看记录是很不方便的，这时候用记录指针可以方便地找到所需要的记录。

Visual FoxPro 在向表中输入记录时为每个记录指定记录号，第一个记录号是“1”，下面的状态栏显示了全部的记录数和当前记录号。

如果想转到某条记录，选择 Table 菜单下 Go To Record（转到记录）下子菜单，如图 2-15 所示。

图 2-15 转到记录子菜单

（1）Top（第一个）转到表的第一条记录；

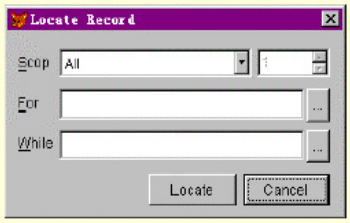
（2）Bottom（最后一个）转到表的最后一条记录；

（3）Next（下一个），Previous（前一个）分别为转到当前记录的下一个和前一个；

（4）如果选择 Record #（记录号），则显示如图 2-16 的对话框输入想转入的记录号，则光标移到此条记录。

（5）如果选择 Locate（定位），弹出如图 1-9 所示的 Locate Record（定位记录）对话框。

图 2-16 转到记录号



2-17 定位记录对话框

在对话框中输入表达式，由 Visual FoxPro 查找满足条件的一个或多个记录。

例如，想查找年龄大于 20 岁的同学，则在对话框中选择“ For ”右边的按钮，这时会弹出 Expression Builder（表达式生成器）对话框。在表达式生成器对话框中输入：年龄>20，按 OK，按定位记录对话框的 Locate，这时记录指针就移到满足条件的记录上。

通过上面的操作，可以看出对表进行操作时，基本上选择 View（显示）菜单和 Table（表）菜单下的功能。

2.4.3 设置条件常用的对话框

Expression Builder 表达式生成器是在表的操作和以后对数据库、查询、视图等操作中经常在设置条件时用到的一个对话框，在此对其界面与用法进行说明。

如图 2-18 所示是定位记录的表达式生成器对话框的界面。

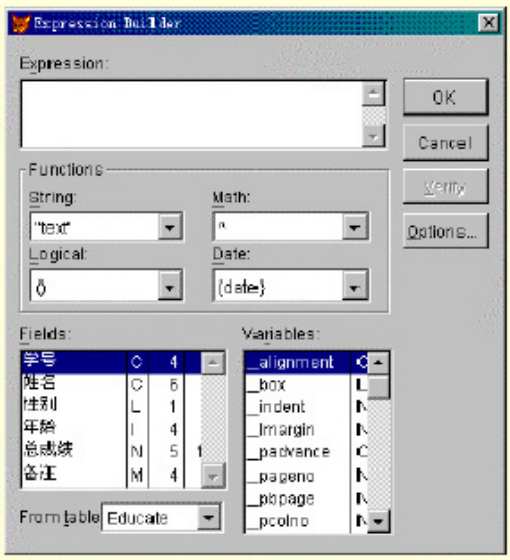


图 2-18 表达式生成器对话框

表达式生成器允许创建并编辑表达式。一个表达式可以简单得象一个字段名，也可以象一个包括 IF 函数。级连和数据类型转换的计算一样复杂。表达式生成器的主要目标是通过提供方法中每一步骤的合适选项的列表

使创建表达式更容易。该对话框可从设计器、窗口、生成器和向导中访问。

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

在表达式中处理字符串时，以下函数非常有用：

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

此对话框中有下列选择项。

1．Function 函数

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

字符串	列出可用的字符串函数
逻辑	列出可用的逻辑函数
数学	列出可用的数学函数
日期	列出可用的日期和时间函数
表达式	显示正在创建或编辑的表达式
字段	列出当前表或视图中的字段

若要将字段粘贴入表达式框，既可双击该字段也可选定该字段并按 ENTER 键。

若要显示不同表中的字段，应在来源于表框中选定另外的表。

2．From Table 源于表

列出打开的表和视图。可以选择表或视图来更新字段框。

3．Variables 变量

列出系统内存变量、数组和已创建的内存变量。

若要将一个变量粘贴入表达式框，既可双击该变量也可选定该变量并按 ENTER 键。

4．Verify 检验

如果相应的表已打开，检查表达式框中表达式的语法。如果表达式是有效的，表达式有效显示于状态栏中。如果表达式是无效的或是相应的表没有打开，Visual FoxPro 显示一条错误信息。该选项不可用于远程视图。

如果用户在表达式中包括一个用户自定义的函数调用，则检验将指示一条错误，但在运行中表达式求值时不会指示错误。

5．Options 选项

显示 Expression Builder Options（表达式生成器选项）对话框，如图 2-19 所示，可在其中设置表达式生成器的参数。

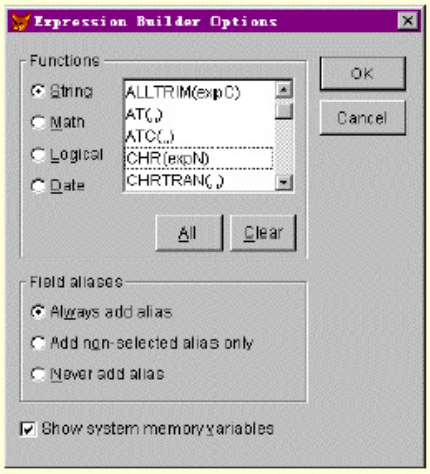


图 2-19 表达式生成器选项对话框

此对话框中有如下选择项：

（1）函数

使用这些选项可以更改显示在表达式生成器中的字符串函数、数学函数、逻辑函数和日期函数的内容。用户可选择把所有这些内容都显示在表达式生成器字符串框中，或仅选择所需要的内容。若要选定或移去多个项，在选择时按下 CTRL 键。

① 字符串：显示框中的所有字符串函数。所有选定的字符串函数都显示于表达式生成器中。

② 数学：显示框中的所有数学函数。所有选定的数学函数都显示于表达式生成器中。

③ 逻辑：显示框中的所有逻辑函数。所有选定的逻辑函数都显示于表达式生成器中。

④ 日期：显示框中的所有日期函数。所有选定的日期函数都显示于表达式生成器中。

通过按 CTRL + 鼠标左键来选择单独的项。用鼠标左键单击第一项然后对最后一项按 SHIFT + 鼠标左键可以选择连续的项组。

（2）全部

选定框中的所有函数，使其作为选项出现于表达式生成器中。

（3）清除

解除框中所有函数的选定，使其不出现在表达式生成器中。

（4）字段别名

这些选项指定了表别名或视图别名是否出现以及如何出现在表达式中。

（5）总是添加别名

指定表达式中使用的字段总是带有表名或视图名。

（6）仅添加未选定的别名

指定有多个表或视图打开的情况下，仅当表达式中使用的字段所在的表或视图没有出现在数据工作期窗口的别名列表中时，该字段才带有表名或视图名。别名列表中选定的表名或视图名不添加在表达式使用的字段中。

（7）不添加别名

指定在创建表达式时，字段名不包含它们所在表或视图的名称。

（8）显示系统内存变量

指定系统内存变量是否显示于表达式生成器中。如果仅需参阅已定义的系统内存变量，应关闭该选项。

2.5 修改表

2.5.1 修改记录

（1）打开一个表，在浏览方式或编辑方式下，将光标移到要修改的记录的字段值上。

（2）键入新的值，FoxPro 会自动保存所作的修改。

（3）若要修改备注型字段，在浏览窗口或编辑窗口中双击该字段，或按下“Ctrl＋Page Down”，这时会打开一个编辑窗口，就可在此窗口中查看、编辑其中的备注内容了。如图 2-20 所示。

（4）若要修改通用型字段中包含或链接的 OLE 对象，双击浏览窗口或编辑窗口中的通用型字段，就可以编辑此对象。可以编辑 Microsoft Word 文档或 Microsoft Excel 工作表，或者双击打开其父类的应用程序。

图 2-20 编辑备注字段

2.5.2 添加记录

在浏览窗口或编辑窗口中，选择 View 菜单下 Append Mode。就可以输入新的记录了。如图 2-21 所示。

图 2-21 追加方式下向表中添加记录

在新记录中填充字段，用 TAB 键可以在字段间进行切换。每次输入完一个记录，在窗口的底端会又出现一个空记录。

2.5.3 导入记录

（1）选择 Table 菜单中 Append Record（追加记录）子菜单，会弹出 Append From（追加来源）对话框。如图 2-22 所示。

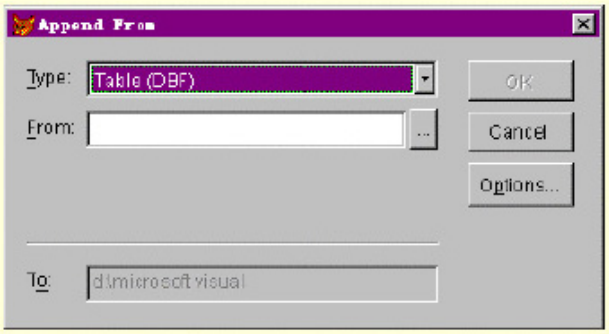


图 2-22 追加来源对话框

（2）在类型选项中选择文件的类型，这些文件的类型有以下几种：

表 2-3 可向 Visual FoxPro 表中追加的文件类型

文件类型	文件扩展名	说 明
Fox 表	.dbf	来源于当前数据库以外的 FoxPro 以 dbf 为扩展名地表文件或者一个 dBASE 文件。
制表符分割	.txt	用制表符分隔每个字段的一种文本文件。
逗号分割	.txt	用逗号分隔每个字段的一种文本文件。
空格分割	.txt	用空格分隔每个字段的一种文本文件。
系统数据格式	.sdf	具有定长记录且记录以回车和换行符结尾的文本文件。
Microsoft Excel	.xls	Microsoft Excel（2.0、3.0、4.0 和 5.0 版本）电子表格格式。列单元转变为字段，行单元转变为记录。
Lotus 1-2-3	wks wk1 wk3	1-A、2.x 和 3.x 版本的 Lotus 电子表格。列单元转变为字段，行转变为记录。
Borland Paradox	.db	3.5 和 4.0 版本的 Paradox 表。

（3）在追加来源对话框中选择 From（来源于）右边的按钮，选择要追加的文件或表。

（4）同时可以选择对话框的 Options（选项）按钮，在弹出的追加来源选项对话框如图 2-23 中指定要追加的字段和记录条件，按 OK 按钮。这时就会在表中看到添加的记录了。

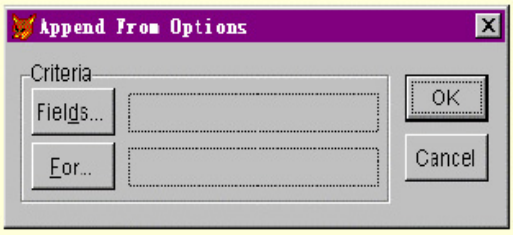


图 2-23 追加来源选项对话框

2.5.4 删除记录

当表中有了一些冗余的记录时，就需要删除它们。在 Visual FoxPro 中，删除记录需要两个步骤：先选定要删除的记录，再删除。

（1）打开表，在浏览窗口单击要删除记录的左边的小方框，标记要删除的记录。通过在此再次单击，可以取消所作的选择。

图 2-24 为删除的记录作删除标记

（2）选择 Table 菜单下的 Remove Deleted Records（彻底删除），才能将选择的记录删除。

通过以上的步骤删除记录后，就不能再恢复了。除非再次输入删除的记录，所以删除记录时一定要小心，确认确实是要删除的记录后，再进行删除。

如果要删除一些具有一定条件的记录时，可以通过按删除条件进行选定和删除。方法是：

选择系统主菜单“ Table ”下“ Delete Records（删除记录）”，系统会弹出如图 2-25 所示的对话框。

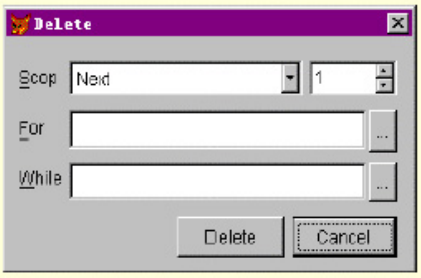


图 2-25 删除记录对话框

此对话框的界面与前面的介绍过的定位记录对话框一样，其用法也差不多。通过选择 Scop（作用范围）确定要删除的记录的范围。

通过在 For 输入框输入删除条件或点击 For 文本框右边的按钮，在弹出的“ 表达式生成器 ”对话框中输入删除条件，例如输入：

性别 = .T. AND 年龄 < 20

可以为年龄小于 20 岁的男同学加上删除标记。

在 While 输入框中也能输入一个条件判断语句，它的作用与 For 输入框基本相同。

加上删除标记的记录并没有从表中真正删除，浏览表时它仍存在。通过选择 Table 菜单下的 Remove Deleted Records（彻底删除），系统会提示是否真的要删除，如图 2-26 所示，按 Yes 就可以把有删除标记的记录全部彻底删除了。

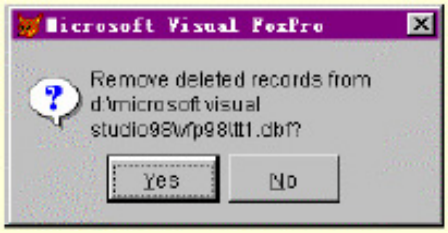


图 2-26 确认是否真的删除

2.6 修改表的结构

（1）打开一个表，在此处打开上面建立的表；

（2）单击 View（显示）菜单下的 Table Designer（表设计器），表设计器就会显示出来，如图 2-27 所示。

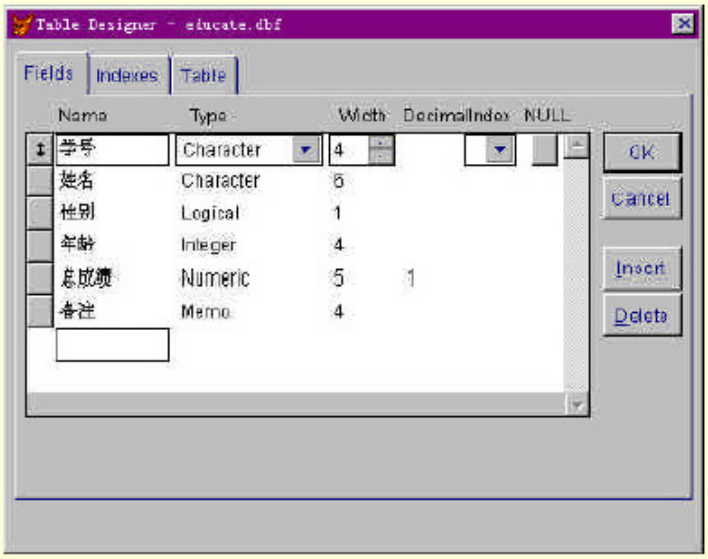


图 2-27 在表设计器中修改表的结构

（3）如果想添加新字段，将光标移到插入位置，单击 Insert 插入，键入字段名等相应项目。

（4）如果想删除字段，将光标移到要删除的字段上，单击 Delete 删除。

（5）如果想重命名字段，在字段的 Name 字段名文本框内键入新的字段名。

2.7 筛选表的字段和记录

如果当前已有一个记录了大量数据的表，但是在浏览的时候，并不是所有字段都需要浏览，或者有些字段并不重要，在浏览时它们只会添加不必要的麻烦。有时有些字段却又很重要的，需要限制对这些字段的访问权限。这时就需要定制一下表。

筛选表的操作用于浏览时只显示某些类型的记录。可以通过设置筛选器对窗口中显示的记录进行限制。这样就可以只看到所需要的记录了。

从 Table 菜单选择 Property（属性）子菜单，就会看到如图 2-28 所示的 Work Area Properties（工作区属性）对话框。

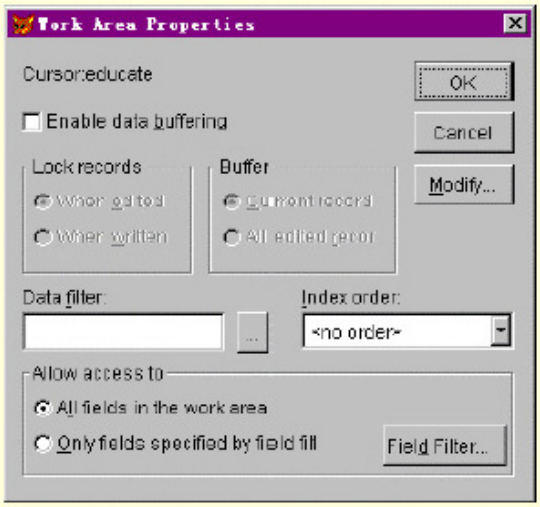


图 2-28 工作区属性对话框

单击 Data filter（数据过滤器）文本框右边的按钮，在弹出的表达式生成器对话框中输入过滤条件，这样表中就会只显示符合条件的记录。

如果单击 Modify（修改）按钮，会弹出表设计器对话框，对表的结构进行修改。

如果只想浏览某些字段，可以设置 Field Filter（字段筛选）来限制对某些字段的访问。其他的字段就不会再显示了。步骤如下：

（1）在 Allow access to（允许访问）区选中 Only Fields Specified by field fill（字段筛选指定的字段）。

（2）按下 Field Filter（字段筛选）按钮，就会弹出如图 2-29 所示的 Field Picker（字段选择器）对话框。

（3）在字段选择器对话框中选出想浏览的字段，如图中“学号”和“姓名”。

（4）确定后，在表中就会只看到所选择的字段了。如图 2-30 所示。

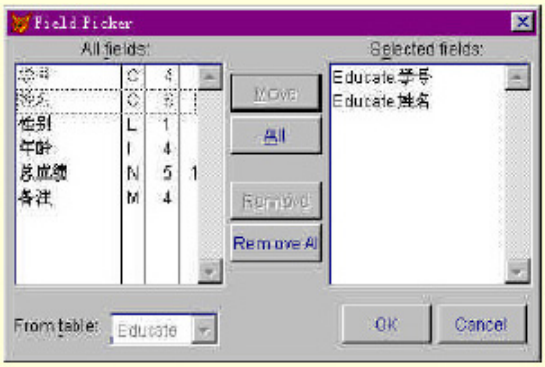


图 2-29 字段选择器对话框

图 2-30　经过筛选字段后的表

2.8　工作区的使用

有时要同时打开多个表进行浏览，例如，除了学生成绩表以外，还有家庭情况表、体育成绩表等等。这时就要用到工作区，以同时访问这些表。

如果要同时查看或访问多个表中的内容，就要在不同的工作区打开不同的表，也可以在不同的工作区中打开同一个表。

一个工作区是一个编号区域，它标识一个打开的表。Visual FoxPro 6.0 可以在 32767 个工作区中打开和操作表。工作区通常在应用程序中，通过使用该工作区中打开的表的别名来标识。表的别名是一个名称，它可以引用在工作区中的表。

使用 File 菜单下 Open 菜单项一次只能打开一个表。同时查看或访问多个表，或要查看打开的表的列表，需选择 Window 菜单下 Data Session（数据工作期）项。这时显示如图 2-31 所示的 Data Session（数据工作期）对话框。

下面是对此对话框中按钮功能的说明：

Open（打开）按钮可以添加新的表；

Browse（浏览）按钮浏览当前选择的表；

Properties（属性）按钮显示当前表的属性窗口

Close（关闭）按钮关闭当前选择的表；

Relations（关系）按钮使用表达式生成器，定义表或视图之间的关系。当两个表之间没有定义索引顺序时，对话框会在表达式生成器对话框之前显示设置索引次序对话框。

One to many（一对多）按钮显示创建一对多关系对话框，在该对话框中可以在子表和父表之间建立一对多临时关系。在 View（视图）对话框中选择两个或多个表，并建立它们之间的关系，然后单击 Relation（关系）按钮，这时该按钮就可用了。

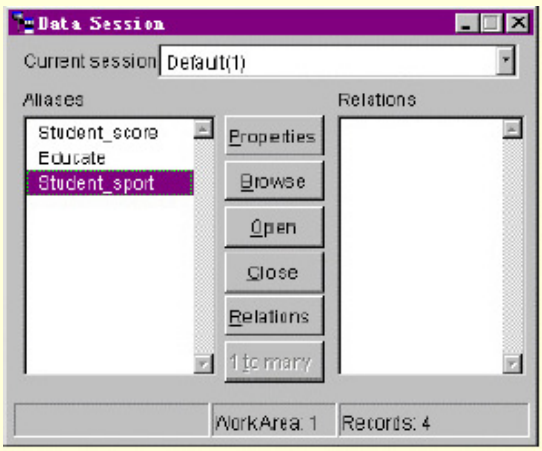


图 2-31　数据工作期对话框

2.9 使用命令操作表

2.9.1 操作表的基本命令

在此简要介绍一下一些较常用的操作表的命令的基本用法。对于其详细用法，请参考 Visual FoxPro 命令方面的参考书。

1. ADD TABLE

功能：向一个当前打开的数据库加入一个自由表。

格式：ADD TABLE <**表名**>| ? [NAME<**长表名**>]

说明：<**表名**> 指定加到数据库的表名。 ？ 打开对话框，选择一个表加到数据库里。NAME <**长表名**> 指定表的长表名。

2. BROWSE

功能：打开浏览窗口

3. CLOSE MEMO

功能：关闭一个或多个备注窗口。

格式：CLOSE MEMO<**备注字段** 1>[<**备注字段** 2>..]|ALL

说明：<**备注字段** 1>[<**备注字段** 2>..] 指定要关闭的备注编辑窗口所在的字段名。
ALL 代表各工作区打开的表中的所有备注字段的编辑窗口。

4. CREATE [<**文件**>| ?]

功能：建立一个新表

格式：CREATE [<**文件**>| ?]

说明：<**文件**>指定要创建的表的名字。？打开对话框，选择表的路径和表名。

5. DISPLAY STRUCTURE

功能：显示一个表的结构。

6. DISPLAY TABLES

功能：显示包含在当前数据库中的所有表及有关这些表的信息

7. LIST

功能：显示表的记录

8. MODIFY STRUCTURE

功能：显示表设计器，允许修改表的结构。

9. PACK

功能：永久地将当前表中有删除标识的所有记录删除，减少与此表相联系的备份文件的大小。

格式：PACK [MEMO][DBF]

说明：MEMO 选项删除备份文件的无用空间但不删除表中有删除标记的记录。DBF 选项删除表中有删除标记的记录。

10. RENAME TABLE

功能：为一个表重新命名。

格式：RENAME TABLE <**表名** 1> TO <**表名** 2>

11. SORT

功能：将当前表的记录进行排序，并将排序后的记录输出到一个新的表中。

格式：SORT TO <**表名**> NO <**字段名** 1>[/A|/D][C] [<**字段名** 2>[/A|/D][C]..]
[ASCENDING|DESCENDING][<**范围**>][FOR<**表达式** 1>]
[WHILE <**表达式** 2>][FIELDS<**字段列表**>|FIELDS LIKE<**框架**>|
FIELDS EXCEPT<**框架**>]
[NOOPTIMIZE]

说明：<表名>指定排序后新表地表名。
ON<字段名 1>根据此字段进行排序。
[A/D]按升序还是降序排列。
[/C]是否忽略大小写。
FOR<表达式 1>所有使表达式 1 为真地记录都参加排序。
WHILE<表达式 2>从当前记录到使表达式 2 不为真为止地所有记录均参加排序。
FIELDS <字段列表>指定由 SORT 命令创建的新表中包含的原始表中的记录。
FIELDS LIKE<框架> 指定原始表中匹配框架的字段包含在新表中。
FIELDS EXCEPT<框架>指定原始表中不匹配框架的字段包含在新表中。
<框架>是由字母和通配符组成的字符串。

12. USE

功能：打开一个表及其相关索引文件。
格式：USE[<表名>][IN<工作区>][EXCLUSIVE][SHARED][NOUPDATE]
说明：<表名>指定要打开的表。
IN<工作区>打开表所在的工作区域。
[EXCLUSIVE]以独占方式打开表。
[SHARED]以共享方式打开表。
[NOUPDATE]不允许对表作修改。

13. ZAP

功能：将所有的记录从表中移去，只留下表的结构。
格式：ZAP [IN<工作区>|<表别名>]
说明：IN<工作区>所有的记录要从其中移去的表所在的工作区。
IN<表别名>所有的记录要从其中移去的表的别名。

14. APPEND

功能：在当前表的末尾追加新的记录。
格式：APPEND [BLANK][IN<工作区>|<表别名>][NOMENU]
说明：BLANK 在当前表的末尾加入空记录。
IN<工作区>指定加入新记录所在的工作区。
IN<表别名>是加入新的记录的表别名。

15. DELETE

功能：标记要删除的记录。
格式：DELETE[<范围>][FOR<表达式>][WHILE<表达式>]
[IN<工作区>|<表别名>][NOOPTIMIZE]
说明：<范围>标记指定要删除的记录的范围。
范围选项是：ALL，NEXT<记录量>，RECORD<记录号>，REST。
其他选项见前面命令所述。

16. INSERT

功能：将一个新的记录插入到当前表中并显示该记录等待编辑。
格式：INSERT [BEFORE][BLANK]
说明：BEFORE 选项在当前记录之前插入记录。
BLANK 插入一个空记录，不显示编辑窗口。

17. SKIP

功能：在当前或指定的表中移动记录指针。
格式：SKIP [<表达式>][IN<工作区>|<表别名>]
说明：<表达式>指定记录指针移动的记录个数。
如果忽略则记录指针移到下一个记录。

除了以上命令与函数以外，下面的命令与函数也是在表的操作中常用的，关于其详细用法，可以查阅有关书籍。

AVERAGE、CHANGE、COPY MEMO、COPY STRUCTURE、COPY TO、COUNT、CREATE FROM 、CREAT TABLE、DISPLAY、DISPLAY STUCTURE、EDIT、EXPORT、FLUSH、IMPORT、PACK、RECALL、SUM、TOTAL、DELET、INSERT、LOCATE、SEEK、SET FILTER、SET KEY、SET NEAR、UPDATE。

2.9.2 创建表

在 Visual FoxPro 6.0 默认目录下创建“成绩表”，则在命令窗口中输入：

```
CREATE 成绩表
```

如果在其他目录下创建表，则输入：

```
CREATE ?
```

这时系统会弹出 New（新建）对话框，选择路径和文件名后，系统会弹出表设计器，这时可按照前面讲的，继续创建表。

上面的方法都是在表设计器中创建表，可以使用下面的命令创建表，并且指定表的字段，包括字段的类型，长度等。这里创建的“成绩表”，和前面的 educate 表具有同样的结构。可以使用如下命令：

```
CREATE TABLE 成绩表 ;
    ( 学号 N（4），姓名 C（10），;
    年龄 N（2）， 性别 L， ;
    总成绩 N（5，1），备注 M ）
```

2.9.3 浏览表的信息

用如下命令打开“成绩表”：

```
USE 成绩表
```

可以用如下命令在浏览窗口中浏览打开的表：

```
BROWSE
```

用如下命令在编辑窗口中浏览打开的表：

```
EDIT
```

用如下命令可以把表的记录列出在 Visual FoxPro 6.0 的主屏幕中：

```
LIST
```

以上命令的执行如图 2-32 所示。

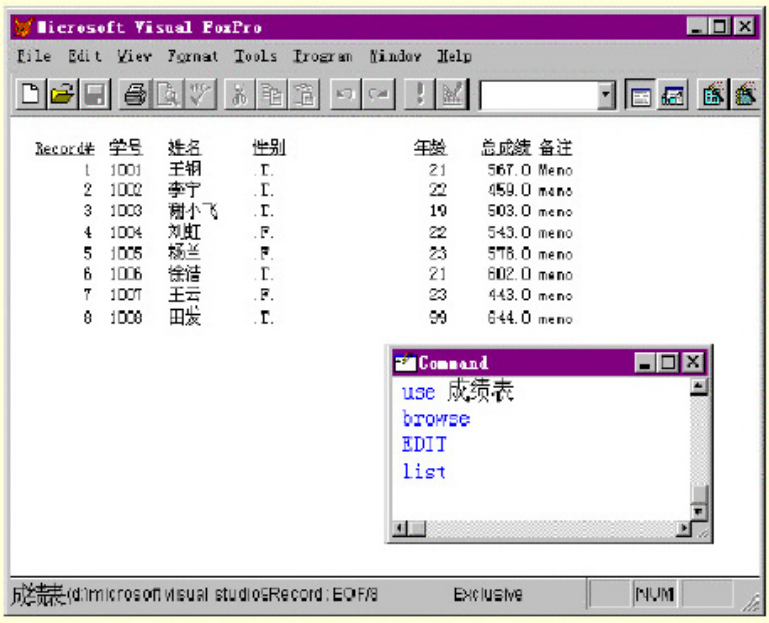


图 2-32 使用 LIST 命令浏览表的内容

2.9.4 在表中移动指针

将指针移动到第一条记录：

GO TOP

将指针移动到最后一条记录：

GO BOTTOM

将指针移到第三条记录：

GO 3

将指针向后移动 2 个记录：

SKIP －2

将指针定位到姓名为“杨兰”的记录

LOCATE FOR 姓名＝“杨兰”

2.9.5 修改表的内容

1．添加记录

（1）使用 INSERT－SQL 命令

INSERT－SQL 命令可以将指定的值，或者将来自于数组或内存变量的值添加到表中。执行该命令时，Visual FoxPro 先在表中增加一条空的记录，然后将数据加入此空记录。例如，向成绩表中学号和姓名字段加入新记录：

INSERT INTO 成绩表（学号，姓名）VALUE（10009，‘杨楠’）

使用此命令后，再使用 LIST 命令，可以看到表中多了一条记录，如图 2-33 所示。

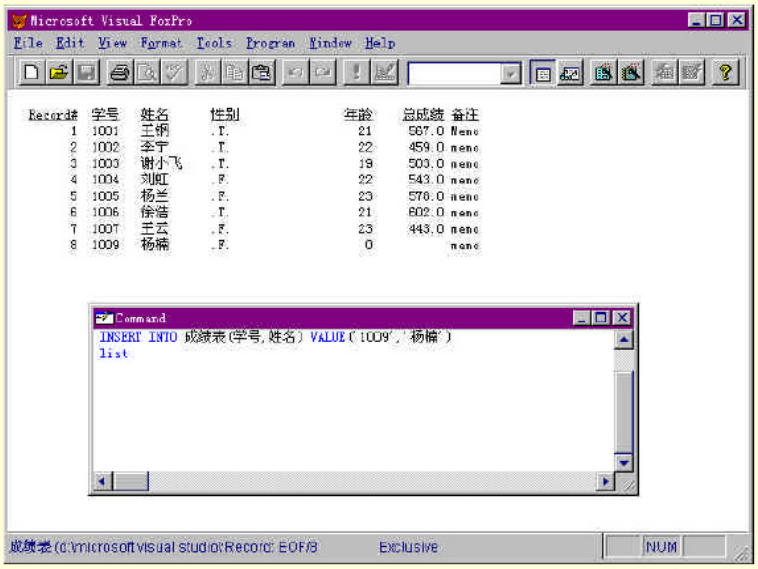


图 2-33 使用 INSERT 命令添加的记录

（2）使用 REPLACE 和 APPEND BLANK 命令

REPLACE 命令不会在表中添加记录，只是将当前记录的一个字段值替换为指定的值。例如：

REPLACE 姓名 WITH ‘刘冰’

可以将当前记录的姓名字段改为‘刘冰’。

APPEND BLANK 命令可以向表中添加一个空记录。所以，可以将 APPEND BLANK 和 REPLACE 命令连用，先用 APPEND BLANK 命令向表中添加一个空记录后，用 REPLACE 命令向当前空记录中加入数据。例如：

APPEND BLANK

REPLACE 姓名 WITH ‘刘冰’

2．从其他表中追加记录

Visual FoxPro 6.0 不但可以向表中逐条添加记录，而且还可以方便的从别的表中或文件中将所需要的数据追加到一个表中。这种成批的追加记录的命令有两种：APPEND FROM 命令和 IMPORT 命令。IMPORT 命令主要从其他格式的文件中提取数据，并将这些数据转化为 FOX 表中的数据格式，然后将这些数据添加到表中。

3. 删除记录

(1) 给表作删除标记。

给表作删除标记的命令如下：

使用 DELETE－SQL 命令，该命令可以指定要删除的记录的范围和条件。例如要为总成绩少于 500 分的记录作删除标记，可使用如下命令：

```
DELETE ALL FOR 成绩表.总成绩 <500
```

使用此命令后可以看到，满足条件的记录被作了删除标记。

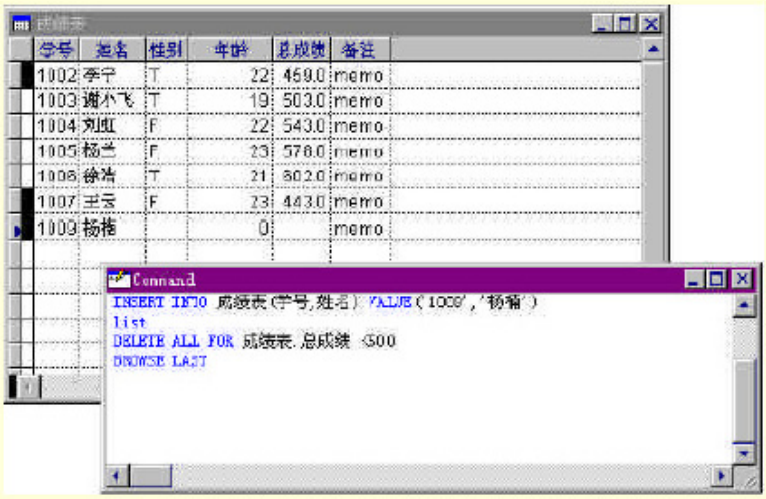


图 2-34 使用 DELETE 命令作删除标记

(2) 撤销记录的删除标记

在使用 PACK 命令或 ZAP 命令之前，可以使用 RECALL 命令来撤销记录的删除标记。同时，RECALL 命令还可以指定撤销删除标记的记录范围和条件。例如，使用命令：

```
RECALL ALL FOR 成绩表.总成绩>450
```

可以把总成绩大于 450 的删除标记撤销。

(3) 从磁盘上删除记录：

```
PACK
```

PACK 命令可以删除已有删除标记的记录：

```
ZAP
```

(4) ZAP 命令可以彻底删除表中的所以记录，只保留该表的结构。此命令相当于 DELETE ALL 命令后加上 PACK 命令。只是 ZAP 更方便。

2.9.6 修改表的结构

1. 修改表结构

修改表结构可以使用 ALTER TABLE 命令以编程方式来修改表的结构。也可以使用如下命令打开表“成绩表”，并使用命令打开表设计器来修改表的结构。

```
USE 成绩表
MODIFY STRUCTURE
```

2. 复制表结构

复制表结构就是用当前前的表的结构的所有字段来创建一个新表。

复制表的结构可以按照如下步骤和命令来实现：

(1) 使用前面讲过的命令打开一个表。

(2) 使用 COPY STRUCTURE EXETENDED 命令产生一个含有旧表结构信息的新表。

(3) 编辑包含结构信息的新表以更改任何根据此结构信息创建的新表的结构。

(4) 使用 CREATE FROM 命令来创建一个新的空表。

还可以使用 APPEND FROM 或数据复制命令来填充新表。

例如，创建一个名为 TEST 的新表，使用如上命令复制“成绩表”，使 TEST 具有“成绩表”相同的结构。

```
USE 成绩表
COPY STRUCTURE EXTENDED TO TEST
USE TEST
BROWSE
```

这时，就可以看到如图 2-35 所示的 TEST 表了。它具有和“成绩表”完全相同的字段和结构。

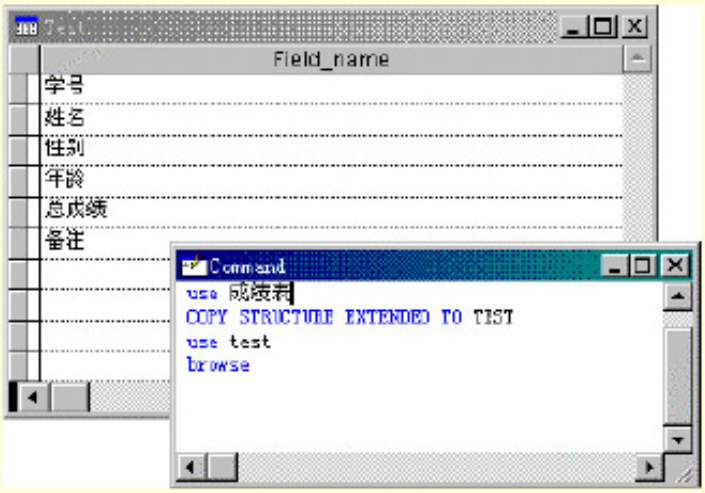


图 2-35 用复制表结构命令创建的新表

3. 添加字段

如果为当前表添加一个字段，可以使用 ALTER TABLE 命令中的 ADD[COLUMN]子句。例如，为上面建立的 TEST 新表增加一列“籍贯”字段，并且允许该字段为空值。可使用如下命令：

```
ALTER TABLE TEST ADD COLUMN 籍贯 C（20）NULL
```

4. 删除字段

可以使用 ALTER TABLE 中的 DROP [COLUMN]子句。例如，使用如下命令把上面添加的“籍贯”字段删除：

```
ALTER TABLE TEST DROP COLUMN 籍贯
```

5. 重命名字段

如果要重命名字段，可以使用 ALTER TABLE 中的 RENAME [COLUMN]子句。例如，可以使用如下命令把 TEST 表中的“总成绩”改为“成绩”：

```
ALTER TABLE TEST RENAME COLUMN 总成绩 TO 成绩
```

2.9.7 同时使用多个表

1. 在工作区中打开表

在命令窗口中使用带有 IN 的 USE 命令，如 USE 成绩表 IN 0，这样，就可以在最小的工作区中中打开成绩表。

使用 SELECT 命令切换工作区，然后用 USE 命令在该工作区中打开一个表。例如：

```
SELECT 0
USE 成绩表
```

2. 关闭工作区中的表

用 USE 命令的 IN 子句来指示要关闭的表所在的工作区，然后关闭此表。例如：

```
USE 成绩表
BROWSE
USE IN 成绩表
```

2.10 本章小结

自由表是存储数据的一个重要单位，本章向读者讲述了自由表的各种操作，包括创建表到多个表的使用，以及这些操作的各种方法。

本章主要讲述了如下内容：

- 操作表的各种命令
- 使用表向导创建表
- 使用表设计器创建表
- 查看表
- 修改表的记录和结构
- 操作多个表

第三章　为表创建索引

创建了一个表以后，如果表的记录特别的多，那么寻找所需要的记录将是一件很麻烦的事。怎样才能快速的找到需要的记录呢？为了解决这个问题可以建立索引。

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

可使用索引加速要排序记录或搜索记录的显示或打印速度。索引对于数据库内表之间创建关系的创建也很重要。

3.1　索引概述

前面已对索引的用途作了简介，表的索引类似于一本书的目录。一本书的目录列出了书的章节，以及每一章节所在的页码，使读者能方便的找到每一章节所在的位置，而不必一页页翻阅全书直至找到所需的内容。利用索引可对表的数据进行排序，表索引是一个记录号的列表，是包含指针的文件，它指向表的记录，并确定了记录的处理顺序。

若要按特定的顺序定位、查看或操作表中记录，就可以使用索引，对查看和访问的顺序进行控制。也可使用索引快速显示、查询或者打印记录。Visual FoxPro 6.0 使用索引作为排序机制，为开发应用程序提供灵活性和更多的功能。根据应用程序的要求，可以灵活地对同一个表创建和使用不同的索引关键字，按不同顺序处理记录。也能根据这些索引创建自定义表间关系，以便能准确地访问所需要的记录。

Visual FoxPro 6.0 的索引是由指针构成的文件，这些指针逻辑上按照索引关键字的值进行排序。索引文件和表的.DBF 文件分别存储，并且不改变表中记录的物理顺序。实际上，创建索引是创建一个由指向.DBF 文件记录的指针构成的文件。若要根据特定顺序处理表记录，可以选择一个相应的索引。使用索引不但可以加速对表的查看和访问，还可使用筛选索引把访问的记录限制在指定的数据上。为了在两个以上的表间建立关系，也必须先对建立关系的字段建立索引。

可以为一个表建立多个索引，每一个索引代表一种处理记录的顺序，索引保存在一个复合结构索引文件中。在使用表时，该文件被打开并更新。复合结构索引文件名与相关的表同名，扩展名为.CDX。

一个表中的索引也不是越多越好，不常用的索引反而会降低程序的执行速度，所以最好不要把表的每个字段都建立索引，更好的方法是应该用其他类型的索引文件来保存不常使用的索引。

3.2　自由表的三种索引

在 Visual FoxPro 6.0 中，系统为数据库表提供了四种类型的索引：主索引、候选索引、惟一索引和普通索引，为自由表提供了三种类型的索引：候选索引、惟一索引和普通索引。可使用表设计器中的索引选项卡来建立索引，如图 3-1 所示。

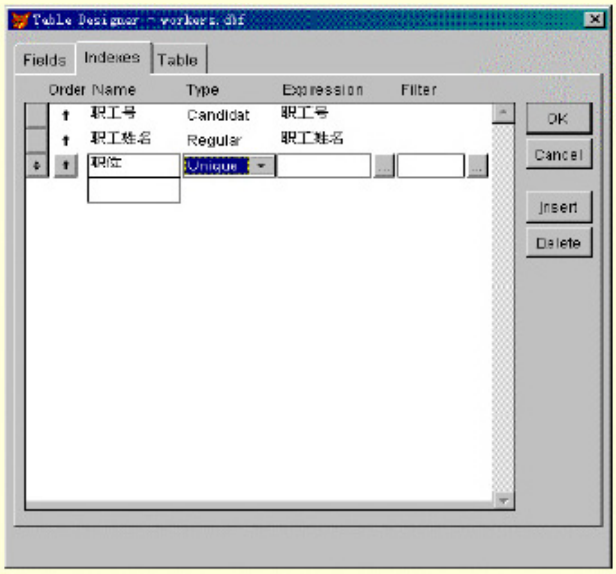


图 3-1 用表设计器为一个自由表建立索引

3.2.1 主索引

在 Visual FoxPro 6.0 中，主索引是一个永远不允许在指定字段和表达式中出现重复值的索引，也就是在数据库表的永久关联中建立参照完整性时主表和被引用表使用的索引。对于每一个表只能建立一个主索引，只有数据库表才能建立主索引，自由表不能建立主索引。另外，如果将一个字段指定为建立主索引的关键字段，那么当这个字段中出现重复值时，系统将会出现一个错误。

3.2.2 候选索引

候选索引也是在一个指定字段和表达式中不能出现重复值的索引，之所以将它命名为候选索引，主要是因为这种索引是作为一个表中主索引的候选者出现的。

对一个表而言，尽管只允许有一个主索引，但它的候选索引却可以有许多。而且也可以用它们在永久关联中建立参照完整性。

数据库表和自由表都可以有候选索引。

对于一个表，其主索引和候选索引都储存在.CDX 结构复合索引文件中，同时也存储在数据库的 Primary 和 Candidate 特性中，但是它不能存储在.CDX 独立复合索引文件和.IDX 索引文件中。这主要是因为主索引和候选索引必须和表同时打开和同时关闭，而.CDX 独立复合索引文件和.IDX 单项索引文件却不能做到这一点。

3.2.3 惟一索引

在 Visual FoxPro 6.0 中，惟一索引无法防止重复值记录的建立，但是，在惟一索引中，系统只在索引文件中保存第一次出现的索引键值，即只能找到同一个关键值第一次出现时的记录。对于重复键值的其他记录，尽管他们仍然保留在表中，但在惟一索引文件中却没有包括它们。惟一索引主要是为了向下兼容而提供的。

数据库表和自由表都可以有惟一索引。

3.2.4 普通索引

普通索引是一个简单的索引，是惟一索引、主索引和候选索引之外的另一种索引。在普通索引中，索引关键字段和表达式允许重复值的出现，可以用普通索引进行表中记录的排序和搜索。正是由于普通索引文件中不排斥重复索引关键值的出现，因此不但适合可重复键值表中的定序和搜索，也适合于一对多永久关联中“多”的一边（子表）的索引。

普通索引主要用于逻辑排序，以便快速查询。

在 Visual FoxPro 6.0 中，只有惟一索引和普通索引可以存放在.CDX 独立复合索引文件和.IDX 单项索引文件中。

3.3 索引文件

在介绍索引文件之前，现在先介绍两个概念：

1. 索引关键字

索引关键字是用来建立索引的字段表达式。

Visual FoxPro 6.0 使用索引关键字来显示和访问表中的记录。如果令某一字段为索引关键字，则在浏览表时，记录则按此字段的顺序排列。索引关键字通常是一个字段或字段表达式。虽然不提倡，但是也可以用以下各项的组合作为索引关键字：函数、内存变量、其他工作区中的字段、数组元素。

2. 索引标识

索引标识是关键字的名称。

索引标识可以自己指定，但必须以下划线、字母或汉字开头，而且不能超过 10 个字节。例如：可以建立“职工号＋姓名”为索引标识，此索引标识也可写成“职姓”。

Visual FoxPro 6.0 的索引文件有两种：传统的.IDX 索引文件和复合索引文件.CDX 文件。.IDX 索引文件只有一个索引关键字表达式，即只有一个入口。复合索引文件包含了多个索引关键字表达式，就好像是把多个.IDX 索引文件合并成为一个文件。

3.3.1 结构复合索引文件

结构复合索引文件的扩展名为.CDX，它是在表设计器中创建索引时系统自动生成的，如图 3-2 所示建立的索引就是结构复合索引，而且系统会把表设计器的索引保存在该表中。

结构复合索引文件名与相关的表同名，而且随着表的打开关闭而打开关闭。当用户对表中的记录进行添加、修改或删除等操作时，系统会自动维护.CDX 结构复合索引文件，使其和新的.DBF 文件相匹配。因为 Visual FoxPro 6.0 能自动维护修改改索引文件，使用户对表记录的操作简化。同时这种索引也是数据库表之间建立永久关系的基础，所以结构复合索引文件是 Visual FoxPro 6.0 的表中用的最多的一种索引结构。

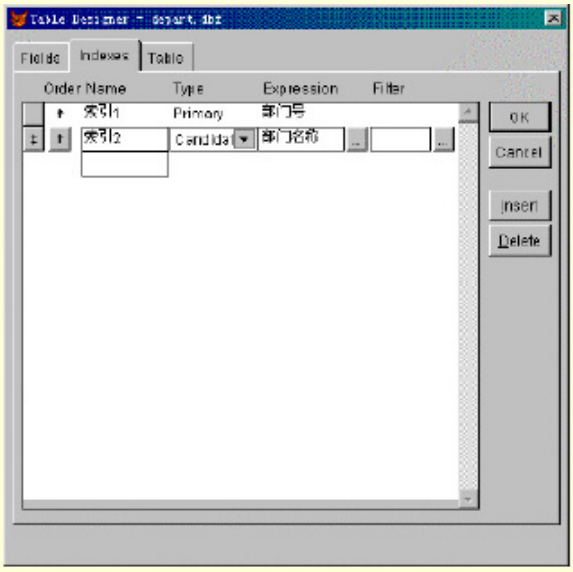


图 3-2 在表设计器中建立的结构复合索引

3.3.2 独立复合索引

独立复合索引又叫非结构化复合索引。

独立复合索引文件是另外建立的，它不象结构复合索引文件一样可以在表设计器中建立。

独立复合索引文件.CDX 可以看作是多个.IDX 文件的组合，实际上.IDX 索引文件完全可以加到.CDX 索引文件中去。独立复合索引文件和结构复合索引文件不同的是，当用户对表的记录进行修改时，独立复合索引文件不会自动打开。而且只有当该文件打开时，系统才能对它进行维护，如果要打开独立复合索引文件，则要用带

INDEX 子句的 USE 命令。

3.3.3 独立单项索引文件

独立单项索引文件的扩展名为.IDX，其主文件名不能和相关表同名，该索引文件和独立复合索引文件一样，不能随着相关表的打开而打开。

独立单项索引文件基于单字节表达式。通常在程序中使用独立索引作为临时索引，在需要时再重建或重新对索引排序，以用来优化应用程序的运行性能。一个表中可以有多个.IDX 独立索引。

另外需要说明的是，建立.IDX 独立单项索引文件的主要目的是为了原来的 FoxBASE+和 FoxPro 的索引文件格式相兼容，此时在用 INDEX 命令创建.IDX 文件时应该省略 COMPACT 子句。

3.4 创建结构复合索引

利用表设计器可以方便地创建结构复合索引。步骤如下：

（1）有如图 3-3 所示为一个自由表，现在为它建立一结构复合索引文件。选择 File（文件）主菜单下的 Open（打开），选择 “ Stuent_score ”表并单击“ 确定 ”。

图 3-3 将创建索引的自由表

（2）选择 View（浏览）主菜单下的 Table Designer（表设计器），进入表设计器对话框中，选择索引选项卡。如图 3-4 所示。

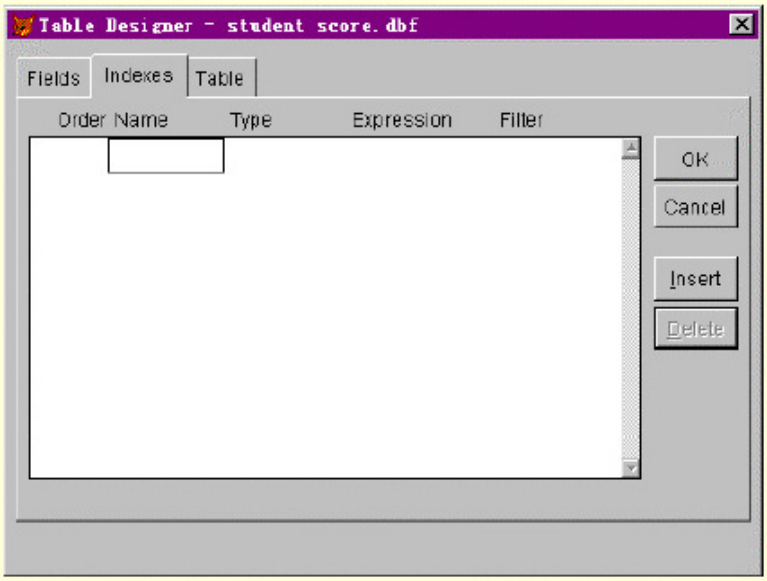


图 3-4 表设计器中的索引选项卡

（3）在如图 3-5 所示的索引选项卡中的 Name（索引名）中输入“ 索引 1“ ，在类型中选择 Candidate（候选索引），在表达式中输入“ 学号 ”或单击表达式右边的按钮，在弹出的表达式生成器中输入或选择“ 学号 ”。在此选择候选索引是为了使该字段必须具有惟一性，不能重复。在第二个索引“ 索引 2 ”中类型选择 Regular。

（4）按确定后系统保存建立的索引。

在 Filter 中可以输入筛选条件，筛选条件是对字段内容的过滤，如果有筛选条件，则索引表达式只对满足筛选条件的记录作索引。

Name（索引名）是索引的名称，可以随便指定。

Type（索引类型）即前面所述的四种类型（主索引、候选索引、惟一索引和普通索引），由于此处的表是一个自由表，所以没有主索引，只有 Candidate（候选索引）、Unique（惟一索引）和 Regular（普通索引）。

Expression（索引表达式）是包含要索引的字段及对该字段进行操作和转换的运算符和函数，索引表达式可以只包含一个字段，也可以是复杂的表达式。

在索引名的左边有两列按钮，按钮是为调整索引顺序的，选中后有上下方向的双箭头。如果要调整索引的顺序，就象以前表的操作中调整字段的顺序一样，在按钮上按下鼠标左键，拖到新的位置，释放左键即可。第二列按钮是来确定索引的升序还是降序，选中后会出现单箭头符号，箭头向上表示升序，向下表示降序。此按钮是一个开关按钮，按一下成为升序，再按一次为降序，使索引在升序和降序之间切换。

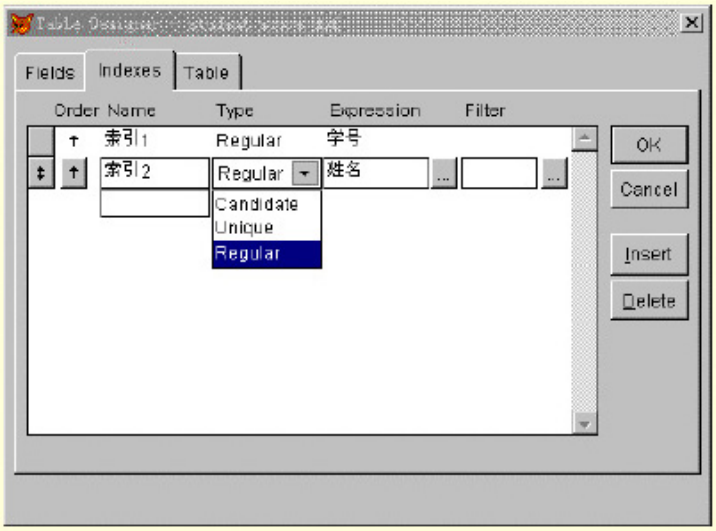


图 3-5 建立索引

3.5 用 INDEX 命令来创建索引

在索引文件中，除了结构复合索引文件可以在建立表时建立外，.CDX 独立复合索引文件和.IDX 单项索引文件都需要用命令来建立。

利用 INDEX 命令可以为当前表建立索引文件，该文件可以是独立索引文件，也可以是结构复合索引文件或非结构复合索引文件。

INDEX 的语法如下：

```
INDEX   ON < eExpression> TO <lINDFileName>| TAG <TagName>
        [OF <CDXFileName>]
[ FOR <lExpression>]
[COMPACT]
[ASCENDING|DESCENDING]
[UNIQUE|CANDIDATE]
[ADDITIVE]
```

在 INDEX 命令中<eExpression>是索引关键字表达式。索引要根据该表达式来建立。索引关键字表达式可以由表中的字段、内存变量、数组来构成字符、数值、日期或逻辑表达式。下面对此 INDEX 命令的各个命令子句及参数进行详细说明。

3.5.1 建立独立索引的命令

命令为：

INDEX ON <eExpression> TO <lIDXFileName>

现在为 Student_score **表根据“高数”字段建立独立索引文件，并且文件名为** Maths.IDX**，则命令如下：**

USE “Student score.dbf”

INDEX ON **高数** TO Maths.IDX

在以上命令行中，其中“高数”为关键字表达式 eExpression **，** Maths.IDX **为独立索引文件名。**

一旦用 INDEX ON **命令建立了一个独立索引文件后，该索引文件就自动处于打开的状态，成为当前索引。**

独立索引只能是升序的，不能建立一个降序的独立索引，但可以利用 INDEX **命令的索引关键字表达式来建立一个逻辑上的降序独立索引文件。**

假如要为上述 Student_score **表的“高数”字段建立逻辑上的降序独立索引文件，文件名为** DecMaths.IDX**，则命令为：**

USE “Student score.dbf”

INDEX ON －1*高数 TO DecMaths.IDX

建立好 DecMaths.IDX **独立索引文件以后，使用** LIST **命令来显示** Student_score **表中的记录，就会在屏幕上看到记录按照高数成绩按照由高到低的降序方式排列。图** 3-6 **显示了以上命令的执行结果。**

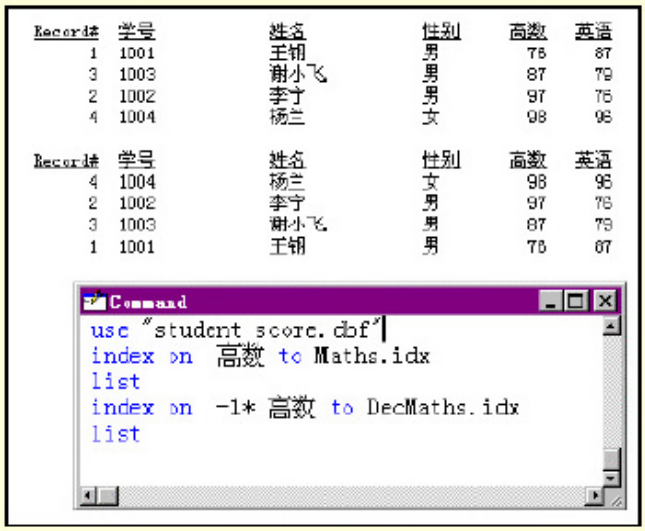


图 3-6 **建立独立索引文件**

3.5.2 建立结构复合索引文件

建立结构复合索引的命令：

INDEX ON <eExpression> TAG <TagName>

如果要为 Student_score **表根据“姓名”字段建立结构复合索引文件，则命令如下：**

USE “Student score.dbf”

INDEX ON **姓名** TAG **索引** 1

在以上命令行中，其中“姓名”为关键字表达式 eExpression **，“索引** 1**”为结构复合索引的索引标识。**

结构化复合索引文件名与表同名。

可以多次利用 INDEX ON **命令为结构化复合索引创建其他的索引标识。**

假如要为上述 Student_score **表的“英语”字段建立结构化复合索引文件，索引标识为“索引** 2**”，则命令为：**

USE Student_score

INDEX ON **英语** TAG **索引** 2

这样，在 Student score **表中利用两次** INDEX ON **命令建立了具有两个索引标识（索引** 1 **和索引** 2**）的结构化复合索引文件（**Student score.CDX**）。表设计器中的索引选项卡中可以看到建立的两个索引，如图** 3-7 **所示。**

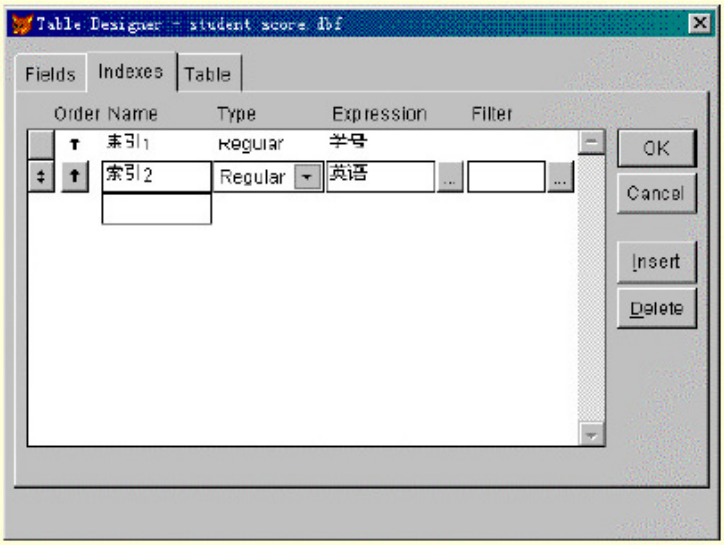


图 3-7　创建的结构化复合索引

3.5.3　创建非结构复合索引

创建非结构复合索引的命令如下：

INDEX ON <eExpression> TAG <TagName> [OF<CDXFileName>]

如果要为 Student score 表根据高数字段建立非结构化复合索引，此索引文件名为 Maths.CDX，索引标识为 Maths，则命令为：

USE “Student score.dbf”

INDEX ON 高数 TAG Maths　OF　Maths.CDX

由于非结构化复合索引与当前表不同名，所以必须使用[<OF<CDXFileName>]参数指明非结构化复合索引名。与结构化复合索引一样，多次利用 INDEX ON 命令可以为非结构化复合索引文件创建其他的索引标识。

非结构化复合索引和独立索引一样，都不能随着文件的打开而打开。如图 3-8 所示 MATHS 索引就是上面建立的非结构化复合索引。而“索引 1”和“索引 2”是结构化复合索引，它随着表的打开而打开。当此表再次打开时，MATHS 索引就不会再显示在此对话框中了，除非用户打开此 Maths.CDX 文件。

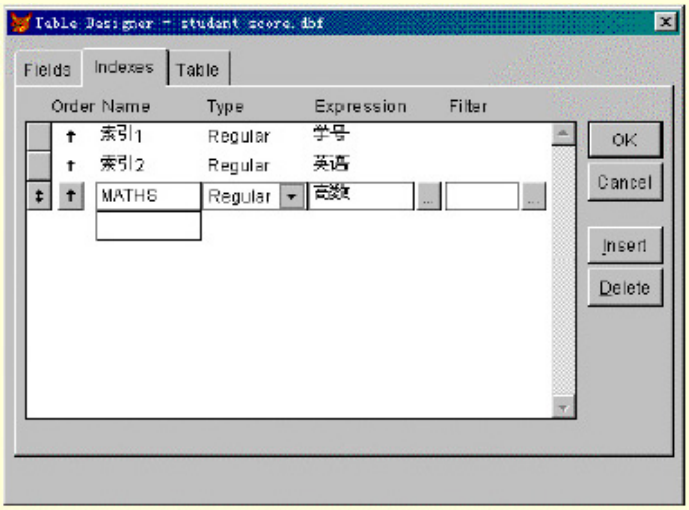


图 3-8　Maths 为非结构化复合索引

3.5.4　设置复合索引排序方式

在用 INDEX 为表建立复合索引文件时，[ASENDING/DESCENDING]参数是为了指定某一索引标识是按升序（ASCENDING）还是降序（DESCENDING）方式进行排序。前面已经说过，独立索引文件只能按照升序方式进行排序，所以在建立独立索引文件时，不能使用[ASCENDING/DESCENDING]参数。

使用如下命令可以为 Student score 表根据“英语”字段建立结构复合索引文件，索引标识符为 Eng，而且

是按照降序方式排序：

USE “Student score.dbf”

INDEX ON 英语 TAG Eng DESCENDING

通过图 3-9 可以看到设置索引排序方式前后的结果。

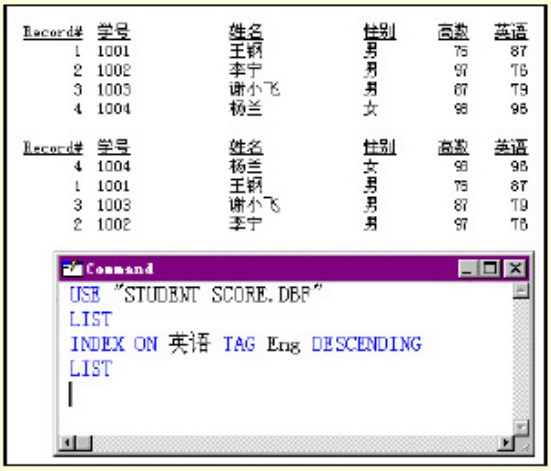


图 3-9 建立复合索引排序方式命令

3.5.5 为索引设置条件

在 INDEX ON 命令中的 [FOR < lExpression >]参数是在所要建立索引的表上建立有条件的索引。表中只有符合 FOR<lExpression>的记录才能出现在索引文件的索引关键字值列表中。

例如为 Student score 表中根据“高数”字段建立独立索引文件，索引文件名为 Math.IDX，并且只希望高数成绩高于 85 分的记录才会出现在索引文件的索引关键字中，则可以使用以下命令来实现：

USE “Student score.dbf”

INDEX ON 高数 TO Math.IDX FOR 高数>85

则用 LIST 命令来显示命令时，就会只能看到高数成绩大于 85 分的记录显示出来，并且记录显示的顺序是是一高数成绩升序方式。如图 3-10 所示。

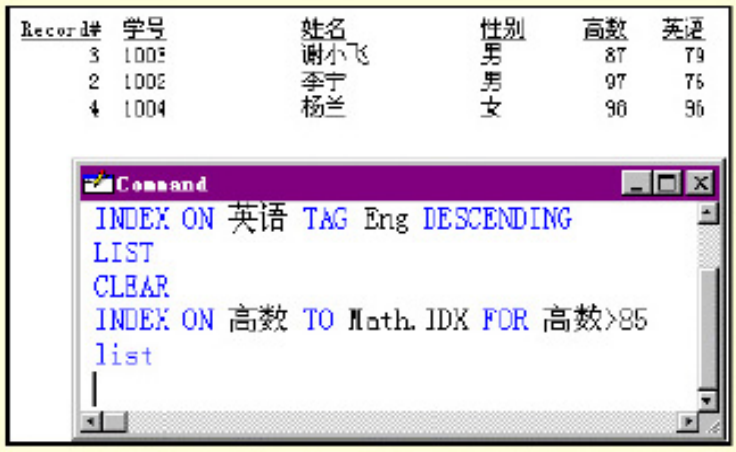


图 3-10 建立有条件独立索引

3.5.6 压缩处理索引文件

如果在建立独立索引文件时，INDEX ON 命令中未使用 COMPACT 关键字，那么，此独立索引文件不能获得压缩处理快速存取索引文件技术的支持。如果用户在建立独立索引文件时希望获得压缩处理快速存取索引文件技术的支持，从而大大提高独立索引文件的使用效率，那么在 INDEX ON 命令中使用 COMPACT 关键字可实现。

例如，在 Student score 表中根据“英语”字段建立独立索引文件，并且希望该索引获得压缩处理快速存取

索引文件技术的支持，可使用如下命令：

USE “Student score.dbf”

INDEX ON 英语 TO Eng COMPACT

另外一点说明的是，COMPACT 关键字仅对独立索引文件有效，对于符合索引，Visual FoxPro 6.0 自动为其应用压缩处理快速存取索引文件技术。

3.5.7 建立惟一索引和候选索引

如果在 INDEX ON 命令中使用了关键字 UNIQUE，则对拥有相同键值的若干个记录，只有第一条记录才会出现在该索引文件的关键字值列表中，其他具有此键值的所有记录都将被排斥在该索引文件之外。如果在 INDEX ON 命令中使用了关键字 CANDIDATE，则相当于在表设计器中建立候选索引。如图 3-11 所示为使用 UNIQUE 参数建立索引文件。

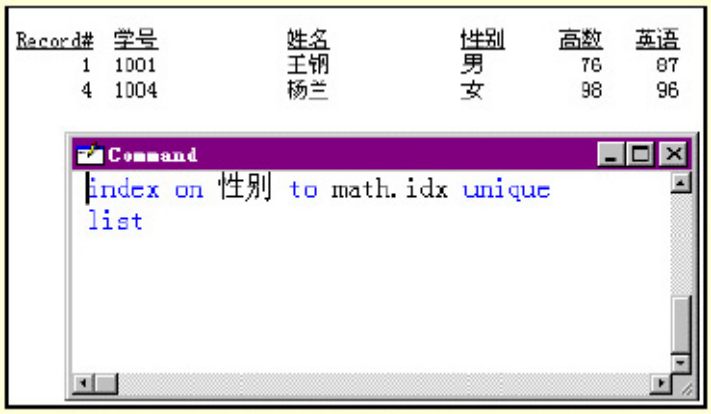


图 3-11 建立惟一索引的命令及显示的记录

3.5.8 关键字 ADDITIVE

当用户使用 INDEX ON 命令建立索引文件时，Visual FoxPro 6.0 通常先关闭除结构复合索引文件以外的已打开的索引文件，然后再建立新的索引文件并将此索引文件置于打开状态。如果希望在建立新的索引文件时不关闭以前已经打开的索引文件，那么应在 INDEX ON 命令中加入 ADDITIVE 关键字。

3.6 使用索引

3.6.1 使用非结构.CDX 索引

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

3.6.2 使用独立索引

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

通常可以使用独立索引作为临时索引，在需要它们时再创建或重新编排这些索引。例如，有一个只用于每季度或每年度总结报表的索引。如果把这个不经常使用的索引包括到结构 .CDX 文件中，便要求在每次使用表时都要维护它，因此最好创建一个独立 .IDX 索引。对一个特定的表，可创建任意多个 .IDX 文件。

3.7 维护索引标志

创建了.CDX 复合索引文件和.IDX 独立索引文件后，如果已不再使用.CDX 符合索引中的某些索引标识和某些.IDX 索引文件，最好要删除 .CDX 文件中的标识来删除不再使用的索引，或者通过删除 .IDX 文件本身来删除独立索引。删除无用的索引标识可以提高性能，因为 Visual FoxPro 6.0 不必再去更新无用标识，来反映表中数据的变化。所以为了提高系统的效率，需要及时清理无用的标识和.IDX 索引文件。

3.7.1 从结构 .CDX 文件中删除标识

若要删除在结构 .CDX 文件中的索引标识可以通过表设计器或使用命令从 .CDX 结构文件中删除标识。

（1）在表设计器中使用索引选项卡选择并删除索引。

（2）使用 DELETE TAG 命令。

例如，可以使用以下代码删除 employee 表中包含的 title 标识：

```
USE employee
DELETE TAG title
```

使用 ALTER TABLE 命令中的 DROP PRIMARY KEY 或 DROP UNIQUE TAG 子句。

可以使用 ALTER TABLE 命令删除 employee 表的主关键字标识：

```
USE employee
ALTER TABLE DROP PRIMARY KEY
```

3.7.2 从非结构 .CDX 文件中删除标识

非结构 .CDX 索引和它的标识在“表设计器”中不可见。若要删除非结构 .CDX 文件中的索引，只能使用命令从非结构 .CDX 文件中删除标识。

可以使用 DELETE TAG 命令的 OF 子句，指示 Visual FoxPro 从其他不同于 .CDX 结构文件的 .CDX 文件中删除标识。

例如，可以使用以下命令删除非结构 .CDX 文件 QTRLYRPT.CDX 中的标识 title：

```
DELETE TAG title OF qtrlyrpt
```

可以使用 DELETE TAG 命令的 ALL 子句，删除结构或非结构 .CDX 文件中的所有标识。

3.7.3 删除独立 .IDX 索引文件

由于独立索引文件只包含单索引关键字表达式，若要删除独立 .IDX 文件，可通过使用 DELETE FILE 命令。从磁盘上删除 .IDX 文件来删除表达式。

例如，以下代码删除独立 .IDX 索引文件 ORDDATE.IDX：

```
DELETE FILE orddate.idx
```

也可以使用 Windows 资源管理器等工具删除独立 .IDX 文件。

3.8 维护索引

在表中的索引建立之后，就将当前表的记录放入索引中，以便能从索引中存取这些记录。随着对表的编辑，如添加，删除记录等操作，需要调整这个表的每个索引标记和索引文件，使索引总能反映表中最新状态。在 Visual FoxPro 6.0 中，只有打开的索引标记和索引文件，才能在表的编辑中自动维护。那些没有打开的索引文件是不会得到维护的，所以这些索引文件不能反映表的状态，成为无效的索引文件。当使用这些无效的索引文件时，就会产生错误的结果，甚至可能会导致系统的崩溃。所以当表的记录产生变化后，应当及时对表的索引标识和索引文件进行维护。

在 Visual FoxPro 6.0 中，.CDX 结构复合索引文件是随着表的打开而打开的，其中的索引标识是系统自动维护的。但是.CDX 独立复合索引文件和.IDX 索引文件不能随着表的打开而打开，必须用 USE 命令的 INDEX 子

句或者 SET INDEX 命令打开后才能自动维护。然而，如果打开的索引过多，就会影响系统的维护效率。因此，Visual FoxPro 6.0 提供了一种简单的维护方法，即重建索引。

在 Visual FoxPro 6.0，重建索引的方法有两种，一种是选择 Table（表）菜单下的 Rebuild Indexes 命令，另一种是使用 REINDEX 命令。REINDEX 命令可以重建当前工作区中打开的所有索引文件。而且，Visual FoxPro 6.0 能够自动识别.CDX 和.IDX 索引文件，按顺序索引标记和索引文件进行重建。

由于集中重建索引可能会花大量的时间，所以应避免集中重建索引，仅对应用程序中需要使用的那部分索引在使用之前重建。对于那些暂时用不到的索引仍然保持原状，当用户需要使用这些索引时再重建这些索引.这样，就将重建索引的工作分配到多个应用程序中间，减少了每次重建索引的工作量。

3.9 本章小结

索引对于数据库表来说非常重要，可以利用索引对表中的数据进行排序，以便加速检索数据的速度。可以用索引快速显示、查询或者打印记录。还可以选择记录、控制重复字段值的输入并支持表间的关系操作。

本章主要讲述了如下几个问题：

- 索引的概念
- 索引的类型
- 索引文件的类型
- 创建索引
- 索引的使用
- 索引的维护

熟练掌握索引的使用是熟练掌握数据库操作的一项重要技能，能够大大方便或者提高数据库操作的速度。

第四章 查询与视图

创建数据库的目的是为了存储数据，同时对存储的数据进行利用，而最常用的利用就是查询数据。对于具有大量记录的数据库，特别是大型数据库，往往是查询满足一定条件的数据，对数据库的查询，是对数据库操作的一项重要操作。

与查询相类似的另一种查询数据库的数据的方法是视图，使用视图，可以从表中提取一组记录，改变这些记录的值，并把更新结果送回到源表中。Visual FoxPro 6.0 的视图设计设计器可以通过直观的操作创建视图，从而快速地从本地表、其他视图、存储在服务器上的表或远程数据源中筛选出满足条件的记录，并可像查询一样对记录进行排序和分类汇总。

4.1 数据库基本操作

Visual FoxPro 是一个具有处理数据库的强大功能的系统。它主要实现了对数据库操作的功能。在学习查询与视图之前不能不对数据库有所了解。

4.1.1 关系数据库

关系数据库模型将数据表示为表的集合。通过建立简单表之间的关系来定义结构，而不是根据数据的物理存储方式建立在数据中的关系。如图 4-1 中所示的关系数据库 Testdata，产品、客户、订货等表各存放不同的数据，它们之间通过表之间的连接来确定表之间的关系，
具有直接关系的表之间有一个相同的字段，通过此字段把两个表连接起来。

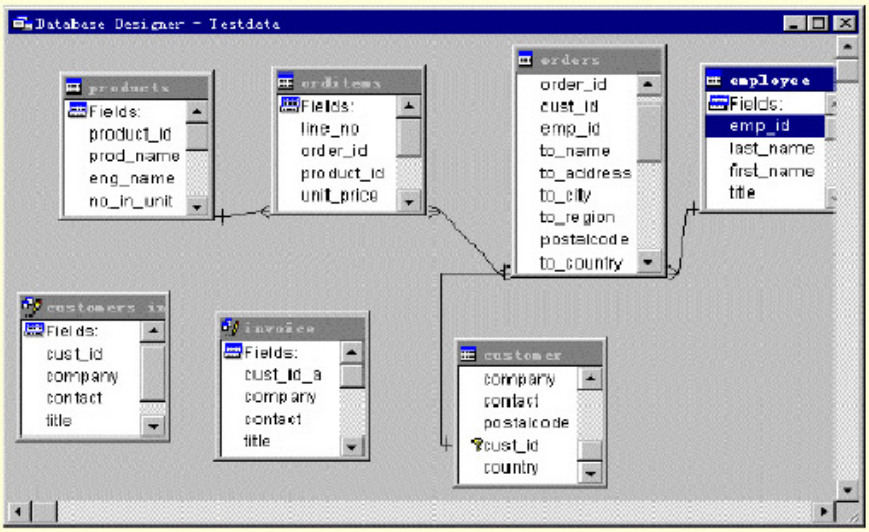
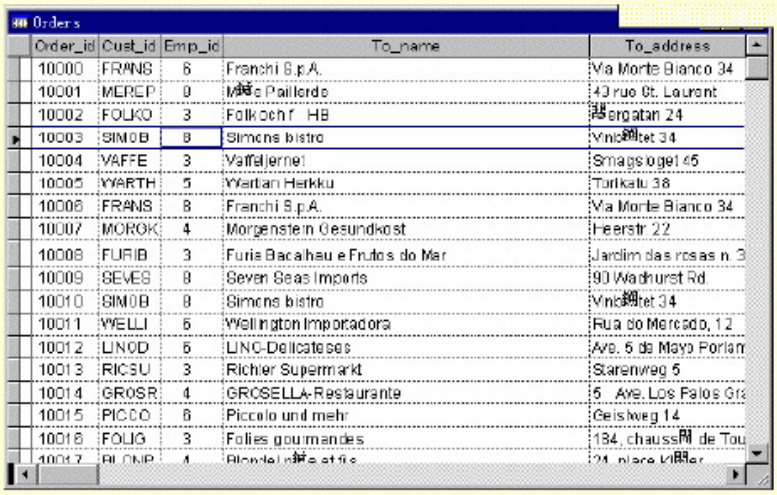


图 4-1 一个关系数据库的实例 Testdata

如产品表和定单有直接的关系（定单表上记录了订货的产品），而客户表和定单表之间有直接的关系（客户有不同的定单）。这样，客户与产品之间也就具有了关系，它们在同一个关系数据库中。

数据库是由多个数据库表组成，表与表之间可以用不同的方式相互关联。例如，Testdata 数据库还可以有一个包含某个客户的所有定单的表。它只用一个字段来引用该定单的客户，而不在定单表中的每项重复所有客户信息，如图 4-2 所示。


Orders

Order_id	Cust_id	Emp_id	To_name	To_address
10000	FRANS	6	Franchi S.p.A.	Via Monte Bianco 34
10001	MEREP	9	Mère Paillarde	43 rue St. Laurent
10002	FOLKO	3	Folkoch f HB	Åbergatan 24
10003	SIMOB	8	Simons bistro	Vinbæltet 34
10004	VAFFE	3	Vaffeljernet	Smagsloget 45
10005	WARTH	5	Wartian Herkku	Torikatu 38
10006	FRANS	8	Franchi S.p.A.	Via Monte Bianco 34
10007	MOROK	4	Morgenstern Gesundkost	Heerstr 22
10008	FURIB	3	Furia Bacalhau e Frutos do Mar	Jardim das rosas n. 3
10009	SEVES	8	Seven Seas Imports	90 Wadhurst Rd
10010	SIMOB	8	Simons bistro	Vinbæltet 34
10011	WELLI	6	Wellington Importadora	Rua do Mercado, 12
10012	LINOD	6	LINO-Delicateses	Ave. 5 de Mayo Porlam
10013	RICSU	3	Richter Supermarkt	Starenweg 5
10014	GROSR	4	GROSELLA-Restaurante	5 Ave. Los Palos Gra
10015	PICCO	6	Piccolo und mehr	Geislweg 14
10016	FOLIG	3	Folies gourmandes	184, chaussée de Tou
10017	BLONP	4	Blondel père et fils	24, place Kléber

图 4-2　定单表

在这个表中，Cust_Id **字段引用了** Customer **表中的** Cust_Id **字段，从而把定单和客户联系起来了。可以看到，客户** SIMOB **与** Simons bistro **有过业务来往。用来建立关系的关键字段叫做外部关键字段，因为它与“外部”表（**Customers **表）的主关键字段关联。**

在数据库中将多个表的数据组织在一起，就要确定数据库表之间的关系。如上面所示，使用户可以利用这些关系来查找所需要的有联系的信息，并能方便地对这些信息进行操作和处理。

以数据库 Testdata **为例，图** 4-1 **所示的该公司主管想知道某个客户向该公司定购产品的情况时，需要同时得到** Customer、Orders、Products **和** Orditems **四个表的数据。首先，可以在** Customer **表中通过主关键字段** Cust_Id **找到该客户自身的信息；然后通过** Orders **表中的** Cust_id **字段和** Customer **表中的** Cust_id **字段相关联，在** Orders **表中查找** Cust_id **字段的值是该顾客的** Cust_id **值的所有记录，得到有关定单的信息。同理，通过** Orders **表和** Orditems **表中的** Order_id **字段的关联，在** Orditems **表中找到相关记录和** Product_id**；接着又通过** Product_id **找到** Products **表中的相关记录。这样，通过表之间的关联，可以在仅知道** Cust_id **的情况下，得到该顾客向公司定购产品的所有信息。这些信息本来存在不同的表中，包括了顾客的公司名、地址、电话等顾客信息和订货的时间、产品数量、金额等定单信息，以及产品的名称、单价等产品的信息。**

从上面可以看出，表与表之间具有多种联系方式，为了正确关联不同表中的数据，表与表之间应采用正确合适的关系相关联。Visual FoxPro 6.0 **的数据库表之间有三种关系：**

1．一对一关系

在一对一关系中，表 A **的一个记录在表** B **中只能对应一个记录，而表** B **中的一个记录在表** A **中也只能有一个记录与之对应。这种关系并不经常使用，也需要在设计数据库时做某些更改。**

2．一对多关系

一对多关系是关系型数据库中最普通的关系。在一对多关系中，表 A **的一个记录在表** B **中可以有多个记录与之对应，但表** B **中的一个记录最多只能有一个表** A **的记录与之对应。**

3．多对多关系

在多对多关系中，表 A **的一个记录在表** B **中可以对应多个记录，而表** B **的一个记录在表** A **中也可以对应多个记录。这种情况下，在向** Visual FoxPro **正确指定关系之前，需要改变数据库的设计。**

4.1.2　创建数据库

打开 File **菜单中的** New **命令，在** New **对话框中中选择** Database**（数据库），然后单击** New File**（新建文件）按钮。在弹出的** Create **对话框中输入** Business**，确定后就会进入数据库设计器中如图** 4-3 **所示，一个新的数据库创建好之后，里面是空的，没有包含任何相关表或其他对象。向数据库中添加表实际上是建立表文件与数据库容器的双向链接关系：在数据库中保存指向表文件的前链，在表中保存指向数据库容器的后链。**

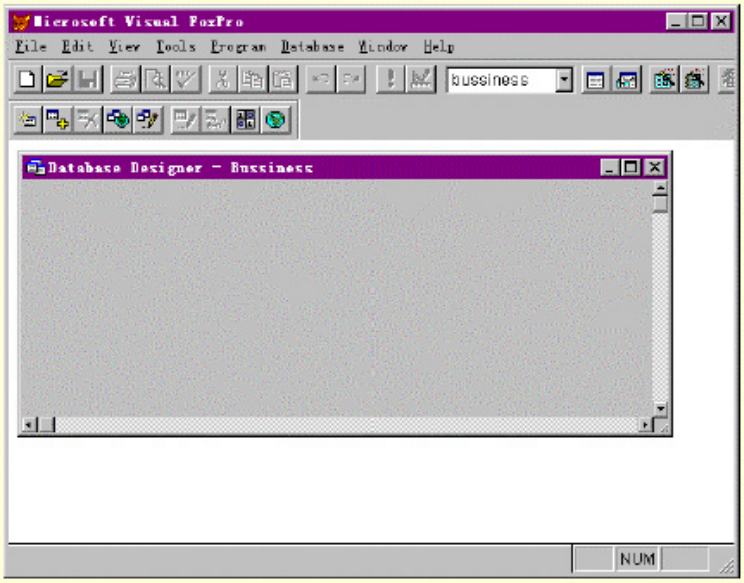


图 4-3 数据库设计器初始界面

或者使用 CREATE DATABASE 命令。

下面的代码创建并以独占方式打开了一个新数据库 Business。

```
CREATE DATABASE Business
```

下面是对数据库的其他常用操作。

1. 在数据库中添加表

选择 Database 菜单中的 Add Table 命令，或者在数据库设计器上单击鼠标右键，选择快捷菜单中的 Add Table，这时系统弹出 Open（打开）对话框，选择索要添加的表，此处选择 Depart 表，单击确定按钮，此时数据库设计器中多了一个表。同样的方法，把 Client 表、Yewu 表添加到数据库中。此时数据库设计器中共有三个表。如图 4-4 所示。此时添加的三个表还不是真正意义上的数据库表，因为这三个表还是互相独立的，表与表之间还没有建立关系。

或者使用 ADD TABLE 命令。

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

```
OPEN DATABASE Business
ADD TABLE Depart
```

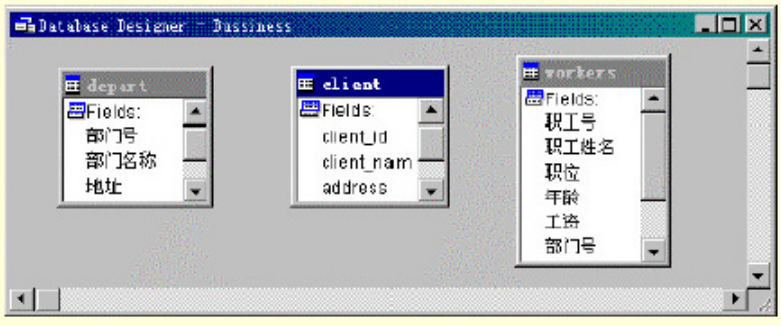


图 4-4 在数据库中加入了三个表

2. 在数据库中新建表

选择 Database 菜单中的 New Table 命令，或者在数据库设计器上单击鼠标右键，选择快捷菜单中的 New Table，这时系统弹出如图 4-5 所示的 New Table（新建表）对话框。

图 4-5 新建表数据库

如果选择 Table Wizard（表向导），则会以表向导的方式来创建一个表。如果选择 New Table（新建表），则系统弹出 Create（创建）对话框，输入要新建的表名例如 Temp，确定后进入表设计器对话框中，在此设计器中创建一个新表。

这时表设计器的 Fields（字段）、Indexes（索引）、Table（表）选项卡已不同于自由表的字段、索引、表选项卡了。如图 4-6、图 4-7 所示。

此对话框中不但具有自由表表设计器的所有功能，而且还可以设置以下内容：

（1）长表名和长字段名
（2）表中字段的标题和注释
（3）表中字段的默认值
（4）设置字段的显示格式
（5）匹配字段类型到类
（6）可以在数据库中设定表间永久关系
（7）在索引中增加了主索引和主关键字
（8）插入、更新和删除事件的触发器

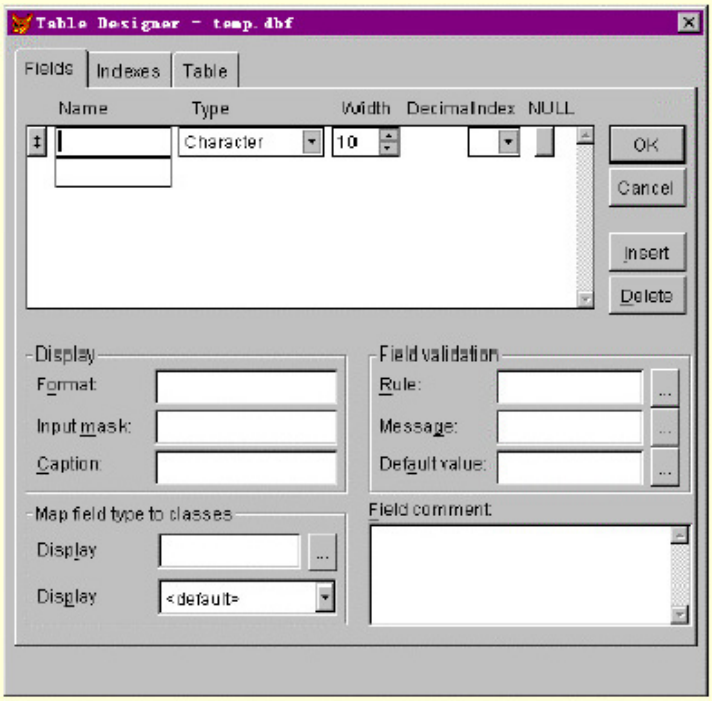


图 4-6 数据库表的表设计器字段选项卡

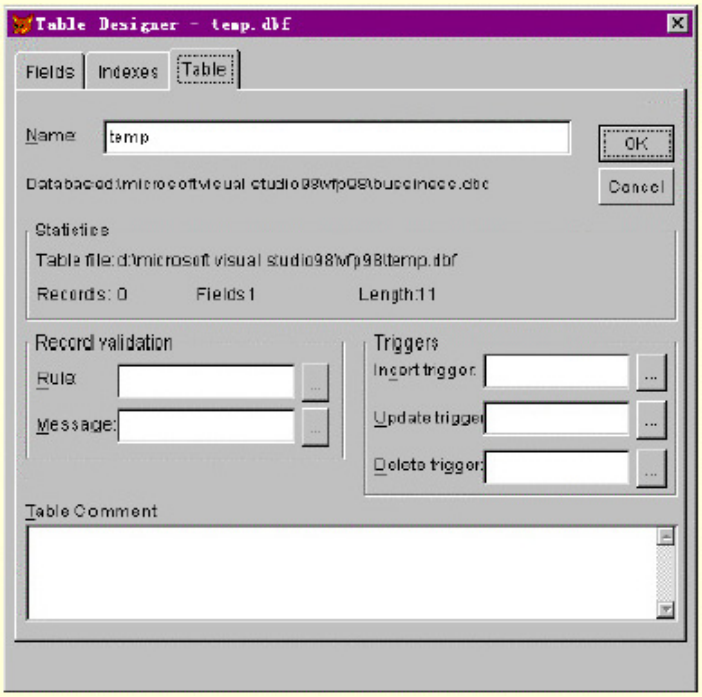


图 4-7 数据库表设计器中表选项卡

3. 在数据库中建立关系

在上面的数据库中，Depart 表和 Client 表并非直接联系，它们通过 Yewu 表间接联系。Depart 和 Client 的多对多关系在数据库中采用两个一对多关系来代替。

在 Visual FoxPro 6.0 中，表之间的关系是通过表中的索引来建立的。因为这种在数据库中建立的关系是作为数据库的一部分保存起来，这种关系是一种永久关系。当在查询设计器或视图设计器中使用表，或者在创建表单时所用的数据环境设计器中使用表时，这些永久关系将作为表的默认连接。

如果上述三个表还没有建立索引，则在数据库表设计器中为其创建索引。

在表 Depart 中，由于字段“部门号”是能够唯一确定公司部门的一个字段，所以把部门号作为其主索引。同理，在表 Client 中，将 Client_id 作为其主索引。由于表 Yewu 与表 Client 和表 Depar 都是一对多的关系，将表内的 Client_id 和“部门号”作为普通索引，以建立外部关系。此时三个表各多了索引字段。如图 4-8 所示。

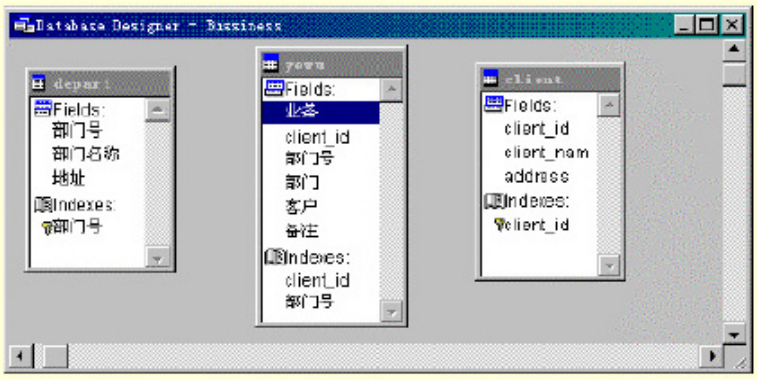


图 4-8 建立索引后的三个表

用鼠标在 Depart 表中的中的部门号索引字段上单击左键，并移动鼠标，此时光标变成长条状，移到表 Yewu 的索引“部门号”后放开鼠标左键，此时在表 Depart 和表 Yewu 之间建立了一对多的关系。用同样的方法在 Client 表和 Yewu 表之间建立一对多的关系。此时数据库设计器中显示了表之间的关系，如图 4-9 所示。

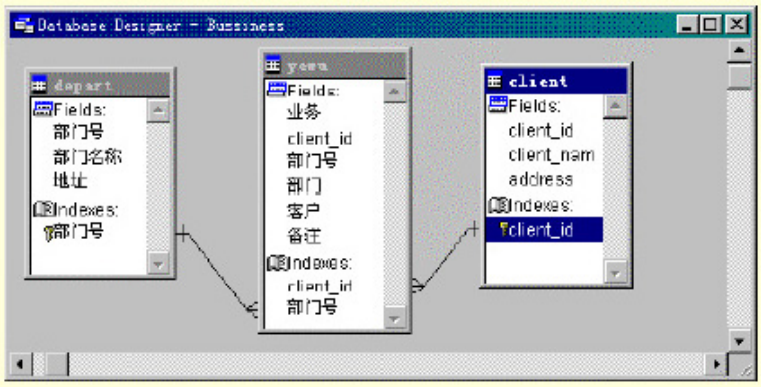


图 4-9　在表之间建立了关系修改数据库

或者在 CREATE TABLE 或 ALTER TABLE 命令中使用 FOREIGN KEY 子句。

例如，下面的命令根据表 Depart 的主关键字“部门号”，添加了 Depart 和 Yewu 表之间的一对多关系，并在 Yewu 表中添加了一个新的外部关键字“部门号”：

```
ALTER TABLE Yewu ；
ADD FOREIGN KEY 部门号 TAG ；
部门号 REFERENCES Depart
```

4. 删除数据库中的表

选中数据库中的一个表，选择主菜单 Database 下的 Remove，或者在选中的表上单击鼠标右键，在弹出的快捷菜单中选择 Delete，这时系统会问是从数据库中移去还是从磁盘上删除，或者取消。

也可选中数据库表后按 Delete 键或者选择数据库设计器工具条上的 Remove 按钮。以下的步骤同上。

使用 REMOVE TABLE 命令。

例如，下面的代码打开了数据库 Business 并移去表 Depart：

```
OPEN DATABASE Business
REMOVE TABLE Depart
```

从数据库中移去表不能自动删除该表文件。如果想要从数据库中移去表，同时从磁盘上删除该表的 .DBF 文件，可使用 REMOVE TABLE 命令的 DELETE 子句或 DROP TABLE 命令。例如从盘上删除表 Depart：

```
REMOVE TABLE Depart DELETE
```

以下代码也从盘上删除表 Depart，但不在 Windows 回收站中保存副本。

```
DROP TABLE Depart NORECYCLE
```

5. 修改数据库表之间的关系

选中数据库表之间的关系连线，此时关系连线变成粗黑线，然后单击鼠标右键，在弹出的快捷菜单中可以删除关系、编辑关系、编辑参照完整性。如图 4-10 所示。

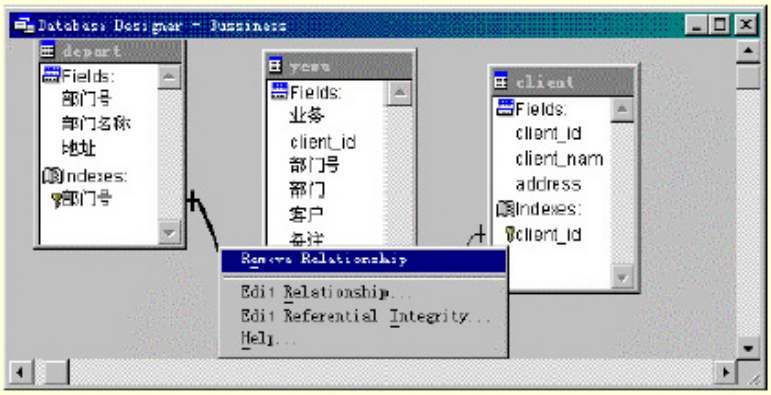


图 4-10　编辑数据库表之间的关系

下面对此快捷菜单的命令进行说明。

（1）Remove Relationship 删除关系

选择删除关系以后，表之间的关系就被删除了，同时表与表之间的关系连线也消失了。也可以选中数据库表之间的关系连线，直接单击 Delete 键。或者在 ALTER TABLE 命令中，使用 DROP FOREIGN KEY 子句。

例如，下面的命令删除了表 Depart 和表 Yewu 之间的一个永久关系，这个关系基于 Depart 表的主关键字“部门号”和 Yewu 表的外部关键字“部门号”：

```
ALTER TABLE Yewu  DROP FOREIGN KEY TAG 部门号 SAVE
```

（2）Edit Relationship 编辑关系

选择编辑关系以后，系统会弹出如下所图 4-11 所示的对话框。在此对话框中可以设置表 Depart 和表 Yewu 的关系，如：表 Depart 中的哪一个索引字段和表 Yewu 中的哪一个索引字段相关。

图 4-11 编辑关系对话框

（3）Edit Referential Integrity 编辑参照完整性

选择编辑参照完整性，会弹出参照完整性生成器对话框，通过此生成器可以建立一些规则，以便控制记录的插入、更新或删除，设置插入、更新或删除父表与子表记录时所遵循的规则。如图 4-12 所示。

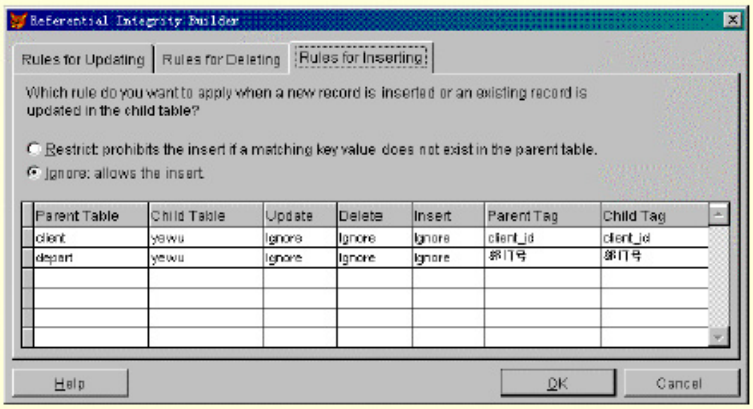


图 4-12 参照完整性生成器对话框

对于更新和删除规则，有三种选择：级联、限制和忽略。而对于插入规则，则只有两种选择：限制和忽略。

（1）更新规则

如果选择级联，则当修改父表中的记录时，子表中的和该记录相关的记录也会改变；如果选择限制，则当修改父表中的记录时，子表中有和该记录相关的记录时，禁止对父表操作；如果选择忽略（缺省），则两表互不影响。

（2）删除规则

如果选择级联，则当删除父表中的记录时，子表中的和该记录相关的记录也会被删除；如果选择限制，则当删除父表中的记录时，子表中有和该记录相关的记录时，禁止对父表操作；如果选择忽略，则两表的删除操作互不影响。

（3）插入规则

如果选择限制，则当在父表中插入记录时，子表中没有和该记录相关的记录时禁止对子表操作；如果选择忽略，则两表的插入操作互不影响。

当设置好以后，单击 OK 后，在系统弹出的如图 4-13 所示的对话框中，单击“是”以后，系统将生成参照完整性代码保存在数据库中，然后退出参照完整性生成器对话框。

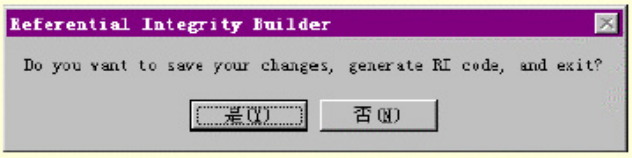


图 4-13 确认是否保存修改

6. 删除数据库

使用 DELETE DATABASE 命令。

例如，下面的代码删除了 Business 数据库：

```
DELETE DATABASE Business
```

要从磁盘上删除数据库，可使用上面两种方法。 DELETE DATABASE 命令能使 Visual FoxPro 从数据库的表中移去指向该数据库的后链，其他文件实用工具（例如 Windows 文件管理器）则不能移去这些后链。

DELETE DATABASE 命令并没有从磁盘上删除和数据库有关联的表，而只是把和数据库有关的表变成自由表。如果想从磁盘上删除数据库及所有关联的表，可在 DELETE DATABASE 命令中使用 DELETE TABLES 子句。

4.2 重要的查询命令

当在应用程序中使用查询或视图时，实际是在使用 SELECT－SQL 语句。这里的 SELECT－SQL 语句可以由查询设计器中定义的查询来创建；也可以是由视图设计器定义的视图来创建；还可以在事件代码或过程代码中创建。

4.2.1 SELECT－SQL 语句语法

```
SELECT [ALL | DISTINCT] [TOP nExpr [PERCENT]]
   [Alias.] Select_Item [AS Column_Name]
   [, [Alias.] Select_Item [AS Column_Name] ...]
FROM [FORCE]
[DatabaseName!]Table [[AS] Local_Alias]
   [[INNER | LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER] JOIN
        DatabaseName!]Table [[AS] Local_Alias]
        [ON JoinCondition ..].
[[INTO Destination]
   | [TO FILE FileName [ADDITIVE] | TO PRINTER [PROMPT]
   | TO SCREEN]]

[PREFERENCE PreferenceName]
[NOCONSOLE]
[PLAIN]
[NOWAIT]
[WHERE JoinCondition [AND JoinCondition ...]
   [AND | OR FilterCondition [AND | OR FilterCondition ...]]]
[GROUP BY GroupColumn [, GroupColumn ...]]
[HAVING FilterCondition]
[UNION [ALL] SELECTCommand]
[ORDER BY Order_Item [ASC | DESC] [, Order_Item [ASC | DESC] ...]]
```

4.2.2 语句参数说明

SELECT 指定显示在查询结果中的字段，常量或表达式。

ALL 默认值，在查询结果中显示所有的行。

DISTINCT 在查询结果中省略列中含有重复数据的记录。

TOP nExpr [PERCENT] 只返回查询结果中的一定数量的记录或一定百分比的记录，如果使用 TOP 子句，则也要使用 ORDER BY 子句。

Alias 别名。限制匹配项目名。

Select_Item 选择项。指定每个项目生成查询结果中的一列。

AS Column_Name 指定查询输出中一列的别名。

FROM 列出需要查询数据的表。如果表没有打开，则系统显示 Open 对话框，让用户选择文件。

FORCE 指定表按照出现在FROM子句中的顺序连接起来。

DatabaseName 指定包含该表的非当前数据库的名字。

Local_Alias 局部别名。指定一个名为 Table 的表的别名。

NNER JOIN 查询结果中仅包含两相关数据表格中彼此相对应的数据记录。

LEFT [OUTER] JOIN 查询结果中将包含位于关键字 LEFT [OUTER] JOIN 左侧的来源数据表格的所有数据记录，但是仅会包含位于关键字 LEFT [OUTER] JOIN 右侧的来源数据表格中相对的数据记录。

RIGHT [OUTER] JOIN 查询结果中将包含位于关键字 RIGHT [OUTER] JOIN 右侧的来源数据表格的所有数据记录，但是仅会包含位于关键字 RIGHT [OUTER] JOIN 左侧的来源数据表格中相对的数据记录。

FULL [OUTER] JOIN 查询结果将会包含位于关键字 FULL [OUTER] JOIN 左右两侧的来源数据表格中所有相对应以及不对应的数据纪录。

ON JoinCondition 指定表连接的条件。

INTO Destination 指定查询结果将储存在何处。如果没有 INTO 选项，则查询结果显示在一个浏览窗口中。Destination 可以是如下子句之一：

（1）ARRAY（数组）将查询结果储存在一个内存变量数组中。

（2）CURSOR（指针）将查询结果储存在一个指针中。

（3）DBF<表名>|TABLE<表名>将查询结果存入一个表。

[TO FILE FileName [ADDITIVE] | TO PRINTER [PROMPT] | TO SCREEN]]

如果使用了 TO 选项而不使用 INTO 选项，则将查询结果送到一个名为< FileName >，送到打印机或主窗口。

选项 ADDITIVE 将查询结果添加到任何已存在的文本文件的末尾。

[PREFERENCE PreferenceName] 将查询结果送到一个浏览窗口，使用[PREFERENCE PreferenceName]来储存浏览窗口特性和选项以备后用。PREFERENCE 将特性长期储存在 FOXUSER 源文件中，并可在任何时候被查询。

NOCONSOLE 阻止查询结果送到文件、打印机或显示在主窗口。

PLAIN 禁止表头出现在查询输出显示中。

NOWAIT 当查询结果送到浏览窗口并打开后，继续执行程序的执行。

When TO SCREEN 将查询结果输送到主窗口或一个用户自定义窗口，如果装满查询结果则系统输出暂停。单击任一个键可翻看下一组查询结果，如果使用了 NOWAIT，则主窗口或用户自定义窗口中的输出结果将滚动，而不会暂停下来等待单击一键。

WHERE 告诉 Visual FoxPro 在查询结果中只显示一定的纪录，从多个表中检索数据时必须有此子句。

JoinCondition 指定连接 FROM 子句中的表的字段。如果在查询中使用了多个表，则应该为每个表中指定连接条件。当有多个连接条件时，要用 AND 连接。每一个连接条件的格式是：

FieldName1 Comparison FieldName2

FieldName1 是表中的字段，FieldName2 是另一个表中的字段。Comparison（比较符）列在表 4-1 中。

表 4-1 比较操作符

比较操作符	比 较
=	相似
==	等于
LIKE	SQL 相似
<>、!=、#	不相似
>	大于
>=	大于等于
<	小于
<=	小于等于

FilterCondition 过滤条件。指定查询结果中的记录必须满足的条件。

GROUP BY GroupColumn [, GroupColumn ...]根据一列或多列的的值将记录分组。

HAVING FilterCondition 指定查询结果中的组所满足的条件。HAVING 应该与 GROUP BY 一起使用。

[UNION [ALL] SELECTCommand]将一个 SELECT 的最后结果与另一个 SELECT 的最后结果合并起来。缺省时，UNION 检查合并结果并去掉重复的行，使用括号可以连接多个 UNION 子句。

ORDER BY Order_Item 根据一列或多列中的数据对查询输出结果进行排序。ASC，对查询结果根据排序项按升序排列。DESC 指定对查询结果降序排列。

4.3 利用查询向导设计查询

使用查询向导可以快捷地创建一个查询：

（1）选择 File 菜单下 New 子菜单，在弹出的 New 对话框中，选择 Query 项，然后单击 Wizard 按钮。系统会显示 Wizard Selection（向导选取）对话框。如图 4-14 所示。

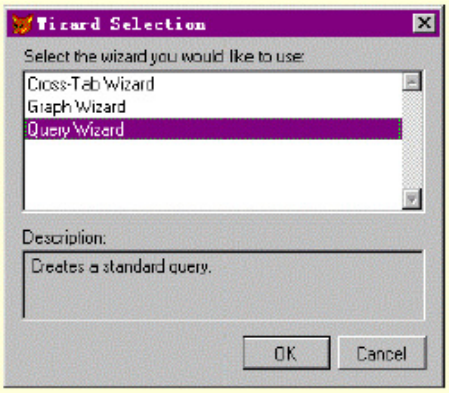


图 4-14 向导选取对话框

此对话框中共有三种向导可以选择：交叉表向导、图形向导和查询向导。交叉表向导是以电子表格的形式显示查询数据；图形向导是在 Microsoft Graph 里创建用于显示 Visual FoxPro 表数据的图形。查询向导是利用一个或多个表创建一个 SQL 查询。选择查询向导。

（2）接着进入查询向导的第一步：字段选取。如图 4-15 所示。

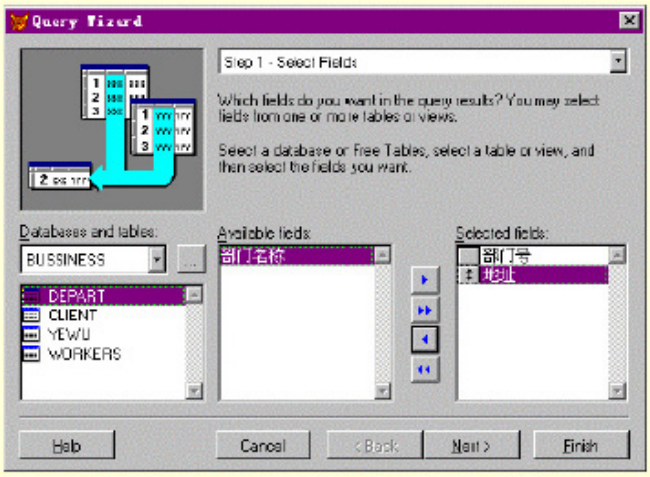


图 4-15 向导第一步：字段选取

如果在当前没有打开任何表、数据库或视图，则先选择数据来源。单击 Database And Table 右边的按钮，选择数据源。在这里选择前面一章建立的数据库 Bussiness，同时要选择将在查询中显示的字段。

如果选择的字段来自不同的表或视图，则按 Next 按钮以后进入向导第二步：关联表。如果选择的字段仅来自一个表或视图。则会直接进入第三步：筛选记录。

选择 Depart 表和 Workers 表的三个字段“职工号”、“职工姓名”和“部门号”后进入下一步。

（3）在查询向导第二步关联表中，如图 4-16 所示，设定表 Depart 和表 Workers 的关系：Depart.部门号＝Workers.部门号。

第二步的作用就是为了为被选取的表建立关系。通过表之间的关系，可以查询到符合复杂查询条件的来自多个表中的信息。单击 Next 后进入下一步。

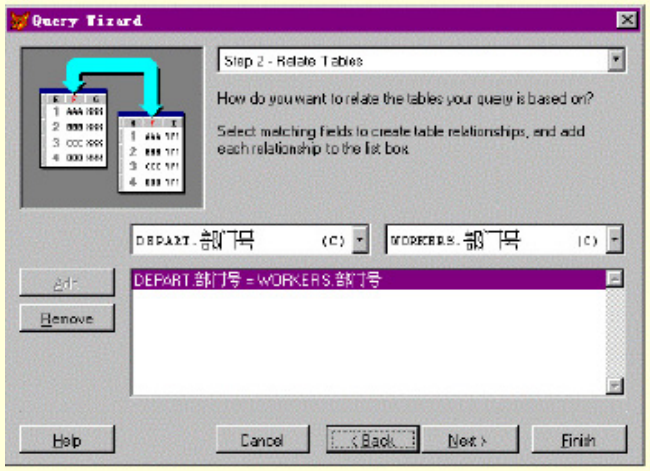


图 4-16 向导第二步：关联表

（4）此一步为第二 A 步：包含记录，如图 4-17 所示。

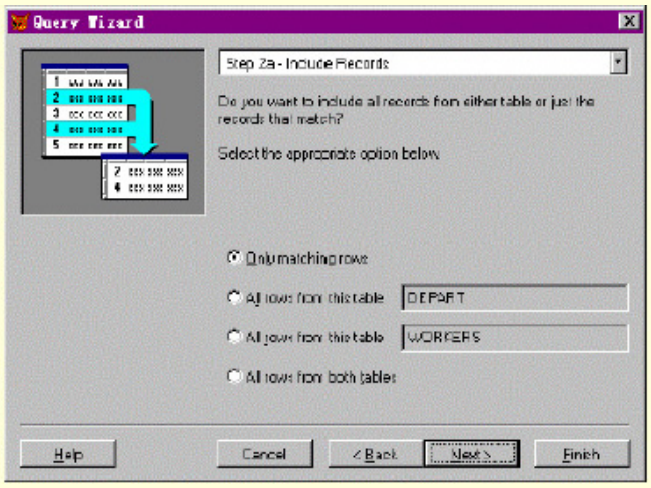


图 4-17　向导第二 A 步：包含记录

在这一步中可以选择：

① “仅匹配行”，查询文件中仅包含两个表之间匹配的记录。此为默认选项。

② Depart 表中的所有行。

③ Workers 表中的所有行。

④ 两个表中的所有行。

（5）选取第一种选择后，单击 Next 进入向导第三步：筛选记录。如图 4-18 所示。在默认的情况下，查询所有的记录。在此一步中可以设置记录满足的条件。

比如要查询公关部的工资多于 1500 元的记录。可以在此对话框中设置如下：

Depart.部门名称＝“公关部” AND Workers.工资>=1500

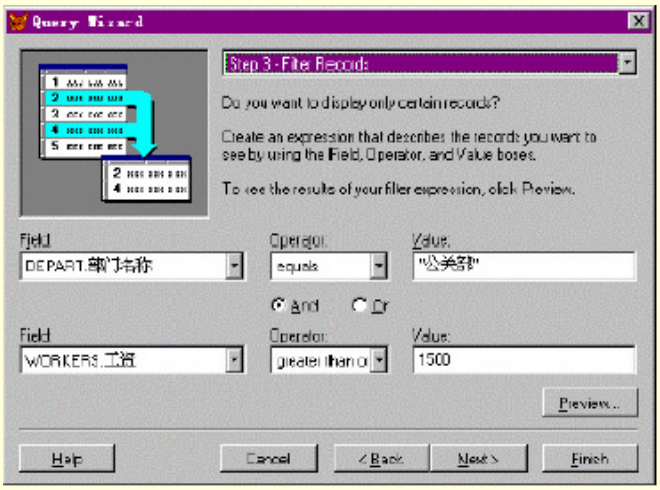


图 4-18　向导第三步：筛选记录

（6）设置好筛选条件后，单击 Next 按钮可以进入下一步，也即第四步：排序记录。如图 4-19 所示。在此一步可以设置显示的记录的次序。选择按照排序的字段和排序方式升序或降序。单击 Next 后进入第四 A 步：限制记录，对记录进行进一步限制。如图 4-20 所示。在此一步可以设置显示记录的数量或者全部记录的百分比。

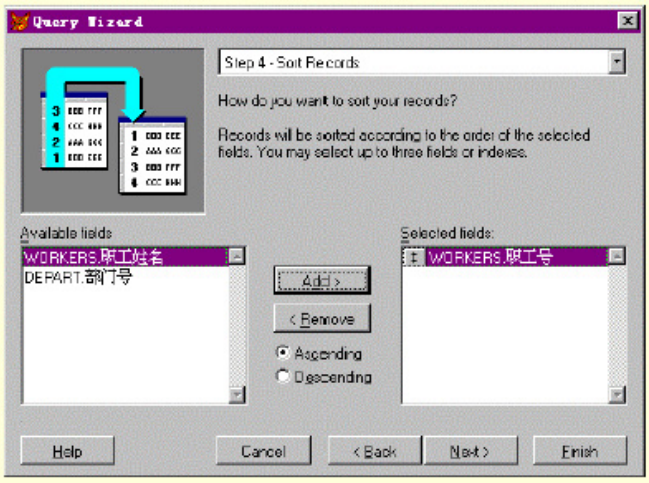


图 4-19　向导第四步：排序记录

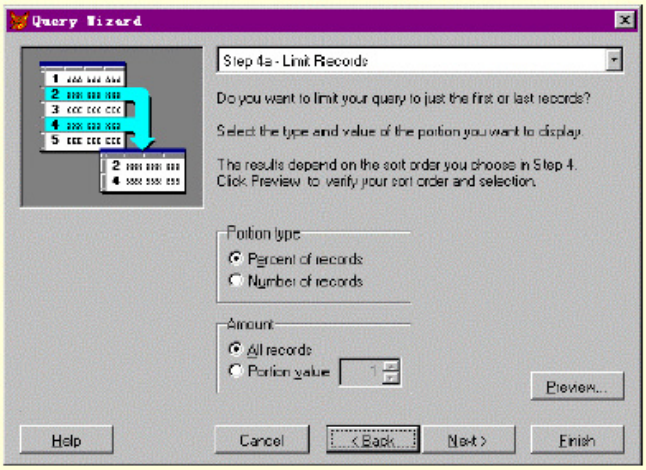


图 4-20　第四 A 步：限制记录

（7）单击 Next 以后，进入向导最后一步。如图 4-21 所示。在此一步中有如下选项：

① 保存查询。

② 保存并运行查询。

③ 保存查询并在查询设计器中修改。

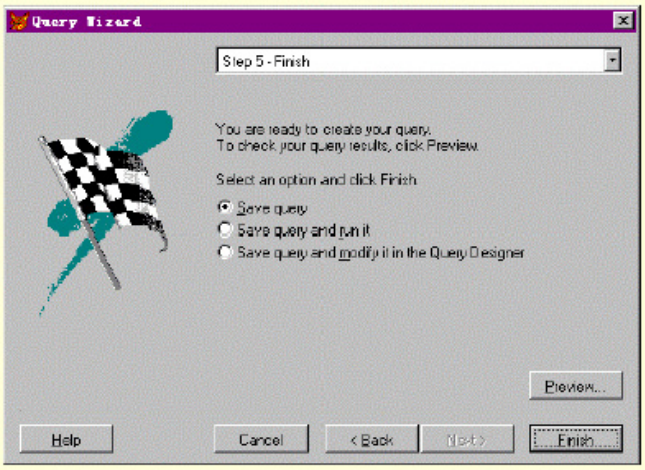


图 4-21　查询向导最后一步

以上查询设计过程，可以由以下 SELECT－SQL 语句来实现：

SELECT Workers.**职工号**, Workers.**职工姓名**, Depart.**部门号**;

```
FROM    bussiness!depart INNER JOIN bussiness!workers ;
    ON    Depart.部门号  = Workers.部门号;
WHERE Depart.部门名称  = "公关部";
   AND Workers.工资  >= "1500";
ORDER BY Workers.职工号
```

（8）选择“保存并运行查询”后，可以在浏览窗口中看到查询的结果。如图 4-22 所示。

通过以上利用查询向导创建查询的过程可以对创建一个查询所要经历的大体步骤有一个初步认识。这对下一节通过查询设计器来设计查询有很大帮助。从以上可以看到，设计查询大体有以下几步：

（1）确定数据源，即从那些数据库或表中查询数据。

（2）确定数据源之间的关系。

（3）要查询哪些数据信息。

（4）查询的信息具有哪些条件。

（5）信息如何排序。

图 4-22　查询运行结果

4.4　通过查询设计器创建查询

利用查询设计器可以创建比向导更加复杂的查询，可以定制更多的查询条件，总之，如果对 Visual FoxPro 6.0 的查询熟悉，那么，用查询设计器更加方便。查询设计器的使用方法与前面创建查询方法差不多。

4.4.1　打开查询设计器

选择 File 菜单下的 New 命令，在新建对话框中选择 Query，并单击 New File 按钮。这时，系统进入查询设计器界面，并且弹出 Open 对话框，让用户选择需要的表或其他数据源。

也可以先打开一个数据库或者表，然后再建立查询，此处打开数据库 Bussiness。这时，查询设计器上弹出添加表或视图对话框，如图 4-23 所示。

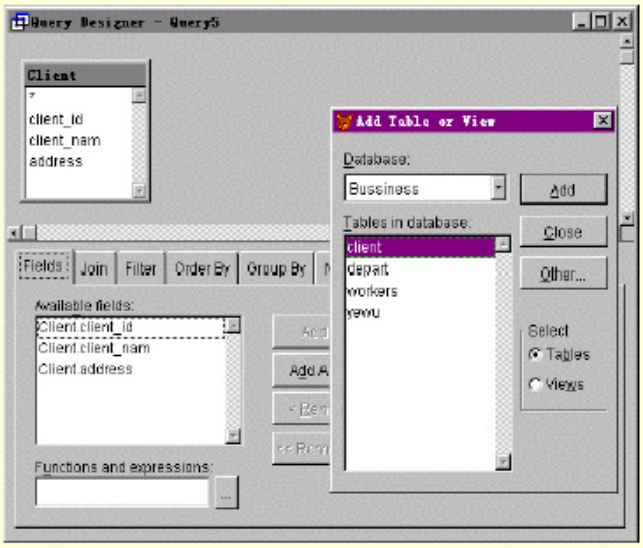


图 4-23　在查询设计器中添加表或视图对话框

在此对话框中选择将要从中查询的表或视图，然后单击 Add 按钮。选择的表就会添加到查询设计器中。当加入第二个表后，如果第二个表与第一个表没有关系，就会弹出连接条件，创建这两个表之间的关系。如图 4-24

所示。关于连接条件中内部连接、左连接、右连接还是完全连接，在介绍 SELECT－SQL 语句时已经讲过。

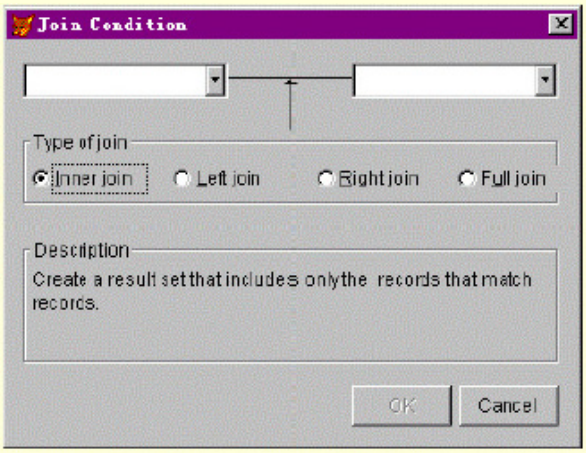


图 4-24　连接条件

因为 Business 数据库表之间已经建立了关系，将需要的表按照相关关系依次加入查询设计器中。则在查询设计器中仍然显示表与表之间的关系。如图 4-25 所示。

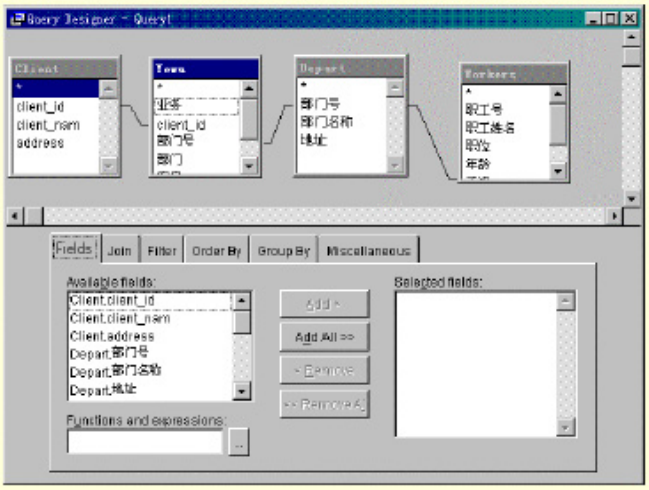


图 4-25　查询设计器中显示的数据源

4.4.2　选择所需字段

在运行查询之前，必须选择表或视图，并选择要包括在查询结果中的字段。在某些情况下，可能会使用表或视图中的所有字段。但在另一些情况下，也许只想使查询与选定的部分字段相关，比如要加到报表中的字段。

如果想用某些字段给查询结果排序或分组，一定要确保在查询输出中包含这些字段。选定这些字段后，可以在输出中为它们设置顺序。在字段选项卡中，字段的出现顺序决定了查询输出中信息列的顺序。

可以使用查询设计器底部字段选项卡来选定需要包含在查询结果中的字段。如图 4-25 所示。假如在此例查询中选择公关部工资多于 1500 元的人员的姓名、工资、年龄、职位、部门名称等信息。则选择姓名、工资、年龄、职位、部门名称等字段。

此查询可以通过如下命令实现：

```
SELECT Workers.职工姓名, Workers.工资, Workers.年龄, Workers.职位,;
    Depart.部门名称;
FROM    bussiness!depart INNER JOIN bussiness!workers ;
    ON    Depart.部门号  = Workers.部门号
```

若要在查询输出中添加字段，选定字段名，然后单击 Add 按钮。将字段名拖到“选定字段”框中。

如果要输出全部字段，可使用名字或通配符选择全部字段。如果使用名字选择字段，查询中要包含完整的字段名。此时若向表中添加字段后，再运行查询，则输出结果不包含新字段名。

如果使用通配符，通配符包含在查询中并包含当前查询的表中的全部字段。如果创建查询后，表结构改变

了，新字段也出现在查询结果中。

若要在查询中一次添加所有可用的字段，选择 Add All（全部添加），按名字添加字段。或者将表顶部的 * 号拖到“选定字段”框中。

如果使用字段的别名则可使查询结果易于阅读和理解，方法是在输出结果字段添加说明标题。例如，可在结果列的顶部显示“年平均工资”来代替字段名或表达式“工资”。

若要给字段添加别名可以使用如下方法：

（1）在 Functions and Expressions（函数和表达式）框键入字段名，接着键入“AS”和别名，例如：

工资 AS “年平均工资”

（2）选择“添加”在“选定字段”框中放置带有别名的字段。

如图 4-26 所示，在选择字段框中的字段。要查看此时定制的查询结果，可以在查询设计器中单击鼠标右键，在快捷菜单中选择 Run Query，可以看到查询结果。如图 4-27 所示。

图 4-26 为字段添加别名

图 4-27 查询结果

由于此时没有定制过滤条件，所以显示的是此公司的所有职员。

4.4.3 选定所需的记录

选定想要查找的记录是决定查询结果的关键。用查询设计器中的筛选选项卡，可以构造一个带有 WHERE 子句的选择语句，用来通知 Visual FoxPro 想要搜索并检索的记录。

如果需要查找一个特定的数据子集，并将其包含在报表或其他输出中。例如，此例中选择公关部工资大于 1500 元的人员的信息。

在 Visual FoxPro 中，使用筛选选项卡可以确定用于选择记录的字段、选择比较准则、以及输入与该字段进行比较的示例值。

若要指定过滤器，可以：

（1）从 Fields Name（字段名）列表中选定用于选择记录的字段。通用字段和备注字段不能用于过滤器中。

（2）从 Criteria（条件）列表中选择比较的类型。

（3）在 Sample（实例）文本框中，输入比较条件。

在输入比较条件时应注意：

① 仅当字符串与查询的表中字段名相同时，用引号括起字符串。否则，无需用引号将字符串引起来。

② 日期也不必用花括号括起来。

③ 逻辑位的前后必须使用句点号，如（.T.）。

④ 如果输入查询中表的字段名，Visual FoxPro 就将它识别为一个字段。

（4）在搜索字符型数据时，如果想忽略大小写匹配，请选择 Case（大小写）下面的按钮。

（5）在最右边的 Logical（逻辑）中选择两个过滤条件是 AND 关系还是 OR 关系。默认情况下是 AND 关系。

（6）.如果对逻辑操作符的含义取反，则可以单击 NOT 下面的按钮。例如，如果将图 4-28 的第二个筛选条件的“＝”左边选择 NOT，则选择的则是非公关部人员的信息。

此例中的筛选条件可以如图 4-28 所示。

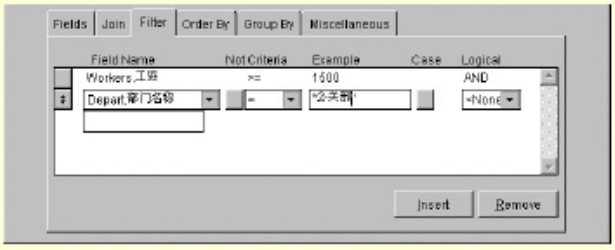


图 4-28 在筛选选项卡中定制筛选条件

定制了查询的筛选条件之后，运行查询，结果如图 4-29 所示。

图 4-29 设定筛选条件后的查询结果

此选择条件相当于如下 SELECT－SQL 语句的 WHERE 子句：

```
WHERE Workers.工资 >= "1500";
    AND Depart.部门名称 = "公关部";
```

4.4.4 对查询结果进行排序

如果从数据库中查询的记录不经过排序就显示出来，当记录特别多时，将显得杂乱无章，对记录进行排序可以更加明晰的查看查询结果。同时对记录按照某一字段进行分组，也是为了达到这种目的。

排序决定了查询输出结果中记录或行的先后顺序。例如，对查询到的职工信息按照职工号进行排序，或按照工资多少进行排序。

可以利用排序 Order By（排序依据）选项卡设置查询的排序次序，如图 4-30 所示，排序次序决定了查询输出中记录或行的排列顺序。

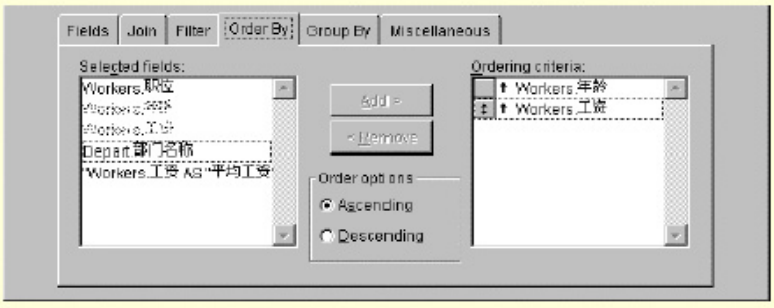


图 4-30 排序依据选项卡

首先，从 Selected fields（选定字段）框中选定要使用的字段，并把它们移到 Ordering criteria（排序条件）框中，然后根据查询结果中所需的顺序排列这些字段。

若要设置排序条件：

（1）在“选定字段”框中选定字段名。

（2）选择“添加”。

若要移去排序条件:

（1）选定一个或多个想要移去的字段。

（2）选择“移去”。

字段在“排序条件”框中的次序决定了查询结果排序时的重要性次序，第一个字段决定了主排序次序。例如，在“排序条件”框中的第一个字段是“年龄”，第二个字段为“工资”，查询结果将首先以“年龄”进行排序，如果 Workers 表中有一个以上的记录具有同样的“年龄”字段值，这些记录则再以工资进行排序。

为了调整排序字段的重要性，可以在“排序条件”框中，将字段左侧的按钮拖到相应的位置上。

通过设置“排序选项”区域中的按钮，可以确定升序或降序的排序次序。在“筛选”选项卡的“选定字段”框中，每一个排序字段都带有一个上箭头或下箭头，该箭头表示按此字段排序时，是升序排序还是降序排序。

为了能看到更多的记录，并按照以上排列顺序，对筛选条件放松一个，选择公司全部职员中工资大于 1500 的人员信息，则查询结果如图 4-31 所示。

图 4-31　按照年龄进行排序后的查询结果

4.4.5　分组查询结果

所谓分组就是将一组具有相同字段的记录按照此字段压缩成一个结果记录，这样就可以完成基于一组记录的计算。例如，若想找到某一特定地区所有订货的总和，不用单独查看所有的记录，可以把来自相同地区的所有记录合成一个记录，并获得来自该地区的所有订货的总和。若要控制记录的分组，可使用查询设计器中的 Group By（分组依据）选项卡。如图 4-32 所示。

分组在与某些累计函数联合使用时效果会更好，诸如 SUM、COUNT、AVG 等等。

例如，若想看到 Workers 表中各部门中职工的工资金额总值，只需将具有相同部门号的记录合成为一个记录，同时寻找公司总和即可。

首先在“字段”选项卡中，把表达式 Depart.部门号、Depart.部门名称、 SUM（Workers.工资）添加到查询输出中，然后利用“分组依据”选项卡，根据部门分组，输出结果显示了每个客户的总净订货量。

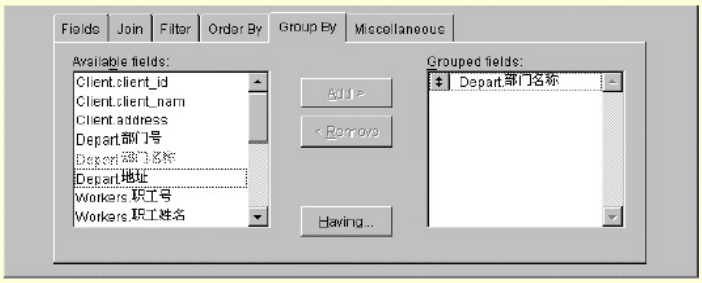


图 4-32　查询设计器分组依据选项卡

若要设置分组选项，可以按如下步骤操作：

（1）在“字段”选项卡中，在“函数和表达式”框中键入表达式：SUM（Workers.工资）。或者选择使用“表达式生成器”的对话框，在“函数和表达式”框中键入表达式。

（2）选择“添加”按钮，在“选定字段”框中放置表达式。

（3）在“分组依据”选项卡中，加入分组结果依据的表达式：Depart.部门号。

也可以在已分组的结果上设置选定条件。

若要对已进行过分组或压缩的记录而不是对单个记录设置过滤器，可在“分组依据”选项卡中选定 Having（满足条件）按钮在弹出的满足条件对话框，如图 4-33 所示。也可使用字段名，字段名中的 aggregate 函数，或者“字段名”框中的另外的表达式。

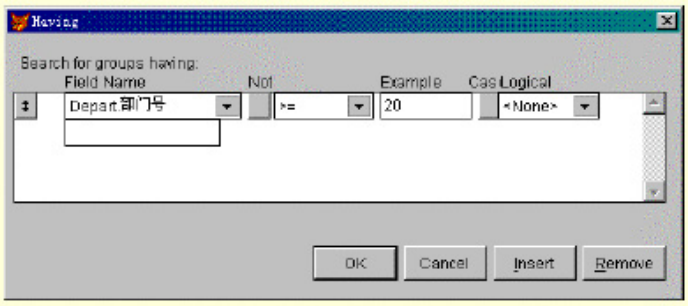


图 4-33 分组满足条件对话框

在上述示例中，按部门号显示工资的总和，可以进一步利用“满足条件”按钮，限制查询的结果中部门号小于 20 的部门信息。

如果为一个组设置条件，则可以使用如下方法：

（1）在“分组依据”选项卡上，选择“满足条件”按钮。

（2）在“满足条件”对话框中，在“字段名”域中选定字段名或定制表达式，同时设置满足条件。

在此例中，满足条件为：Depart.部门号>=20

（3）单击 OK 按钮。

按照上述查询条件的查询结果如图 4-34 所示。

此查询可以通过如下 SELECT－SQL 语句实现：

```
SELECT Depart.部门号, Depart.部门名称, SUM（Workers.工资）;
 FROM    bussiness!depart INNER JOIN bussiness!workers ;
    ON    Depart.部门号  = Workers.部门号;
 GROUP BY Depart.部门名称;
 HAVING Depart.部门号  >= "20";
 ORDER BY Depart.部门号
```

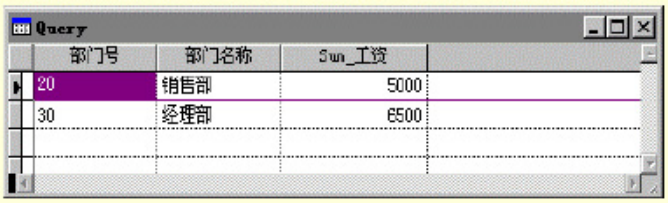


图 4-34 按分组进行的查询结果

4.4.6 查询中删除重复记录

重复记录是指其所有字段值均相同的记录。如果想把查询结果中的重复记录去掉，只需选中 Miscellaneous（杂项）选项卡中的“无重复记录”。否则，应确认“无重复记录”框已被清除。如图 4-35 所示。

如果选中了“无重复记录”，在 SELECT 命令的 SELECT 部分中，字段前会加上 DISTINCT。

如果查询最靠前一定数目或一定百分比的记录，也即可使查询返回指定的特定字段中有多少或百分之几的记录包含最高或最低值。例如，查询可显示指定字段中含 50 个最高或最低值的记录，或者显示含有字段值 50% 的最高或最低记录。

利用“杂项”选项卡的顶端设置，可设置到一定数目或一定百分比的记录。若要设置是否选取顶端或底端，可设置查询的排序顺序，降序可查看顶端记录，升序可查看底端记录。

若要检索一个数目或一定百分比的顶端记录，可按如下步骤进行操作。

（1）在“排序根据”选项卡中，选择要检索的顶端数值，接着选取“降序”显示最高值或“升序”显示最低值。如果还要按其他字段排序，可将其放在字段顶端值顺序表列后面。

（2）在“杂项”选项卡中，在“记录号”框中，键入想要检索的最高或最低值的数目或百分数。若要显示百分比，请选取“百分比”。

（3）如果不希望数目或百分比中含有重复的记录，可选择“无重复记录”。

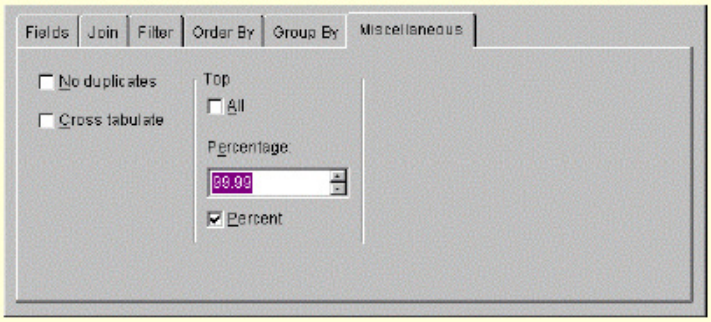


图 4-35 查询设计器杂项选项卡

4.5 利用视图设计器创建视图

利用视图向导可以快速的创建视图，但对于深入了解 Visual FoxPro 的用户来说，仅仅学会使用向导是远远不够的。还应学会使用视图设计器的强大功能来创建视图。

4.5.1 创建本地视图

使用视图设计器设计视图时，在数据库设计器中点击鼠标右键，在弹出的快捷菜单中选择 New Local View 或者在数据库设计器工具栏中单击 New Local View 按钮，在弹出的对话框中选择 New View，如图 4-36 所示。

图 4-36 新建本地视图对话框

系统就会弹出视图设计器，同时在视图设计上面有一个 Add Table or View 对话框，如图 4-37 所示。

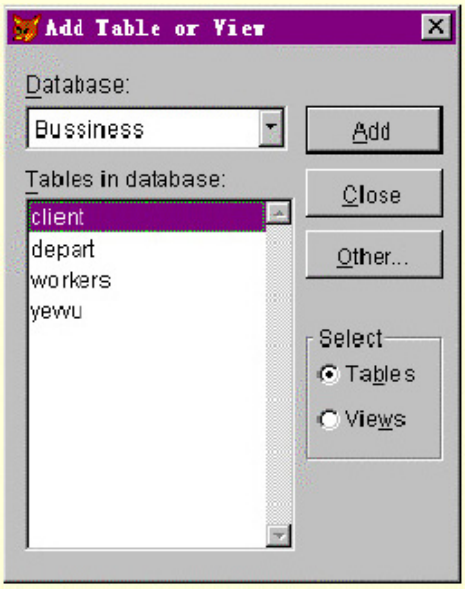


图 4-37 添加表或视图对话框

向视图设计器中添加表或者视图后就会进入视图设计器中。如图 4-38 所示。视图设计器与查询设计器的界

面差不多。它比查询设计器要多一个 Update Criteria（更新条件）选项卡。它可以控制更新。

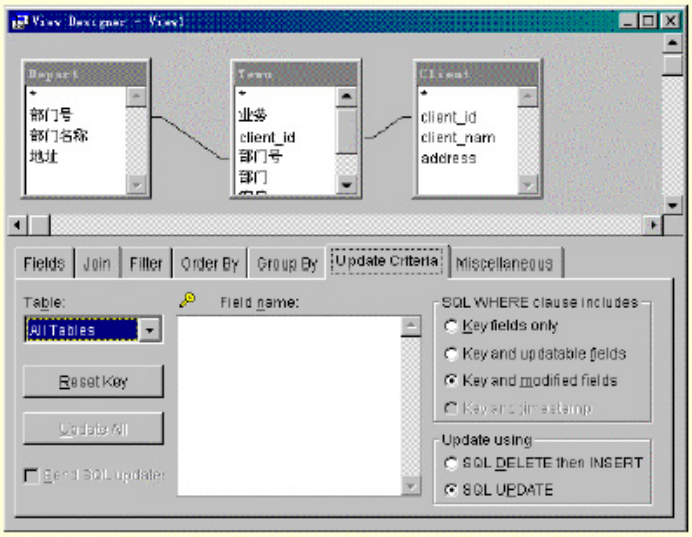


图 4-38　视图设计器

在视图设计器中，各个选项卡的功能与向导对应：

（1）字段选项卡，对应于向导的选取字段。

（2）连接对应与向导中表的关系。

（3）筛选选项卡对应于向导的筛选记录。

（4）排序选项卡对应于向导的记录排序。

（5）分组依据选项卡对记录进行分组处理，向导中无此项。

（6）更新条件在向导中也没有。

（7）杂项对应于向导中限制记录。

下面通过一个示例来说明视图创建的过程。例如，为 business 数据库创建一个视图，此视图中显示的记录为工资少于 3000 元的职员信息。

在字段选项卡中可以选择在视图中显示的字段，如图 4-39 所示为选择的字段。

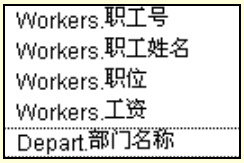


图 4-39　在字段选项卡中选择的字段

在连接选项卡中定制表与表之间的连接关系，如图 4-40 所示。

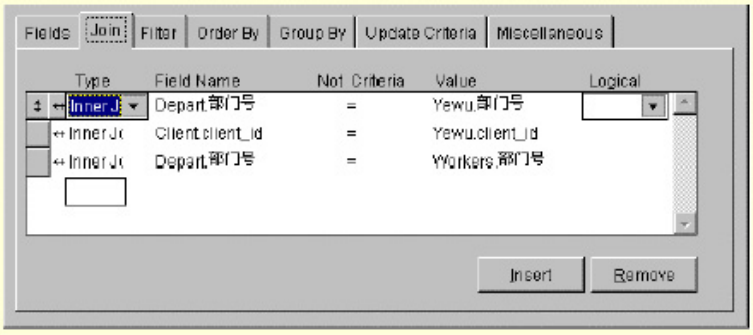


图 4-40　为表建立关系

在过滤选项卡中可以限制出现在视图中的记录所满足的条件。

Workers.工资<=3000

同时按照职工号进行升序排列。在排序选项卡中选择 Workers.职工号，并选择升序。

此处记录属于不同的部门，可以按照部门进行分组。

设置好视图以后，关闭视图设计器，这时 Visual FoxPro 会提示用户输入视图名以在数据库中保存视图。如图 4-41 所示。

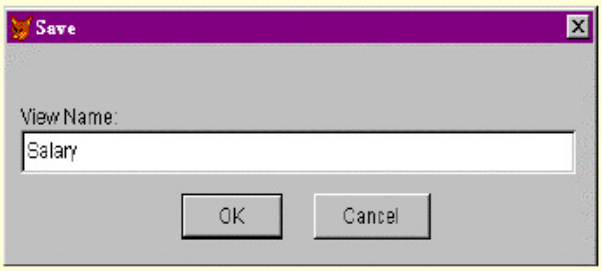


图 4-41 保存视图

这时在数据库中就会多了一个视图。可以浏览此视图中的记录。通过双击视图或者在视图上单击鼠标右键，选择 Browse 命令，可以在浏览窗口中看到视图中的记录。

图 4-42 浏览视图中的记录

在此看来，创建视图和创建查询在很大程度上还是相似的。

创建以上视图也可以通过以下 SQL 命令来实现。

```
SELECT Workers.职工号, Workers.职工姓名, Workers.职位, Workers.工资,;
    Depart.部门名称;
FROM    bussiness!depart INNER JOIN bussiness!workers ;
    ON    Depart.部门号  = Workers.部门号;
    WHERE Workers.工资  <= 3000;
    GROUP BY Depart.部门名称
```

4.5.2 创建远程视图

创建新的远程视图首先必须要有同数据源的连接。

在数据库设计器中单击鼠标右键选择 New Remote View 或者在 File 下的 New 命令，在新建对话框中选择远程视图，并单击 New File 按钮。此时会显示选择连接或数据源对话框，在此对话框中选择连接或者数据源。

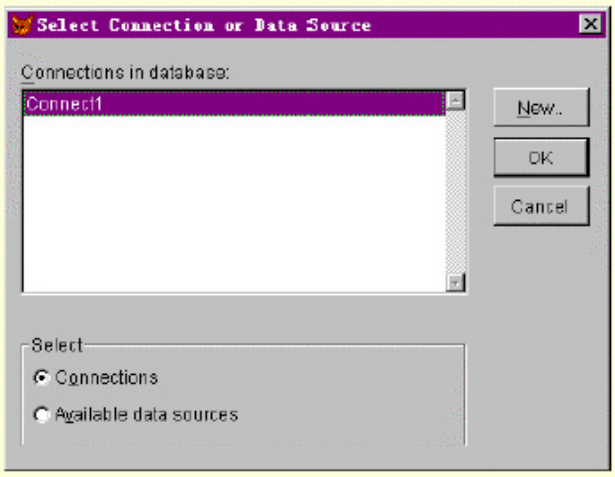


图 4-43 选择连接或数据源对话框

如果已经存在定义的连接，该对话框中就显示所定义的连接。如果要定义新的连接，可单击 New 按钮，在连接设计器中建立连接。选择连接后按下 OK 按钮。如果连接的是 Microsoft Access 数据库，就会显示如图 4-44 所示的选择数据库对话框。

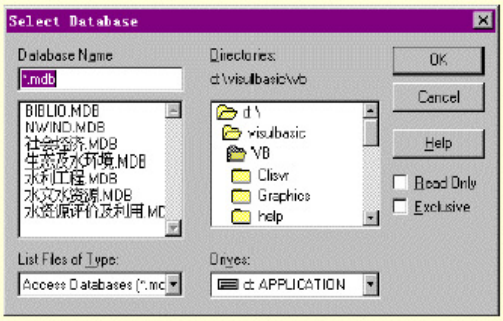


图 4-44 选择数据库

在此对话框中选择一个数据库，单击 OK 按钮，Visual FoxPro 将显示图 4-45 所示的打开对话框，此对话框中列出了上面选择的数据库中所包含的表。

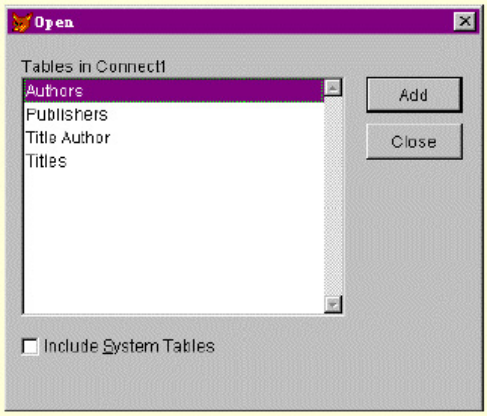


图 4-45 打开对话框

在此需要说明的是 Access 数据库也是一种关系数据库，其数据库也是由多个数据库表所组成的。

选择要加入视图设计器中的表并按下 Add 按钮，可以将表加入视图设计器中。

在视图设计器中对加入的表创建远程视图的操作与建立本地视图的操作一样。此处不再赘述。

4.6 利用视图更新数据

在视图中更新数据的方法与在表中更新数据类似，使用视图可以对源表进行更新。可以通过交互方式更新视图中的数据，也可用命令来更新数据。

通过视图设计器创建的视图，默认设置通常是允许视图更新，可以通过选择“Send SQL Update”复选框来打开更新开关，也可通过选择其他项来打开或关闭对表的指定字段的更新，并设置合适的 SQL 更新方法。

在视图设计器中，更新条件选项卡可以控制把对远程数据的修改（更新、删除、插入）回送到远程数据源中的方式，也可以打开和关闭对表中指定字段的更新，并设置适合服务器的 SQL 更新方法。

4.6.1 使表可更新

如果希望在本地视图上所作的修改能回送到源表中，需要设置“发送 SQL 更新”选项，必须至少设置一个关键字段来使用这个选项。如果选择的表中有一个主关键字段，并且已在字段选项卡中，则“视图设计器”自动使用表中的该主关键字段作为视图的关键字段。

4.6.2 设置关键字段

当在视图设计器中首次打开一个表时，更新条件选项卡会显示表中哪些字段被定义为关键字段。Visual FoxPro 用这些关键字段来唯一地标识那些已在本地修改过的远程表中的更新记录。

若要设置关键字段，则在更新条件选项卡中，单击字段名旁边的“关键列”（钥匙形），图 4-46 所示。如果已经改变了关键字段，而又想把它们恢复到源表中的初始设置，可以选择 Reset Key（重置关键字）。Visual FoxPro 会检查远程表并利用这些表中的关键字段进行更新。

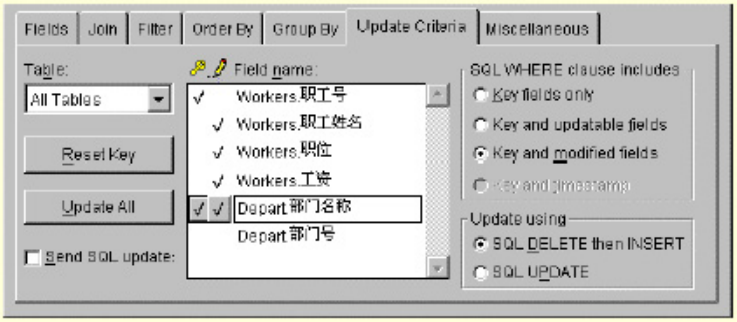


图 4-46 视图设计器更新条件选项卡

4.6.3 更新指定字段

可以指定任一给定表中仅有某些字段允许更新。若使表中的任何字段是可更新的，在表中必须有已定义的关键字段。如果字段未标注为可更新的，用户可以在表单中或浏览窗口中修改这些字段，但修改的值不会返回到远程表中。

若要使字段为可更新的，在更新条件选项卡中，单击字段名旁边的“可更新列”（笔形）。

如果想使表中的所有字段可更新，可以将表中的所有字段设置成可更新的。在更新条件选项卡中，选择 Update All（全部更新）。

4.6.4 控制如何检查更新冲突

如果在一个多用户环境中工作，服务器上的数据也可以被别的用户访问，也许别的用户也在试图更新远程服务器上的记录，为了让 Visual FoxPro 检查用视图操作的数据在更新之前是否被别的用户修改过，可使用更新条件选项卡上的选项。

在更新条件选项卡中，SQL WHERE clause includes（SQL WHERE 子句包括）框中的选项可以帮助管理遇到多用户访问同一数据时应如何更新记录。在允许更新之前，Visual FoxPro 先检查远程数据源表中的指定字段，看看它们在记录被提取到视图中后有没有改变，如果数据源中的这些记录被修改，就不允许更新操作。

在“更新条件”选项卡中可以设置 SQL WHERE 子句。

这些选项决定哪些字段包含在 UPDATE 或 DELETE 语句的 WHERE 子句中，Visual FoxPro 正是利用这些语句将在视图中修改或删除的记录发送到远程数据源或源表中，WHERE 子句就是用来检查自从提取记录用于视图中后，服务器上的数据是否已改变。表 4-2 列出了各个选项的用法：

表 4-2 SQL WHERE 子句选项

选 项	含 意	用 法
Key fields only	关键字段	当源表中的关键字段被改变时，使更新失败
Key and updatable fields	关键字和可更新字段	当远程表中任何标记为可更新的字段被改变时，使更新失败。
Key and modified fields	关键字和已修改字段	当在本地改变的任一字段在源表中已被改变时，使更新失败。
Key and timestamp	关键字和时间戳	当远程表上记录的时间戳在首次检索之后被改变时，使更新失败（仅当远程表有时间戳列时有效）

4.7 操作视图的命令

4.7.1 创建视图

命令：CREATE SQL VIEW

功能：显示视图设计器，创建一个视图。

```
CREATE SQL VIEW [ViewName ] [REMOTE]
[CONNECTION ConnectionName [SHARE]
| CONNECTION DataSourceName]
[AS SQLSELECTStatement]
```

说明：ViewName **指定要创建的视图名，如果没有指定视图名，系统将提示用户输入一个名称。**

REMOTE **指定产生的视图是远程视图，如果忽略，系统默认的是创建的本地视图。**

CONNECTION ConnectionName [SHARE] **表示当视图被打开时建立指定的连接。如果加上** SHARE **子句，**Visual FoxPro **将使用一个共享连接。这就意味着其他视图也可以使用这个连接。如果不能使用共享连接，**Visual FoxPro **将使用一个唯一连接，该连接不能被其他视图使用。**

CONNECTION DataSourceName **表示创建视图时使用一个已经存在的连接。在此情况下，不必使用** REMOTE **子句，系统就会根据定义的连接建立远程视图。**

AS SQLSELECTStatement **为视图指定条件，**AS **后面必须紧跟着一个正确的** SQL SELECT **语句。**SQL SELECT **语句不能用引号括起来。对于本地视图来说，**SQL SELECT **语句里的视图或表名前可以带上所属的数据库名。**

举例：创建一个名为 my_view **的视图，视图中包括工资多于** 1500 **元的一些职工信息。则命令如下：**

```
CREATE SQL VIEW ;
        my_view ;
        AS select 职工号,部门号,工资;
        from workers ;
        where 工资>=1500
```

如果要从 ODBC **数据源** odbcdata **查询所需要的有关信息，则可以在** Business **数据库中利用如下代码创建连接：**

```
CREATE  CONNECTION connection1 DATASOURCE odbcdata
```

如果要创建一个远程视图，将数据库 business **中的** workers **表放到远程服务器上，则可以使用如下命令创建此表的远程视图：**

```
OPEN DATABASE business
CREATE SQL VIEW workers_remote_view ;
  CONNECTION connection1 ;
  AS SELECT * FROM workers
```

4.7.2 建立参数化视图

建立参数化视图的方法很简单，使用命令比使用交互界面更方便，只要在 CREATE SQL VIEW **的命令中的** AS **后面的** SQL SELECT **语句中的** WHERE **中使用？和一个参数即可。**

例如，要创建视图查询工资等于一定值的的参数化视图，则可以使用如下命令：

```
CREATE SQL VIEW ;
   test_view ;
  AS select 职工号,部门号,工资;
    from workers ;
          where 工资=?工资数量
```

上面的命令中“ 工资数量 ”为视图 test_view **的参数。**

当运行此视图，会弹出如图 4-47 所示的对话框提示用户。

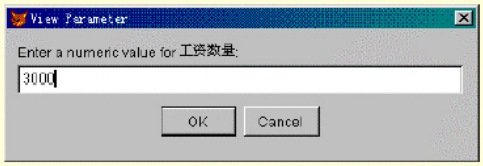


图 4-47　参数化视图运行

运行结果如图 4-48 所示。

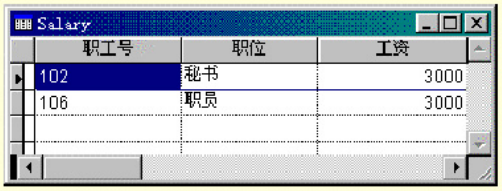


图 4-48　视图运行结果

4.7.3　重命名视图

可以使用 RENAME VIEW 命令重命名视图，必须首先打开视图所在的数据库，下面的命令将 test_view 改名为 para_view

```
RENAME test_view TO para_view
```

4.7.4　删除视图

要删除视图前必须要包含此视图的数据库已经打开，并且设置为当前数据库。

使用 DELETE VIEW 或 DROP VIEW 命令可以删除视图。例如使用如下命令删除视图 para_view：

```
DELETE VIEW para_view
DROP VIEW para_view
```

4.7.5　使用视图

视图可以用来显示和更新数据，而且还可以通过调整视图的属性来提高性能

在数据库中对视图的处理和使用的方法也类似于表，视图的显示方式与表也相同。可以使用 USE 命令来打开一个视图，同时用 USE 命令来关闭视图。也可以在浏览窗口中显示视图中的数据，在表单的文字框、表格控件或者报表中把视图作为数据源。也可在数据工作期窗口中显示打开的视图。

例如，使用 USE 命令显示视图：

```
OPEN DATABASE business
USE para_view
BROWSE
```

视图在使用时，作为临时表在自己的工作区中打开。如果此视图基于本地表，则基表将在另一个工作区中打开。视图的基表是指由使用 CREATE SQL VIEW 命令创建视图时 SELECT SQL 语句访问的表。

如果视图是基于远程表，则基表不在工作区中打开，而只在数据工作期中显示远程视图的名称。

前面已经讲过，在多个数据工作区中打开一个表，视图和表一样，也可以在不同的工作区中打开一个视图的多个实例。不同之处是：在默认情况下，每次使用视图时，都要去取一个新的数据集合。

在数据工作期窗口中打开一个视图的多个实例时，可以选择 OPEN 按钮，然后选择视图名，重复此动作，在其他工作区中打开此视图。

用 USE 命令访问视图时，不必重新查询数据源就可以打开视图的另一个实例。这在多个工作区中打开一个远程视图时特别有用，因为不需要从远程数据源中下载数据。如果想不下载数据再次使用视图，在使用 USE 命令时加上 NOREQUERY 子句或者 AGAIN 子句。

例如，下面的命令在两个浏览窗口中显示从视图 remote_view1 中第一个实例中得到的临时表，而不必重新

查询远程视图：

```
OPEN DATABASE business
USE remote_view1

BROWSE
SELECT 0
USE remote_view1 NOREQUERY
BROWSE
```

使用 NOREQUERY 子句可以指定一个工作期编号，如果不能指定工作期编号，Visual FoxPro 将对全部工作期进行搜索以查到结果集合。如果查到了打开的结果集合，就在这个结果集合中再次打开一个临时表。如果没有找到打开的结果集合，则为该视图取一个新的结果集合。如果没有找到视图，则打开一个新的视图临时表。

使用 AGAIN 子句可以只在当前工作期内搜索已经打开的视图结果集合。例如，在两个浏览窗口中显示 remote_view1 视图。

```
OPENDATABASE business
USE remote_view1
BROWSE
SELECT 0
USE remote_view1 AGAIN
BROWSE
```

使用 AGAIN 子句时，Visual FoxPro 在当前工作期中查找一个已经存在的视图临时表，并打开一个指向该视图临时表的附加别名。

如果要为视图创建本地索引，可使用 INDEX ON 命令，创建过程与表一样。与表的索引不同的是，在使用视图上创建的本地索引非永久保存，它们随着视图的关闭而消失。

可以使用 SET RELATION 命令在视图索引之间或视图索引与表索引之间创建临时关系，在使用 SET RELATION 命令对一个视图与一个表进行关联时，若要获得较好的性能，在关系中设视图为父表，设表为子表。这时因为在临时关系中，子对象必须按其索引排序，表的结构索引是在不断被维护的，因此可以快速访问。而对于视图，Visual FoxPro 要在视图每次激活时为其重建索引，因此很浪费时间，此外，如果使用数据环境，视图将不能被当作子对象，因为子对象的索引必须是定义的一部分，而视图并不支持此功能。

4.8 本章小结

数据库的操作是本书所讲述的重点，熟悉数据库的基本操作是非常必要的。而在数据库的操作中，对数据的查询和更新是所有数据库操作中最重要的两种操作，也就是查询和视图的操作。

本章主要讲述了如下内容：

- 数据库的基本操作
- 查询语句 SQL-SELECT
- 利用查询向导执行查询
- 利用查询设计器执行查询
- 利用视图向导设计视图
- 利用视图设计器设计视图
- 操作视图的各种命令

关于查询结果的输出等等其他内容限于本书篇幅，不作本书重点内容，请读者查阅相关书籍。

对于 SQL－SELECT 语句，是读者要重点掌握的内容，这在所有数据库的操作中都是非常重要的命令，本章列举了大量命令以帮助读者熟练掌握这一命令。

第五章　输出报表

报表是一种用户经常用到的用来输出数据的形式，而且报表的设计也是在应用程序的开发过程中的重要组成部分。

报表的设计包括两个组成部分：数据源和布局。数据源通常是数据库中的表，也可是视图、查询或临时表。报表的布局也就是报表的打印格式。

创建报表的过程包括定义报表的样式以及把这个定义存储在文件中，扩展名为.FRX。每个报表文件还具有一个相关文件.FRT 文件。

标签文件保存在.LBX 文件中，相关文件为.LBT。

5.1　报表向导的使用

报表向导是创建报表的一种常用的快速方法，如果报表的数据源是单一的表或视图，则可以使用报表向导来创建报表。

使用报表向导创建报表时，可以选择 File 菜单下的 New 命令，在新建对话框中选择报表并按下 Wizard 按钮。

或者在项目管理器中选择文档卡片，选择报表后按下新建按钮，在新报表对话框中选择报表向导。

此时进入报表向导中报表类型选择的对话框。如图 5-1 所示。Visual FoxPro 6.0 提供了两种类型的报表：一对多报表向导和报表向导。

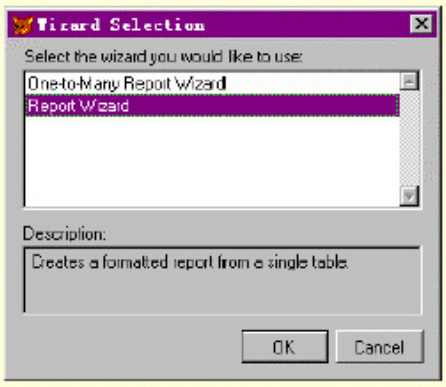


图 5-1　选择报表类型对话框

选择报表向导后按下确定按钮，则进入报表向导的步骤一：选择字段。如图 5-2 所示。可以从数据库中选择表或者视图中的字段作为报表的数据源。从“数据库和表”中选择要使用的表或视图，然后选择要打印的字段到“选定字段”列表框中。选择的字段将在报表中显示出来。

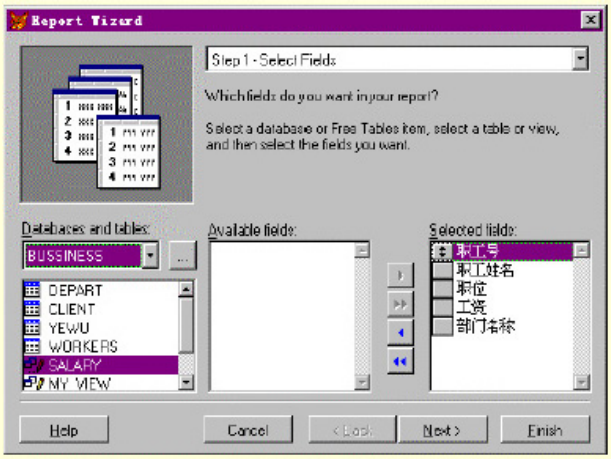


图 5-2 报表向导第一步：选择字段

在此例中选择数据库 Business 中的视图 SALARY 中的所有字段。选用视图作为报表的数据源的优点是显示在报表中的记录为符合一定条件的记录。选择了字段以后，按下 Next 按钮，进入报表向导的第二步：分组记录。如图 5-3 所示。在此一步中可以为记录选定分组依据的字段，可以进行三级分组。

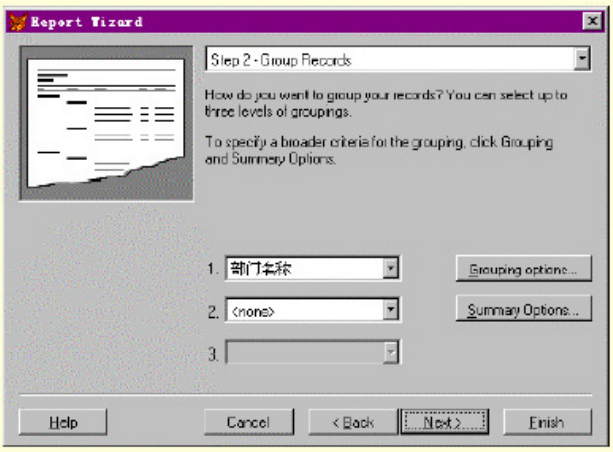


图 5-3 报表向导第二步：分组记录

可以使用数据分组来分类并排序字段，这样能够方便读取。在某个“分组类型”框中选择了一个字段之后，可以选取 Grouping options（分组选项）和 Summary Options（总结选项）来进一步完善分组设置。选择“分组选项”后将打开“分组间隔”对话框，从中可以选择与用来分组的字段中所含的数据类型相关的筛选级别。如图 5-4 所示。

图 5-4 分组间隔对话框

选择“总结选项”将打开分组选项对话框。如图 5-5 所示。

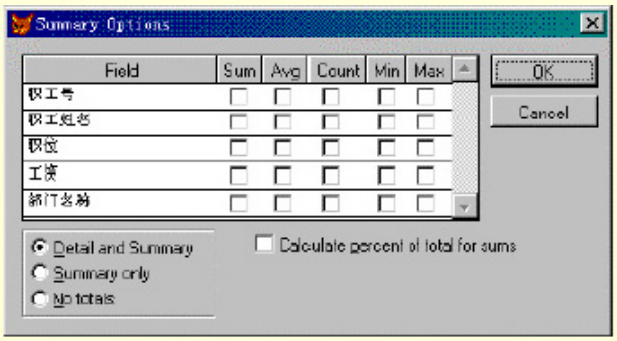


图 5-5 总结选项对话框

可以利用表 5-1 所示的计算类型来处理数值型字段：

表 5-1 处理数值型字段的计算类型

总结选项	返 回
求和	指定的数值型字段值的总和
平均值	指定的数值型字段值的平均值
计数	在指定的字段中，包含非零值的记录的个数
最小值	指定的数值型字段中的最小值
最大值	指定的数值型字段中的最大值

也可以为报表选择“细节及总结”、“只包含总结”或“不包含总计”。

在向导第二步中按下 Next 按钮以后，进入了报表向导的第三步：选择报表样式。如图 5-6 所示。

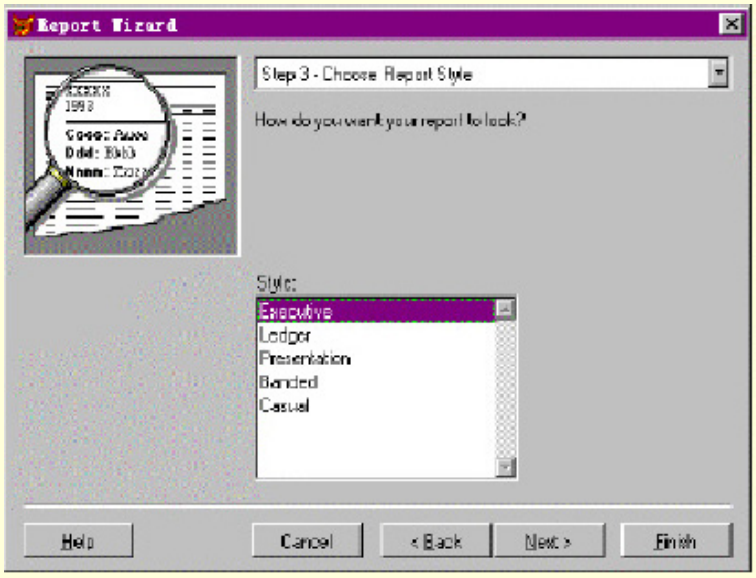


图 5-6 报表向导第三步：选择报表样式

当单击任何一种样式时，向导都在放大镜中更新成该样式的示例图片。

选择好样式，按下 Next 按钮，进入了向导的第四步：

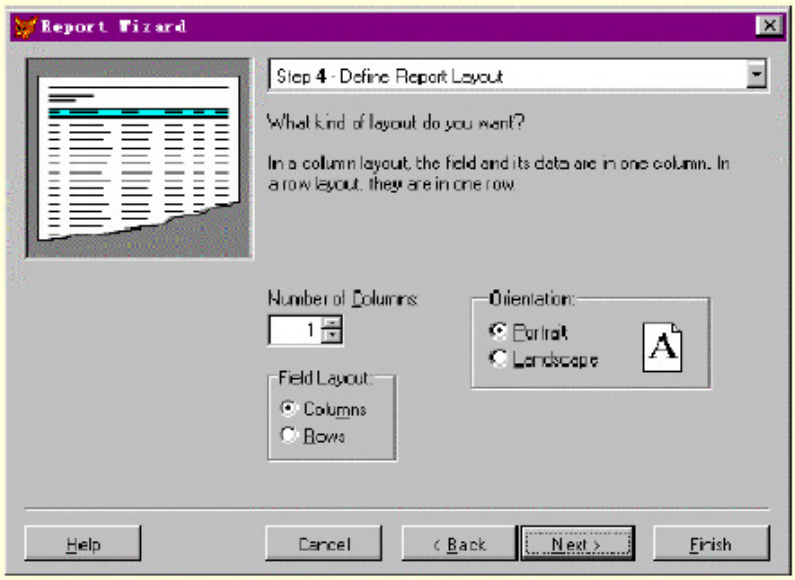


图 5-7 报表向导第第四步：定义报表布局

指定列数或布局时，向导即在放大镜中更新成选定布局的实例图形。

此一步中各个选项的意义如下：

（1）列数：确定报表打印几列，一般根据需要可以设置同时多列打印。

（2）方向：确定纸张打印的方向，是横向还是纵向。

（3）字段布局：确定字段的排列形式，是横排还是竖排。

如果在步骤 2 中指定分组选项，则本步骤中的“列数”和“字段布局”选项不可用。

确定了报表的布局以后，按下 Next 按钮，进入报表向导第五步：排序记录。如图 5-8 所示。在此一步中选择排序的列和排序方式。

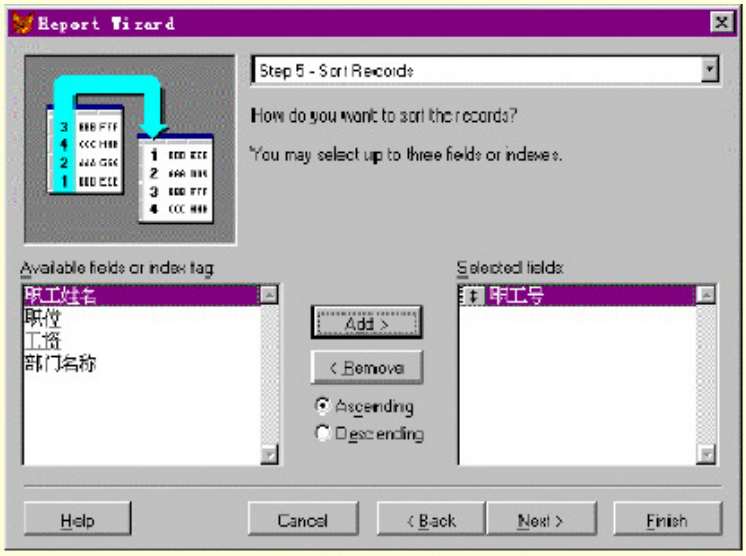


图 5-8 向导第五步：排序记录

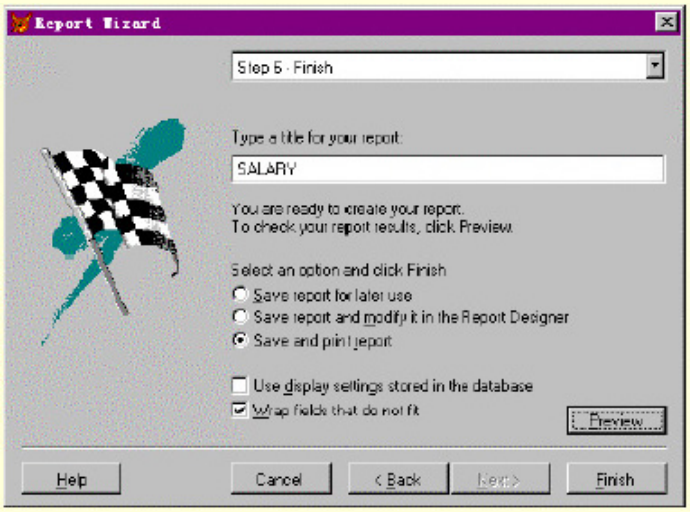


图 5-9　报表向导第六步：完成报表设计

确定记录排序以后，对报表的设计就完成了。按下 Next 按钮以后，进入报表设计的最后一步：完成报表设计。

在此一步中可以完成以下设置：

如果选定数目的字段不能放置在报表中单行指定宽度之内，字段将换到下一行上。如果不希望字段换行，清除“对不能容纳的字段进行折行处理”选项。

如果选定的表来自数据库，则本步骤可以使用数据库中的显示设置。

可以保存报表以后再使用，也可以像其他报表一样在“报表设计器”中打开或修改它，或者保存并打印报表。

单击“预览”按钮，可以在关闭向导前显示报表。

选择保存并打印或者进行预览，可以看到如下预览结果：

SALARY

01/03/99

职工号	职工姓名	职位	工资	部门名称
104	王军	职员	2,700	公关部
102	田小玲	秘书	3,000	经理部
106	王晓宾	职员	3,000	销售部

图 5-10　预览报表

5.2　报表布局

通过设计报表，可以用各种方式在打印页面上显示数据。使用“报表设计器”可以设计复杂的列表、总结摘要或数据的特定子集，比如发票。设计报表有四个主要步骤：

（1）决定要创建的报表类型。

（2）创建报表布局文件。

（3）修改和定制布局文件。

（4）预览和打印报表。

5.2.1　选择类型

创建报表之前，应该确定所需报表的常规格式，报表主要有如上五种类型，如图 5-11 所示。

图 5-11 报表主要的几种类型

报表可能同基于单表的电话号码列表一样简单，也可能复杂得象基于多表的发票那样。另外还可能创建特殊种类的报表。例如，邮件标签便是一种特殊的报表，其布局必须满足专用纸张的要求。所以，选择合适的报表类型对设计报表很重要。

为帮助选择布局，这里给出常规布局的一些说明，以及它们的一般用途及示例。

表 5-2 报表的常规布局

布局类型	说 明	示 例
列报表	每行一条记录，每条记录的字段在页面上按水平方向放置。	分组/总计报表 1 财政报表 存货清单 销售总结
行报表	一列的记录，每条记录的字段在一侧竖直放置	列表
一对多报表*	一条记录或一对多关系	发票 会计报表
多列报表	多列的记录，每条记录的字段沿左边缘竖直放置	电话号码薄 名片
标签	多列记录，每条记录的字段沿左边缘竖直放置，打印在特殊纸上	邮件标签 1 名字标签

*： 与报表向导关联的布局

5.2.2 设计报表

Visual FoxPro 6.0 可以用以下三种方式设计报表的布局：

（1）用“报表向导”创建简单的单表或多表报表。

（2）用“快速报表”从单表中创建一个简单报表。

（3）用“报表设计器”修改已有的报表或创建自己的报表。

以上每种方法创建的报表布局文件都可以用“报表设计器”进行修改。“报表向导”前面已经讲过，它是创建报表的最简单途径，它自动提供很多“报表设计器”的定制功能。“快速报表”是创建简单布局的捷径。此处主要介绍如何用报表设计器创建报表。

创建快速报表的步骤如下：

1．打开报表设计器

在“项目管理器”窗口中，选择“报表”。选择“新建”按钮。在弹出的对话框中选择“新建报表”。

或者选择 File 菜单下的 New 命令，在新建对话框中选择报表，单击新建文件按钮。

此时显示一空白的“报表设计器”。可以使用“报表设计器”的任一功能来添加控件和定制报表。

2．选择快速报表

此时主菜单项已经有了 Report 一项。选择 Quick Report 命令。然后系统提示选择一个数据库或者自由表。

选择好数据源并单击 OK 按钮以后，进入如图 5-12 所示的快速报表对话框。

图 5-12 快速报表对话框

利用快速报表可以自动将选定字段放入一个空的报表设计器窗口中。快速报表对话框中各选项的意义如下：

（1）字段按钮：显示字段选择器对话框，如图 5-13 所示。可以在此对话框中选择要在报表中显示的字段。

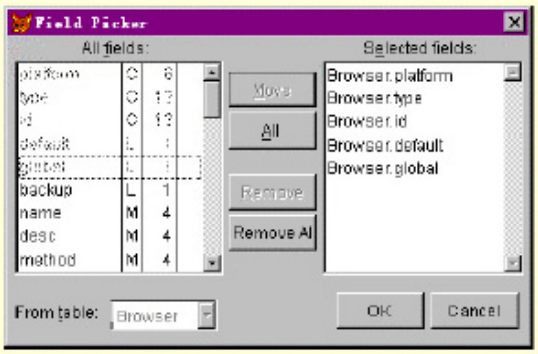


图 5-13 字段选择器对话框

（2）字段布局：左侧按钮显示列布局，右侧显示行布局。列布局可使字段在页面上从左到右排列，选择行布局可以使字段在页面上从上到下排列。

（3）标题：确定是否将字段名作为标签控件的标题置于相应的字段上面或旁边。

（4）添加别名：自动在报表设计器中为所有的字段添加别名。

（5）将表添加到设计环境中：自动将表添加到数据环境中。

单击快速报表对话框中的 OK 按钮后，选择的字段将显示在报表设计器中。如图 5-14 所示。关于使用报表设计器设计报表布局，将在下一节详细介绍。

图 5-14 选定字段后的报表设计器

5.3 报表设计器的使用

利用报表设计器可以设计出美观、实用的报表。特别是复杂报表的设计，更是离不开报表设计器的帮助。在使用报表设计器之前，先熟悉报表设计器是非常必要的。

在“报表设计器”的带区中，可以插入各种控件，它们包含打印的报表中所需的标签、字段、变量和表达式。例如，在电话号码列表布局中，应把字段控件设置成人名和电话号码，同时应设置标签控件和列表顶端的列标题。要增强报表的视觉效果和可读性，还可以添加直线、矩形以及圆角矩形等控件。也可以包含图片/OLE 绑定型控件。

5.3.1 报表带区

如果有报表布局，则可以修改数据在报表页面上的位置。使用“报表设计器”内的带区，可以控制数据在页面上的打印位置。报表布局可以有几个带区。图 5-15 为报表中可能包含的一些带区以及每个带区的典型内容。每个带区下的栏标识了该带区。

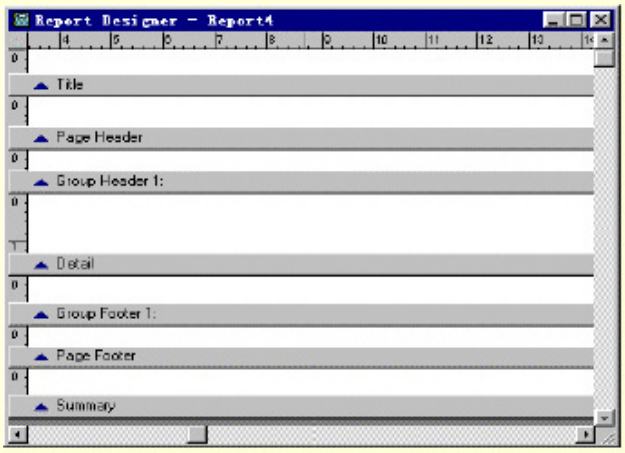


图 5-15 报表设计器带区

报表也可能有多个分组带区或者多个列标头和注脚带区。表 5-3 列出了各个带区的功能与用法。

表 5-3 报表带区的功能与用法

带 区	打 印	使用命令
标题	每报表一次	从“报表”菜单中选择“标题/总结”带区。
页标头	每页一次	默认可用。
列标头	每列一次	从“文件”菜单中选择“页面设置”，设置“列数”>1
组标头	每组一次	从“报表”菜单中选择“数据分组”。
细节带区	每记录一次	默认可用。
组注脚	每组一次	从“报表”菜单中选择“数据分组”。
列注脚	每列一次	从“文件”菜单中选择“页面设置”，设置“列数”>1
页注脚	每页一次	默认可用。
总结	每报表一次	从“报表”菜单中选择“标题/总结”带区

使用“标题”和“总结”带区时，在菜单 Report 下选择“Tile/Summary”命令，将弹出如图 5-16 所示的“标题/总结”对话框。

在此对话框中选中“标题带区”复选框表示往报表里添加一个“标题”带区，同时选中“新页”复选框表示把标题内容单独打印一页。如果选中“总结带区”复选框，则表示往报表里添加一个“总结”带区，同时选中“新页”复选框表示把总结内容单独打印一页。

调整带区大小的一种方法是通过用鼠标选中某一带区并按下鼠标左键，通过上下拖动来调整大小，直到自己满意。也可以双击要调整高度的带区的标识栏，系统将弹出一个对话框。例如，双击页标头带区的标识栏将显示页标头对话框。如图 5-17 所示。

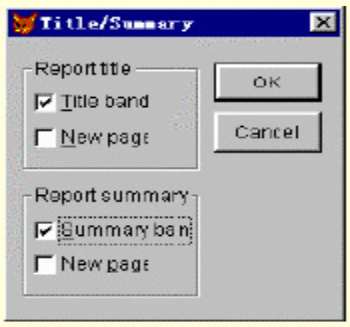


图 5-16 标题/总结对话框

图 5-17 页标头对话框

在此对话框中，可以通过调整高度微调器里的数值来调整页标头的对话框。选中下面的“带区高度保持不变”复选框后，可以防止报表带区因为容纳过长的数据或者从其中移去数据而移动。

在此对话框中还可以设置两个表达式：入口处运行表达式或者出口处运行表达式。如果设置了入口处运行表达式，则在打印该带区内容之前计算表达式；如果设置了出口表达式，则打印该带区内容之后计算表达式。

5.3.2 打印输出设置

当打印报表时，通常会考虑页面的外观。例如页边距、纸张类型和所需的布局。下面将对如何设置页边距、页面方向和报表页面带区的高度进行介绍。

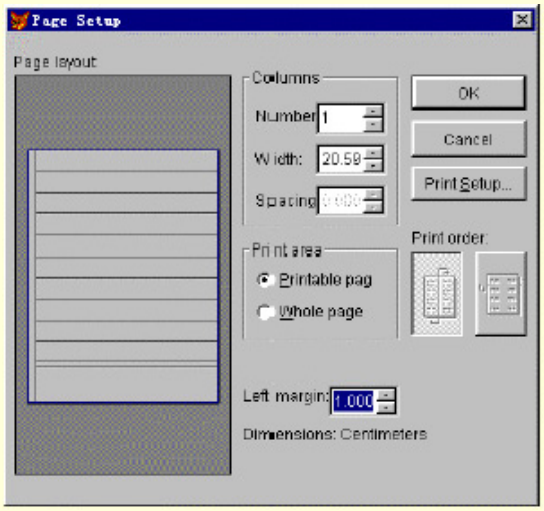


图 5-18 页面设置对话框

可以设置报表的左边距并为多列报表设置列宽和列间距。在这种情况下，“列”一词指的是页面横向上打印的记录的数目，不是单条记录的字段数目。“报表设计器”没有显示这种设置。它仅显示了页边距内的区域，其中包含了页面中包含一条记录的一列。因此，如果报表中有多列，当更改左边距时，列宽将自动更改以显示出新的页边距。

如果更改了纸张的大小和方向设置，请确认该方向适用于所选的纸张大小。例如，如果纸张定为信封，则方向必须设置为横向。

如果要设置页面，选择 File 菜单下的 Page Setup 命令。将显示如图 5-18 所示的对话框。

页面布局模拟框显示当前各个选项的设置结果。列数微调器可调整报表列数，宽度微调器可调整各列的的宽度，间隔微调器调整每列的间隔距离。

在“左页边距”框中输入一个边距数值。页面布局将按新的页边距显示。

若要选择纸张大小，可选择“打印设置”。在“打印设置”对话框中，从“大小”列表中选定纸张大小。若要选择纸张方向，从“方向”区选择一种方向，再选择“确定”。

5.3.3 选择报表数据源

报表中要显示一定的数据，如果该报表总是显示固定的数据源，可以在数据环境中简单地定义报表的数据源，用它们来填充报表中的控件。

报表的数据源可以是数据库表、视图或者自由表。可以添加表或视图并使用一个表或视图的索引来排序数据。

若要向数据环境中添加表或视图，可采用如下步骤：

（1）从“显示”菜单中，选择“数据环境”。或者在报表设计器中单击鼠标右键，在弹出的快捷菜单中选择“数据环境”。将显示报表的数据环境设计器。

（2）从“数据环境”菜单中，选择“添加”。或者在数据环境设计器中单击鼠标右键，在快捷菜单中选择“添加”。

（3）在“添加表或视图”对话框中，从“数据库”框中选择一数据库。或者直接选择一个表。

（4）在“数据库中的表/视图”框中，选取一个表或视图。

（5）选择“添加”按钮。将把选择的表或者视图添加到报表的数据环境设计器中去。如图 5-19 所示。

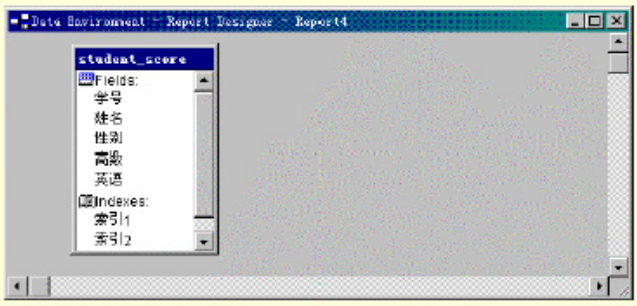


图 5-19 报表设计器数据环境

可以设置各条记录出现在报表中的顺序，方法是先为数据环境设置索引。

若要为数据环境设置索引，采用如下步骤：

（1）在数据环境设计器中单击鼠标右键。从快捷菜单中，选择“属性”命令。将弹出属性窗口，如图 5-20 所示。

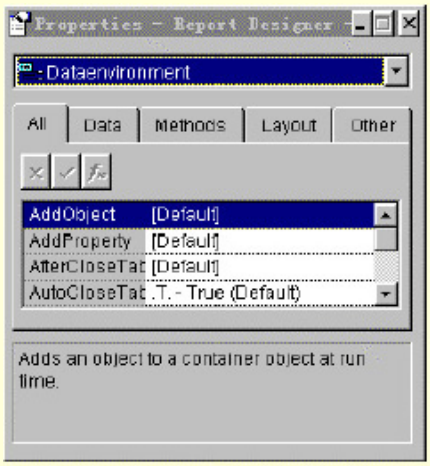


图 5-20 报表设计器属性窗口

（2）在“属性”窗口中，选择对象框中的“Cursor1”。

（3）选择“数据”选项卡，然后，选定“Order”属性。

（4）输入索引名。或者从可用索引列表中选定一个索引。

如果要防止其他设计器对全局数据工作期的更改影响当前报表的数据工作期，则把报表的数据工作期设置为私有。方法是选择 Report 菜单下的 Private Data Session（私有数据工作期），当 Private Data Session 前面有一个对号时，表示当前报表的数据工作期已被设置为私有。

5.3.4 报表控件

可以在报表和标签布局中插入以下类型报表控件。

表 5-4 增添报表控件

显示类型	选用控件
表的字段、变量和其他表达式	域控件
原义文本	标签
直线	线条
框和边界	矩形
圆、椭圆、圆角矩形和边界	圆角矩形
位图或通用字段	图片/ActiveX 绑定控件

设置控件后，可以更改它们的格式、大小、颜色、位置和打印选项。也可仅出于参考目的而为每个控件添加注释，实际上在报表内并不打印。

1．域控件

报表或标签可以包含域控件，它们表示表的字段、变量和计算结果。

若要从数据环境中添加表中字段，采用如下步骤：

（1）打开报表的数据环境。

（2）选择表或视图。

（3）拖放字段到布局上。

若要从工具栏添加表中字段，可以采用如下步骤：

（1）从“报表控件”工具栏中，插入一个“域控件”。

（2）在“报表表达式”对话框中，选择“表达式”框后的对话按钮。

（3）在表达式生成器对话框中双击所需的字段名。

（4）表名和字段名将出现在“报表字段的表达式”内。如图 5-21 所示。

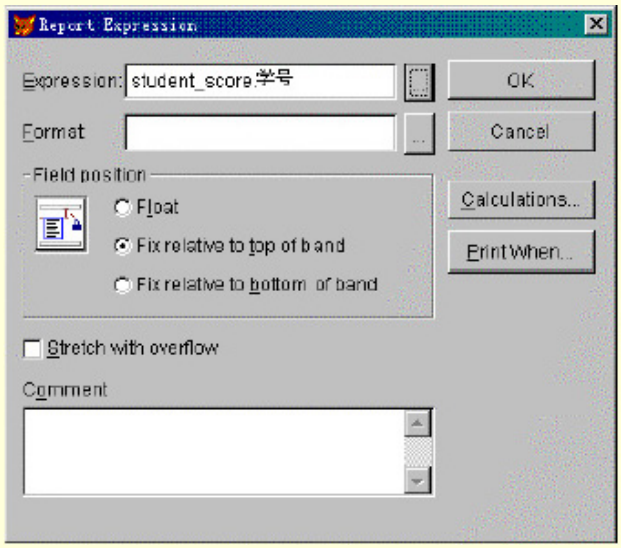


图 5-21　报表表达式对话框

注意：如果“字段”框为空，则应该向数据环境添加表或视图。不必保持表达式中表的别名。可以删除它或者清除“表达式生成器”对话框选项。

（5）单击“确定”按钮。

（6）在“报表表达式”对话框中，选择“确定”按钮。

对于一些类型的字段，像字符型字段，在报表中显示时也许会产生一些不必要的空格，此时如果把多个字段合并到一起，可以在表达式对话框中输入一个或者多个字段，例如下面的表达式：

ALLTRIM（student_score.学号）＋“ ”＋ALLTRIM（student_score.姓名）

（1）可以在格式文本框中定义域控件的格式，或者单击格式文本框右边的的按钮，将显示格式对话框，如图 5-22 所示。在此对话框中定制域控件的格式。

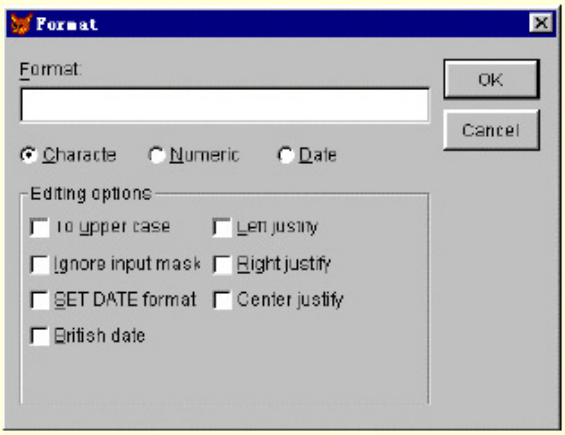


图 5-22 域控件格式对话框

格式让用户输入格式表达式或输入格式模板而不对下面的选项进行设置。

字符型选项指定字符数据类型，并显示相应的“编辑选项”。

数值型选项指定数值数据类型，并显示相应的“编辑选项”。

日期型选项指定日期数据类型，并显示相应的“编辑选项”。

编辑选项此选项组随数据类型（字符、数值、日期）的改变而改变。通过选择这些选项可将格式代码置于“格式”框中。

全部大写（字符型）：将所有字符转化为大写。

忽略输入掩码：显示但不存储不符合格式的字符。

左对齐（数值型和字符型）：从选定字段位置的最左端开始显示数值。

右对齐（字符型）：从最右端开始显示数据。

居中对齐（字符型）：将数据放在字段中央。

如果为零保持为空（数值型）：如果字段输出为零，则不打印。

负数加括号（数值型）：将负数放入括号中。

SET DATE 格式（所有类型）：使用 SET DATE 当前格式显示日期数据。

英国日期格式（所有类型）：使用欧洲（英国）日期格式显示数据。

如果为正，加 CR（收入）（数值型）：在正数后显示 CR（贷方）。

如果为负，加 DB（支出）（数值型）：在负数后显示 DB（借方）。

前导零（数值型）：打印全部的前导零。

货币型（数值型）：按“选项”对话框的“区域”选项卡中指定格式显示货币格式。

科学计数法（数值型）：以科学计数法显示数据（当数值很大或很小时使用）。

（2）单击报表表达式对话框中计算按钮将显示“计算字段”对话框，以便选择一个表达式来创建一个计算字段。如图 5-23 所示。

计算字段对话框允许选择一个数学操作符，创建一个计算结果字段。该对话框中有如下选项：

① 重置

允许指定把表达式重置为初始值。默认值是“报表尾”，也可选择“页尾”或“列尾”。如果使用“数据分组”对话框在报表中创建组，“重置”框为报表中的每一组显示一个重置项。

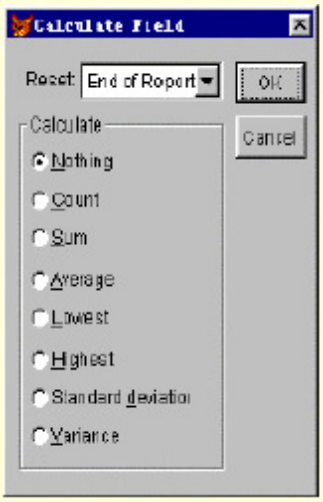


图 5-23 计算字段对话框

② 计算

这些选项指定在报表表达式中执行的计算。

不计算：指定不计算此表达式（默认设置）。

计数：计算每组、每页、每列或每个报表（取决于在“重置”框中的选择）中打印变量的次数。此计算操作基于变量出现的次数，而不是变量的值。

求和：计算变量值的总和。求和操作在运行时对每组、每页、每列或每个报表（取决于在“重置”框中的选择）进行变量值的求和计算。

平均值：在组、页、列或报表（取决于在“重置”框中的选择）中计算变量的算术平均值。

最小值：在组、页、列或报表（取决于在“重置”框中的选择）中显示变量的最小值。将组中第一个记录的值放入变量，当更小的值出现时，此变量的值随之更改。

最大值：在组、页、列或报表（取决于在“重置”框中的选择）中显示变量的最大值。将组中第一个记录的值放入变量，当更大的值出现时，此变量的值随之更改。

标准误差：返回组、页、列或报表（取决于在“重置”框中的选择）中变量的方差的平方根。

方差：衡量组、页、列或报表（取决于在“重置”框中的选择）中各个字段值与平均值的偏离程度。

（3）打印条件

显示“打印条件”对话框以便精确设置何时在报表或标签中打印文本。如图 5-24 所示。

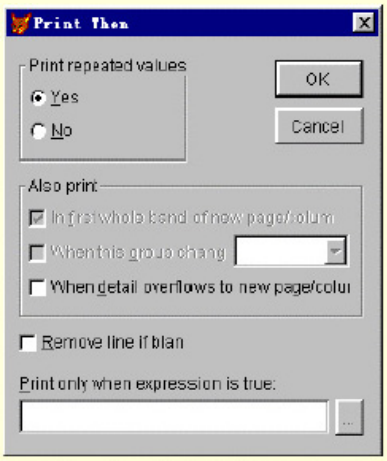


图 5-24 打印条件对话框

可利用此对话框设置表达式，根据表达式的值对一个对象进行条件打印，以及控制如何打印各信息带中的对象。

对于字段对象，此对话框控制各个信息带中的重复值是否打印。对于非字段对象，此对话框控制其在各连

续信息带中的打印方式。

在标签或报表中放置控件，并在此控件的相应对话框中选择“打印条件”，显示此对话框。

当在标签或报表中画一个矩形或线条并双击鼠标，然后从“矩形/线条”对话框中选择“打印条件”时，所得到的“打印条件”对话框与字段对象的“打印条件”对话框稍有不同。

此对话框中有如下选项：

① Also print

此选项仅用于非字段对象（如 OLE 对象、标签、线条、矩形和圆角矩形等控件）。

如果选 Yes，则开始打印包含此对象的信息带时，打印此对象。

如果选 No，则只有选中以下选项中的一个或全部时，才打印对象：“在新页/列的第一个完整信息带内打印”和“当细节区数据溢出到新页/列时打印”。

② 打印重复值

此选项仅用于“字段”对象。

若选是，则打印重复值。

若选否，则不打印重复值。

③ 有条件打印

在新页/列的第一个完整信息带内打印：在新页或新列的第一个完整信息带内而不是在前一页或前一列溢出的信息带内打印此字段。如果“打印重复值”项选择“是”，则此项被自动选定；如果“打印重复值”项选择“否”，则此项为可选项。

当此组改变时打印：当选定的组改变时，打印此字段。只有当数据组存在，并且从列表框中选定了一个数据组时，此选项可用。

当细节区数据溢出到新页/列时打印：当“细节”带区中的某一信息带溢出到新页或新列时，打印此字段。

④ 若是空白行则删除

如果没有对象在打印，同时又没有其他对象位于同一水平位置上，就删除对象所在行。

⑤ 仅当下列表达式为真时打印

可输入表达式，或显示“表达式生成器”对话框，从而定义在打印字段之前要计算的表达式。如果使用了一个表达式，则此对话框中，除“若是空白行则删除”选项外的所有选项都不可用。

（4）在报表表达式对话框中还可以控制域控件位置

浮动：指定选定字段相对于周围字段的大小移动。

相对于带区顶端固定：使字段在“报表”或“标签设计器”中保持您指定的位置，并维持其相对于带区顶端的位置。

相对于带区底端固定：使字段在“报表”或“标签设计器”中保持指定的位置，并维持其相对于带区底端的位置。

（5）溢出时伸展

使字段伸展到域控件宽度以外地地方，显示字段或表达式中的所有数据。

（6）备注

向 .frx 或 .lbx 文件中添加注释。注释仅用于参考，并不出现在打印的报表或标签中。

2．添加标签控件

标签控件是希望出现在报表中的原义文本字符。它在报表中应用相当广泛，例如在每一字段前加说明性文字，或者报表的标题等，都是用标签。

若要添加标签控件，可以从报表控件工具栏中，选择“标签”按钮。在“报表设计器”中单击，可将一个标签控件放置在报表中。键入该标签的字符即可。

可以在文本编辑器内随意编辑：使用 ENTER 键换行或使用“编辑”菜单剪切、复制和粘贴文本。

若要编辑标签控件，可以在“报表设计器”中选择“标签”按钮，然后单击需编辑的标签。键入修改内容即可。

设置文本后，可以更改字体、文本颜色、背景色以及打印选项。

3．添加通用字段

可以在报表中插入包含 OLE 对象的通用型字段。一个 OLE 对象可能是图片、声音或者 Word、Excel 文档等。Visual FoxPro 6.0 的表中可以包含这些对象，则报表也可以包含这些对象。利用添加 OLE 对象，可以在报表中显示人员、产品的照片，或者录音等等。

若要插入通用型字段，按照如下步骤：

（1）在报表设计器中，添加图片/ ActiveX 绑定控件。将显示 Report Picture（报表图片）对话框，如图 5-25 所示。

（2）在 Picture from（图片来源）区域，选择 Field 或 File.

（3）在 Field 框中，键入字段名。或者选择使用对话按钮来选取字段或变量。

（4）单击 OK 按钮。

通用字段的占位符将出现在定义的图文框内。默认情况下，图片保持其原始大小。

在图片来源选择中，可以选择两种形式的图片，来自文件或者来自字段。如果图片不随记录的变化而变化则选择 File，选择一个图片文件或者输入文件名；如果要显示的是表中通用字段的图片内容，则选择字段，并在“选择字段/变量”对话框中选择表中的字段或者直接输入字段名。如果通用字段的内容不是图片或者图表，则代表此对象的图标将显示在报表中。

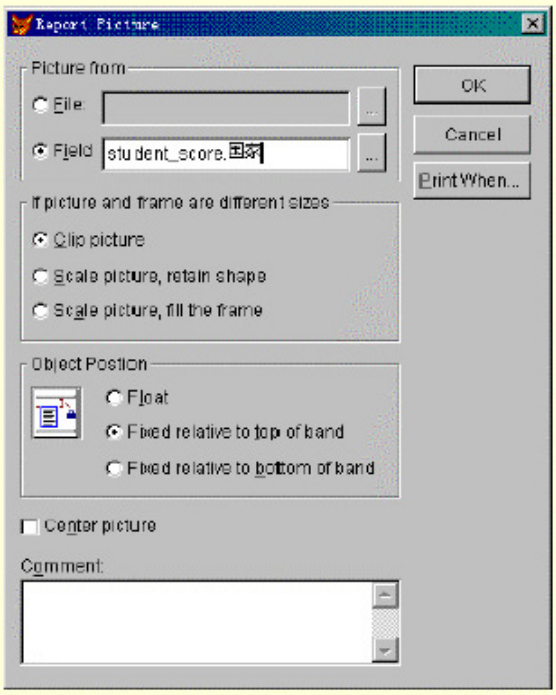


图 5-25　报表图片对话框

在 If picture and frame are different sizes（假如图片和图片框大小不一致）选项中可以进行如下设置：

■Clip picture（剪裁图片）：以图片框的大小显示图片，有时可能会因为图片框太小而只有部分图片被显示出来。

■Scale picture，retain shape（缩放图片，保持形状）：在图片框中显示一个完整的、不变形的图片，这种情况下可能会因为图片无法填满整个图片框。

■Scale picture，fill the frame（缩放图片，填充图片框）：图片填满整个图片框，但图片的比例可能会改变。

如果希望图片在居中位置放置，可以选择“图片居中”复选框。这样图片框中比图片框小的图片在控件的正中显示。如果图片的来源是文件，则此复选框不可用，因为储存在文件中的图片的形状和大小都是固定的，不需要居中放置。

在 Object Position（图片位置）选项中可以有三种选择：

■Float（浮动）：图片相对于周围的字段和大小移动。

■Fixed relative to top of band（相对于带区顶端固定）：使图片保持在报表中指定的位置上，并保持其相对于带区顶端的位置。

■Fixed relative to bottom of band（相对于带区底端固定）：使图片保持在报表中指定的位置上，并保持其相对于带区底端的位置。

在注释编辑框中可以为图片添加注释文本，这些文本不会在报表中打印。

4．添加线条、矩形和圆形

直线、矩形和圆形增强了报表布局的视觉效果，当报表中仅包含数据时，报表是不够美观的。可用它们分割或强调报表中的部分内容。

使用“线条”控件，可以在报表布局中添加垂直和水平直线。通常，需要在报表主体内的详细内容和报表的页眉和页脚之间划线。

若要绘制线条，从“报表控件”工具栏中，选择“线条”按钮。在“报表设计器”中，拖动光标可以调整线条。

绘制线条后，可以移动或调整其大小，或者更改它的粗细和颜色。方法是选择“格式”菜单下的“Pen”（绘图笔）菜单。如图 5-26 所示。

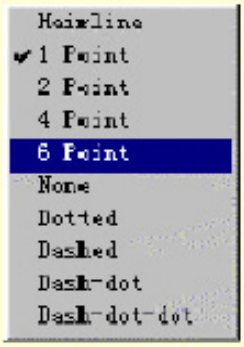


图 5-26　绘图笔菜单

在此菜单中可以设置线条的形状和粗细。

可以在布局上绘制矩形，从而以醒目的方式组织打印在页面上的信息。也可以把它们作为报表控件、报表带区或者整个页面周围的边框使用。

若要绘制矩形，从报表控件工具栏中选择“矩形”按钮。在“报表设计器”中，拖动光标可以调整矩形的大小。

若要绘制矩形或圆形，采用如下步骤：

（1）从报表控件工具栏中，选择“圆角矩形”按钮。在“报表设计器”中，拖动光标以调整该控件。

在“样式”区域中可以选择想要的圆角样式。

如果需要，设置矩形的位置。

可以更改垂直、水平线条、矩形和圆角矩形所用线条的粗细（从细线到六磅粗的线），也可以更改线条的样式（从点线到点线和虚线的组合）。

若要更改线条的大小或样式，选定希望更改的直线，矩形或圆角矩形。从“格式”菜单中选择“绘图笔”。从子菜单中选择适当的大小或样式。如图 5-26 所示。

双击该控件，将显示“圆角矩形”对话框。如图 5-27 所示。

在圆角矩形对话框中，一共有 5 种样式可选择，当圆角矩形的范围不足以显示指定的数据时，如果选择：

■不伸展：表示当带区伸展显示数据时，圆角矩形不伸展。

■相对于组中最高的对象伸展：表示圆角矩形伸展到组中最高对象的高度。

■相对于带区高度伸展：表示指定矩形伸展到带区的大小。

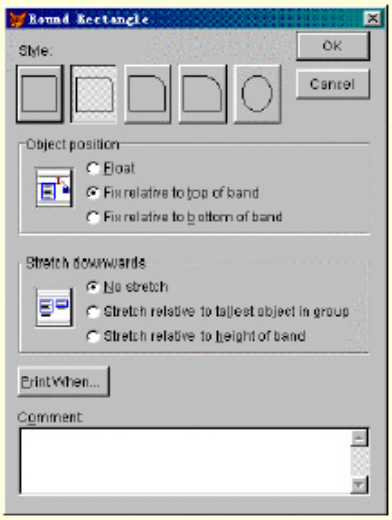


图 5-27 圆角矩形对话框

（2）设置标签控件、域控件的字体及颜色

可以更改每个域控件或标签控件中文本的字体和大小。也可以更改报表的默认字体。

若要更改报表中的字体和大小，可以按照如下步骤：

（1）选定要更改的控件。

（2）从 Format 菜单中选定 Font。

（3）此时显示“字体”对话框。如图 5-28 所示。

图 5-28 字体对话框

（4）选定适当的字体和磅值，然后选择“确定”按钮。

若要更改默认字体，从 Report （报表）菜单中选择 Default font（默认字体）。在“字体”对话框内，选择想要的适当字体和磅值作为默认值，然后选择“确定”按钮。

只有改变默认字体后，插入的控件才会反映出新设置的字体。

5.3.5 报表变量

若要在报表中操作数据或显示计算结果，可以使用报表变量。使用报表变量，可以计算各种值，并且可以用这些值来计算其他相关值。

例如，在学生成绩表中，如果要在报表中统计每个同学的总成绩和平均成绩，并打印出来。这时，使用报表变量可以很方便地完成设计要求，并且不需要修改表的结构。

若要定义报表变量，采用如下步骤：

（1）首先打开或创建一个报表。

（2）从 Report（报表）菜单中选择 Variables（变量）命令。

（3）在 Report Variables（报表变量）对话框中，选择 Variables 框并键入一个变量名。如图 5-29 所示。

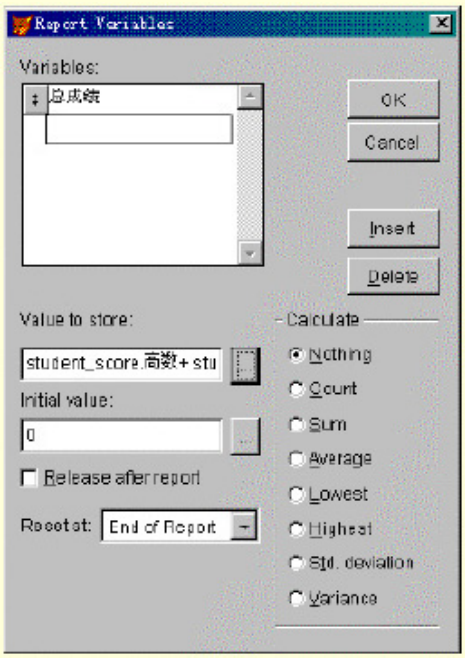


图 5-29 报表变量对话框

例如，在此处建立一个报表变量“总成绩”。

在“要存储的值”框中，键入一个字段名或其他表达式。

例如，此处“总成绩”的值为：则键入的表达式为：student_score.高数 + student_score.英语

（5）如果需要，选择一个计算选项。

（6）如果需要，在 Initial value（初始值）框中键入一个设置初始值的表达式。

（7）单击 OK（确定）按钮。

报表变量对话框允许创建报表中的变量。Visual FoxPro 使用变量来保存打印报表时所计算的结果。使用此命令可以添加新的变量，改变或删除已有变量，或者更改变量的计算顺序。

在报表变量对话框中，可以进行下面的选项设置：

（1）变量

显示当前报表中的变量，并为新变量提供输入位置。

（2）要存储的值

显示存储在当前变量中的表达式，也可以在文本框中输入表达式。要创建一个存入变量的表达式，可选择对话按钮，显示“表达式生成器”对话框。

（3）初始值

在进行任何计算之前，显示选定变量的值以及此变量的重置值。可以直接在文本框中输入一个值，或者选择对话按钮，显示“表达式生成器”对话框，为初始值创建表达式。

（4）报表输出后释放

在报表打印后从内存中释放变量。如果未选定此选项，那么除非退出 Visual FoxPro 或使用 CLEAR ALL 或 CLEAR MEMORY 命令来释放变量，否则此变量一直保留在内存中。

（5）重置

指定变量重置为初始值的位置。“报表尾”是其默认值，也可选择“页尾”或“列尾”。如果使用“数据分组”命令在报表中创建组，“重置”框将为报表中的每一组显示一个重置项。

（6）插入

在“变量”框中插入一个空文本框，以便定义新的变量。

（7）删除

在“变量”框中删除选定的变量。

（8）计算

用来指定变量执行的计算操作。从其初始值开始计算，直到变量被再次重置为初始值为止。此选项中的选

择的意义同计算计算字段对话框中的设置的意义。见图 5-23。

报表变量是根据它们出现的先后顺序来计算，并且会影响引用了这些报表变量的表达式的值。例如，如果用变量 1 定义变量 2 的值时，那么变量 1 应该在变量 2 之前出现。例如，在前一个示例中，如果再定义一个变量“平均成绩”，平均成绩＝总成绩/2，则变量“平均成绩”应出现在变量“总成绩”之后。

5.3.6 为记录分组

设计基本布局后，若要根据给定字段或其他条件对记录分组，会使报表更易于阅读。可以添加一个或多个组，更改组的顺序，重复组标头或者更改或删除组带区。分组允许明显地分隔每组记录和为组显示介绍和总结性数据。组的分隔基于分组表达式。这个表达式通常由一个以上的表字段生成，但也可以相当复杂。

分组之后，报表布局就有了组标头和组注脚带区，可以向其中添加控件。一般地，组标头带区中包含组所用字段的“域控件”，可以添加线条、矩形、圆角矩形或希望出现在组内第一条记录之前的任何标签。组注脚通常包含组总计和其他组总结性信息。

也可以为组指定如下选项：

（1）在标头和注脚内打印文本以标识特定组。

（2）在新页面上打印每一组。

（3）当组打印到新页面时重置页码。

如果数据源是表，记录的顺序可能不适合于分组。通过为表设置索引，或者在数据环境中使用视图，或者使用查询作为数据源，可以把数据适当排序来分组显示记录。报表布局实际上并不排序数据，它只是按它们在数据源中存在的顺序处理数据。排序必须使用视图、索引或布局外的其他形式的数据操作来完成。

例如，可以将成绩表按照男生和女生分为两个组，然后再按照每个学生的籍贯进行二次分组。

若要添加组，可以采用如下步骤：

（1）从 Report（报表）菜单中，选择 Data Grouping（数据分组）。将显示 Data Grouping 对话框。如图 5-30 所示。

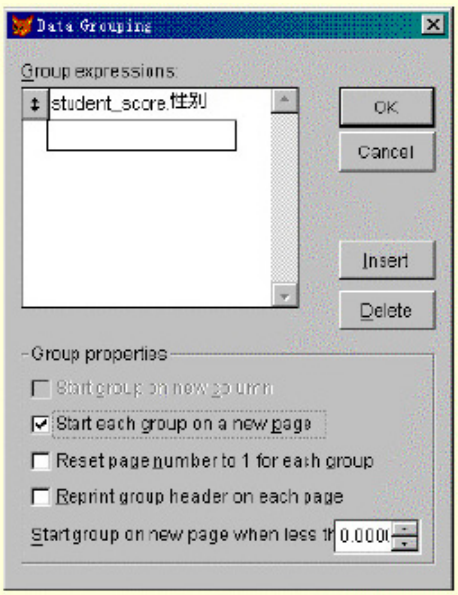


图 5-30 数据分组对话框

（2）在第一个“分组表达式”框内键入分组表达式。或者选择对话按钮，在“表达式生成器”对话框中创建表达式。

（3）在“组属性”区域，选定想要的属性。

（4）选择“确定”按钮。

添加表达式后，可以在带区内放置任意需要的控件。通常，把分组所用的域控件从“细节”带区移动到“组标头”带区。

在报表内最多可以定义 20 级的数据分组。嵌套分组有助于组织不同层次的数据和总计表达式。

若要选择一个分组层次，请先估计一下更改值的可能频度，然后定义最经常更改的组为第一层。例如，报

表可能需要一个按地区的分组和一个按城市的分组。城市字段的值比地区字段更易更改，因此，城市应该是两个组中的第一个，地区就是第二个。在这个多组报表内，表必须在一个关键值表达式上排序或索引过，例如 Region+City 。

在报表中的组定义之后，可以更改它们的次序。当组重新排序时，组带区中定义的所有控件都将移到新的位置。重新排序组并不更改以前定义的控件。如果框或线条以前是相对于组带区的上部或底部定位的，那么它们仍将固定在组带区的原位置。

若要更改组的次序，在“报表”菜单中，选择“数据分组”。选中想移动的组左侧的移动按钮，并把它拖到新位置。

5.3.7 报表实例

要创建一个基于表 Student_score（如图 5-31 所示）的报表“学生成绩表”。要求报表中的记录按照性别进行分组，同时要统计每一个同学的总成绩。

学号	姓名	性别	高数	英语	国家
1001	王钢	男	76	87	Gen
1002	李宁	男	97	76	Gen
1003	谢小飞	男	87	79	Gen
1004	杨兰	女	98	96	Gen

图 5-31 表 Student_score

下图是基于表 Student_score 创建的报表“学生成绩表”，如图 5-32 所示。

图 5-32 创建学生成绩表报表

报表标题为“学生成绩表”，出现在页标头中的各个字段名均用标签控件。除此之外，还用到了线控件，圆角矩形控件。而小狐狸图标是使用了 OLE 控件，图片来自文件 Fox.bmp.

细节带区中，姓名、英语、高数、学号都是使用表中的字段，使用域控件。其中国家字段是通用型字段，使用 OLE 控件，图片来自表中字段“国家”。

报表中使用了一级分组，按照性别进行分组。字段性别放在组标头带区中。

而字段总成绩使用的是报表中变量“总成绩”，此变量的要储存的值为字段高数加上英语的和。对报表的预览结果如图 5-33 所示。

图 5-33 报表“学生成绩表”预览结果

5.4 输出报表

设计报表的最终目的是为了将数据输出。在输出之前可以预览报表。

开始设置报表或标签的布局后，也可以预览工作结果或打印一份报表或标签。定制期间可以随时进行预览。

5.4.1 预览

通过预览报表，不用打印就能看到它的页面外观。例如，可以检查数据列的对齐和间隔，或者查看报表是否返回所需的数据。有两个选择：显示整个页面或者缩小到一部分页面。“预览”窗口有它自己的工具栏，使用其中的按钮可以一页一页地进行预览。如果得到提示“是否将所做更改保存到文件？”，那么，在选定关闭“预览”窗口时一定还选取了关闭布局文件。此时可以选定“取消”按钮回到“预览”，或者选定“保存”按钮保存所做更改并关闭文件。如果选定了“否”，将不保存对布局所做的任何更改。

可以按照如下步骤预览报表布局：

(1) 从“显示”菜单中，选择“预览”。或在报表设计器中的快捷菜单中选择预览命令。

(2) 在打印预览工具栏中，选择“上一页”或“前一页”来切换页面。

(3) 若要更改报表图象的大小，选择“缩放”列表。

(4) 若要打印报表，选择“打印报表”按钮。

(5) 若想要返回到设计状态，选择“关闭预览”按钮。或者直接关闭预览窗口。

5.4.2 打印报表

使用“报表设计器”创建的报表或标签布局文件只是一个外壳，它把要打印的数据组织成令人满意的格式。它按数据源中记录出现的顺序处理记录。如果直接使用表内的数据，数据就不会在布局内按组排序。在打印一个报表文件之前，应该确认数据源中已对数据进行了正确的排序。如果表是数据库的一部分，可创建视图并且把它添加到报表的数据环境中，该视图将排序数据。如果数据源是一个自由表，可创建并运行查询，并将查询结果输出到报表中。若要打印报表，从“文件”菜单中，选择“打印”，将显示打印对话框，如图 5-34 所示。

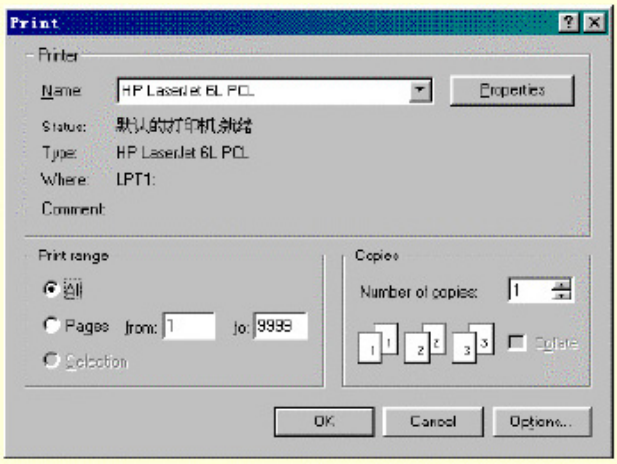


图 5-34 打印对话框

如果未设置数据环境，则显示“打开”对话框，并在其中列出一些表，从中可以选定要进行操作的一个表。然后，Visual FoxPro 把报表发送到打印机上。

此打印对话框是 Windows 应用程序的常用对话框。在此对话框中“打印机名”列表框列出了当前操作系统安装的打印机。

属性按钮主要用于设置纸张的大小与尺寸、打印精度等等选项。打印范围和打印分数选项用于设置打印的页码范围和打印多少份。

单击选项对话框，将显示一个打印选项对话框。如图 5-35 所示。

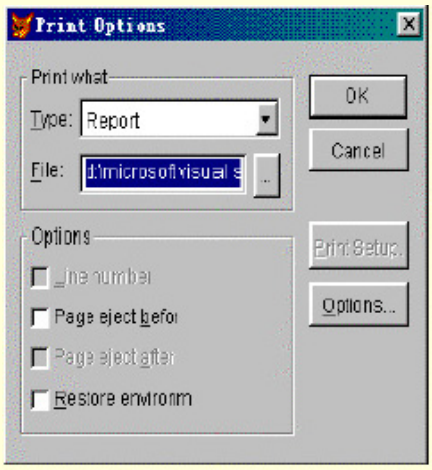


图 5-35 打印选项对话框

在 Print Options（打印选项）对话框中，可以有如下选项：

（1）打印内容

可在类型列表框中选择所需要打印的文件类型，可以是报表、标签、命令窗口、文件、ASCII 文本和剪切板。

文件文本框中可以输入或者选择一个文件。

（2）选项

行号：打印对应的行号。

打印前走纸：打印开始前先走一张纸。

打印后走纸：打印结束后走一张纸。

还原环境：打印完毕后恢复打印前的所有设置。

如果打印的文件类型是报表或者标签，则可以选择对话框中的选项按钮。将弹出如图 5-36 所示的“报表和标签打印选项”对话框，在此对话框中可以设置打印的记录范围以及记录条件。

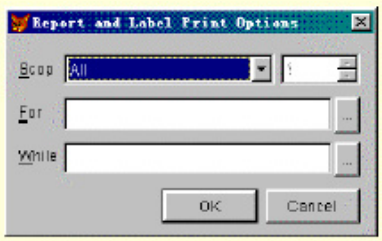


图 5-36 报表和标签打印选项对话框

例如，要打印英语成绩在 80 分以上的同学的记录信息，则可以在 For 文本框中输入如下表达式：

Student_score.英语 >=80

在 While 文本框中可以输入一个逻辑表达式，只有当逻辑表达式为“真”时才开始打印，直至逻辑表达式为“假”时才停止打印。

5.5 创建邮件标签

标签是多列报表布局，为匹配特定的标签纸而具有相应的特殊设置。在 Visual FoxPro 里，可以使用标签向导或标签设计器迅速创建标签。

5.5.1 使用标签向导

利用标签向导是创建标签的简单方法。用向导创建标签文件后，可用报表设计器定制标签文件。

若要使用“标签向导”创建标签，可以在“项目管理器”窗口中，选定“标签”。或者选择文件菜单下的新建命令。在新建对话框中选择“标签向导”。

按照向导屏幕上的指令操作， 即可创建出标签。

可以原样使用标签布局，也可以按定制报表的方法定制标签布局。可以在“标签向导”的第二步中单击“新建标签”按钮，创建一个标签。

5.5.2 启动标签设计器

如果不想使用向导来创建标签，也可以使用“标签设计器”来创建布局。“标签设计器”是“报表设计器”的一部分，它们使用相同的菜单和工具栏。两种设计器使用不同的默认页面和纸张。“报表设计器”使用整页标准纸张。“标签设计器”的默认页面和纸张与标准标签的纸张一致。

若要使用“标签设计器”创建标签，在“项目管理器”窗口中，选定“标签”。选择“新建”。选择“新建标签”，显示“新建标签”对话框，如图 5-37 所示。 标准标签纸张选项出现在“新建标签”对话框中。

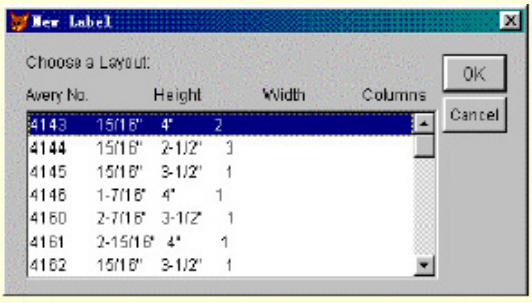


图 5-37 新建标签对话框

从“新建标签”对话框中，选择标签布局，然后选定“确定”按钮。“标签设计器”将出现刚选择的标签布局所定义的页面。标签设计器的界面同报表设计器的界面很相似，如图 5-38 所示，用法也相似。

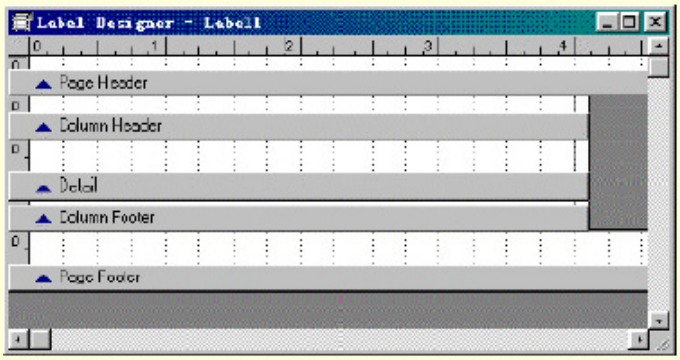


图 5-38 标签设计器

可以像处理报表一样给标签指定数据源并插入控件。此时不再赘述。

5.6 创建报表的命令

根据前面所述，使用系统菜单和设计器设计报表是比较方便的。同时，Visual FoxPro 也提供了若干条命令，如 REPORT 命令等。其中，CREATE REPORT 命令主要用于创建一个报表；MODIFY REPORT 命令主要用于修改一个报表；REPORT 命令则显示或打印由 CREATE REPORT 或 MODIFY REPORT 命令所建立报表格式文件及其控件。由于 CREATE REPORT 或 MODIFY REPORT 命令的使用较为简单，这里主要描述 REPORT 命令的使用方法。

REPORT 命令的语法如下：

```
REPORT FROM FileName1 | ?
[ENVIRONMENT]
[Scope] [FOR lExpression1] [WHILE lExpression2]
[HEADING cHeadingText]
[NOCONSOLE]
[NOOPTIMIZE]
[PLAIN]
[RANGE nStartPage [, nEndPage]]
[PREVIEW [[IN] WINDOW WindowName | IN SCREEN]
[NOWAIT]]
[TO PRINTER [PROMPT] | TO FILE FileName2 [ASCII]]
[NAME ObjectName]
[SUMMARY]
```

在 REPORT 命令中，FROM 子句指定一个报表格式文件（其扩展名为.FRX）。文件名由 FileName1 指定。可以包含路径名。若不使用 FileName1 而指定参数“？”，则会在屏幕上显示一个“打开”对话框，要求用户选择打开一个报表文件。例如：

```
REPORT FROM MyReport        &&打开报表文件 MyReport
REPORT FROM ?               &&显示对话框，从对话框中选择一个报表文件
```

ENVIRONMENT 子句主要是为了兼容 FoxPro 2.X 的报表文件，它提供了一种恢复报表数据环境状态的方式。在 Visual FoxPro 里，可以使用报表定义文件保存当前 Visual FoxPro 的数据环境，保存的范围包括所有打开的表、索引文件、索引次序及表之间所建立的任何关系。为了恢复与报表相关联的数据环境，可以把数据环境属性 AutoOpenTables 设置为“真”(.T.)（缺省值）。为了确保当报表打印完毕时报表的环境被关闭，要把数据环境属性 AutoCloseTables 设置为“真”(.T.)（缺省值）。

若使用 REPORT 命令打开早期版本的报表文件，包含 ENVIRONMENT 子句则可以恢复在数据环境里的所有表和关系，而不管 AutoOpenTables 属性的设置值如何。

Scope 子句指定包含在报表的记录范围。它的取值有下列几种：

ALL（缺省值）

NEXT nRecords

RECORD nRecordNumber

REST

FOR 子句指定包含在报表里的记录必须要满足的条件。只有表达式<IExpression1>计算结果为“真”(.T.)的记录才能包含在报表里。例如：

REPORT FORM MyReport FOR Students.性别=“男”

如果逻辑表达式< IExpression1>是一个可优化的表达式，Rushmore 技术将优化 REPORT 命令。

WHILE 子句为报表打印指定一个条件，只有当逻辑表达式<Iexpression2>的计算结果的返回值为“真”(.T.)，才打印数据，直至逻辑表达式的计算结果的返回值为“假”时停止打印。

HEADING 子句为报表的每一页指定一个额外的标头。标头文本<cHeading Text>一定要用引号括起来。

NOCONSOLE 子句表示打印报表或把报表输出到一个文件时不要把报表显示在 Visual FoxPro 主窗口或用户自定义窗口里。

NOOPTIMEZE 子句指定不用 Rushmore 技术优化 REPORT 命令。尽管 FOR 子句可能包含一个可优化的表达式。

PLAIN 子句表示只在报表的第一页上显示页标头。若 REPORT 命令同时包含 HEADING 和 PLAIN 子句，则 PLAIN 子句比 HEADING 子句优先。

在打印报表时，RANGE 子句指定页号范围。其中<nStartPage>指定开始打印的页号；而<nEndPage>指定最后打印的页号。如果省略了<nEndPage>，最后的页号缺省为 9999。

PREVIEW 子句指定预览而不是打印一个报表。加上 WINDOW 子句可以指定在一个自定义窗口里预览报表。窗口<WindowName>可以由 DEFINE WINDOW 命令或一个表单创建。

在预览报表时，系统变量将被忽略。若要打印报表，必须使用 TO PRINTER 子句。

在预览报表时，若 REPORT 命令包含了 NOWAIT 子句，在运行时打开预览窗口预览报表后，系统将执行后面的程序行，而不管预览窗口是否关闭。若在一个分布式环境下使用 NOWAIT 子句，一定要确保 View 菜单是可用的，否则“打印预览”工具栏在关闭之后将无法恢复。

若要把打开的报表文件输出到打印机上，使用 TO PRINTER 子句。若同时包含 PROMPT 子句，则会在打印报表时首先显示一个“打印设置”对话框，对打印选项进行设置。PROMPT 子句必须紧跟在 TO PRINTER 子句后面。

若希望把报表文件输出到一个文件里，请使用 TO FILE 子句。<FileName2>指定输出的文件名。其扩展名缺省为.TXT。TO PRINTER 子句和 TO FILE 子句不能同时使用。

在把报表输出到指定的文件时，若包含了 ASC II 子句，报表中的 PostScript 和其他打印机代码、线条、任何图像、矩形或圆角矩形等均不包含在输出的文本文件里。若不包含 ASC II 子句，上述所列的内容将包含在输出的文件里。

在 ASC II 文本文件里每一页的行数和列数由系统变量_ASC II ROWS 和_ASC II COLS 决定。_ASC II ROWS 的默认值是 63，而_ASC II COLS 默认值是 80。

NAME 子句为报表的数据环境指定一个对象变量名。在报表的数据环境里，数据环境和对象均有自己的属性和方法。使用 NAME 子句指定对象变量名可以访问到这些属性和方法。若未指定对象变量名，Visual FoxPro 将指定一个缺省的变量名。

SUMMARY 子句使得在执行 REPORT 命令时只打印总计和分项值，而不打印各细节行的信息。

5.7 本章小结

报表是一种常用的数据输出方式，在各行各业的数据计算中经常出现。Visual FoxPro 提供了一种非常方便的设计和输出报表的工具。熟练掌握报表的设计是一项非常实用的技巧。

本章主要讲述了以下内容：

- 报表向导的使用
- 报表设计器的使用
- 报表的输出
- 创建邮件标签
- 创建报表的命令

其中报表设计器的使用是本章的重点，希望读者熟练掌握。

第六章　创建可视化交互程序

表单（Form）设计是 Windows 应用程序不可或缺的部分，它是人机对话的载体。一个好的应用程序的界面都应该具有美观、易操作等特点。Visual FoxPro 6.0 的表单设计器继承了以往 Visual FoxPro 3.0 及 Visual FoxPro 5.0 版本的各种优点：可视化操作、面向对象程序设计（OOP）、功能强大的各种向导等。Visual FoxPro 6.0 的新增功能又使其功能更强大、操作更便捷。

表单设计器是 Visual FoxPro 6.0 程序设计中相当重要的工具，它是 Visual FoxPro 6.0 作为面向对象的程序设计工具的集中体现，它通过表单控件的属性、事件和方法来完成程序的交互功能，而表单设计器提供的可视化环境使得程序设计变得直观简单。

Visual FoxPro 6.0 系统提供的各种表单设计控件与其他可视化编程工具基本类似，如果曾经学习过 Visual Basic、Delphi 等其他可视化编程工具，学习 Visual FoxPro 6.0 表单设计会相当容易。这些控件在 Windows 应用程序界面中都相当常见，用户能在日常的程序操作中深入理解这些控件的属性、事件和方法，所有这些都使读者的学习过程变得相当轻松。

6.1　使用向导设计表单

即便用户是个刚刚接触到 VFP 的新手，仍完全可以利用表单向导制作令人满意的应用程序界面。不过在此之前，用户应首先建立一个数据库文件（.dbc）或自由表（.dbf）并打开它。

（1）设计一个如图 6-1 所示的自由表 star.dbf。

Star

编号	姓名	年龄	住址	年收入
001	乔丹	34	芝加哥	36000000.0000
002	马龙	33	盐湖城	7000000.0000
003	佩顿	34	西雅图	12500000.0000
004	希尔	26	底特律	18000000.0000
005	皮蓬	33	芝加哥	5000000.0000
006	罗德曼	35	芝加哥	5000000.0000
007	尤因	31	纽约	18000000.0000
008	邓肯	22	圣安东尼奥	8000000.0000
009	奥尼尔	25	洛杉矶	22000000.0000

图 6-1　表“Star.dbf”

（2）现在启动表单向导，选择“Form Wizard（表单向导）”，单击“OK”按钮，如图 6-2 所示。

图 6-2 选择表单向导类型

(3) 选择表“ star.dbf ”,并选择希望在界面上显示的字段,如“ 姓名 ”、“ 住址 ”、“ 年收入 ”, 然后单击“ Next〉”按钮,如图 6-3 所示。

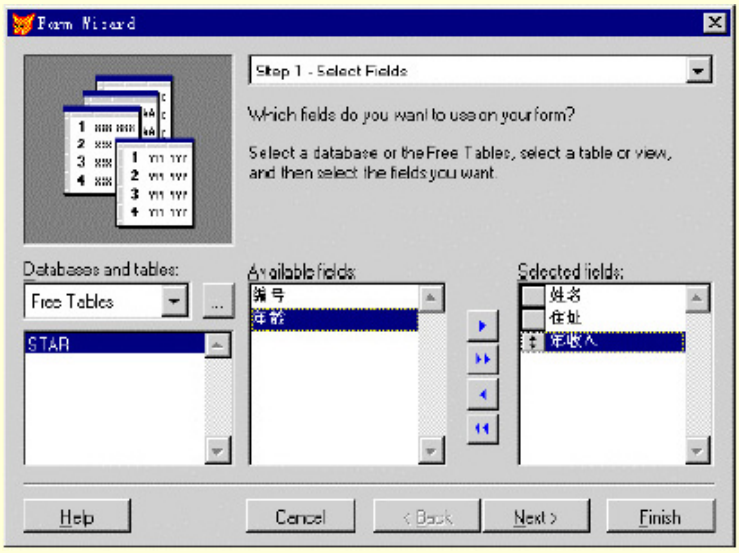


图 6-3 选择表单基于的表及显示的字段

(4) 这里让用户来选择一种屏幕风格及三种命令按钮的类型(图 6-4),屏幕左下角“ Style (样式)”对话框中选择界面风格,选择的风格类型情况直接在左上角的预览窗口得到反映,这里用户选择“ Stone (浮雕式)”风格,并选择按钮类型为“ Text Buttons (文本按钮)”,选择好后,单击“ Next ”按钮,进入下一步。

(5) Visual FoxPro 6.0 作为优秀的数据库管理系统,它能在输入纪录时对其排序,在图 6-5 中通过单击“ Add ”按钮选择 1 至 3 个字段用作排序的依据,并可选择升序或降序排列,这里用户选择“ 年收入 ”字段,并选“ Descending (降序)”后,单击“ Next ”按钮,进入下面的步骤。

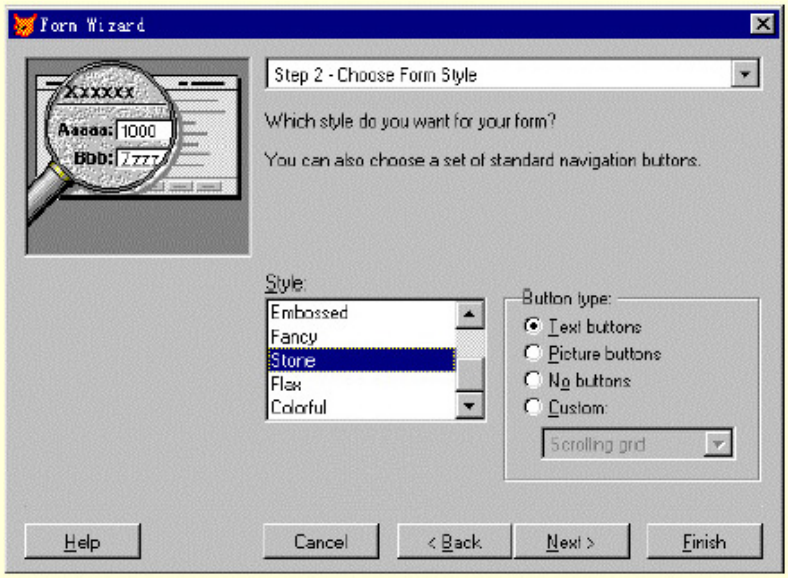


图 6-4　选择一个表单样式

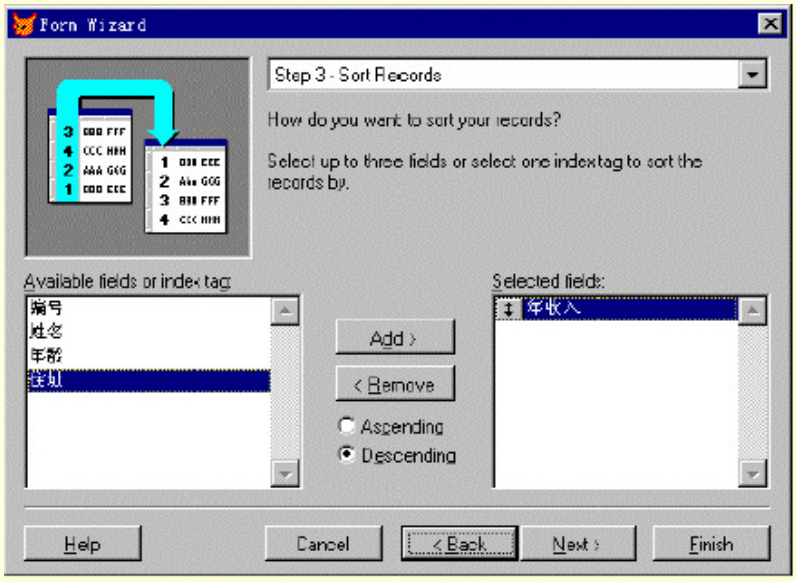


图 6-5　选择排序字段

（6）预览文件，输入要显示的表单标题为“ NBA Stars ”，单击“ Preview ”按钮，用户可以进入模拟的工作状态（图 6-6）。

（7）保存表单：在预览状态下（图 6-7），点击各按钮，观察界面内的变化。按“ Return to Wizard! ”按钮返回设计状态，如果你对预览的结果不满意，可以通过屏幕上方的下拉列表框改变到前面步骤，也可以点击“ Back ”按钮一步步实现。修改完成后，选择保存方法为“ Save form for late use（保存表单供以后使用）”，在按下“ Finish ”按钮，进 Windows 程序常用的“ Save as ”对话框，保存好文件。

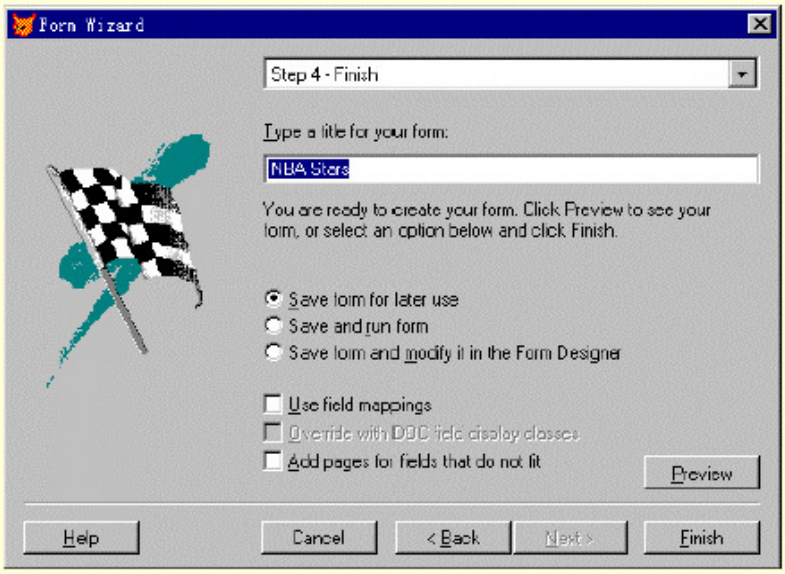


图 6-6　输入表单标题

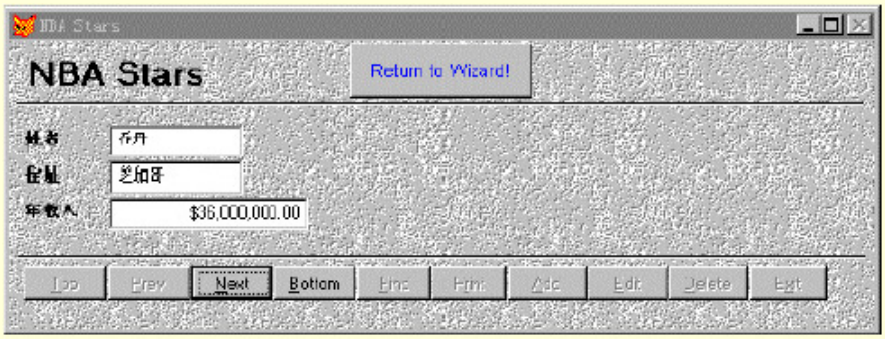


图 6-7　预览表单

6.2　各种表单控件

6.2.1　表单设计器

打开 Form Controls（**表单控件**）**工具栏，**Visual FoxPro 6.0 **主窗口中便显示出表单设计器提供的各种控件，不过，此时该工具栏的各项都是虚的，并不能使用。你仍必须先进入表单设计器，用户先来熟悉一下表单设计器（图** 6-8）：

■ **表单设计区**（Form Designer）：**通过添加各种控件建立应用程序表单，可视地添加、移动、删除、复制、粘贴控件。**

■ **属性窗口**（Properties Window）：**察看和设置表单的属性。**

■ **数据环境**（Date Environment）：**当被设计的表单用于显示一个或多个表或视图的数据时，用于保存运行表单时所需的一个或多个表、视图或表与表之间的关系。**

■ **代码窗口**（Code Window）：**为对象的事件或方法添加响应程序**

■ **表单控制工具栏**（Form Controls Toolbar）：**提供** 19 **种表单控制。**

■ **表单设计器工具条**（Form Designer Toolbar）：**表单设计所需的全部工具。可以用它来打开其他各种窗口和工具栏。**

■ **调色板**（Color Palette）：**可直观地改变选中的对象的前景和背景色。**

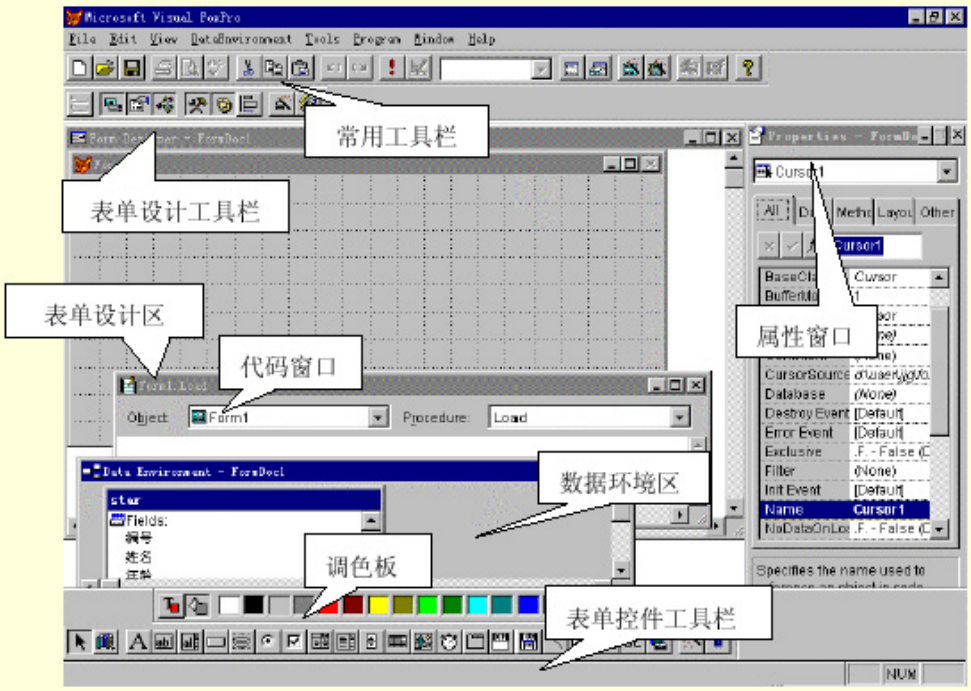


图 6-8 “表单设计器”窗口

6.2.2 表单控件

表单控件工具栏中提供了各种控制，用鼠标单击其中某一控制，使其处于被压下的状态，此时将鼠标移入表单设计区，鼠标指针已变成“+”字状，选择一位置按住鼠标并拖动，就在表单中加入了一个控件。

6.2.3 表单对象的层次

在面向对象的程序设计中，各种对象的引用是通过对象的层次实现的。

1. 由高到低逐层引用

它的顺序为：“表单集（formset）.表单（form）.容器控件（container）.控件（control）”。例如现在用户建立了一个名为 form1 的表单（Form），其中有一个命令按钮（Commandbutton）名为 cmdOk，用户想在 form1 的某个事件代码中找到这个按钮，可以键入： form1.cmdOk 。

2. 由低到高引用

通过 Visual FoxPro 6.0 提供的关键字 Parent 实现，即：“对象名（Object）.Parent”。上例中，若从按钮 cmdOk 中引用 form1 表单，则可以键入：cmdOk.parent。

在子类中调用父类时，可以使用操作符 :: 实现，为“子类 :: 父类”。

3. 直接引用

应用下面一些 Visual FoxPro 6.0 关键字：

Activeform	当前活动表单
ActiveControl	引用对象上的活动控件
This	所在对象
Thisformset	当前对象所在的表单集
Thisform	当前对象所在的表单

4. 使用包容器对象的属性引用

许多包容器对象都有引用其子对象的属性，见表 6-1。

表 6-1 包容器对象

容器对象	引用其子对象的方法
FormSet	Forms
Form	Controls
CommandGroup	Commands
OptionGroup	Options
PageFrame	Pages
其他	Controls

6.3 对象的属性、事件和方法

6.3.1 属性窗口

在表单中先加入一些控件，选中某一控件后，单击鼠标右键，弹出快捷菜单，选中“属性”项，显示出属性窗口（图 6-9），也可以直接使用“表单设计器”工具栏。

在属性窗口中，最上部的下拉列表框就是对象选择框，本图例中表单中的微调钮“Spinner1”和单选钮都是一个对象，另外，本表单也是对象，你可以通过下拉列表框来切换它们。被选中对象的属性被分为四部分：

- 数据：这些属性储存关于对象的数据环境信息和操作这些数据。
- 方法程序：显示对象的事件。
- 布局：对象位置和外观。
- 其他：在此中存储自定义属性和一些特定属性。

方法程序部分属性的设置就是编辑代码，左键双击这些属性，就进入了代码编辑框，来编辑响应事件的代码。其他属性设置语法为：

对象（object）.属性（Property）= 属性值

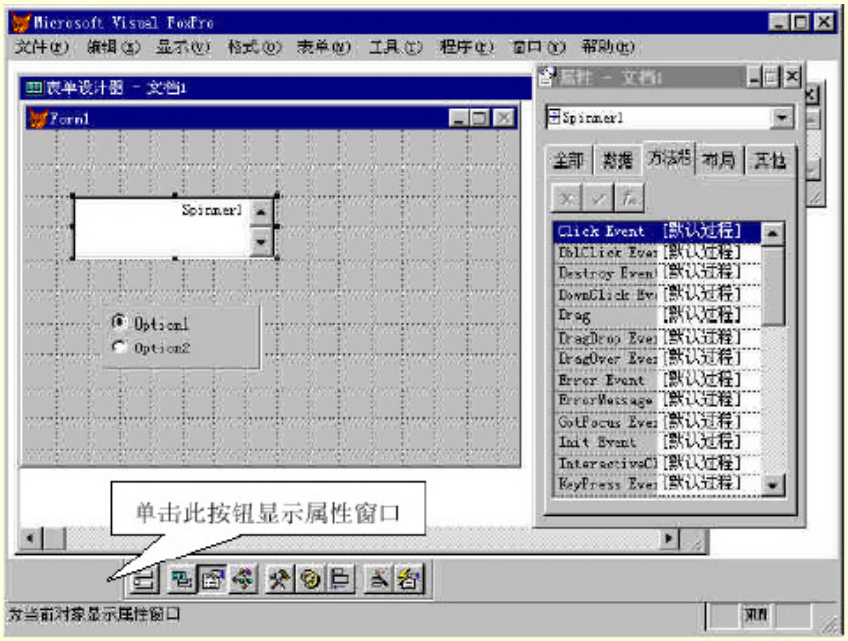


图 6-9 属性窗口

6.3.2 常用属性、事件和方法

1. 数据属性

Comment	存储关于一个对象的信息
ControlSuorce	指定对象的数据源
Tag	为应用程序存储额外的数据
Value	表示用户的当前状态

2. 事件

Click	当用鼠标单击一个对象时执行该事件
DblClick	当用鼠标双击一个对象时执行该事件
Destroy	当释放一个对象时，在释放对象前该事件发生
Dragdrop	在对象上按住鼠标并拖动后松开鼠标时发生
Dragover	当拖动一个对象到目标对象之上时发生
Error	当对象的方法有运行错误时发生
Gotfocus	激活一个对象时发生该事件，在代码中执行事件的 setfoucs 后发生
Init	在创建对象时发生该事件
Keypress	当用户按下并放开一个键时发生
Lostfocus	当对象失去焦点时发生，与 gotfocus 事件正好相反
MouseDown	在对象上按下鼠标时发生
MouseMove	在对象上移动鼠标时，该对象的该对象发生
MouseUp	单用户松开按下的鼠标键时发生
Upclick	单击一个控件的上滚箭头时发生，用在微调器、列表框、组合框中

3. 方法

AddObject（）	在运行时向容器对象种中加入一个新对象
Drag（）	开始、结束或取消一个拖动操作
Move（）	移动某一对象，与直接改变对象的 top、left 属性值的效果相同
Refresh	重画对象并更新所有的值
Release	释放内存中的表单或表单集
SaveAsClass（）	将对象的实例作为类的定义存储到类库
SetFocus	设置对象为当前焦点

4. 布局属性

AutoSize	指定是否根据对象的内容自动调整尺寸
BackColor	设置对象的背景色
BackStyle	指定对象的背景透明与否
Caption	指定在对象的标题中显示的文本
ColorSource	指定如何设置对象的颜色
DrawIcon	指定再拖放操作过程，指针显示的鼠标
Font	指定显示文本的字体特征（大小、颜色等）
ForeColor	设置对象的前景色
Height	设置对象的高度
Left	设置对象的左边缘位置
StatusBarText	指定当焦点在对象上时，显示在状态栏中的文字
Top	设置对象的上边缘位置
Visable	设置对象是否可见
Width	设置对象的宽度

5. 其他属性

BaseClass	指定基于引用对象的 VFP 基类名

Class	返回一个基于对象的类名字
Enabled	设置对象是否可用
Name	指定对象的名字
Parent	对象的容器对象
TabIndex	运行时在用 Tab 键切换时界面中对象的焦点切换顺序

6.4 添加控件到表单

表单设计通常就是分析要实现的目的，通过添加控件，编辑控件的事件响应代码来实现。下面分别讲述各种控件的设计方法。

6.4.1 添加命令按钮（CommandButton）

命令按钮通常完成特定的操作，操作的代码通常放置在命令按钮的 Click 事件中。当命令按钮获得焦点后，这个按钮比其他命令按钮多一个粗的边框，这时若用户按 ENTER 键或空格键，将执行这个命令按钮的 Click 事件。将命令按钮的 Default 属性设置为“真”(.T.)，或使用命令按钮的 Setfocus 方法可使该命令按钮获得焦点。

设计时命令按钮的重要属性有：

Click	当用户单击按钮时，执行的事件
Cancel	指定当用户按下 ESC 时，执行命令按钮的 Click 事件
Caption	在按钮上显示的文本
DisabledPicture	当按钮失效时，显示的 .BMP 文件
DownPicture	当按钮按下时，显示的 .BMP 文件
Enabled	此按钮是否有效
Picture	显示在按钮上的 .BMP 文件

通过菜单新建表单，或者在命令窗口键入“Create Form”，进入表单设计器，屏幕中出现了新建的“form1”表单，如果“表单控件”工具栏（Form Control Toolbar）不存在，则打开它。

下面往表单中加入三个命令按钮（CommandButton），其中两个开始时是隐藏的，通过单击第一个命令按钮显示它们，再利用它们来移动表单和退出表单。

（1）按下工具栏中“CommandButton”钮，逐个加入三个命令按钮，如果按下工具栏中“按钮锁定”钮，此步骤能更快完成。此时表单上显示“command1”、“command2”、“command3”三个按钮。利用鼠标移动并调整这些控件的大小，使整个表单更美观。

（2）打开属性窗口，选择“布局”选项卡，改变表单外观：

把表单变为较小的窗口状，选择“form1”对象，找到下列属性，设置为

Width=400

Height=250

Windowstate=0（普通）

使两个命令按钮控件在运行是不可见，选择“command1”、“command3”对象，设置

visible=.f.

（3）改变显示在命令按钮上的文字，分别选择“form1”、“command1”、“command2”“command3”，找到属性 caption，设置为

form1.caption=命令按钮设计实例

command1.caption=退出

command2.caption=单击此钮显示其他按钮

command3.caption=移动本表单

注意：在这里，form1，command1，command2，command3 都是对象的名字，即对象的 name 属性，只有它才代表对象，它与用户在表单设计器中对象显示的内容（caption 属性）是不同的。在以后用户引用对象时，用

户用到的都是对象的名字，即 name 属性。

（4）现在用户可以运行此表单，单击“常用工具栏”中红色“ !”钮，或者单击右键弹出快捷菜单，选择“执行表单”，系统弹出“ Save as ”对话框，选取存储文件后，用户设计的表单出现了（如图 6-10 所示）。单此时单击按钮，表单并无任何反应，这是因为用户还没有编写事件响应代码。

图 6-10 向表单中添加命令按钮

（5）添加代码：点击表单右上角关闭按钮，关闭此表单，返回表单设计器。在用户单击命令按钮后，表单将响应的是 click 事件。

在表单中，左双击 command2 按钮，弹出代码窗口，在“过程”下拉列表中选择 click，输入代码：

```
thisform.command1.visible=.t.    &&显示 command1 按钮
thisform.command3.visible=.t.    &&显示 command3 按钮
this.enabled=.f.                 &&设置自身（command2 按钮）不能用
```

同样，编写 command1 按钮的 click 代码：

```
release thisform                 &&关闭此表单
```

Command3 按钮的 click 代码为：

```
thisform.left=thisform.left+10 &&点击鼠标使表单向右移动
```

（6）为按钮设计访问键，此时表单已基本设计完毕，为在操作中更可靠更方便，为这些按钮设计访问快捷键，方法如下：

图 6-11 命令按钮设计实例

按上面所述方法，改变对象的 caption 属性设置：

command1.caption=退 出（\<X）

command2.caption=单击此钮显示其他按钮（\<S）

command3.caption=移动本表单（\<M）

至此已完成了这个表单的设计，现在运行这个表单。单击“单击此钮显示其他按钮”后，表单又出现了两

个命令按钮“移动此表单”和“退出”，而原按钮“单击此钮显示其他按钮”变成虚的，不能再被点击了（如图 6-11 所示）；点击按钮“移动此表单”后，此表单向屏幕右边移动；点击按钮“退出”在会关闭此表单。你也可以使用快捷键来实现：按住“Alt”键，在按按钮上设置的加下划线的字母键，与单击鼠标单击此按钮效果相同。

6.4.2 添加标签（Label）

标签常用来显示字符串，一般用于文本提示的设计上。在程序中可以改变标签的 Caption 和 Visible 属性，让标签更好地显示不同内容。

在设计时标签的重要属性有：

Caption	指定由标签显示的文本
AutoSize	确定是否根据标题的长度来自动调整大小标签
BackStyle	确定标签是否透明
ForeColor	显示文本的颜色
WordWrap	确定标签上显示的文本能否换行

下面是标签设计的例子，在这个表单中，设计了两个标签来显示鼠标在表单中的位置，并增加另外两个标签来作为它们的提示，最后，让这些标签都具有立体感效果。

设计步骤如下：

（1）用新建表单方式打开表单设计器，并打开“Form Controls（表单控件）”工具栏，在此工具栏中，按下“标签”钮，移至表单设计区，将一个标签“Label1”添加到表单中。并设置下列属性：

Form1.Width	400	
Form1.Height	300	
Form1.AutoCenter	.T.	&&表单运行时在屏幕上自动居中
Label1.Alignment	1	&&设置其上标题文本靠右
Label1.AutoSize	.t.	&&设置此标签大小可随其标题自动变化
Label1.fontname	黑体	&&设置显示在此标签上文字的字体为黑体
Label1.BackStyle	0	&&设置对象透明，以使各对象不互相遮盖
Label1.Fontsize	24	&&设置显示在此标签上文字的字体大小

（2）添加其他标签。选中标签“Label1”（它的四周出现 8 个控制点）后，单击常用工具栏中“Copy”钮（可单击右键，弹出快捷菜单，选择“Copy”），再移到表单设计区中，选一合适位置，再单击常用工具栏中“Paste”钮（也可用快捷菜单选择），此时表单中又出现了一个标签，它的外形与刚才设计的完全相同，其上也显示“Lable1”，你也可以查看此新对象的其他属性，可以发现，除其 Name 属性为 Label2 外，其他属性与第一个标签相同。重复此步骤，在表单中再加入“Lable3”、“Lable4”两个标签。如图 6-12 所示。

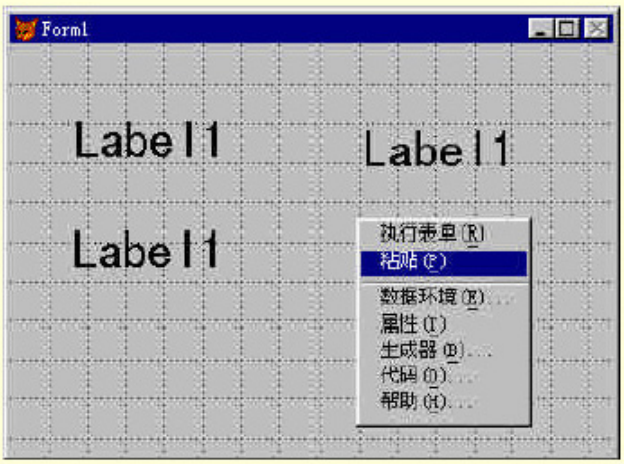


图 6-12 添加对象

（3）改变各对象的标题即 Caption 属性值。依次选中各标签，找到 Name 属性和 Caption 属性，分别改为：

label1.caption	X=
Label2.caption	100

Label3.caption Y=

Label4.caption 100

添加代码：当用户在表单上移动鼠标时，将触发表单的 mousemove **事件，此事件传递四个参数** nButton，nShift， nXCoord， nYCoord，**其中** nXCoord **返回鼠标在表单中的** x **坐标（像素数）**， nYCoord **返回鼠标在表单中的** y **坐标。**Form1.mousemove **事件的代码如下：**

```
LPARAMETERS nButton， nShift， nXCoord， nYCoord
this.label2.caption=str（nxcoord）
this.label4.caption=str（nycoord）
```

注意：对象的 caption **属性值的数据类型为字符型，而参数** nXCoord、nYCoord **返回数字型值，因此，要把它用** VFP **提供的函数** STR（）**来把它转换为字符型。**

现在运行此表单，看看结果如何，可能各对象间的位置不协调，关闭表单，返回设计状态，调整各对象位置，直到满意为止。如图 6-13 **所示。**

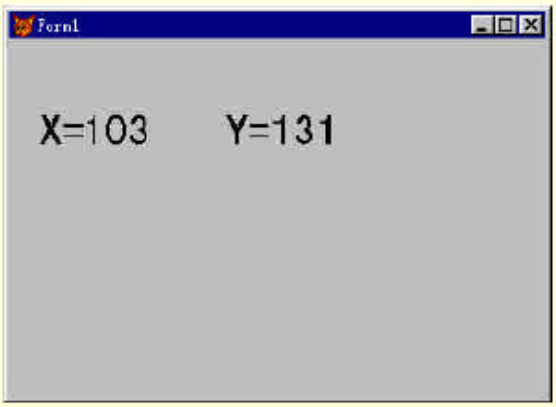


图 6-13 **设计标签**

（4**）使各标签具立体效果：回到设计状态，同时选中表单上的四个标签，可以按住** SHIFT **键，一个个点中对象，也可在表单中按住鼠标左键拖动出一个方框，使各标签都在框中，松开鼠标，则选中它们，选中的对象四周都有** 8 **个黑点。按第③步的方法复制这些对象，再在表单的另一位置粘贴这些对象。**

（5**）设计立体图案的最基本思想就是利用色彩的明暗形成阴影，使之具有立体感，因此，可以把新复制的四个对象的颜色变为白色，用户可用“** 调色板 **”工具条来完成这项操作。在“** View **”菜单中选中“** Color Palette ToolBar**（调色板）”，确定后，**Visual FoxPro 6.0 **主窗口中显示出调色板工具条。**

在调色板工具栏中，按钮  **代表前景色，与对象的** Forecolor **属性对应，按钮**  **代表背景色，与对象的** Backcolor **属性对应，在这里只须改变这四个标签的前景色，因为标签的前景色是指在运行时显示在标签上的文本的颜色，而现在已经把这些对象的背景类型即** BackStyle **属性设置为透明，再设置背景色已无任何意义。按下**  **钮，并选中这四个标签（每个标签周围都有** 8 **个黑点），然后在“** 调色板 **”工具条的右侧选择一种颜色，这里选择白色，按一下白色按钮**  **，可以看到表单中这四个标签已变为白色。**

（6**）形成阴影：选中四个白色标签，把它移至原来的四个黑色标签附近（注意不要完全重叠），可以通过键盘上的方向键来操纵，按一次方向键，这些标签移动一个象素，一边移动一边观察，直到产生较好的视觉立体感。**

别忘了，在事件 form1.click **中添加代码如下：**

```
this.label6.caption=str（nxcoord）
this.label8.caption=str（nycoord）
```

把表单的标题更改为“ 标签设计实例 **”：在属性窗口中找出对象** form1**，设置其** caption **属性：**form1.caption=**“** 标签设计实例 **”，存盘后，运行该表单。结果如图** 6-14 **所示。**

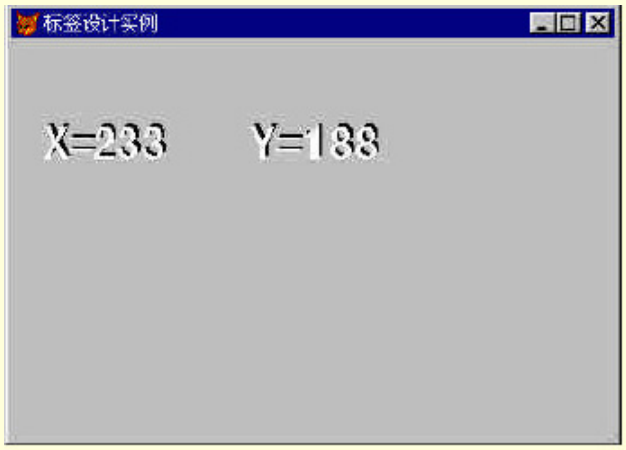


图 6-14 执行结果

6.4.3 添加文本框（Text）

文本框能接受数据的输入和输出，接受的数据可以是任何数据类型，但是，它接受的数据字数要受到文本框自身大小的限制，并且不能超过 275 个，所以，在简单的输入输出中才用到它。文本框的重要属性有：

InputMask	接受数据的输入输出格式（如 99.9 表示数据由 2 位整数、一位小数组成）
Format	接受数据的类型，如整型（N）、字符型（C）、日期型（D）
Value	文本框接受或输出的数据值
SetFocus 方法	接受焦点，执行此方法后光标移到这个文本框
InteractiveChange 事件	通过键盘输入或鼠标操作改变文本框的值时发生
ProgrammaticChange 事件	在程序中通过赋值语句改变文本框的 Value 属性时发生

文本框常用来让用户输入信息，它也可以显示信息，如在前面用表单向导设计的表单即图 6-7 中，就用了三个文本框来显示或输入信息。

在许多应用软件中，为了让不同的用户在它自己的权限范围内操作，在一个被保护的操作前，常让用户登入密码以核实权限。例如在 Microsoft Word 中用户给一个 .DOC 文档设置了“打开权限密码”，则当再试图打开这个文件时，Microsoft Word 将提示用户输入密码。（图 6-15）。

图 6-15 Microsoft Word 中的“密码”窗口

现在读者可以通过向表单中添加文本框对象来设计出这种界面，步骤如下：

通过菜单操作“File/New/Form”打开表单设计器。

（1）可视化地向表单中添加对象：添加一个标签“Label1”、一个文本框“Text1”、两个按钮“Command1”、“Command2”。基本布局如图 6-16 所示。

（2）改变对象的外观设置，设置属性：

Form1.Caption	密码录入	
Label1.Caption	请输入密码：	
Label1.Autosize	.T.	
Label1.Fontsize	18	
Text1.Fontsize	18	
Text1.Passwordchar	*	&&指定输入信息时显示在文本框中的字符

Command1.Caption 确定

Command2.Caption 取消

现在运行此表单，读者可以发现，当从键盘输入字母或数字时，显示在文本框的内容始终是“*”，如图 6-17 所示。

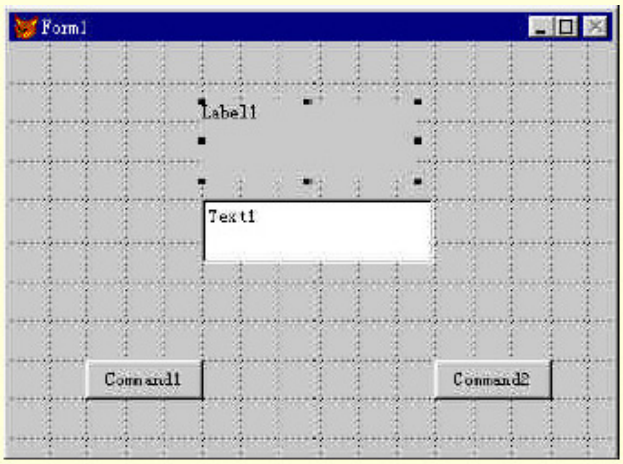


图 6-16 向表单中添加对象

图 6-17 属性 Passwordchar 指定显示的字符

（3）添加代码，使此表单实现其检验密码的功能。

在 Form1.Init 事件中编辑代码如下：

```
thisform.text1.setfocus
```

在 Command1.Click 事件中编辑代码如下：

```
IF thisform.text1.value='111'
    =messagebox（"欢迎你进入系统!"，0+64+0，"密码正确"）
    release thisform
ELSE
    =messagebox（"对不起，请重新输入!"，0+16+0，"密码错误"）
    thisform.text1.value=''
    thisform.text1.setfocus
ENDIF
```

在 Command2.Click 事件中编辑代码如下：

```
thisform.text1.value=''
thisform.text1.setfocus
```

这时再运行表单，当输入正确时，出现信息窗口，提示密码正确，确定后，退出本表单；输入错误时，也出现提示信息，让用户重新输入。如图 6-18 所示。

图 6-18 “密码录入”表单的运行情况

（4）优化表单：现在这个表单并不能令人放心，因为表单中设置的密码是一个字符型的‘111’，一个稍微高明的用户或者使用一个简单的解密程序就能破译这个密码。这里，可以把代码稍稍改变以下，再破译就不会这么容易了。

这里把密码设置成可以改变的值，它可以随着系统时钟的变化而变化， 把事件 Command1.Click 改变为：

```
IF thisform.text1.value=alltrim（str（year（date（））））
  =messagebox（"欢迎你进入系统!"，0+64+0，"密码正确"）
  release thisform
ELSE
  =messagebox（"对不起，请重新输入!"，0+16+0，"密码错误"）
  thisform.text1.value="
  thisform.text1.setfocus
ENDIF
```

注意：代码中第一行连用了四个函数：

date（ ） 调用系统日期，返回日期型值如“12/07/98”

year（ ） 调用日期型值“12/07/98”的年份数，返回数值型，即 1998

str（ ） 把数值 1998 变为字符型

Alltrim（ ） 消除字符型值中的空格（在这里 str（year（date））返回的字符值中含有空格）

现把此表单的 Closeable 设置为 .F. ，使表单右上角的关闭按钮 ✕ 不可用。这样如果你输入的密码有误，表单将不能退出。

6.4.4 添加计时器

计时器是个很有用的对象，它能周期性地执行某一动作。计时器控件与用户的操作独立。它对时间作出反应，可以让计时器以一定的间隔重复地执行某种操作。

它的重要属性有：

Interval 它指定重复执行动作的周期

Timer 事件 它指定周期性执行的动作

下面使用计时器对象来设计两种类型的时钟：

用菜单操作“文件/新建/表单”，打开一个新建的表单 Form1，然后进行下述操作：

（1）添加对象。添加一个计时器“Timer1”，两个文本框“Text1”、“Text2”和一个按钮“Command1”，并可视地调整各对象的位置。并设置属性如下。如图 6-19 所示。

Label1.Caption 现在的时间是：

Label1.Fontsize 6

Label2.Fontsize 6

Label2.Caption 从开始计时到现在的秒数是：

```
Text1.Alignment      2（中间）    &&使显示在它上面的文本居中
Text1.Fontsize       18
Text1.Readonly       .T.          &&使此文本框只读，不接受输入信息
Text2.Alignment      2（中间）    &&使显示在它上面的文本居中
Text2.Fontsize       18
Text2.Value          0            &&设置文本框 Text2 显示的初值为 0
Text2.readonly       .T.          &&使此文本框只读
Timer1.Interval      1000         &&设置事件循环的周期为 1 秒（1000 毫秒）
Command1.Caption     \<Exit       &&本按钮访问键是“ ALT+E ”
```

图 6-19　在表单中使用计时器

（2）运行它，读者可以看到，计时器对象在运行时是不可见的，因此它的各种布局属性是无关紧要的，所以在第（1）步中没有改变它的布局属性。现在再添加事件代码：

① 事件 Form1.Init 的代码：

```
this.text1.value=time（）          &&表单初始化时给出 Text1 的初始值
```

② 事件 Timer1.Timer 的代码：

```
thisform.text1.value=time（）                    &&每隔 1 秒改变它显示的值
thisform.text2.value=thisform.text2.value+1      &&每隔 1 秒其值加 1
```

③ 事件 Command1.Click 的代码：

```
release  thisform
```

（3）存盘后，再运行本表单，表单中有两种时钟，一种直接显示时间，另一种显示从开始计时算起的秒数，如图 6-20 所示。

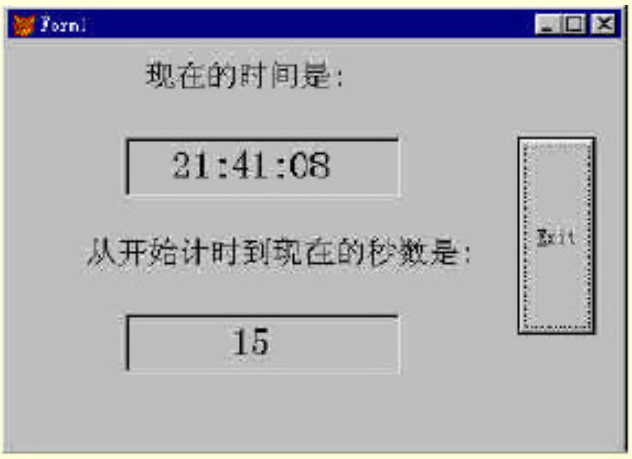


图 6-20　“ 时钟 ”表单的运行情况

注意：计时器对象的 interval 属性不要设置得太小，因为它太小时，事件重复的频率太快，这将较多地占用

CPU 资源，使系统运行速度变慢，同时也将影响周期循环的时间精度。由于系统每秒钟产生 18 次时钟跳动，所以虽然 Interval 属性是以毫秒作为计量单位，但间隔的真正精确度不会超过 1/18 秒。

6.4.5 添加编辑框（EditBox）

编辑框与文本框一样，能够接受数据的输入输出，但是编辑框只能接受字符型数据，它接受的字符数据个数不受编辑框大小的限制，它最多能接受 267483647 个字符，当字符个数多时，它可以出现水平或垂直的滚动条，能使编辑框中的文本内容滚动输出。在设计中，也常用编辑框来输入输出备注型字符段。

它有下列重要属性：

ControlSouce	此编辑框所基于的数据源
InteractiveChange 事件	与文本框的 InteractiveChange 事件相同
ProgrammaticChange 事件	

6.4.6 添加选项组（Optiongroup）与复选框（CheckBox）

选项组是包含选项按钮的容器。通常，选项按钮允许用户指定对话框中几个操作选项中的一个，而不是输入数据。例如，在安装 Visual FoxPro 6.0 时，选项按钮指定用户是典型安装、自定义安装还是便携式安装，用户只能选择其一。

在表单中创建一个选项组时，它默认地包含两个选项按钮，改变 ButtonCount 属性可以设置选项组中的选项按钮数目。

根据选项组的 Value 属性可以判断用户选定了哪个按钮。如果按钮的控件源为数值型，如果选定了第三个按钮，则选项组的 Value 属性值为 3。如果组的 ControlSource 属性是一个字符型字段，或者如果在运行表单之前将 Value 属性设置为一个字符值，则组的 Value 属性就是被选中的选项按钮的标题。选项组中的选项按钮也有自己的 Value 属性，如果没有选定选项按钮，选项按钮的 Value 属性值为 0，如果被选中，该项按钮的 Value 属性为 1。

选项组的重要属性有：

ButtonCount	选项组中的选项按钮的数目
Value	指示用户选中哪个按钮的属性值
DisabledForeColor	当废止了选项按钮时，选项按钮的前景色
DisabledBackColor	当废止了选项按钮时，选项按钮的背景色
SetAll 方法	用这个方法可以同时改变选项组中所有按钮的属性

复选框让用户指定一个问题的一个判断，用户可以使用多个复选框，与选项组不同，这些复选框的操作互不干扰。

Value 属性可以决定复选框有三种可能的状态：

0 或 .F.	复选框被选中并且可再对其操作
1 或 .T.	复选框没被选中但可再对其操作
2	复选框被选中但不能再对其操作
.NULL.	复选框没被选中且不能再对其操作

Value 属性的数据类型反映最近一次指定的数据类型。如果该属性设置为“真”(.T.)或“假”(.F.)，类型为逻辑型，直到属性重新设置为数值型值。

下面举一个选项组（Optiongroup）与复选框（CheckBox）相结合运用的例子，请读者练习一下，尽快熟悉它们的设计方法。

图 6-21 是设计完成后运行的结果，它同时利用选项组合复选框对编辑框中的输入的文本进行操作，使它显示不同的字体和字型。

图 6-21 添加选项组的表单执行情况

看起开，它很象在文字处理软件中用到的一些工具栏产生的效果，下面就来实现它：

（1）先打开一个新表单，把各种对象添加进去，它包括一个标签（Label1），一个编辑框（Edit1）、一个选项组（Optiongroup1）、三个复选框（Check1、Check2、Check3），然后设置每个对象的属性：

Label1.Caption	请输入文本：
Label1.fontsize	12
Text1.fontsize	18
Text1.fontname	宋体
Check1.caption	粗体
Check2.caption	斜体
Check3.caption	下划线
Optiongroup1.buttoncount	3
Optiongroup1.option1.caption	宋体
Optiongroup1.option2.caption	楷体
Optiongroup1. option3.caption	隶书

（2）手动调整各对象的大小和位置，还可以改变对象的颜色，使表单界面更友好，完成后，基本的界面布局如图 6-22 所示。

图 6-22 设计选项组对象

（3）编写实现功能的事件代码。

当用户刚开始执行这个表单时，需要先向编辑框 Edit1 中输入文本，因而，可在 Form1.Click 事件中添加代码：

```
This.Edit1.Setfocus          &&把焦点设置在编辑框上
```

当用户改变选项组中的选项时，要单击要选择的选项，这就触发了选项组 Optiongroup1 的 Click 事件，可编写下述代码。

```
DO case
    CASE this.value=1
      thisform.edit1.fontname="宋体"
    CASE this.value=2
      thisform.edit1.fontname="楷体_GB2312"
    CASE this.value=3
      thisform.edit1.fontname="隶书"
  ENDCASE
```

同样，分别添加下列代码：

事件 Check1.Click：

```
IF this.value=1
    thisform.edit1.fontbold=.t.
    ELSE
    thisform.edit1.fontbold=.f.
ENDIF
```

事件 Check2.Click ：

```
IF this.value=1
    thisform.edit1.fontitalic=.t.
    ELSE
    thisform.edit1.fontitalic=.f.
ENDIF
```

事件 Check1.Click：

```
IF this.value=1
    thisform.edit1.fontunderline=.t.
    ELSE
    thisform.edit1.fontunderline=.f.
ENDIF
```

表单设计完成后，结合它的运行情况，想一下选项组与复选框的各个属性的具体含义，还要特别注意它们的 Value 属性。

6.4.7 添加命令组（Command Group）

命令组是一个容器对象。在应用程序中，命令组（Command Group）的作用与命令按钮是相同的，只是在设计时，如果多个命令按钮要完成的功能类似时，用一个命令组来代替这几个命令按钮能减少应用程序的代码量，使设计过程更简单。

与命令按钮一样，在通常情况下，只使用命令组的 Click 事件，要让命令组中所有命令按钮的 Click 事件代码都用同一个方法程序过程，可将代码加入命令按钮组的 Click 事件代码中，并通过命令组的 Value 属性指明单击了哪个按钮。

如果为命令组中某个按钮的 Click 事件编写了代码，当选择这个按钮时，将执行编写的代码而不是命令组的 Click 事件代码。

下面列出了在设计时命令按钮组的重要属性。

ButtonCount	命令组中命令按钮的数目
BackStyle	命令组是否具有透明或不透明的背景。一个透明的背景与命令组下面对象颜色相同，通常是表单或页面的颜色

Value　　它的值就是用户选择的按钮在命令组中的位置

Buttons（）　　可以使用它来指定命令组中的命令按钮的对象名

Addobject（）方法　　在运行时动态地向命令组中添加按钮

Removeobject（）方法　　在运行时动态地从命令组中移走按钮

下面举一个实例，在这个实例中，建立了一个容纳 2 个命令组的表单，利用一个命令组来动态地向另一个命令组中添加和移去命令按钮。设计步骤如下：

（1）打开一个新表单，向里面添加 2 个命令组 CommandGroup1、Commandgroup2，命令组 CommandGroup1 的外观设置不用改变，将在表单的初始化事件（Init 事件）中给它的各项属性赋值。先来设置属性如下：

Form1.AutoCenter	.T.
Form1.Width	400
Form1.Height	300
CommandGroup2.Left	200
CommandGroup2.Top	100
CommandGroup2.buttoncount	3
CommandGroup2.autosize	.T.
CommandGroup2.command1.caption	添加按钮
CommandGroup2.command1.caption	移去按钮
CommandGroup2.command1.caption	退　出

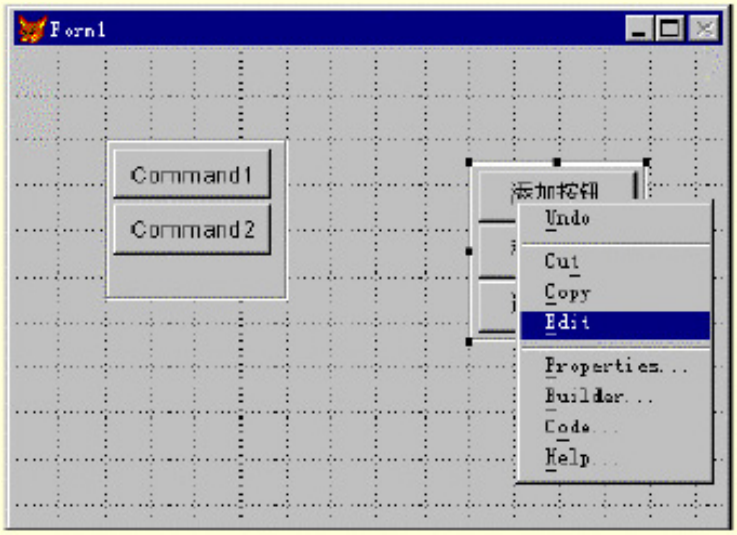


图 6-23　使用快捷菜单编辑容器控件中的对象

设置命令组中的按钮的属性时，用户可以先选中该命令组，然后单击鼠标右健，在弹出的快捷菜单中选择“Edit”（如图 6-23 所示），这时，命令组的四周出现高亮度的蓝色框，表示正在编辑这个容器控件所包含的对象，这时用户可以用鼠标选择命令组中的命令按钮。用户也可以直接在属性窗口（“Properties Window”）中的对象列表框中选择要设置的命令组中的按钮（如图 6-24 所示）。

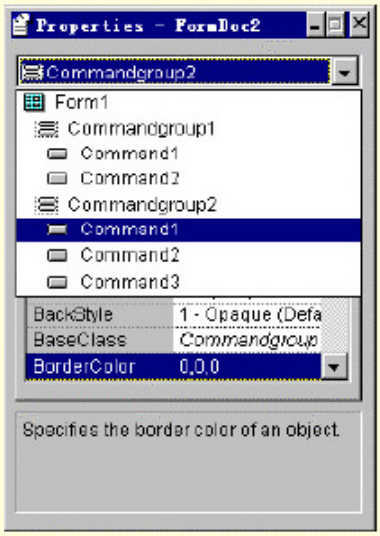


图 6-24　属性窗口中的对象列表

（2）在上述各对象的属性设置完成后，这个表单的布局如图 6-25 所示。

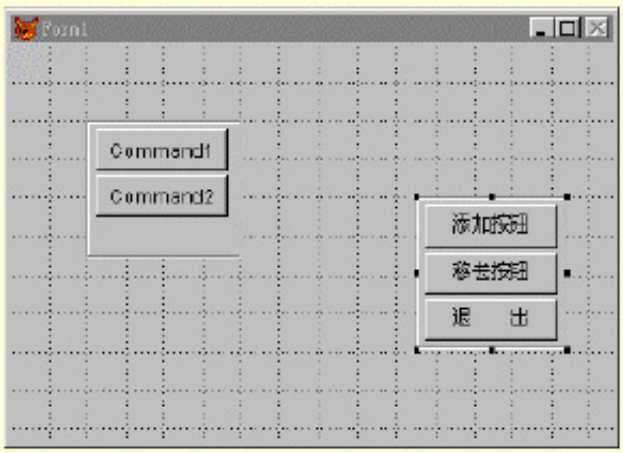


图 6-25　命令组设计实例

（3）添加代码。

先在表单 Form1 的 Init 事件中声明一些公共（Public）变量，并且给命令组 CommandGroup1 的各种布局属性赋初值。下面就是这个事件的程序代码：

```
PUBLIC i
i=1
this.commandgroup1.autosize=.t.
this.commandgroup1.buttoncount=1
this.commandgroup2.autosize=.t.
WITH this.commandgroup1.buttons（1）
    .caption="第 1 个按钮"
    .top=10
    .left=20
    .height=20
    .width=60
ENDWITH
```

代码前两行定义了一个公共变量，在一个表单中定义公共变量时常把其声明语句写在表单的加载（Load）事件或初始化（Init）事件中，因为在运行时，这两个事件是较早发生的，能保证用户在其他事件代码中引用这个变量时不致找不到它而产生错误。但对表单中对象的属性的赋值语句不能写在表单的 Load 事件中，因为此时

表单中对象的 Load 事件还没有发生，表单中对象还没有在内存中加载，执行时将找不到这些对象。

给命令组 CommandGroup2 的 Click 事件中添加代码，当用户点击这个命令组中的任一个命令按钮时，这个事件都会发生。添加代码如下：

```
DO CASE
CASE   THIS.VALUE=1
IF i=10
    WAIT WINDOWS "命令组中已经有了 10 个按钮!不能再添加了!" AT 10，50
ELSE
    i=i+1
    THISFORM.commandgroup1.ADDOBJECT（"command"+   ；&& 下一行是续句
                                ALLTRIM（STR（i）），"commandbutton"）
    WITH THISFORM.commandgroup1.BUTTONS（i）
        .LEFT=THISFORM.commandgroup1.BUTTONS（1）.LEFT
        .WIDTH=THISFORM.commandgroup1.BUTTONS（1）.WIDTH
        .TOP=THISFORM.commandgroup1.BUTTONS（i—1）.TOP+25
        .HEIGHT=20
        .CAPTION="第"+ALLTRIM（STR（i））+"个按钮"
        .VISIBLE=.T.
    ENDWITH
ENDIF
CASE THIS.VALUE=2
IF i=1
    WAIT WINDOWS "命令组中只剩下 1 个按钮!不能再减少了!" AT 10，50
ELSE
    THISFORM.commandgroup1.REMOVEOBJECT（"command"+ ;
                                ALLTRIM（STR（i）），"commandbutton"）
    THISFORM.commandgroup1.HEIGHT=THISFORM.commandgroup1.HEIGHT—25
    i=I—1
ENDIF
CASE THIS.VALUE=3
RELEASE THISFORM
ENDCASE
```

事件代码中用 Case 语句与命令组的 Value 属性结合来区别用户点击的三个不同按钮，然后视不同的情况执行不同的语句。值得注意的是命令组的 Buttons 属性，不论用户是否知道命令组中的命令按钮的名字，都可以通过命令组的 Buttons 属性来引用这些命令按钮，属性 Buttons（）需要一个参数，这个参数就是按钮在命令组中的位置。

（4）表单执行的结果如图 6-26 所示。

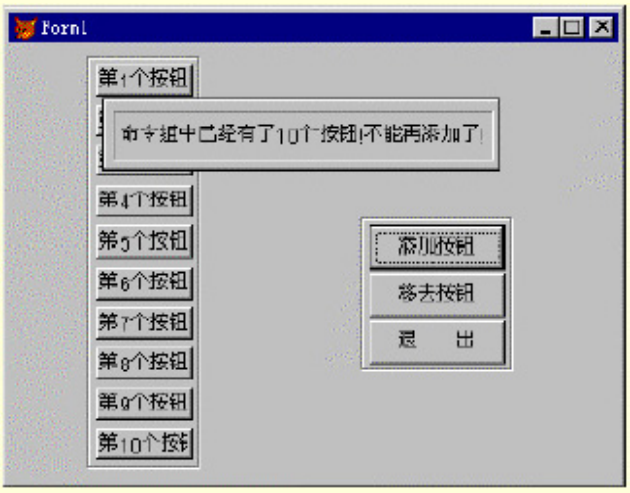


图 6-26 命令组设计实例的执行结果

6.4.8 添加线条（Line）与形状（Shape）

线条和形状对象用以在表单上添加一个线条、方框、圆或椭圆形状。它们常被用来将表单中的多个控件归成组，这将能帮助有助于用户学习和了解界面，更容易使用应用程序。线条和形状也常用来美化应用程序的界面。

它的重要属性有：

BackColor	方框或圆的边界颜色
FillColor	方框或圆中填充的颜色
FillStyle	填充线的类型
Move（）方法	把这个对象移到新位置
Curvature	指定方框四个角的弯曲程度，为 0 时是矩形，为 99 时是圆

下面这个实例帮助读者理解这个对象的基本设计方法。在这个例子中，先建立了一个演示程序，它模拟乒乓球的下落过程，然后用户可以关闭这个演示，进行手动移动操作。设计步骤见下面：

（1）添加对象，添加一个形状“ Shape1 ”、一个计时器“ Timer1 ”、一个命令按钮“ Command1 ”，并设置下列属性：

Shape1.backcolor	255，255，255	&&用调色板可以获得
Shape1.bordercolor	0，0，0	
Shape1.curvature	99	&&此对象是一个圆
Timer1.interval	100	
Command1.caption	转到手动操作	

改变后，所设计的表单布局如图 6-27 所示。

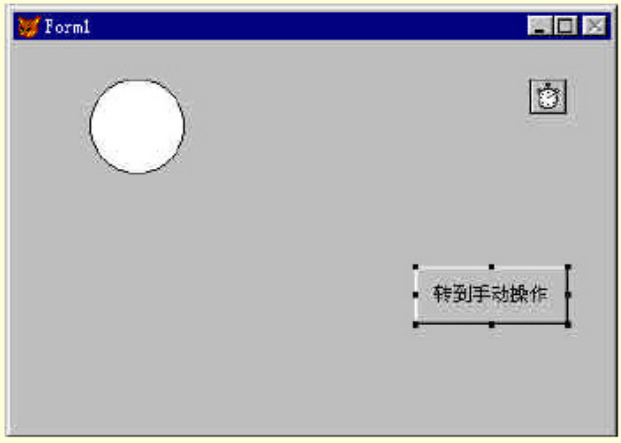


图 6-27 形状控件的设计实例

① 设计由计时器“Timer1”控制的形状“Shape1”的自动移动，给计时器“Timer1”添加 Timer 事件：

```
Y=thisform.shape1.top                    &&创建变量 Y 并给它赋值
IF y<180
    thisform.shape1.top=Y+0.1*（10+Y）  &&下移速度随位置而变，模拟物体下落时逐渐加速
ELSE
    thisform.shape1.top=0                &&下移超过一定位置后重新返回表单顶部
ENDIF
```

② 添加按钮“Command1”的 Click 事件代码如下：

```
IF this.caption='转到手动操作'
    this.caption='转到形象演示'
    thisform.timer1.enabled=.f.          &&使计时器无效，其 Timer 事件不再发生
ELSE
    this.caption='转到手动操作'
    thisform.timer1.enabled=.t.
ENDIF
```

③ 添加用键盘操作这个形状对象的代码，即在“Form1”的 Keypress 事件加入：

```
LPARAMETERS nKeyCode， nShiftAltCtrl
IF this.timer1.enabled=.f.      &&关闭演示后才能用键盘控制。
   DO case
      CASE   nKeyCode=5          &&  按下向上箭头键
         this.shape1.top=this.shape1.top－10
      CASE nKeyCode=24           &&  按下向下箭头键
         this.shape1.top=this.shape1.top+10
      CASE nKeyCode=4            &&  按下向右箭头键
          this.shape1.left=this.shape1.left+10
      CASE nKeyCode=19           &&  按下向左箭头键
          this.shape1.left=this.shape1.left－10
    ENDCASE
ENDIF
```

运行表单，开始时，上面设计的形状对象自动地从上落下，此时，按键盘上的方向键，屏幕上没有任何效果。此时单击“转到手动操作”钮，这个圆不再移动，再按方向键，这时已经可以操纵了，按钮上的提示文字变成了“转到形象演示”，圆又开始自动移动。运行时如图 6-28 所示。

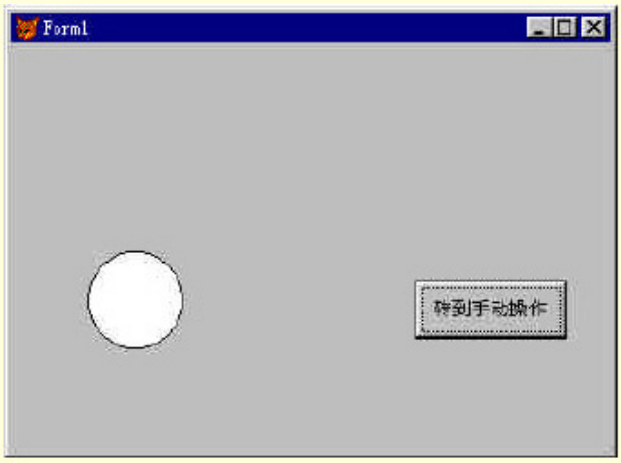


图 6-28　添加形状控件的表单运行实例

6.4.9 添加图像（Image）

图像对象用来显示一个来自文件的图片，所显示的图像由属性 picture 指定，在 Visual FoxPro 6.0 之前的版本中，它只支持 .BMP 及 .ICO 图形文件，在 Visual FoxPro 6.0 中，它还支持 .DIB、.GIF、.JPG、.ANI 等图形格式。

图像控件和其他控件一样，也有自己的属性、事件和方法程序。因此，在运行时可以动态地更改它。用户可以用各种操作交互地使用图像。

图像对象的重要属性有：

Backstyle	指定图像的背景类型，为 0 时背景透明，为 1 时背景不透明
Picture	指定显示的图形文件
Stretch	指定图像的显示方式（为 0 时按原来的大小，超出的部分不显示；为 1 时等比填充，高宽变化的比例相同；为 2 时变比填充，按对象的大小自动调整）

6.4.10 添加微调钮（Spiner）

微调钮控件通常被用在数据变化范围不大的数据输入中，使用微调钮可以让用户通过“微调”来改变数据，或直接在微调框中键入值。

微调控件值一般为数值型，也可以使用微调控件和文本框相结合来微调多种类型的数值。

微调控件的重要属性有：

Incerment	用户每次单击向上或向下按钮时增加和减少的数值
KeyboardHighValue	用户能键入到微调文本框中的最高值
KeyboardLowValue	用户能键入到微调文本框中的最低值
SpinnerHighValue	用户单击向上按钮时，微调控件能显示的最高值
SpinnerLowValue	用户单击向上按钮时，微调控件能显示的最低值
Upclick 事件	用户单击向上按钮时，响应的事件
Downclick 事件	用户单击向下按钮时，响应的事件

下面的实例将用到上述的全部属性，在实例表单中，用两个微调器来分别控制一个图像对象的宽度和高度，使图片放大或缩小，并且还添加了一个选项组来改变图片放大或缩小的方式（是否等比）。这个表单的设计过程如下：

（1）添加对象：

打开表单设计器，向新表单中添加两个微调器（“Spinner1”、“Spinner2”）、一个图像（“Image1”）、一个选项组（“Optiongroup1”）和两个标签（“Label1”“Label2”），并设置属性：

Label1.caption	图片宽度
Label2.caption	图片高度
Spinner1.fontsize	6
Spinner1.value	100
Spinner1.Increment	10
Spinner1.KeybordHighValue	180
Spinner1. KeybordLowValue	10
Spinner1.Alignment	2（Center）
Spinner2.fontsize	6
Spinner2.value	100
Spinner2.Increment	10
Spinner2.KeybordHighValue	180
Spinner2. KeybordLowValue	10
Spinner2.Alignment	2（Center）
Spinner2.enabled	.F.

Image1.picture	c：\program files\vfp6\fox.bmp
Image1.Stretch	2（Stretch）
Image1.Width	100
Image1.height	100
OptionGroup1.bottoncount	2
OptionGroup1.option1.Caption	等比变化
OptionGroup1.option2.Caption	任意变化

这个表单设计时的基本情况如图 6-29 所示。

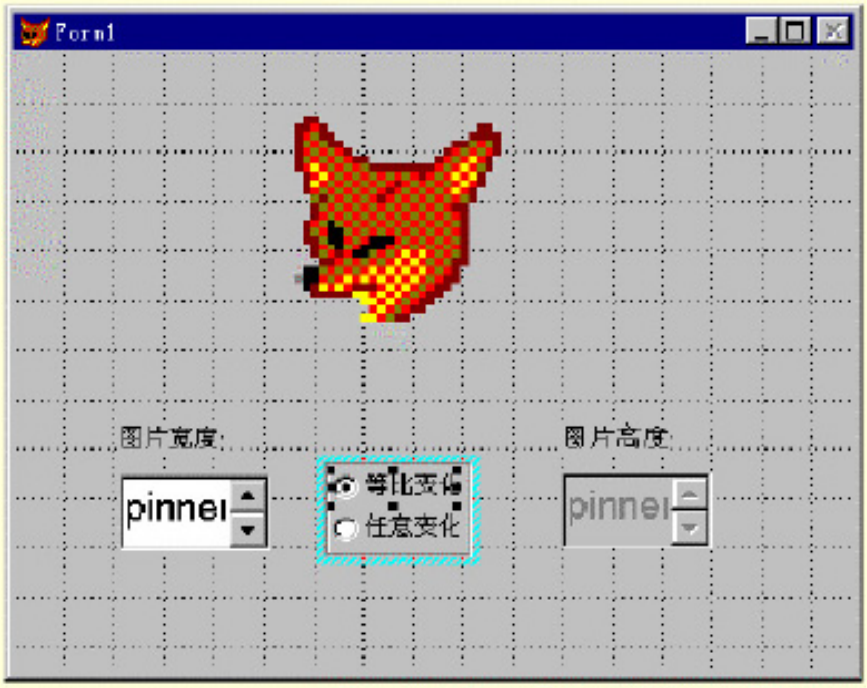


图 6-29 微调钮设计实例

（2）分析程序过程、添加代码：

选项组 OptionGroup1 是用来控制微调器和图像的，对它的操作与其他对象的状态无关，可先设计它的 Click 事件代码如下：

```
IF THIS.VALUE=1
    THISFORM.spinner2.ENABLED=.F.
    THISFORM.spinner2.VALUE=THISFORM.spinner1.VALUE
    THISFORM.image1.HEIGHT=THISFORM.spinner1.VALUE
ENDIF
IF THIS.VALUE=2
    THISFORM.spinner2.ENABLED=.T.
ENDIF
```

在编写代码后，为便于阅读和修改代码，可以使用 Visual FoxPro 6.0 提供的美化工具“ Beautify ”来美化程序代码，用户在代码窗口中单击鼠标右键，可以弹出如图 6-30 所示的快捷菜单，选择“ Beautify ”，进入“ Beautify Option（美化工具选项）”，如图 6-31 所示。

框架“ Capitalization ”是代码中字母大写的选项，其中“ KeyWards ”列表框让用户选择 Visual FoxPro 6.0 关键字大小写的格式，“ Symbols ”列表框提供其他符号大小写的格式。框架“ Indentation ”是代码行的缩进选项。

设置好这些些选项后，单击“ Run ”按钮，代码窗口中的代码就被美化了，可以明显看到，程序代码的可读性大大增强了。

图 6-30　代码窗口中的快捷菜单

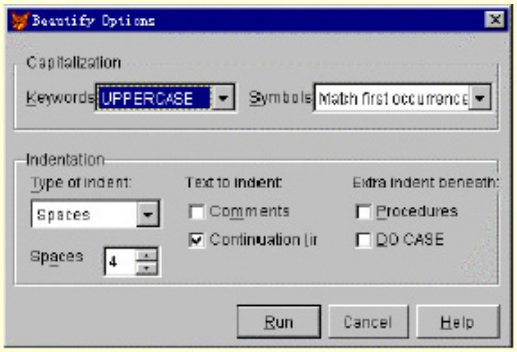


图 6-31　美化代码工具选项窗口

（3）同样，在事件 Spinner1.DownClick **中加入代码，用于减小图像对象的宽度，当用户选择选项组中的第一个选项按钮（“等比变化”）时，也同时减小图像对象的高度。**

```
IF THIS.VALUE=10
    WAIT WINDOWS "图像不能再被缩小" AT 10，40 TIMEOUT 1
ELSE
    IF THISFORM.optiongroup1.VALUE=1
      THISFORM.image1.WIDTH=THIS.VALUE
       THISFORM.image1.HEIGHT=THIS.VALUE
       THISFORM.spinner2.VALUE=THIS.VALUE
    ELSE
       THISFORM.image1.WIDTH=THIS.VALUE
    ENDIF
ENDIF
```

（4）同样添加代码到事件 Spinner1.UpClick **中，用于增大图像对象的宽度，当用户选择选项组中的第一个选项按钮（“等比变化”）时，也同时增大图像对象的高度。**

```
IF THIS.VALUE=180
    WAIT WINDOWS "图像不能再被放大" AT 10，40 TIMEOUT 1
  ELSE
    IF THISFORM.optiongroup1.VALUE=1
      THISFORM.image1.WIDTH=THIS.VALUE
      THISFORM.image1.HEIGHT=THIS.VALUE
      THISFORM.spinner2.VALUE=THIS.VALUE
    ELSE
      THISFORM.image1.WIDTH=THIS.VALUE
```

```
        ENDIF
    ENDIF
```

（5）再编写对象 Spinner2 的事件代码，用于改变图像对象的高度。

UpClick 事件：

```
    IF THIS.VALUE=180
        WAIT WINDOWS "图像不能再被放大" AT 10，40 TIMEOUT 1
    ELSE
        THISFORM.image1.HEIGHT=THIS.VALUE
    ENDIF
```

DownClick 事件：

```
    IF THIS.VALUE=10
        WAIT WINDOWS "图像不能再被缩小" AT 10，40 TIMEOUT 1
    ELSE
        THISFORM.image1.HEIGHT=THIS.VALUE
    ENDIF
```

这里，Spinner2 的事件代码与 Spinner1 的事件代码比较相似，并且，在前面的属性设置中，这两个对象的各种属性也是近似的，可以在设计是直接先把对象 Spinner1 的属性和事件代码编好后，复制 Spinner1 对象后再粘贴为 Spinner2 对象，然后修改 Spinner2 的事件代码。这样可以大大减少设计时的工作量。

（6）运行本表单，结果如图 6-32 所示。

图 6-32　微调钮设计实例的运行结果

6.4.11　添加列表框和下拉列表框

列表框和下拉列表框可以提供一系列的选项供用户选择。在列表框中，任何时候都能看到多个项，而在下拉列表中，只能看到一个项，用户可单击向下按钮来显示可滚动的下拉列表框。

下面是设计时重要的列表框属性：

ColumnCount	列表框的列数
ControlSource	用户从列表中选择的值的数据源
MoverBars	是否在列表项左侧显示移动钮栏，有助于用户更方便地重新安排列表中各项的顺序
Multiselect	用户能否从列表中一次选择一个以上的项
RowSource	列表中显示的值的来源
RowSourceType	确定 RowSource 是下列哪种类型：一个值、表、SQL 语句、查询、数组、文件列表或字段列表
Value	列表框的值，可以是数值型，也可以是字符型，默认值为数值型
Requery 方法	当 RowSource 中的值改变时更新列表

在这一节中，要重点讲一下如何用生成器（Buider）来设计列表框和下拉列表框。在 Visual FoxPro 6.0 中，表单（Form）、文本框（Textbox）、编辑框（Editbox）、选项组（OptionGroup）、命令组（CommandGroup）、列表框（ListBox）、组合框（ComboBox）和网格（Grid）都有它自己的生成器，其他几种对象的生成器的使用方法与列表框相似。

（1）打开一个新表单，在表单设计器的“表单设计区”中添加一个列表框，选中这个列表框，使它周围出现 8 个小黑点后，单击鼠标右键，在弹出的快捷菜单（图 6-33）中选择“Builder”，出现“列表框生成器”页框，如图 6-34 所示。

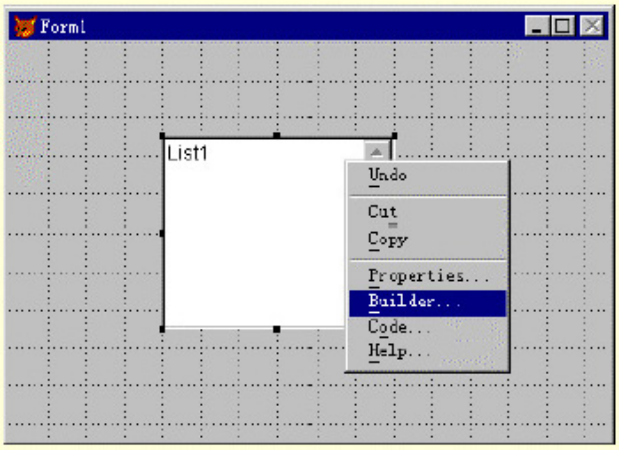


图 6-33　使用列表框生成器

（2）在“列表框生成器”页框（如图 6-34 所示）中，含有四个页面：列表项（List Items）、样式（Style）、布局（Layout）、值（Value）。

① 列表项（List Items）：如图 6-35 所示。填充列表（“Fill the list with”）项供用户选择怎样在列表框中填入数据，它有三个选项：表或视图中的字段（Fields form a table or view）、手工输入数据（Date entered by hand）和数组中的值（Values form an array），在本例中，选择“Fields form a table or view”。“Datebases and tables”项让用户选择列表框基于的数据库或自由表，这里仍选择本章第一节建立的自由表“Star.dbf”，选择后，“Available fields”框中出现可供选择的字段名，单击 或其他几个钮，选择用户希望显示在列表框中的字段，选择“姓名”“年龄”“住址”三个字段。

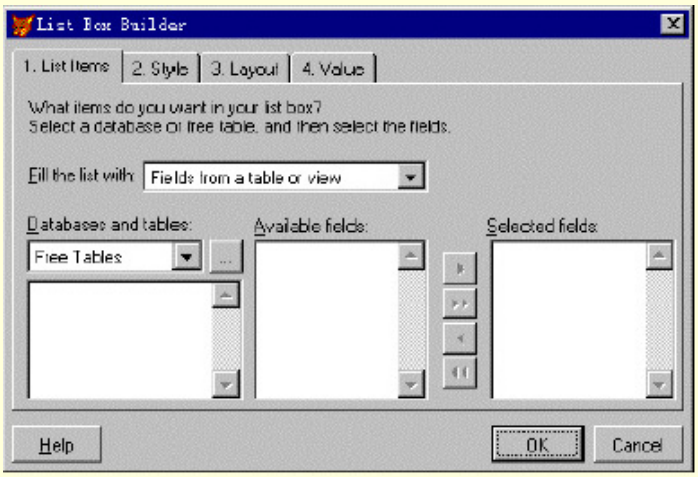


图 6-34　列表框生成器页框

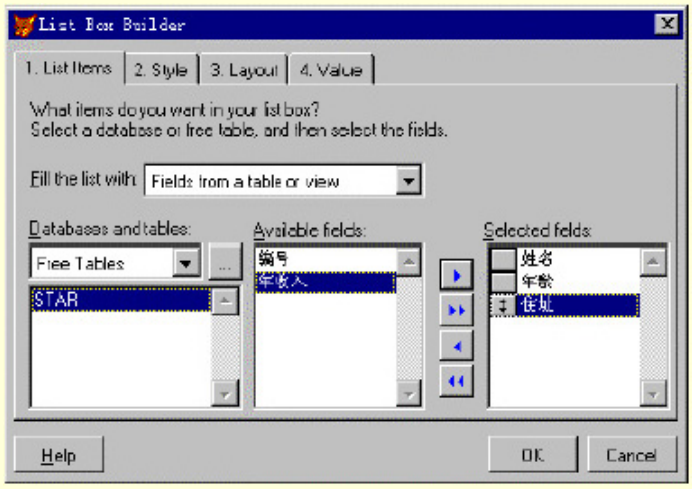


图 6-35　列表框生成器中的列表项对话框

② 上一步完成后，按一下页框上部的页面“2. Style”，就进入了样式对话框。在这个对话框中，有两个选项组和一个微调器。第一个选项组让用户选择所设计的列表框是三维的（Three-dimensional）还是平面（Plain）的。第二个选项组让用户选择在列表框中是否允许递增搜寻。微调钮“Rows to display”表示列表框输出的行数。在本例中仍保持它们的默认值。如图 6-36 所示。

③ 在第三个页面“Layout”中，主要调整列表框输出的布局信息，用户可以选择让系统自动调整大小以显示所有列，也可以手动调整各列的宽度。本例中，手动调整各列宽度，使输出更整齐。如图 6-37 所示。

④ 最后是值（Value）对话框，在这个页面中，它含有两个下拉组合框，在第一个组合框中，生成器询问运行时当用户选择列表框中的某一行时，这个列表框返回的值，让用户在数据表的字段中选择一个。在第二个下拉组合框中，让用户选择返回值的保存位置。实例中，返回值选择字段“姓名”，保存位置项不选择。如图 6-38 所示。

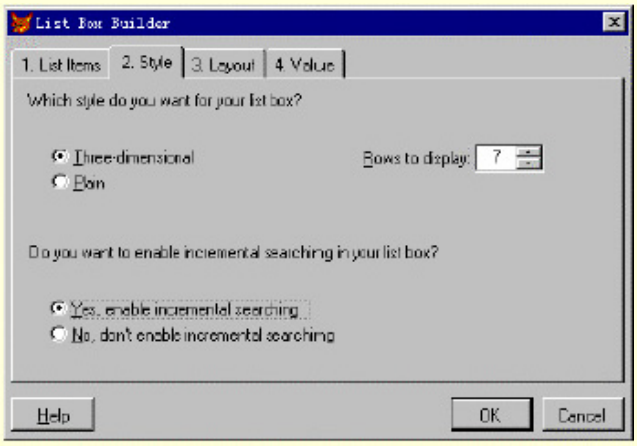


图 6-36　列表框生成器中的样式对话框

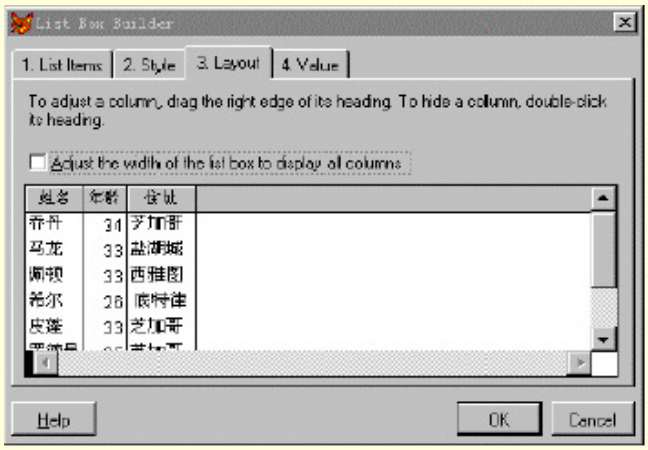


图 6-37　列表框生成器中的布局对话框

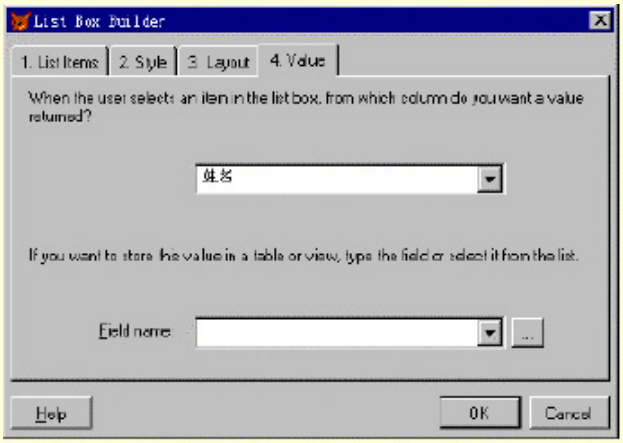


图 6-38 列表框生成器中的值（Value）对话框

⑤ 列表框生成器各项选择完成后按“ OK ”按钮，回到表单设计状态。读者可以运行这个表单，欣赏一下设计的列表框。如图 6-39 所示。

图 6-39 列表框设计实例的执行结果

6.4.12 添加组合框

组合框是列表框和文本框的结合，它可以让用户直接键入数据，或在列表框中选择一个选项作为输入。

组合框的重要属性：

ControlSource	指定用于保存用户选择或输入值的表字段
InputMask	对于下拉组合框，指定允许键入的数值类型
IncrementalSearch	指定在用户键入每个字母时，控件是否和列表中的项匹配
RowSource	指定组合框中项的来源
RowSourceType	指定组合框中数据源类型。与列表框的这个属性相同

仍举一个组合框设计的实例来加深对组合框各种属性的使用方法。在下面的实例中，在上节设计的下拉列表框的基础上，在添加一个下拉组合框来实现对表中记录的控制，并通过下拉列表框的变化反应出来。步骤见下：

（1）先打开自由表 STAR.DBF，在上面设计的基础上，添加一些新对象：五个标签和一个组合框，标签的 caption 属性分别为“ 请选择球员姓名 ”、“ 他的基本信息 ”、“ 姓名 ”、“ 年龄 ”、“ 城市 ”，用来在运行时提示用户进行操作，在改变组合框“ Combo1 ”、列表框“ List1 ”的属性：

Combo1.Fontsize	18
Combo1.DisplayValue	1
Combo1.RowsourceType	6（Fields）

Combo1.Rowsource	star.姓名
List1.Fontsize	18
List1.Height	36

设置后，这个表单的基本布局如图 6-40 所示。

图 6-40 组合框设计实例

（2）如果不加入事件代码，在用户在下拉组合框中选择不同的姓名是，在列表框中并没有任何变化，要达到设计目的，可在组合框“ Comobo1 ”的 Click 事件中输入下列代码：

```
THISFORM.REFRESH
THISFORM.list1.VALUE=THIS.VALUE
THISFORM.REFRESH
```

代码中的 refresh 语句用来刷新表单屏幕，其目的是为了保证当列表框的数据源变化时能在表单上显示出来。然后运行此表单，当用户在下拉组合框中选取不同的姓名时，表的记录指针随之发生相应变化。然后下拉列表框显示出相应的信息，运行时的情况如图 6-41 所示。

图 6-41 组合框设计实例的运行结果

6.4.13 添加页框

页框在 Windows 中是一种常见的控制方式，它提供了多个不同内容的页，通过选择不同的页面，切换到不

同的页，可以获得不同的页内容，因此，页框大大扩充了可利用的界面空间。

图 6-42 所示是 Windows95（98/NT）中设置任务栏的页框，它包括“任务栏选项”、“开始菜单程序”两个页面，虽然它在屏幕上占有的空间不变，但它能同时处理两项不同的任务。

图 6-42 Windows 任务栏属性页框

在设计时，选项卡是包含页面的容器对象，页面又可包含控件。可以在页框、页面或控件级上设置属性。

常用的重要属性：

Tabs	确定页面的选项卡是否可见
TabStyle	是否选项卡都是相同的大小，并且都与页框的宽度相同
PageCount	页框的页面数
TabStretch	指定页突是单行或多行显示
ActivePage	指定当前活动的页面
Pages（）	该数组是页框所包含页面的对象数组，使用它能访问这些页面

当用户要在页框的一个页面中添加控件时，要先使这个页框处于编辑（Edit）状态，在编辑状态下，用户可以象在运行时切换各个页面一样通过点击页突来编辑各个页面。用户可以通过快捷菜单中的“Edit”来实现，也可以在属性窗口中的对象列表框中直接选择，如图 6-43 所示。

一个表单不要添加多个页框，若需要实现的功能较多时，可增加页框的页面数（PageCounts 属性），表单中页框尺寸也不要太小，它应占有表单的大部分空间，而把多个功能分别放到各个页面中来完成，这样才能充分利用页框对象的优点，提高表单的界面使用效率。

每个页面的设计与一个表单的设计方法完全相同，在这里不再赘述。

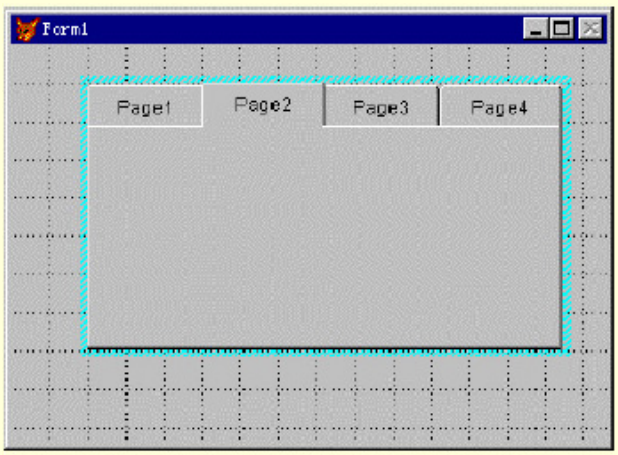


图 6-43 页框在编辑状态下可以设计它的各个页面

6.4.14 添加表格（Grid）

表格对象与本书前文表操作中的“Browse”窗口类似，它能在表单或页面中显示并操作行和列中的数据。表格也是一个容器对象，和表单集包含表单一样，表格也能包含列。这些列除了包含标头和控件外，每一个列还拥有自己的一组属性、事件和方法程序，从而提供了对表格单元的大量控件。

表格对象可以用最小的编程量将数据库或自由表中的数据加入表单中。

表格对象的重要属性有：

RecordsourceType　　表格数据源的类型，可以是表、别名、查询等

Recordsource　　表格的数据源

下面是一个例子，它可以完成“Browse”窗口的操作。这个表单使用一个表格对象来浏览自由表（.dbf 文件），使用一个命令按钮来让用户选择自由表的名字。设计步骤如下：

（1）打开一个新表单，添加一个表格对象“Grid1”，一个命令按钮“Command1”，把它们的属性设置为：

Grid1.caption　　点击此处选择要浏览的表

Grid1.autosize　　.T.

Command1. RecordsouceType　　0（表）

（2）现在，这个表格的数据源是一个表或数据库，但并没有与任何具体的数据表相联系，只需使用 Visual FoxPro 6.0 内部函数 GETFILE（）就可完成，在 Command1.Click 事件中编写代码：

```
USE                          &&若打开了其他的表，则先关闭
myfile=GETFILE（"dbf" ）      &&读取要显示的表文件
THISFORM.grid1.RECORDSOURCE=myfile
```

（3）运行表单时，单击按钮后，弹出“打开文件”对话框（图 6-44）。在文件列表中选择一个文件名，例如“Foxuser.dbf”，后按“确定”，返回表单“Form1”，表单中的表格显示了这个表的浏览情况，如图 6-45 所示。

图 6-44　由 Getfile（ ）弹出的打开文件对话框

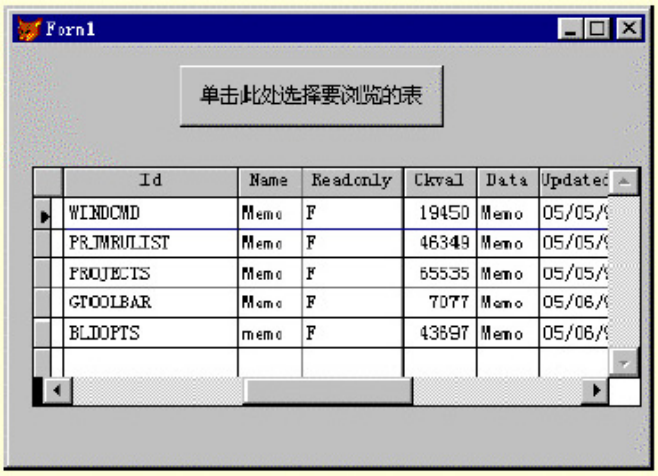


图 6-45　表格设计实例的运行结果

6.4.15　添加 ActiveX 容器控件

ActiveX 容器控件在 Visual FoxPro 3.0 或 Visual FoxPro 5.0 中被称作 OLE 容器控件，用户点击“ ActiveX 容器控件 ”按钮，并在表单设计区中将其拖至所期望的大小，将弹出如图 6-46 所示的“ Inser Object（插入对象）”对话框，选择可以将 ActiveX（OLE）对象添加到表单。

用户可以添加诸如“ 画笔图片 ”、“ 媒体剪辑 ”或“ Microsoft Excel ”等服务对象。另外，如果 Windows 的 SYSTEM 目录下包含 ActiveX 控件（带有 .OCX 扩展名的文件），这些文件是封装好的 ActiveX 控件，也可以添加入表单。

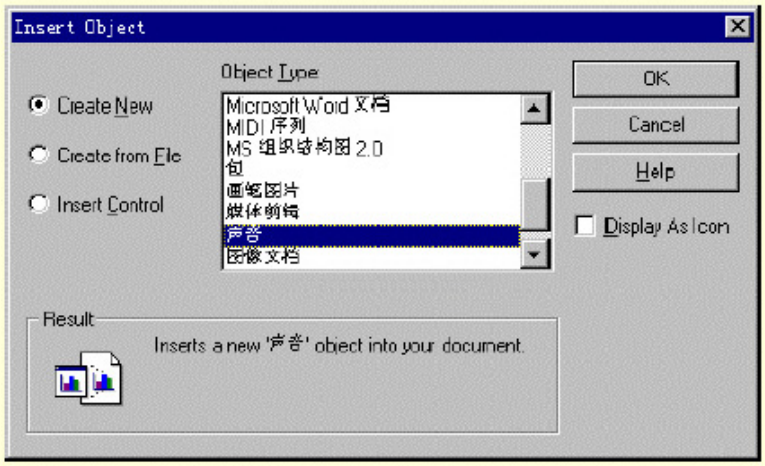


图 6-46　选择 ActiveX 控件的对象类型

关于 ActiveX 控件的使用方法，本书将在第八章中详细介绍，在这里，只是作为一个简单的例子讲一下如何向表单中添加一个声音对象。步骤如下：

（1）打开一个新表单，点击“ Form Controls（表单控件）”工具栏中的“ OLE 容器控件 ”按钮，并在表单设计区中将其拖至所期望的大小后松开鼠标，在弹出如图 6-46 所示的“ 插入对象 ”对话框，在左边的选项组中选择“ Create New ”，然后在对象类型（Object Type）列表中选择“ 声音 ”，按“ OK ”按钮后返回表单设计器。

（2）系统将弹出“ 音频对象 ”对话框，如图 6-47 所示。这与在 Windows 平台上新建一个声音文件时完全相同，通过菜单“ 编辑/插入文件 ”向这个 ActiveX 对象插入声音文件，在新出现的“ 插入文件 ”对话框中选择用户想要插入的文件，例如 Windows 98 的启动声音“ C：\Win98\Media\The Microsoft Sound ”，返回表单设计器。

图 6-47 在声音类型的 ActiveX 控件中插入文件

（3）表单上这个对象的位置上出现了一个喇叭形状的图标，查看属性窗口，对象的名字是“ Olecontrol1 ”，可以通过它的属性、事件、方法来操纵它，现在就添加一个命令按钮来使用户在运行表单时播放“ Olecontrol1 ”中的声音。命令按钮的名字是“ Command1 ”，标题即 Caption 属性为“ 播放声音 ”，在这个命令按钮的 Click 事件中调用“ Olecontrol1 ”的 Doverb 方法，“ Command1 ”的 Click 事件代码为：

```
THISFORM.olecontrol1.DOVERB（0）
```

方法 Doverb（0）就是执行对象“ Olecontrol1 ”的默认动作。

（4）执行这个表单，单击“ 播放声音 ”按钮后看看有什么结果 ，执行时如图 6-48 所示。

图 6-48 添加一个声音对象的表单实例

对 OLE 控件的操纵是通过其 DOVERB（）方法实现的，DOVERB（）方法接受的参数来决定该控件的动作，这个参数的值可以在 Windows 中的“ 注册表编辑器 ”来查看。“ 注册表编辑器 ”的文件名为“ Regedit.exe ”，它存放在“ ..\Windows\ ”目录下，用户可以直接运行这个程序。下面简要说明一下怎样使用“ 注册表编辑器 ”查看 OLE 控件的命令动作对应的值。

1. 打开“ 注册表编辑器 ”

在 Windows 资源管理器中找到文件“ Regedit.exe ”,或在开始菜单中选择“ 运行 ”,并键入文件名“ Regedit.exe ”，打开“ 注册表编辑器 ”窗口，如图 6-49 所示。

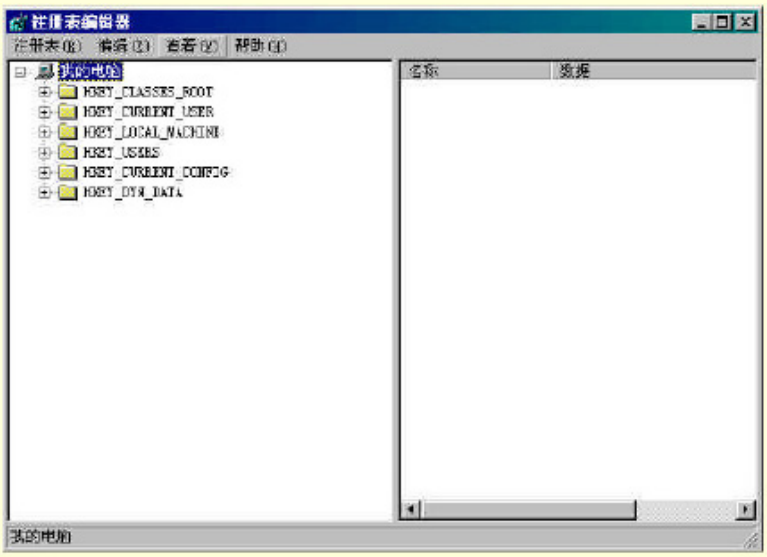


图 6-49 “注册表编辑器”窗口

2. 查找所用的 OLE 对象在注册表中的数据

选择“编辑”菜单中的“查找”项，在弹出的“查找”对话框中键入 OLE 对象基于的文件类型的后缀名，如一个声音型的 OLE 对象就输入“.WAV”，在“查看”项中只选择“主键”，并选中“只匹配整个字符串”，如图 6-50 所示，单击“查找下一个”按钮。

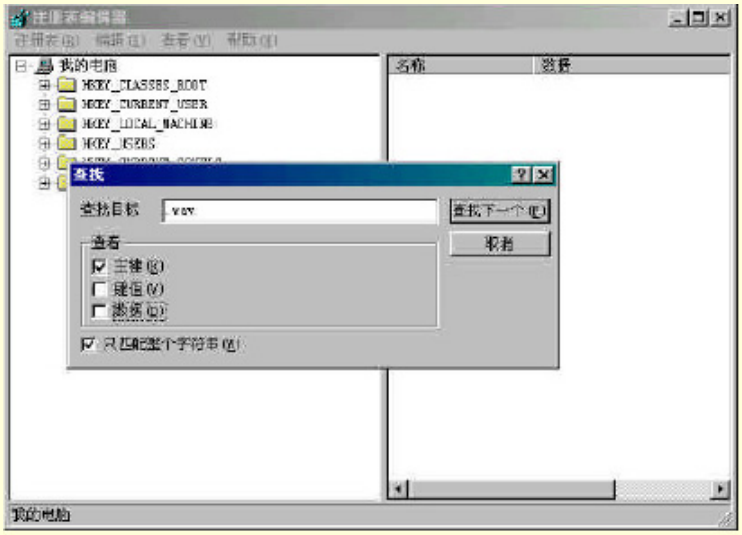


图 6-50 查找 OLE 对象在注册表中的数据

3. 查找 OLE 对象主题

上面的查找结束后，“注册表编辑器”屏幕左侧移动到“.WAV”项，屏幕右侧的“数据”列表下显示了 OLE 对象在注册表中的数据，如图 6-51 所示，图中“数据”列表下的“SoundRec”和“Audio/wav”就是查到的数据。

再次选择“编辑”菜单下的“查找…”，与第 2 步基本相同，键入“数据”列表下查出的 OLE 对象的数据（如“SoundRec”或“Audio/wav”），如图 6-51 所示，之后单击“查找下一个”按钮。

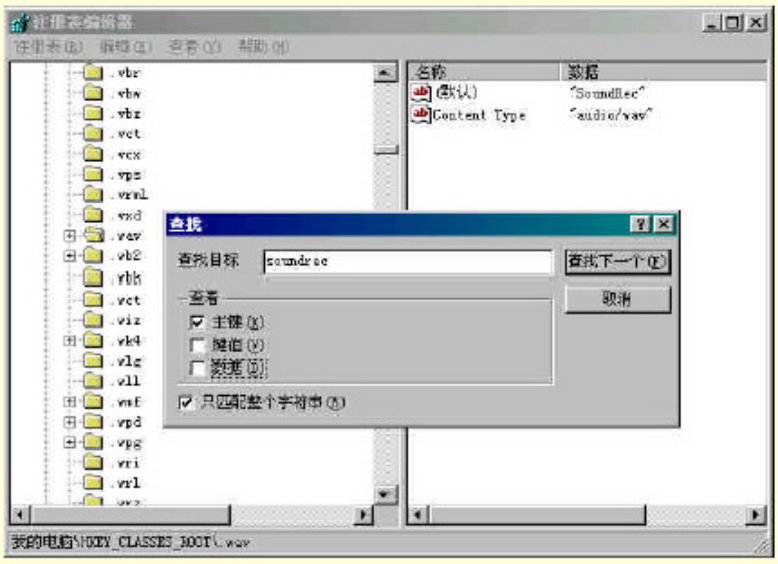


图 6-51 查找 OLE 对象主题

4. 查出命令动作对应的值

在查出“SoundRec”项后（这一过程可能花费几分钟时间，要耐心等待），如图 6-52 所示，展开这一项，这个操作与“资源管理器”中的操作完全相同，找到“Verb”项及其子项，“Verb”项的子项是用数字表示的，这些数字就是 OLE 控件 Doverb（）方法中的参数，分别选中各个子项，屏幕右侧的“数据”列表中就会出现这个的参数对应的动作。有时不太容易直接找到“Verb”项，仍用“查找”对话框来查找，在对话框中键入“Verb”就可以了。

最后的结果是 Doverb（0）表示播放声音，这是缺省项，当 Doverb（）中不带任何参数时执行此项；Doverb（1）表示编辑声音。

通常情况下，Doerb（）方法能接受的参数都较少，只能实现很简单的操作，要实现其他复杂一些的操作，需要调用 Windows API 函数，本书在后面将有阐述。

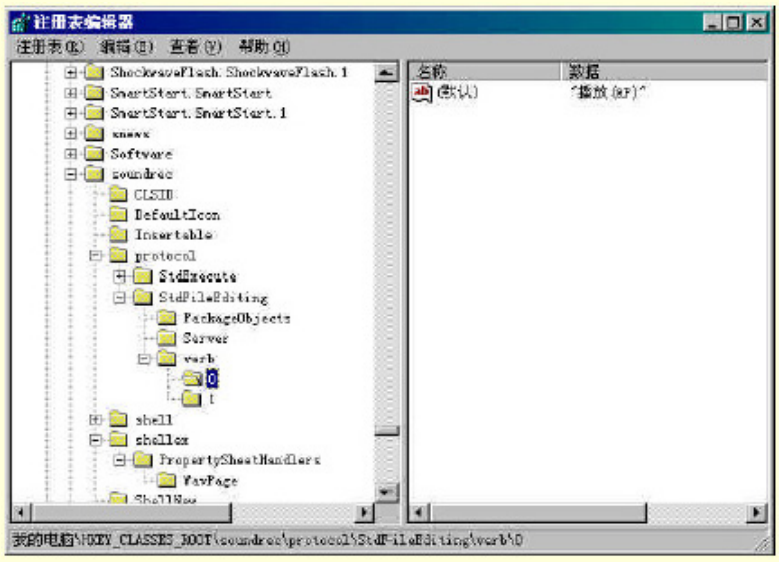


图 6-52 Doverb 方法的参数及其对应的动作

6.4.16 添加 ActiveX 绑定型控件

ActiveX 绑定型控件又称 OLE 绑定型控件，它的使用方法与 Active 容器控件基本相同，只不过它的数据源是表中的通用型字段。

在设计表的结构时，允许用户定义类型为通用型（General）的字段，此字段用于保存 OLE 对象的数据。在

设计表单时，可以先打开设计好的表，添加 ActiveX 绑定型控件后，把它的 ControlSource 属性设置为表的通用型字段，如一个自由表名为“职工信息”，它的一个通用型字段为“照片”，则可设置 ActiveX 绑定型控件的 ControlSource 属性为“职工信息.照片”。

下面是一个实例，它应用 ActiveX 绑定型控件来控制音乐的播放。

（1）新建一个自由表“Sound.dbf”，如图 6-53 所示。表中的字段“音乐”是通用型字段，分别存储了四个 MIDI 序列文件，在“\Windows\media\”中可以找到这些文件。

图 6-53 “Sound”表

（2）新建一个新表单，并把自由表“Sound.dbf”设置为这个表单的数据环境。具体方法是，通过单击鼠标右键选择“Date Environment”或通过菜单打开表单的“数据环境”窗口，如图 6-54 所示，在“数据环境”窗口中单击右键，在弹出的快捷菜单中选择“Add”，在新打开的“Open”对话框中选中表“Sound.dbf”。

图 6-54 设置表单的数据环境

（3）向表单中添加控件。添加一个 ActiveX Bound Control 控件，两个标签和四个命令按钮，各个对象的名称（即 Name 属性）如图 6-54 所示。并可视地调整对象的大小和位置，ActiveX Bound Control 控件对象 Ole1 是用来播放 MIDI 序列音乐的，它不是可视对象，因此在运行表单时它并不显示。

（4）添加事件代码。先添加表单的 Init 代码，这些代码用来设置表单及其中对象的初始属性，也可以象前面几节的例子那样，直接在表单的“属性窗口”设置，并且在代码中设置的属性不能是运行时只读的属性。另外给各个按钮的 Click 事件添加代码。

```
Procedure  Form1.Init
WITH THIS.label1
    .ALIGNMENT=2        && 位于中央
    .CAPTION=SPACE（20）
    .FONTSIZE=20
    .FONTNAME="楷体_GB2312"
ENDWITH
```

```
        WITH THIS.label2
            .ALIGNMENT=2        && 位于中央
            .CAPTION=SPACE（30）
            .FONTSIZE=30
            .FONTNAME="楷体_GB2312"
        ENDWITH

        THIS.CAPTION="播放音乐"

EndProc

Pocedure form1.cmdplay.Click
        thisform.ole1.doverb（0）
        thisform.label1.caption="现在正在播放第"+alltrim（str（recno（））））+"首乐曲!"
        thisform.label2.caption=alltrim（作者）+"的"+alltrim（音乐名）
        thisform.refresh
Endproc

Procedure form1.CmdPrev.Click
        IF !Bof（）
            Skip  —1
            thisform.ole1.doverb（0）
            thisform.label1.caption="现在正在播放第"+alltrim（str（recno（））））+"首乐曲!"
            thisform.label2.caption=alltrim（作者）+"的"+alltrim（音乐名）
        ELSE
            WAIT windows（"现在播放的是第一首乐曲!"）
        ENDIF
Endproc

Procedure form1.Cmdnext.Click
        IF !Eof（）
            Skip 1
            thisform.ole1.doverb（0）
            thisform.label1.caption="现在正在播放第"+alltrim（str（recno（））））+"首乐曲!"
            thisform.label2.caption=alltrim（作者）+"的"+alltrim（音乐名）
        ELSE
            WAIT windows（"现在播放的是最后一首乐曲!"）
        ENDIF
Endproc

Procedure form1.CmdExit.Click
        Release thisform
Endproc
```

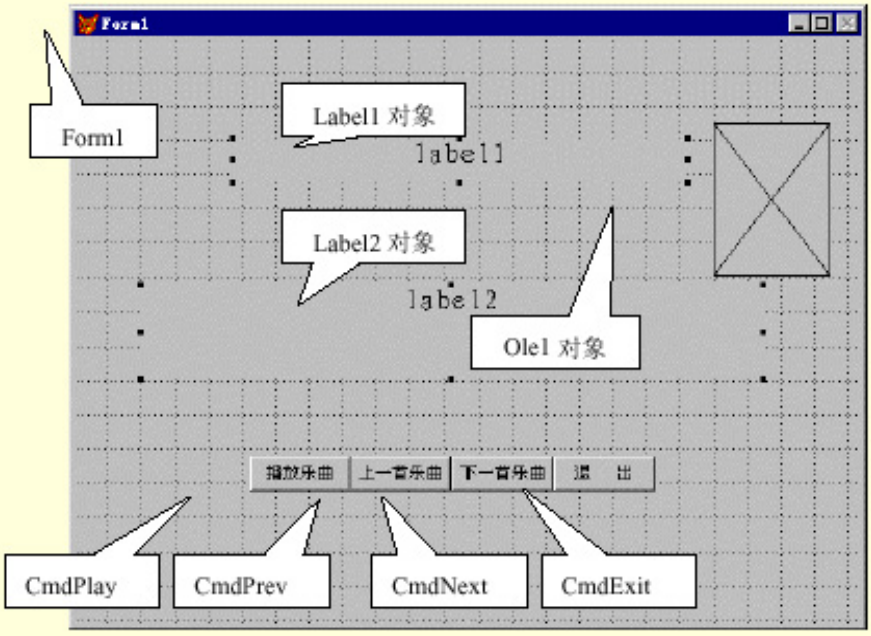


图 6-55 表单中对象的布局和名称

（5）运行表单。情况如图 6-56 所示。

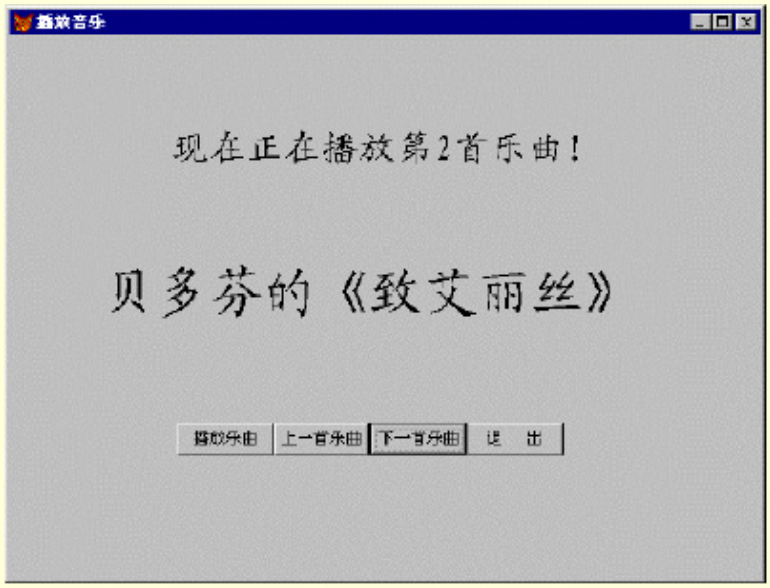


图 6-56 “播放音乐”表单运行情况

6.4.17 添加自定义容器（Container）

虽然 Visual FoxPro 6.0 提供了许多容器对象，如命令组（CommandGroup）、选项组（OptionGroup）、表格（Grid）、页框（PageFrame）等，但它们的使用远不如自定义容器灵活，自定义容器控件能把属性根本不相似的对象组合起来。

在设计时，在自定义容器控件处于编辑（Edit）状态时，可以把自定义容器当作一个小的表单来看待，用户可以把各种控件添加到自定义容器中。

如果表单中的对象较多时，使用自定义容器把表单中分隔成多个单元，这便于应用程序的使用者快速掌握它的使用方法，并且自定义容器能形成不同的凸凹效果，也能起到美化界面的作用。图 6-57 是一个自定义容器的实例。

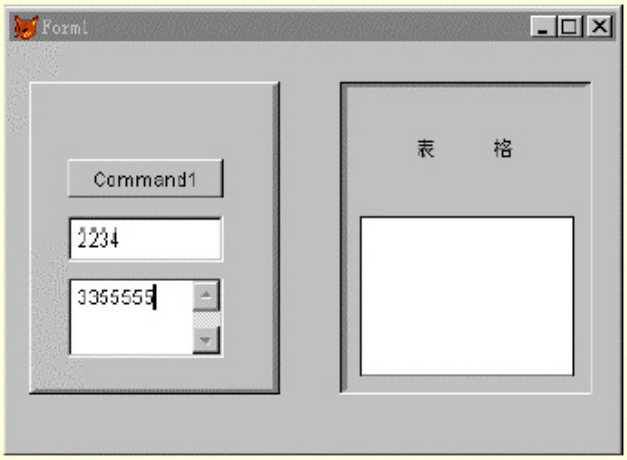


图 6-57 自定义容器的设计实例

6.5 使用自定义类

6.5.1 注册类库

在使用自定义类之前，需要注册类所在的类库，注册类库的方法有两种：

1. 在“Option”对话框中注册

从 Visual FoxPro 6.0 主菜单“Tools”中选择“Option”，在“Option”对话框中选择“Control”选项卡，如图 6-58 所示。

选择“Add”按钮，系统将打开“Open”对话框中，选择一个用户已经设计好的自定义类库文件（.VCX 文件），单击“Open”按钮，把这个自定义类库加入到“Selected”列表框中，至此，就完成了自定义类库的注册。

用同样的方法注册其他的自定义类库。

在“Selected”列表框中列出的类库中的类，在“表单设计器”中可以和 Visual FoxPro 基类一样方便地使用。

也可以把一个已注册的类库从“Selected”对话框移去，选中要移去的类库名，选“Remove”按钮，就移去了这个类库。

如果用户希望在每次运行 Visual FoxPro 时“表单控件”工具栏中的类库都有效，可在“Option”对话框中，选择“Set As Default”把它设置为默认值”。

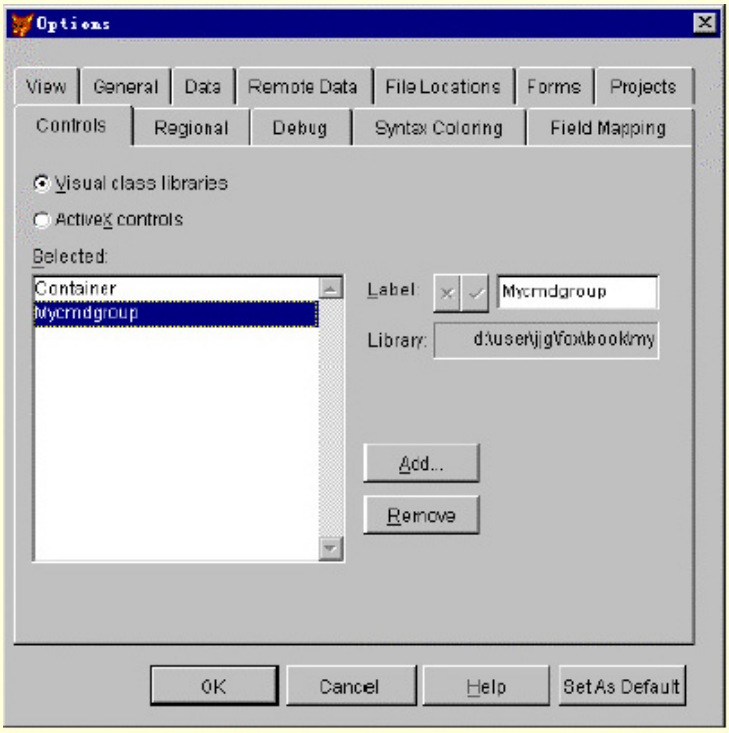


图 6-58 “ Option ”对话框中的“ Controls ”选项卡

2. 在“ 表单设计器 ”中直接注册类库

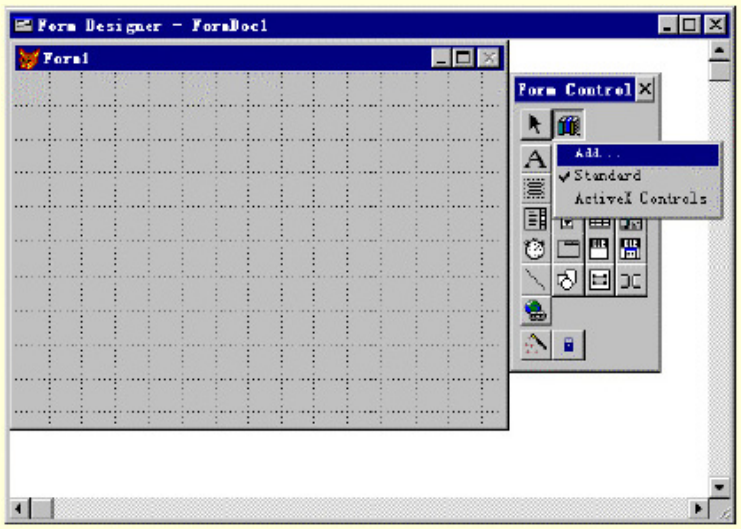


图 6-59 “ View Classes ”的子菜单

在“ 表单控件 ”工具栏中选择“ 查看类（View Classes）”按钮，在出现的菜单中选择“ 添加（Add ..）”项（如图 6-59 所示），系统打开“ Open ”对话框中，选择一个用户已经设计好的自定义类库文件（.VCX 文件），单击“ Open ”按钮后，回到表单设计器中，这时“ 表单控件 ”工具栏上的“ 查看类（View Classes）”的子菜单已经发生了变化，同样，可注册其他的自定义类库。“ 查看类 ”中的子菜单如图 6-60 所示。

图 6-60 “查看类”的子菜单显示出注册的类库

6.5.2 向表单中添加自定义类的对象

自定义类库注册后，用户就能在“表单设计器”中把它们添加到表单上。在“查看类”的子菜单中选择一个自定义类库后，“表单控件”工具栏中的按钮发生了变化，自定义类库中类的图标出现在其中，这样用户就能够像使用系统基类一样使用它们。如图 6-61 所示。

图 6-61 “表单控件”中显示用户自定义类控件

现在举一个例子来说明向表单中添加自定义类对象的方法。

在类设计器中设计一个时钟类，它基于 Visual FoxPro 6.0 的基类自定义容器（Container），它包含一个计时器和一个文本框，它的类设计代码为：

```
DEFINE CLASS mytimer AS container

Width = 120
Height = 35
BorderWidth = 0
Name = "mytimer"

ADD OBJECT text1 AS textbox WITH ;
    FontSize = 20，  ;
    Height = 35，  ;
    Left = 0，  ;
    ReadOnly = .T.，  ;
    TabStop = .F.，  ;
    Top = 0，  ;
    Width = 120，  ;
    Name = "Text1"

ADD OBJECT timer1 AS timer WITH ;
    Top = 0，  ;
    Left = 84，  ;
    Height = 61，  ;
```

```
        Width = 24，  ;
        Interval = 1000，  ;
        Name = "Timer1"

    PROCEDURE timer1.Timer
        this.parent.text1.value=time（）
    ENDPROC
    ENDDEFINE
```

读者可根据上述代码来设计这个时钟类，然后把它存储到 Timerlib.VCX **文件中，并用上面讲的方法注册这个类库。**

在“ View Classes **”的子菜单中选择“** Timerlib **”项，工具栏中出现“** MyTimer **”按钮，按下它，像添加其他** Visual FoxPro 6.0 **基类一样，在表单上拖动一定范围来放置下这个对象。同样办法，再在表单上放置四个“** MyTimer **”对象，并调整各个对象的位置。表单的执行的结果如图** 6-62 **所示。**

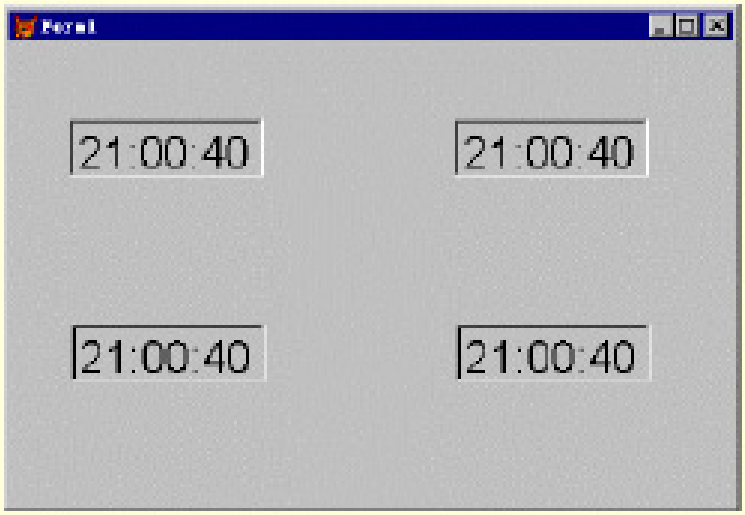


图 6-62　**在表单中添加了自定义时钟类对象**

6.6　使用 ActiveX 对象

除了可以在表单中添加 ActiveX **容器控件和** ActiveX **绑定型控件以外，**Visual FoxPro 6.0 **中还能使用更广泛的** ActiveX **对象，在主菜单“** Tools **”中的“** Option **”对话框中的“** Controls **”选项卡中列出了大约一百种** ActiveX **对象（如图** 6-63 **所示），可供用户选择使用。**

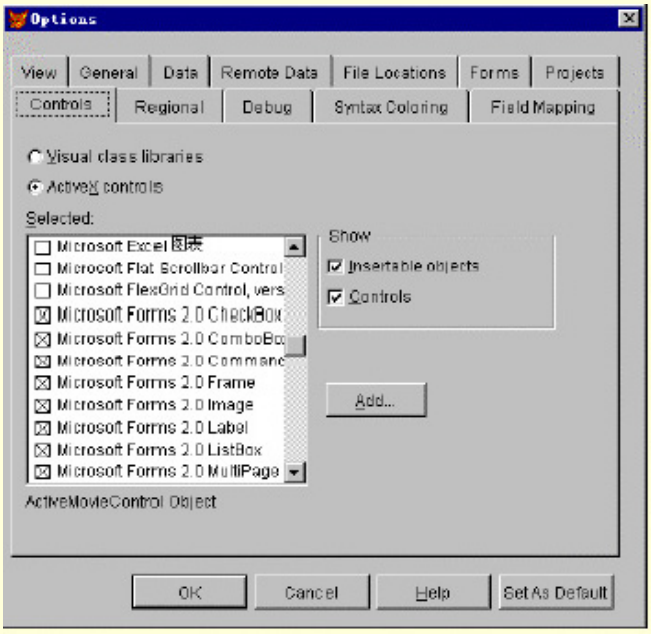


图 6-63 “ Option ”对话框中列出的 ActiveX 对象

若需要使用一个 ActiveX 对象，就选中它前面的选择钮，可同时选择多个 ActiveX 对象，选择完后，按“ OK ”钮返回。

打开表单设计器，在“ 表单控件 ”工具栏中选择“ View Classes ”钮，选择它的菜单中的“ ActiveX ”（如图 6-58 所示），就在工具栏中显示出 ActiveX 对象（如图 6-64 所示）。

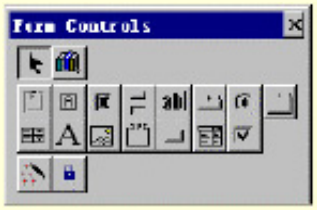


图 6-64 ActiveX 对象

现在使用 ActiveX 对象来给表单添加一个新对象，在 Visual Basic 的“ 窗体控件 ”工具栏中，有一种在应用程序中应用很广的“ 框架（Frame）”控件，它用于给界面的对象分组。虽然 Visual FoxPro 6.0 的基类中没有这种对象类，但可以使用 ActiveX 对象方便地在表单中添加它。

图 6-65 Visual Basic 中的“ 窗体控件 ”工具栏

在图 6-64 的 ActiveX 对象列表框中找到“ Microsoft Forms 2.0 Frame ”项并选中它，并按前面的方法使之显示在“ 表单控件（Form Controls）”工具栏中，如图 6-66 所示。

图 6-66 显示一个 ActiveX 控件时的“ 表单控件 ”工具栏

在表单中添加一个这种对象，并设置其 Caption 属性为“ Visual FoxPro 6.0 中的框架 ”，相应地把表单的 Caption 属性设置为 “ ActiveX 对象的应用 ”。如图 6-67 所示。

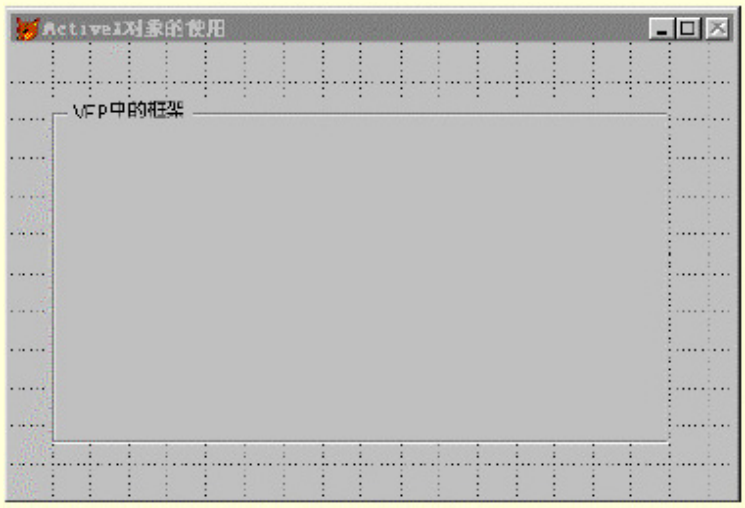


图 6-67　在表单中添加一个 ActiveX 对象

用户可以在这个添加的框架中添加许多其他的新控件，把鼠标放在框架对象上，单击鼠标右键，在弹出的快捷菜单中选择“ Frame Edit ”，此时表单上框架对象的部分打开了一个编辑窗口，并新出现了一个新的“ Controls ”工具栏，这个窗口很象是 Visual Basic 中的窗体设计窗口，如图 6-68 所示。用户可以在其中添加 “ Controls ” 工具栏中的任何控件，当然也可以添加 Visual FoxPro 6.0 中的控件。

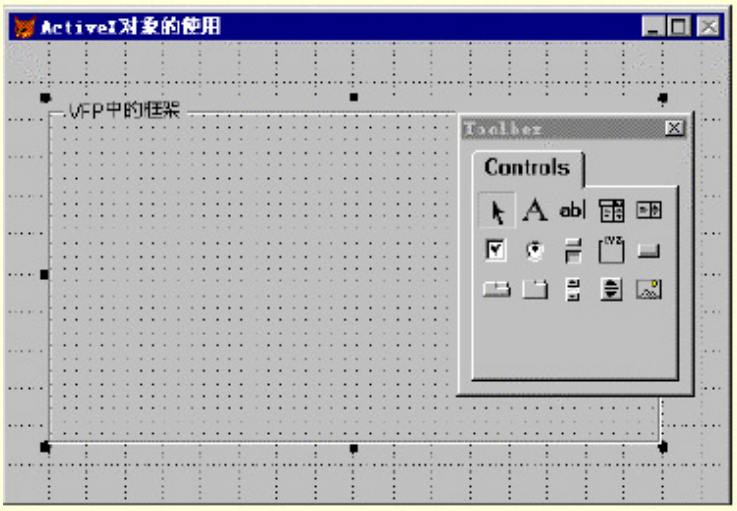


图 6-68　编辑 “ 框架 ” 对象

向其中添加一个 “ Toggle Button ” 对象，这也是一个按钮对象，它能保持被按下的状态，Microsoft Word 中给文本加粗、加下划线的工具栏按钮就是这种对象，调整它的大小和位置后，改变它的其他属性，鼠标指在这个对象时，单击右键，在弹出的快捷菜单中选择 “ Properties …”，把 Caption 属性改为""（空字符），把 Picture 属性选择为 “ ..\Visual FoxPro 6.0\Fox.bmp ”，返回后，运行这个表单，看看结果怎样。

6.7 表单设计过程

6.7.1 设置数据环境

数据环境（Date Environment）包括表单、表单集要使用的数据库、自由表、视图、索引以及表或视图的关系等。一般情况下，一个表单指定了数据环境后，它的数据环境将随着表单的打开或执行自动打开，随着表单的关闭而自行关闭。

“数据环境”窗口是 Visual FoxPro 6.0 提供的设置数据环境的工具，在“数据环境”窗口中用户可以给表单添加（Add）要使用的数据库、表、视图、索引及各种关联，也可以把其中不需要的部分删除。

打开“数据环境”窗口的方法有：

（1）单击“表单设计（Form Design）”工具栏中的数据环境（Date Environment）按钮。

（2）通过菜单“View/Date Environment”进入。

（3）通过快捷菜单，在表单设计区中点击鼠标右键，在弹出的快捷菜单选择“Data Environment”选项。

进入“Data Environment （数据环境）”窗口后，可以看出“数据环境”窗口类似一个数据库设计器，如图 6-69 所示，用户可以使用鼠标拖动来直接添加数据库或表到数据环境，也可以使用鼠标拖动来给数据环境中的数据表之间建立关联。

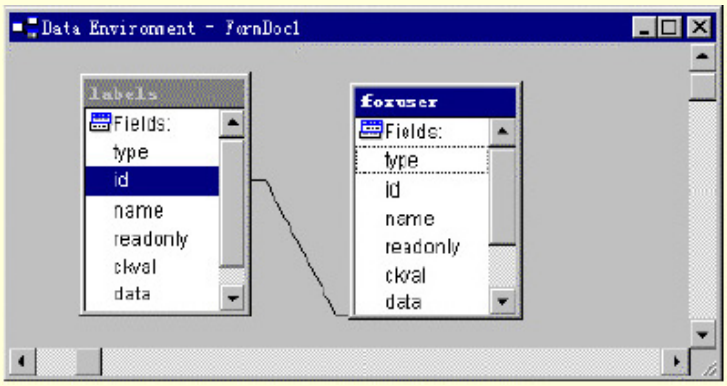


图 6-69 表单的数据环境

在这个数据环境中，它包含了两个自由表“Labels”和“Foxuser”，这个数据环境还包含一个表间的关联，它可视地表示为表间的一条连接线，它们的对象层次与属性如图 6-70 所示。

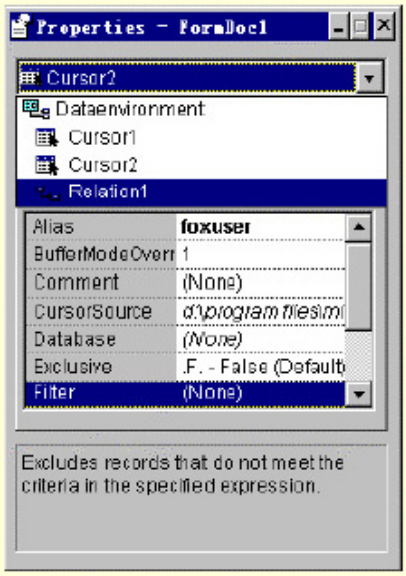


图 6-70 数据环境的对象层次及属性

显然，Visual FoxPro 6.0 把数据环境、数据表、视图、表或视图之间的关联等都看作是一个对象。可以在表

单运行时通过程序中的控制语句动态地改变它们。

在设计表单时，数据环境不是必须的，在本章的许多实例中，都没有设置数据环境。

6.7.2 根据目的选择适当的控件

在应用程序的开发过程中，设计表单添加控件有其基本的出发点，它一方面考虑应用程序的使用者的需求，力求功能更完备；另一方面也要让应用程序操作简单，便于用户掌握使用方法。因此要求开发者在使用控件时一定要使用它的约定俗成的事件，并尽量给文本框、列表框等显示信息的控件旁添加标签来提示。假若一个开发者给一个直线（Line）控件指定 Click 事件，试图让用户在使用时单击它来完成一项操作，试想那是多么滑稽可笑。

6.7.3 添加新属性、新方法

虽然 Visual FoxPro 6.0 中表单及每一个控件都有多种属性、事件和方法，在一般情况下能满足用户设计时的要求，但是它们毕竟是有限的，每个属性、事件或方法都只能完成系统指定了的操作，开发者的创造性受到了相当的局限。为此 Visual FoxPro 6.0 允许用户给设计的表单和控件添加新属性、新方法，这些新的属性、方法能同系统给定的属性、方法一样使用。

打开表单设计器后，Visual FoxPro 6.0 主菜单中又出现了“Form”项，点击它，弹出的下拉菜单如图 6-71 所示。它有两项是“New Property”、“New Method”，用来添加新属性和新方法。

图 6-71 “Form”菜单的选项

单击“New Property”，进入新属性对话框（如图 6-72 所示）。在“Name”项中填入新属性的名字，注意不要和对象原有的属性、事件或方法重名。“Description”项是新属性的描述，在“属性”窗口中，当用户选择了新属性时，它将显示在状态栏里。

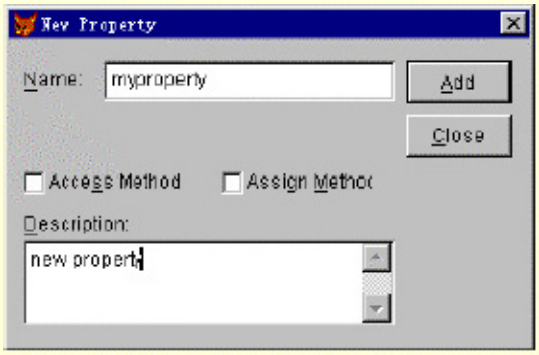


图 6-72 添加新属性对话框

Visual FoxPro 6.0 也允许用户创建数组属性，数组属性和其他属性一样都属于表单，不同的是可用 Visual FoxPro 的数组命令和函数处理它。创建一个数组属性的方法与其他属性基本相同，在“新建属性”对话框的“名称”框中键入数组属性的名称，但需要包括数组的大小和维数。例如，要创建一个 5 行的二维数组，应在“新建属性”对话框的“属性名”框中键入：

Propname[5，2]

在表单中添加完数组属性后，数组属性在属性窗口中以只读方式显示，不能在设计设改变它的值。用户可以在运行时管理数组属性，重新设置数组属性的维数，也可对数组属性的元素赋值。

与添加新属性一样，用户通过“Form”菜单下的“New Method”来添加新方法。用户添加了新方法后，需要在“代码窗口”中设计它，然后可以在其他代码中任意调用。

上面所说的是给表单添加新属性和方法，在表单设计器中不能给其他控件添加新属性和方法，但用户可以在类设计器中添加。

6.7.4 设置表单的属性

在 Visual FoxPro 6.0 中，表单也被作为一个对象处理，在“属性窗口”的对象列表中，它通常处于对象层次的顶层，它也同其他的表单控件一样，拥有自己的属性、事件、方法。下面列出了在设计时表单对象的重要属性：

AlwaysOnTop	控制表单是否总是处在其他打开窗口之上
AutoCenter	控制表单初始化时是否让表单自动地在 Visual FoxPro 6.0 主窗口中居中
BackColor	决定表单窗口的颜色
BorderStyle	决定表单是没有边框，还是具有单线边框、双线边框或系统边框
Caption	决定表单标题栏显示的文本
Closeable	控制用户是否能通过双击“关闭”框来关闭表单
DataSession	控制表单或表单集里的表是在全局都能访问的工作区中打开还是对表单或表单集私有
MaxButton	控制表单是否能最大化，为.F.时表单的“最大化”按钮不可用
MinButton	控制表单是否具有最小化按钮
Movable	控制表单是否能移动到屏幕的新位置
WindowState	控制表单运行时的默认值是最小化、最大化还是正常状态
WindowType	控制表单是非模式表单（默认）还是模式表单。如果表单是模式表单，用户在访问应用程序用户界面中任何其他单元前必须关闭这个表单
Icon	指定表单的图标；当 Windows 中的窗口最小化或当显示为标题栏时显示这个图标，它是一个 .ICO 文件的文件名

表单对象的属性设置相当重要，它直接关系到用户开发的应用程序界面的总体布局。例如若表单的 Closeable 属性设置为.F.时，在表单的或表单中的某一对象的一个事件中必须有释放表单的语句“Release thisform”或“Thisform.release”，使之在运行时不致不能关闭。再如表单的 ShowWindow 属性设置表单运行时在 Windows 桌面上显示的类型，在默认情况下，它是在 Visual FoxPro 6.0 主窗口中显示的（如图 6-73 所示），当最小化 Visual FoxPro 6.0 主窗口时，表单也最小化，但通常用户建立的应用程序界面并不想在 Visual FoxPro 6.0 主窗口中显示，这时就需要把表单的 ShowWindow 属性改变为 2（As Top-lever Form），这时表单与任何 Windows 应用程序的界面一样，直接显示在 Windows 桌面上（图 6-74），当它最小化时，在任务栏中显示它的图标（由其 Icon 属性指定）。

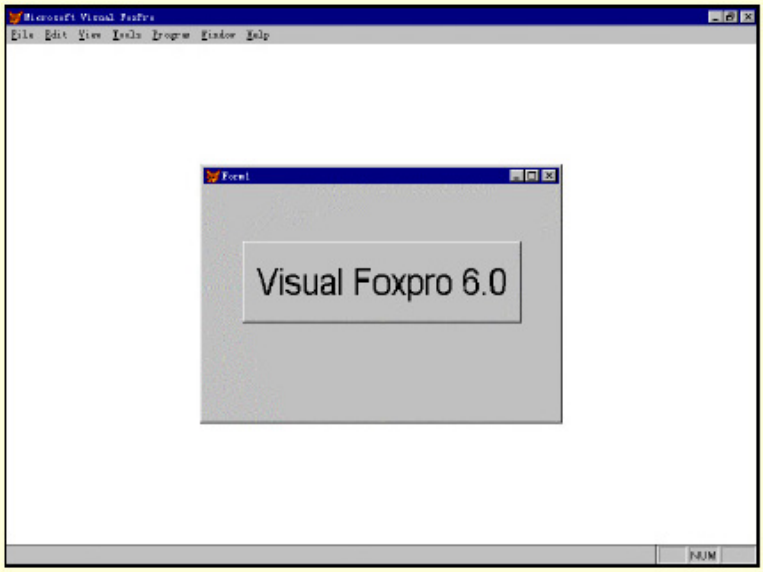


图 6-73　表单的 ShowWindow 属性为 0 时显示在 Visual FoxPro 6.0 主窗口中

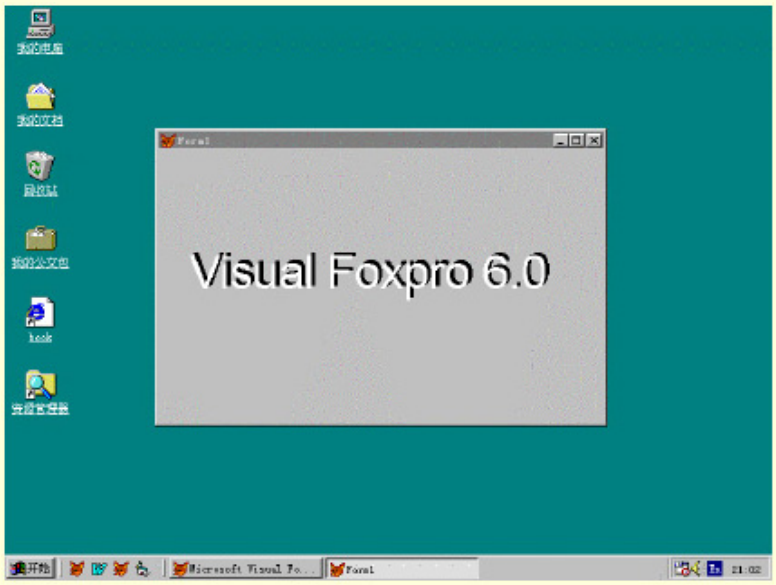


图 6-74　表单的 ShowWindow 属性为 2 时显示在 Windows 桌面上

6.8　创建表单集

有时候，需要使用多个表单来处理问题，这时可以将多个表单包含在一个表单集中，以便同时处理它们。表单集是表单对象的包容器对象，它有以下优点：

（1）可同时显示或隐藏表单集中的全部表单。

（2）能可视地调整多个表单以控制它们的相对位置。

（3）可以在一个表单中方便地操纵另一个表单及其中的对象。

（4）因为表单集中所有表单都是在单个 .scx 文件中用单独的数据环境定义的，可自动地同步改变多个表单中的记录指针。如果在一个表单的父表中改变记录指针，另一个表单中子表的记录指针则被更新和显示。

6.8.1　创建表单集

表单集是一个包含有一个或多个表单的父层次的容器。可在“表单设计器”中创建表单集。步骤如下：

打开“表单设计器”后，系统主菜单出现了一个新选项“Form”，选择并打开“Form”菜单，选择“Create FormSet”选项。这样就创建了一个表单集。

如果表单集中只有一个表单，则可删除表单集而只剩下表单。若要删除表单集，用户需从“ Form ”菜单中选择“ Remove FormSet ”。

6.8.2 添加和删除表单

新创建了表单集时，表单集中仍只有一个表单，需要添加一些新表单。这样表单集的存在才有意义。

从“ Form ”菜单中，选择“ Add New Form ”，可发现在表单设计器中新出现了一个表单。

同样，也可从表单集中删除表单。在“ 表单设计器 ”“ 属性 ”窗口的对象列表框中，选择要删除的表单对象名，或者使用鼠标选中要删除的表单（表单标题栏的颜色变深时），从“ Form ”菜单中选择“ Remove Form ”，这个表单就会从表单设计器中消失。

当运行表单集时，有时不想在最初让表单集里的有些表单是可视的，可以设置其 Visible 为“ .F. ”。也可在表单集运行时，将不希望显示的表单的 Visible 属性设置为“ .F. ”。将希望显示的表单，其 Visible 属性设置为“ .T. ”。

下面是一个表单集的实例.

表单集的基本布局如图 6-75 所示。

图 6-75 表单集实例

读者可根据这个图来设置表单集中的对象属性：

Leftform.Caption="左表单"
Leftform.Closeable=.F.
Leftform.label1.Caption="请选择右表单中显示的图片："
Leftform.Label2.Caption="显示方式"
Leftform.Command1. Caption="..".
Leftform.Command1.FontSize=20
Leftform.OptionGroup1.Option1.Caption="显示全图"
Leftform. OptionGroup1.Option2.Caption="按原尺寸"
Rightform.Caption="右表单"
Rightform.Command1.Caption="退出表单集"
Rightform.Image1.Width=60
Rightform.Image1.Height=60

并添加事件代码：

```
Procedure   Leftform.Command1.Click
      filename=GETFILE（"bmp，jpg，ico，gif"）
      THISFORM.text1.VALUE=filename
      THISFORMSET.rightform.image1.PICTURE=filename
EndProc

Procedure   Leftform.Text1.InteractiveChange
```

```
        thisformset.rightform.image1.picture=this.value
    EndProc

    Procedure    Leftform.OptionGroup1.Option1.Click
        WITH THISFORMSET.rightform.image1
            .WIDTH=60
             .HEIGHT=60
            .STRETCH=1
        ENDWITH
    EndProc
     Procedure    Leftform.OptionGroup1.Option2.Click
        THISFORMSET.rightform.image1.STRETCH=0
    EndProc

    Procedure RightForm.Command1.Click
        THISFORMSET.Release
    EndProc
```

应当注意，运行表单集时，将加载表单集中的所有表单和表单的所有对象，无论用户是否使用，它们都存在于内存中，因此使用表单集会使程序运行的速度比较慢。如果不需要将多个表单处理为表单组，例如当各个表单间互相的操作不太频繁时，则尽可能不创建表单集，最好把多个表单分别存储在单个.SCX **文件中，在使用表单时用命令“** DO FORM **表单名”来加载表单。**

6.9 单文档界面与多文档界面

Visual Foxpro **允许创建两种类型的应用程序：单文档界面（**SDI**）和多文档界面（**MDI**）。**

单文档界面（SDI**）应用程序由一个或多个独立窗口组成，这些窗口均在** Windows **桌面上单独显示。许多小型软件都是一个** SDI **应用程序，在这些软件中打开的每条消息均显示在自己独立的窗口中。例如** Windows **“附件”中的“扫雷”游戏界面，自始至终只有一个窗口，是一个很简单的** SDI **界面。**

多文档界面（MDI**）各个应用程序由单一的主窗口组成，且应用程序的窗口包含在主窗口中或浮动在主窗口顶端。**Microsoft Word **就是一个** MDI **应用程序，在它的主窗口中可以同时打开多个文档窗口。**

还有一些应用程序综合了 SDI **和** MDI **的特性。程序中的某一部分是一个** SDI **应用程序，而程序总体上是一个** MDI **程序。**

6.9.1 三种表单类型

为了支持这两种类型的界面，Visual FoxPro **允许创建以下几种类型的表单：**

1. 子表单

包含在另一个窗口中，用于创建 MDI **应用程序的表单。子表单不可移至父表单（主表单）边界之外，当其最小化时将显示在父表单的底部。若父表单最小化，则子表单也一同最小化。**

2. 浮动表单

属于父表单（主表单）的一部分，但并不是包含在父表单中。而且，浮动表单可以被移至屏幕的任何位置，但不能在父窗口后台移动。若将浮动表单最小化，它将显示在桌面的底部。若父表单最小化，则浮动表单也一同最小化。浮动表单也可用于创建 MDI **应用程序。**

3. 顶层表单

没有父表单的独立表单，用于创建一个 SDI **应用程序，或用作** MDI **应用程序中其他子表单的父表单。顶层表单与其他** Windows **应用程序同级，可出现在其前台或后台，并且显示在** Windows **任务栏中。**

6.9.2 指定表单类型

创建各种类型表单的方法大体相同，但需设置特定属性以指出表单应该如何工作。如果创建的是子表单，则不仅需要指定它应在另外一个表单中显示，而且还需指定是否是 MDI 类的子表单，即指出表单最大化时是如何工作的。如果子表单是 MDI 类的，它会包含在父表单中，并共享父表单的标题栏、标题、菜单以及工具栏。非 MDI 类的子表单最大化时将占据父表单的全部用户区域，但仍保留它本身的标题和标题栏。

1. 建立顶层表单

用“表单设计器”创建一个表单，然后将表单的 ShowWindow 属性设置为 2（As Top—Level Form）。顶层表单与各种 Windows 应用程序窗口一样，独立运行在 Windows 平台上，最小化时出现在 Windows 任务栏中。

2. 建立子表单

用“表单设计器”创建一个表单，然后将表单的 ShowWindow 属性设置为 0 或者 1。设置为 0 时，这个子表单的父表单将为 Visual FoxPro 主窗口，在本章前面的许多实例的表单就是这种类型的子表单，它显示在 Visual FoxPro 主窗口中；设置为 1 是，它的父表单是一个顶层表单，有时用户希望子窗口出现在另一个顶层表单窗口内，而不是出现在 Visual FoxPro 主窗口内时，这时可选用该项设置。

如果希望子表单最大化时与父表单组合成一体，可设置表单的 MDIForm 属性为“.T.”，如图 6-76 下所示；如果希望子表单最大化时仍保留为一独立的窗口，可设置表单的 MDIForm 属性为“.F.”，如图 6-76 上所示。

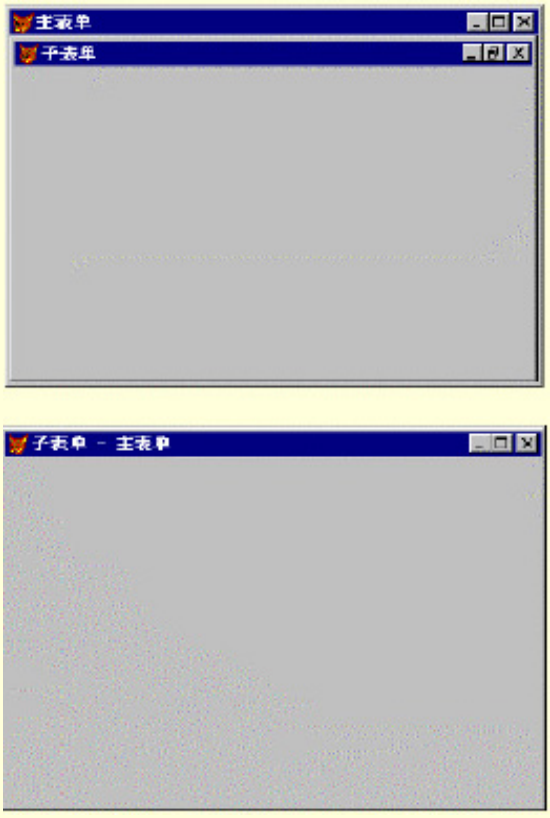


图 6-76 子表单“MDIForm”属性能改变子表单最大化时的显示结果

下面讲一下如何显示顶层表单中的子表单：

先按上文所述的方法创建顶层表单和子表单，然后在顶层表单的事件代码中添加“ DO FORM 子表单名”命令，指定要显示的子表单。但要注意不要把这个命令添加在顶层表单的 Load 或 Init 事件，因为这时顶层表单还未激活，并且要保证显示子表单时顶层表单是可视的。

图 6-77 是一个在顶层表单中显示子表单的例子。

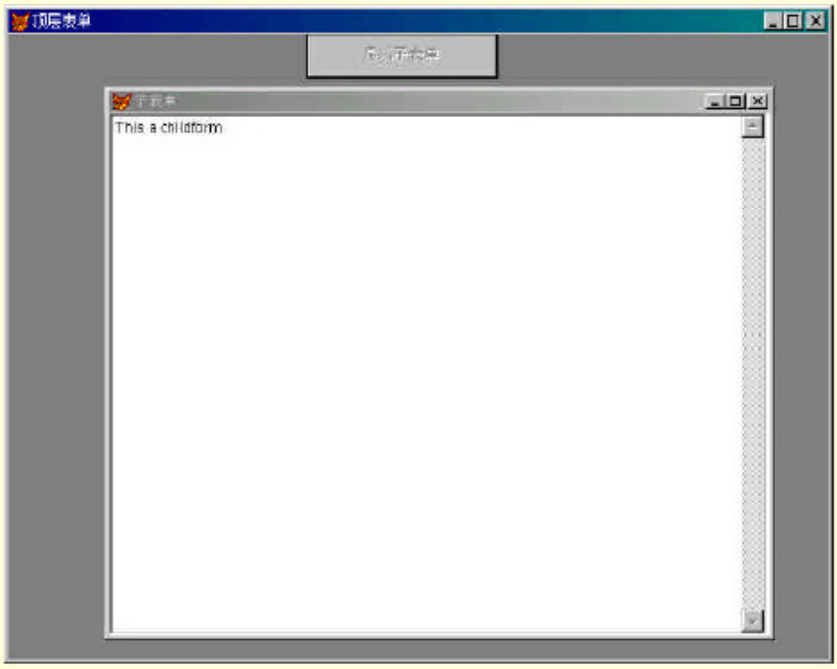


图 6-77 在顶层表单中显示子表单

先建一个顶层表单，然后在顶层表单中建立一个按钮，在按钮的 Click 事件代码中包含如下的命令：

```
DO FORM MyChild
```

应当注意：在显示子表单时，顶层表单必须是可视的、活动的。因此，不能使用顶层表单的 Load 事件或 Init 事件来显示子表单，因为此时顶层表单还未激活。

3. 建立浮动表单

浮动表单是由子表单变化而来，但它属于父表单（主表单）的一部分，但并不是包含在父表单中，因此浮动表单可以被移至屏幕的任何位置，但不能在父窗口后台移动。若将浮动表单最小化时，它将显示在 Windows 任务栏中。若父表单最小化，则浮动表单也一同最小化。

如果把一个表单设置为浮动表单，可将表单的 Desktop 属性设置为.T.（真），并且将表单的 ShowWindow 属性设置为 0 或者 1。

当 ShowWindow 设置为 0（在屏幕中 ）时，浮动表单的父表单将出现在 Visual FoxPro 主窗口；为 1（在顶层表单中）时，浮动表单显示在活动的顶层表单中。

6.10 本章小结

本章介绍了创建一个交互性应用程序的一些基本单位——表单及其控件的使用。熟练掌握这些控件的使用是创建一个应用程序的基础，即使是一个大型的应用程序也是这些控件的组合使用的结果。

本章主要讲述了以下内容：

- 表单向导的使用
- 表单设计器的使用
- 各种表单控件的使用
- ActiveX 控件的使用
- 创建表单集
- 单文档界面和多文档界面

第七章　为应用程序添加菜单和工具栏

菜单和工具栏在应用程序中相当常见。如果把菜单和工具栏设计得很好，那么只要根据菜单的组织形式和内容，用户就可以很好地理解应用程序。恰当地计划并设计菜单和工具栏，将使应用程序的主要功能得以体现，让用户在使用应用程序时不致受挫。

现在几乎所有的 Windows 应用程序都使用下拉式菜单，下拉菜单能较好地组织应用程序的各种操作，为用户提供了一个结构化的访问途径。快捷菜单在 Windows 95/98 中应用广泛，用户在控件或对象上单击右键时，就会显示快捷方式菜单，可以快速展示当前对象可用的所有功能。工具栏常常通过易读的图标来快速实现某一功能，它能够快捷地访问程序，为用户节省操作时间。

本章要讲述的就是设计菜单和工具栏方面的知识，主要包括创建下拉式菜单、快捷菜单和设计自定义工具栏。

7.1　菜单设计器的使用

7.1.1　概述

创建菜单系统的工作主要是在“菜单设计器”中完成的，在那里可创建实际的菜单、子菜单和菜单选项。

在“New File（新建文件）”对话框中，选择“Menu”项，进入“New Menu（新建菜单）”窗口，如图 7-1 所示。它有两个选项，选项“Menu”用来建立一个一般的下拉式菜单，选项“Shortcut”用来建立一个快捷菜单，可单击这两个按钮中的任何一个，现在选择“Menu”按钮，就进入了菜单设计器，如图 7-2 所示。

图 7-1　“New Menu（新建菜单）”窗口

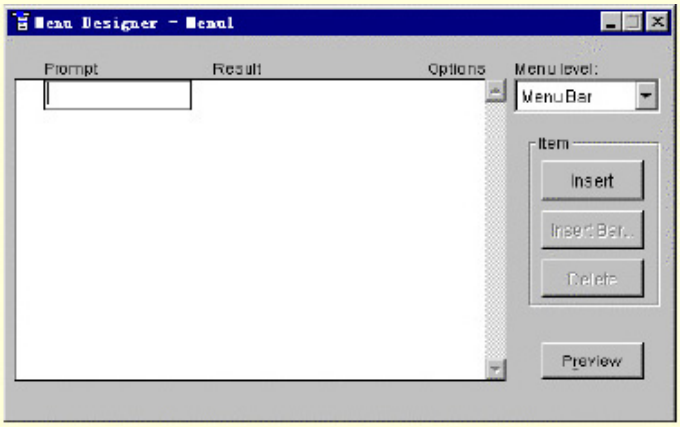


图 7-2　“Menu Designer（菜单设计器）”窗口

7.1.2 菜单设计器的使用

在出现的“菜单设计器”窗口中，左侧屏幕中有三个选项：

1. Prompt 项

此项中输入的是菜单标题，它是在菜单运行时显示的内容，而不是在设计时引用菜单的内部名字，它相当于表单设计时对象的 Caption 属性。

用户在设计此项时，应当给它输入一个有意义的菜单标题，并且这个标题应能准确地描述该菜单项的功能。通常要使用简单的常用词语，使用英文标题时要混合大小写。

2. Result 项

指定该菜单项结果，它有四个选项：“Command”、“Pad Name”、“Submenu”和“Program”，如图 7-3 所示。

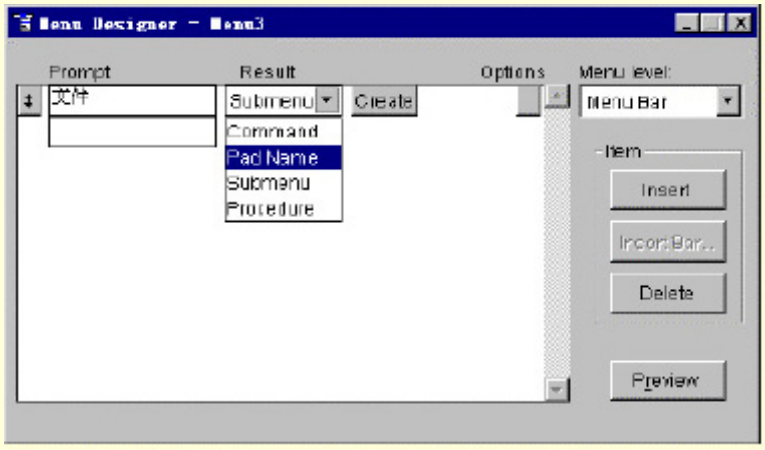


图 7-3 菜单结果的四个选项

（1）Command：表示该菜单项用来执行一条命令语句，选择这一项后，“Result”列右侧出现一个输入框，让用户输入命令语句，如图 7-4 所示。

（2）Pad Name：设计菜单时，在缺省状态下，Visual FoxPro 6.0 会自动给每个菜单项添加一个唯一的名称（Pad Name），它类似于表单控件的 Name 属性，Visual FoxPro 6.0 通过它来引用这个菜单项，但若用户选择此项，“Result”列右侧出现一个输入框，用户便可输入自己定义的菜单名称，如图 7-4 所示。此名称的命名规则与 Visual FoxPro 6.0 中变量的命名规则相同。但在实际设计时并不需要选用此项，因为无论用户选择的菜单项结果是“Submenu”、“Command”还是“Procedure”，都可以为该菜单项设置自定义名称，关于这一点在下文有阐述。

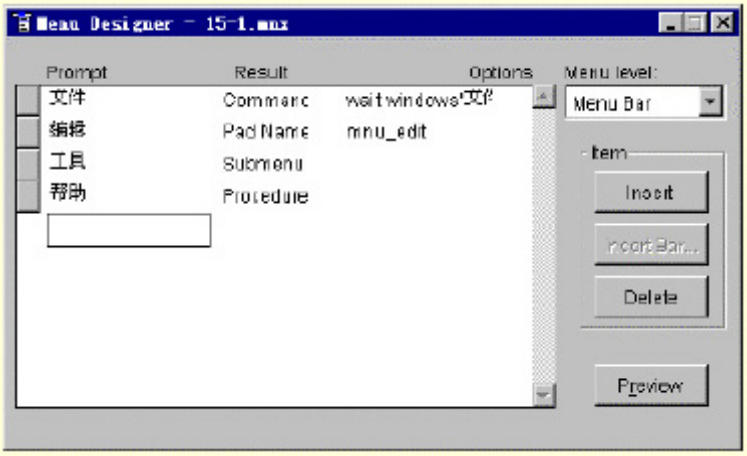


图 7-4 选取不同选项的菜单实例

（3）Submenu：选择此项，表示该菜单项包含一个子菜单，当菜单运行时用户选取该菜单项，将弹出它的子菜单。选择此项后，“Result”列右侧出现“Create”按钮，单击它进入编辑子菜单状态，如图 7-5 所示，除“Menu Level”项与图 7-2 不同外，其余各项完全相同，它各项的意义与图 7-2 也完全相同。

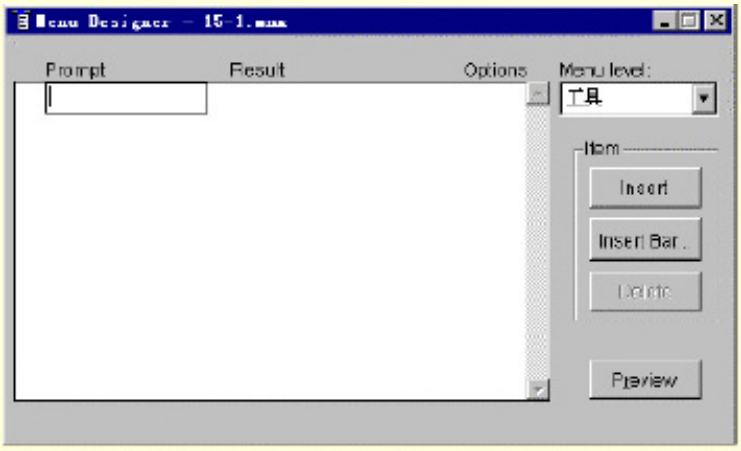


图 7-5 一个子菜单编辑窗口

（4）Procedure：这种菜单项在执行时被选取后，会执行一个程序，它与“ Command ”菜单项有共同之处，当要执行的程序中只有一条语句时，使用这两者都可，但它的优点是能执行含多个语句的一个过程。选择这一选项后，“ Result ”列右侧也会出现一个“ Create ”按钮，单击后，会出现一个过程编辑窗口。

在编辑过程时，不必在开始和结束处书写“ Procedure ..EndProc ”语句，这是由于在生成菜单程序文件.MPR 时，系统会自动把这一语句添加到过程的开始和结束处。这个编辑窗口与程序文件（.PRG）的编辑窗口没有两样，也使用程序文件的设计方法来设计这个过程。如图 7-6 所示。

图 7-6 过程编辑窗口

3. Options 项

选择这个选项列的按钮后，出现“ Prompt Options ”窗口，如图 7-7 所示。用来设置菜单项的各种高级属性，其中各项的使用方法在本章后面有详述。

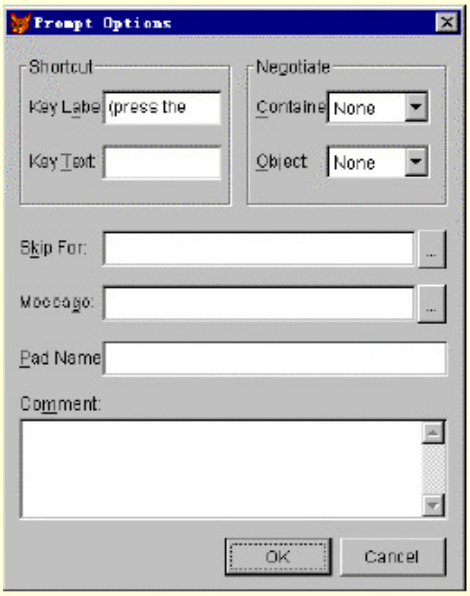


图 7-7 “ Prompt Options（提示选项） ”窗口

除了上述的“ Prompt ”、“ Result ”和“ Option ”三项外，菜单设计器窗口中还有三项：

1. Menu Level

这个下拉组合框用来指示或改变当前设计的子菜单在菜单系统层次结构中的位置，若为“ Menu Bar ”时则表示正在设计系统主菜单。

2. Item

“ Insert ”、“ Insert Item ”、“ Delete ”按钮分别用来插入一个菜单项、插入一个 Visual FoxPro 6.0 系统菜单条、删除一个菜单项。其中“ Insert Item ”按钮只用于设计子菜单时（“ Menu Level ”不能是“ Menu Bar ”）。

3. Preview

用来预览设计的菜单，预览时设计的菜单直接显示在 Visual FoxPro 6.0 主窗口中，能帮助用户更好地设计菜单。图 7-4 菜单的预览结果如图 7-8 所示。注意预览时只是显示菜单的各个菜单项，并不能响应菜单命令和程序。

图 7-8 预览菜单

7.2 创建菜单

7.2.1 快速菜单

打开“ 菜单设计器后 ”，Visual FoxPro 6.0 系统主菜单中出现了一个新的“ Menu ”菜单项，选取它，弹出其子菜单，如图 7-9 所示。

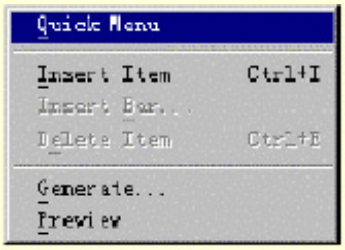


图 7-9 “ Menu ”的子菜单

如果用户没有设计菜单的经验，可以选取“ Quick Menu ”来自动生成一个菜单系统，这就是快速菜单，如图 7-10 所示。快速菜单其实就是 Visual FoxPro 6.0 的系统主菜单，读者可以通过查看这个快速菜单，来学习设计菜单的一般方法。当然用户也可以修改这个快速菜单来达到更满意的效果。

Quit Menu（快速菜单）是设计菜单的捷径，它很类似于一种向导，只不过它的灵活性更差，只能生成单一形式的菜单，因此直接使用它的情况较少。通常情况下要修改它后再使用，例如可以删除不必要的菜单项，或者把这个菜单汉化，修改菜单的方法与下文创建下拉式菜单相同。

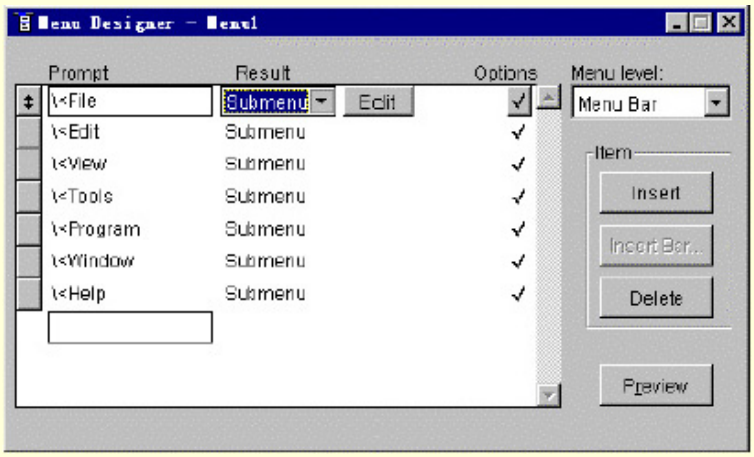


图 7-10 Quit Menu（快速菜单）

7.2.2 规划、形成菜单系统

一个菜单系统是各个菜单项的有机组合，要根据应用系统的要求，来完成下面的几项工作。

1. 规划主菜单

主菜单项应囊括应用系统的各种功能，它们通常又包含自己的子菜单。例如现在要设计一个银行帐务管理系统的主菜单，它由“ 存款 ”、“ 取款 ”、“ 打印 ”、“ 查询 ”、“ 帮助 ”和“ 退出 ”六个主菜单项组成，如图 7-11 所示。图中除“ 退出 ”一项外，其余各项的“ Result ”都选择“ Submenu ”，即表示都有子菜单。

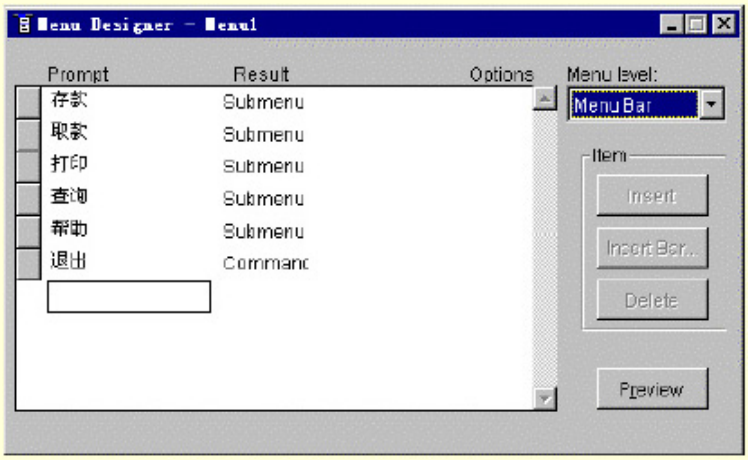


图 7-11 “ 银行帐务管理系统 ”的主菜单项

2. 设计子菜单

不仅子菜单子项中能包含子菜单，子菜单项也可以再有它的子菜单，一个菜单项的子菜单的内容一定要与它本身联系紧密，让使用者操作时不致感到混乱。如现在给图 7-11 中“ 查询 ”项设计子菜单，它包含“ 账号余额 ”、“ 存款日期 ”、“ 取款日期 ”、“ 户主地址 ”四项，分别代表要查询的内容，如图 7-12 所示。

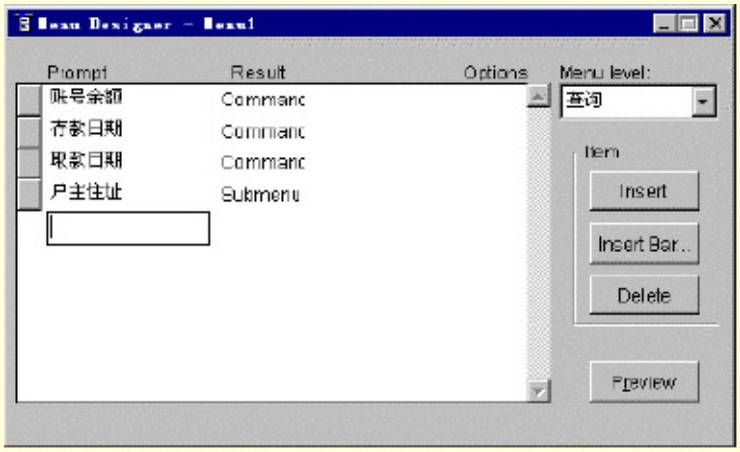


图 7-12 查询菜单项的子菜单

3. 给菜单项排序、分组

在一个子菜单中，各项的使用频率不同，要根据频率估计来给菜单项排序，让使用频率高的菜单排在前面。例如“存款”的子菜单中包含“活期”、“定期”、“ 零存整取”、“ 定活两便”、“ 新开账户”五项，显然，“活期”与“定期”两项的使用频率要比其他几项高的多，把他们排在前面，如图 7-13 所示。

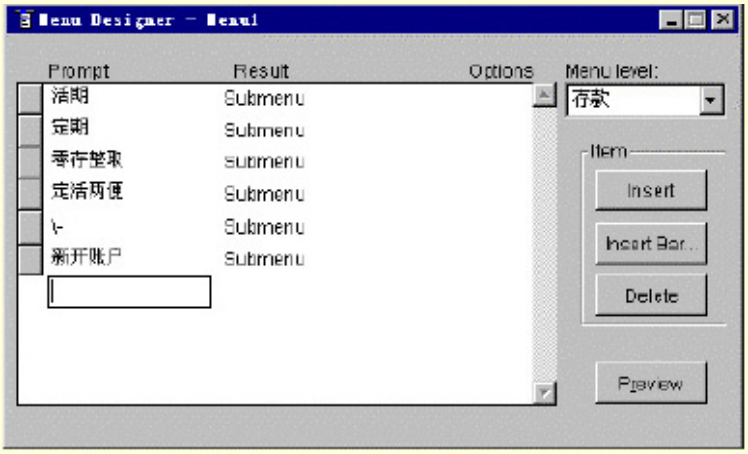


图 7-13 给菜单排序、分组

移动菜单项改变顺序的方法是：先选中要移动的项，按住它前面的按钮“ ↕ ”，把它拖动到新位置后在松开左键，这两个菜单项的位置就调换了。

有时候各个菜单项的功能类别有些差异，就需要给菜单项分组，以增加操作时的可读性。图 7-13 中前面四项都是存款的类别，应与最后一项“新开账户”分开。分组的方法是：在要分组的两个菜单项（即“定活两便”与“新开账户”）之间插入一个新菜单项，并在其“Prompt”列中输入“\-”，如图 7-13 所示。

7.2.3 指定菜单项名称和提示信息

上文讲过，在菜单设计器的“Result（结果）”项中选择“Pad Name”能给一个菜单项命名，但更广泛的命名方式是通过“Prompt Options（提示信息）”窗口中的“Pad Name”输入框来命名，例如给上面“银行帐务管理系统”中主菜单项“存款”命名为“in_money”，如图 7-14 所示，在程序或过程中使用这个名称就能够引用这个菜单项。

添加提示信息的工作是在“Prompt Options”窗口的“Message”输入框中完成的，添加的提示信息在运行时显示在 Visual FoxPro 6.0 的状态条（位于屏幕最下面）中。输入的提示信息可以是任意的变量和常量，显示一串文字时须用" "括起来，以表示是字符型常量。如图 7-14 所示。选取它右侧的触发式按钮“ … ”出现“Expression Builder（表达式生成器）”，然后用户输入复杂的提示信息。

图 7-14 指定菜单名称和添加提示信息

7.2.4 指定快捷键与键盘访问键

专业化的菜单系统都提供快捷键或键盘访问键，以满足各类使用者的要求。快捷键是用“Alt+字母键（或数字键）”作为访问键，键盘快捷键的设置更灵活一些，可以使用“Ctrl”、“Alt”、“Shift”或它们的任意组合与其他键来访问，此外，还可以使用“F1～F12”和“Insert”、“Delete”、“Home”、“End”、“Page Up”和“Page Down”六个键访问。

快捷键的指定方法是在菜单设计器的“Prompt”列中的内容添加“\<快捷键名”，例如“\<A”表示键组合“Alt+A”能访问该菜单项。现在给上面设计的“银行帐务管理系统”主菜单项添加快捷键，如图 7-15 所示。

如果一个菜单系统主菜单项的“Prompt”以英文字母开头，在默认情况下，第一个字母就是这个菜单项的快捷键，例如主菜单项中有一个“File”项，键组合“Alt+F”是这个菜单项的默认访问键。

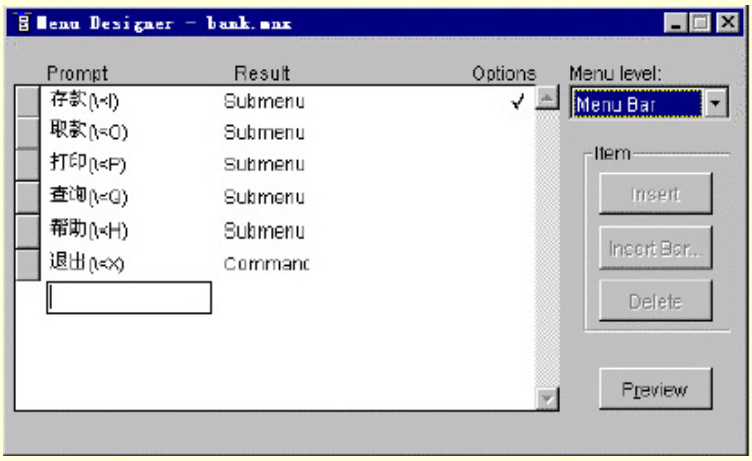


图 7-15 给菜单指定快捷键

键盘访问键通常来便捷地执行菜单中的某一项操作，它一般不用于打开子菜单，即它访问的菜单项的“Result”特性是“Command”或“Procedure”。但这并不是说不能给包含子菜单的菜单项来指定访问键，只是这不符合人们的习惯罢了。

指定键盘访问键在“Prompt Option”窗口中完成。在这个窗口中，如果这个菜单项没有指定访问键时，“Key Label”输入框提示用户“(Press the key)”，按下希望设置的键，“Key Label”框中显示出输入的访问键名。例如现在给上面的“银行帐务管理系统”中的“帮助”项下的“使用方法”设置访问键“F1”，如图 7-16 所示。

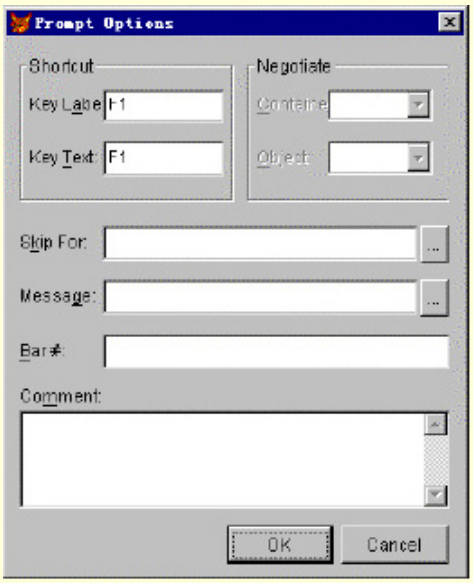


图 7-16 为菜单项指定键盘访问键

图中的“ Key Text ”输入框是在执行时显示在菜单项右侧的提示文字，默认情况下与“ Key Label ”框中相同，一般不要改动它，当把这个框中的文字改为“ FFF1 ”时，预览的结果如图 7-17 所示，这会导致使用者不知道它的访问键到底是什么。

图 7-17 访问键的提示文字

7.2.5 生成菜单程序

在上述的各项工作完成后，这个菜单文件（.Mnx）并不能使用，还需要把它编译为菜单程序文件（.Mpr），然后才能在程序中使用它。选取系统主菜单“ Menu ”（图 7-9）下的“ Generate …”，系统便编译这个菜单系统，并提示用户保存.MPR 文件。

编译后，在命令窗口中键入“ Do 菜单程序文件名（.Mpr）”，就执行这个菜单，执行结果如图 7-18 所示。

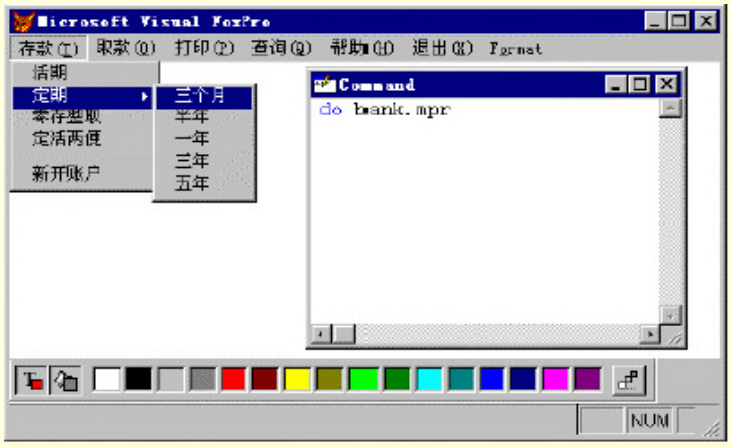


图 7-18 “ 银行帐务管理系统 ”菜单执行情况

7.3 给菜单指定任务

7.3.1 指定菜单项的任务

在上一节中，即便用户已经编译了设计的菜单系统，运行时，除了那些有子菜单的菜单项外，单击其他菜

单项没有任何动作发生，这是由于还没有给菜单项指定任务。并且只能给“ Command ”或“ Procedure ”指定任务，指定任务实质上就是编写响应代码。

例如，在上面设计的“银行帐务管理系统”中，给“存款”下的“新开账户”菜单项添加一个命令语句“ Do form bank1.scx ”，“ bank1.scx ”是已经设计好的表单。这样在运行时选取这一菜单项后，程序将显示并执行表单。

编写代码的方法与前面讲的程序文件或表单事件的代码书写方法一样，这里不再详述。

7.3.2 建立各菜单项的共用过程

共用过程就是在选取菜单项的子菜单项都要执行的过程。如果一个菜单项的子菜单中含有相同的代码，把这些共同代码书写在这个菜单项的共用过程是一个好办法。

建立菜单项的共用过程步骤是：

（1）选取系统主菜单“ View ”中的“ Menu Option…”，打开“菜单选项（Menu Option）”窗口。

（2）在“ Procedure ”输入框中键入共用过程代码，也可单击“ Edit ”按钮调用文本编辑器进行编辑。如图 7-19 所示。

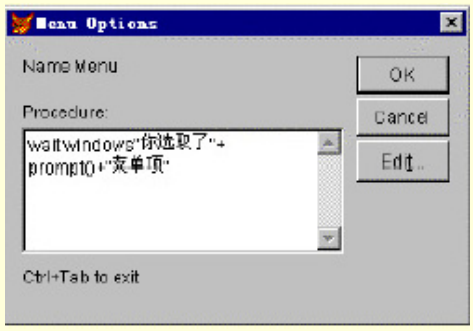


图 7-19 建立共用过程

在“ Menu Designer（菜单设计器）”中改变当前菜单项所在的位置，再重复上述操作，就能依次给各个菜单项添加它所需的共用代码。“ Menu Option ”窗口中的“ Name ”项指示了当前菜单项名称。

7.3.3 建立默认过程

默认过程是一个全局过程，应用于整个菜单系统，运行时如果选定一个没有指定过程的菜单，就运行此过程。

用户在开发一个应用系统时，其中有的菜单可能还没有设计好子菜单或过程。这时，您可以为这些菜单创建一个临时占位过程，当选定这些菜单时，执行此程序，而当给这菜单添加子菜单或过程后，默认过程便不再被执行，这样有利于菜单系统的调试。

创建默认过程的步骤是：

（1）从“显示”菜单中，选择“ General Option …”命令，系统将显示“ General Option（常规选项）”窗口，如图 7-20 所示。

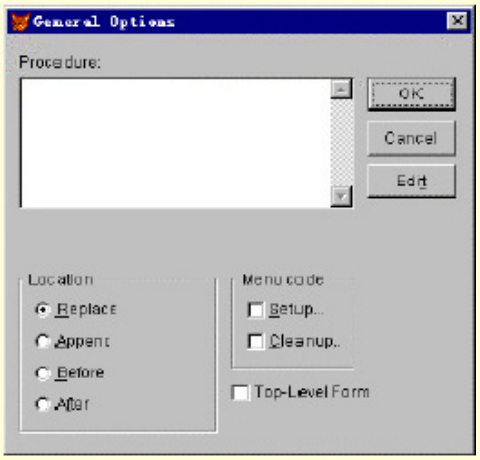


图 7-20 “ General Option（常规选项）”窗口

（2）在“ Procedure ”框中编写过程代码。或选择“ Edit ”，再选择“ Ok ”，打开代码编辑窗口，编写或调用过程。

7.3.4 添加初始化代码

初始化代码是在 .MPR 文件中菜单定义代码之前执行的程序部分，它是菜单文件中最先执行的代码，类似于表单控件的“ Init ”事件。初始化代码可以用来打开所需文件、声明内存变量和设置环境。

添加初始化代码的步骤如下：

（1）在系统主菜单“ View ”中选取“ General Option…”，打开“ General Option ”窗口，如图 7-20 所示。

（2）在窗口的“ Menu Code ”栏中选取“ Setup …”项，并按“ OK ”钮后，出现了一个代码编辑窗口来让用户输入初始化代码。

（3）在编辑窗口中输入初始化代码。图 7-21 是一个菜单的初始化代码实例。

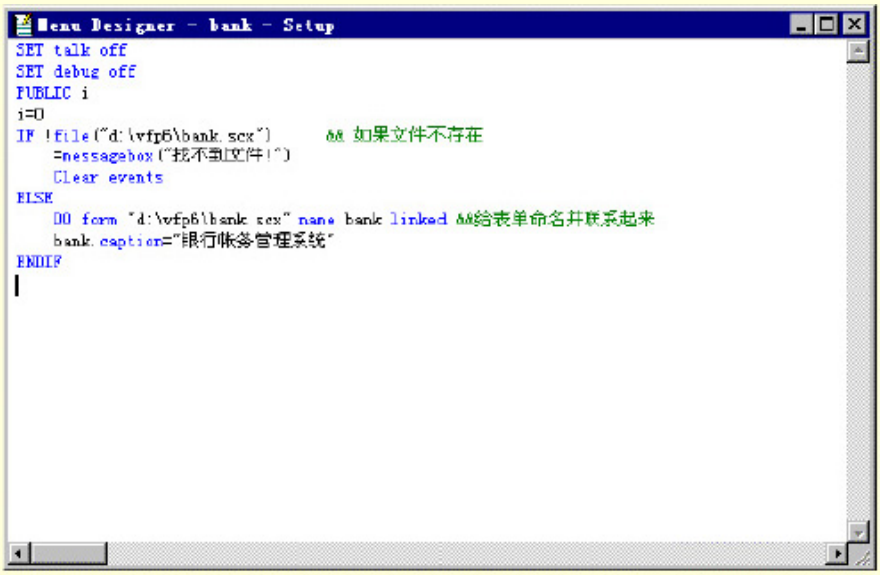


图 7-21 一个初始化代码实例

7.3.5 添加清理代码

清理代码在 .MPR 文件中处于初始化代码和菜单定义代码之后，但在为菜单项指定的过程或命令代码之前。清理代码一般包含启动或跳过菜单项和用户自定义函数、过程。

添加清理代码的步骤与初始化代码基本相同如下：

（1）从系统主菜单“ View ”下选取“ General Option…”，弹出“ General Option ”窗口。

（2）在这个窗口的“ Menu Code ”栏中选取“ ClearUp…”，后按“ Ok ”。

（3）在新出现的代码编辑窗口中输入清理代码。

7.4 动态菜单

7.4.1 菜单项的启用与废止

在一些常见的软件中，有时用户常见到菜单系统的某些项是不能用的，如在 Visual FoxPro 6.0 中，如果没有打开任何文件，则主菜单“ File ”中的“ Close ”项是虚的。现在就讲一下怎样让自己设计的菜单根据逻辑条件启用或废止。

设置菜单项启用或废止的方法有：

（1）给菜单标题（即 Prompt 栏中的文字）前添加“ \ ”后，这个菜单项就处于废止状态。

（2）使用 SET SKIP OF 命令启用或废止菜单项。

（3）选定菜单项，在其“Prompt Option”对话框。在“Skip for”输入框中输入表达式，或按其右面的生成器按钮，打开“Expression Builder（表达式生成器）”，在其中输入表达式。当表达式的值为真（.T.）时菜单项废止，为假（.F.）时则菜单项启用。

在设计时，通常使用第三种方法来控制菜单项，例如在前面设计的“银行帐务管理系统”菜单的初始化代码中定义一个变量 Statu_exit，用它来标记“退出”菜单项的应用与废止状态：

```
Public Statu_exit
Statu_exit=.T.
```

然后在“Prompt Option”对话框的“Skip for”中输入“Statu_exit”，如图 7-22 所示。在运行菜单时，当变量 Statu_exit 的值改变时，这个菜单项就能够自动启动或废止。

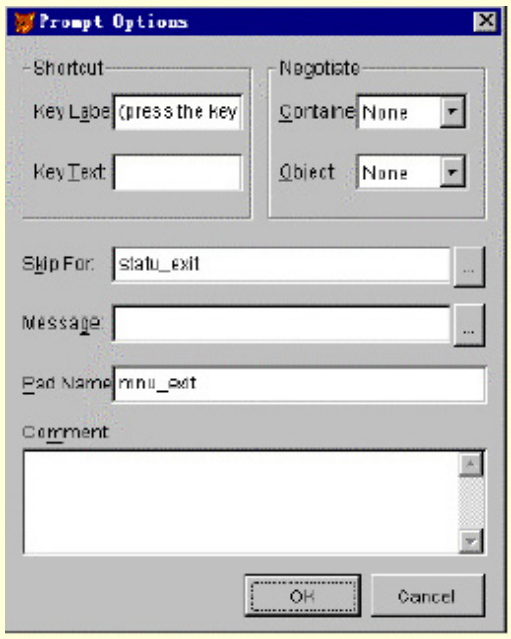


图 7-22 设置启用和废止菜单的逻辑条件

7.4.2 标记菜单项的状态

在菜单中，如果菜单项左边有一个复选标记，则表明此菜单项对应的功能正在使用。例如，在系统主菜单“Window”中（如图 7-23 所示），在最下面的三个菜单项中有一个加复选标记菜单项，它标志了正在活动的窗口。

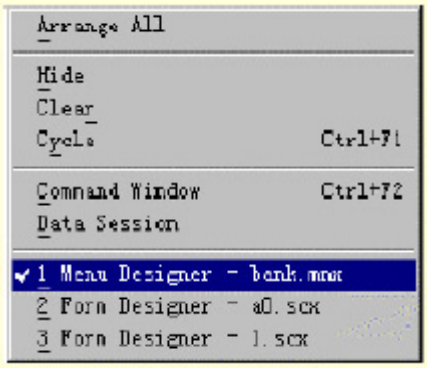


图 7-23 “Window”菜单中加复选标记的菜单项

在菜单代码中使用 SET MARK OF 命令在菜单项旁边设置复选标记，命令“SET MARK OF”的用法为：

```
SET MARK OF MENU MenuBarName1 TO lExpression1
SET MARK OF POPUP MenuName1 TO lExpression2
SET MARK OF BAR nMenuItemNumber OF MenuName2 TO lExpression3
```

例如建一个菜单，它的一个菜单项“Option”下的有一个“sound”项，其子菜单为“On”和“Off”。

并分别给它们添加代码：

```
SET mark of bar 2 of sound to .F.
SET mark of bar 1 of sound to .T.
Set bell On
```

和

```
SET mark of bar 2 of sound to .T.
SET mark of bar 1 of sound to .F.
SET bell Off
```

运行时，当选取“On”时，在它上面添加复选标记；选取“Off”时，消除“On”菜单项上的标记，并给自身添加标记。如图 7-24 所示。

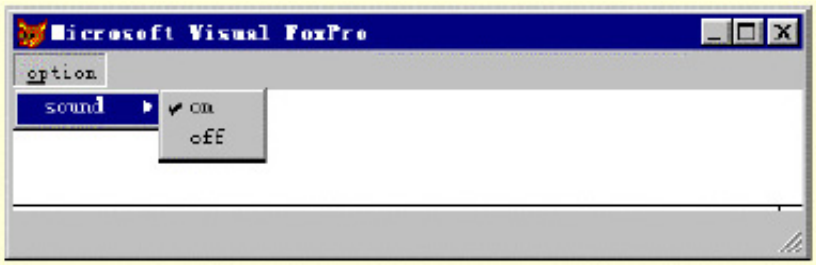


图 7-24 给菜单项加复选标记

7.4.3 改变菜单内容

虽然本章讲解的都是使用菜单设计器来设计菜单，生成 .Mnx 文件后再编译成 .Mpr 文件，但实际上，用户完全可以直接书写 .Mpr 文件，这种文件的书写格式与前面所讲的命令文件没有什么不同。

用户也可以打开一个系统编译的.Mpr 文件来学习用直接编码建立菜单的方法。

一般情况下，并不鼓励使用这种方式来建立菜单，因为它的效率太低且容易出错。但是有时候，程序运行时要对菜单进行复杂的控制，就需要使用一些菜单定义语句，把它书写到用菜单设计器设计的菜单项的过程中去。

创建菜单可使用 DEFINE PAD 命令，创建子菜单可使用 DEFINE POPUP 命令，而在子菜单上创建菜单项则可以使用一组 DEFINE BAR 命令。

7.4.4 释放菜单

（1）关闭菜单

DEACTIVATE MENU 菜单名

（2）释放菜单

CLEAR MENUS 菜单名

或

RELEASE MENUS 菜单名

（3）删除指定的菜单项

RELEASE BAR 菜单项号 OF 菜单名

7.5 创建快捷菜单

在 Windows95/98 中，在控件或对象上单击右键时，就会显示它的快捷方式菜单，可以快速展示当前对象可用的所有功能，方便用户的使用。

用 Visual FoxPro 创建快捷菜单的方法与创建一般的下拉式菜单完全相同，只是在“新建菜单”对话框（图

7-1）中选取“ Shortcut ”，这样建立的菜单就是快捷菜单。

现在建立一个快捷菜单，来实现剪贴板的“ 剪切 ”、“ 复制 ”和“ 粘贴 ”命令。

（1）选择新建菜单，按“ Shortcut ”后，进入菜单设计器。

（2）由于 Visual FoxPro 6.0 系统主菜单“ Edit ”中包含能实现这几种功能的“ Cut ”、“ Copy ”、和“ Paste ”项，因此利用它们来快速完成快捷菜单的设计。选择“ Insert Bar …”，在弹处的“ Insert System Menu Bar ”对话框中选择相应项，如图 7-25 所示。选定后，单击“ Insert ”。

（3）把菜单项的“ Prompt ”分别改为“ 剪切 ”、“ 复制 ”和“ 粘贴 ”，完成菜单的设计。如图 7-26 所示。

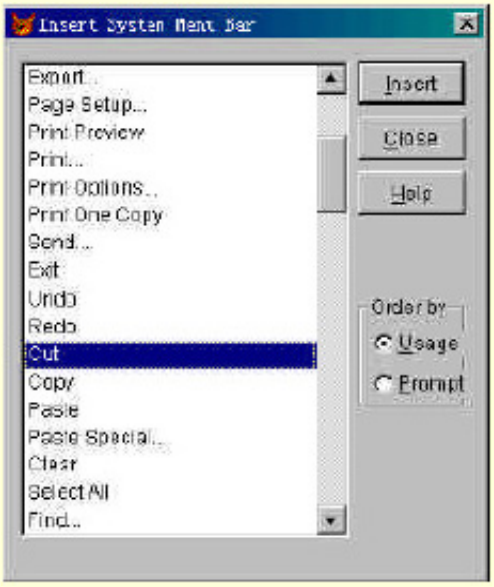


图 7-25 “ Insert System Menu Bar（插入系统菜单条）”对话框

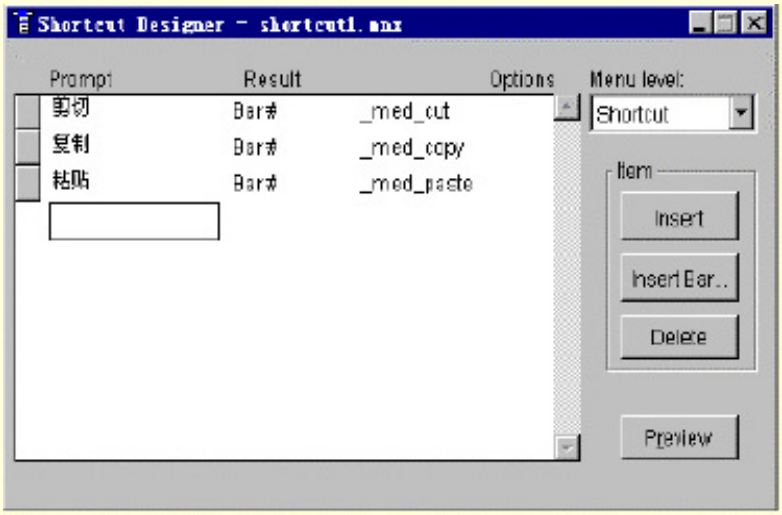


图 7-26 一个快捷菜单实例

7.6 在应用程序中使用菜单

7.6.1 使用菜单作为主界面

现在的许多应用系统都是以下拉式菜单作为主界面，通过菜单命令来执行表单或实现其他操作。在应用系统主程序中执行命令：

Do 菜单文件名（.MPR）

就可以了，这种使用菜单的方法很简单，但应注意所设计的菜单（.MNX）必须编译为菜单文件程序〔.MPR〕后才能执行，并且快捷菜单不能这样使用。

7.6.2 在顶层表单中显示一个 SDI 菜单

在小型软件中，通常只有一个表单或一个主表单和其他几个表单，把菜单附加在表单之中将会显得更为美观。在 Windows 附件中，所有的程序都是在表单中显示菜单。下面就来讲述怎样来实现它。

（1）把表单的 Showwindow 属性设置为 2 ，使它成为顶层表单。

（2）设置菜单为 SDI 菜单，只需在“ General Options ”对话框中选中“ Top-Level Form ”，如图 7-27 所示。

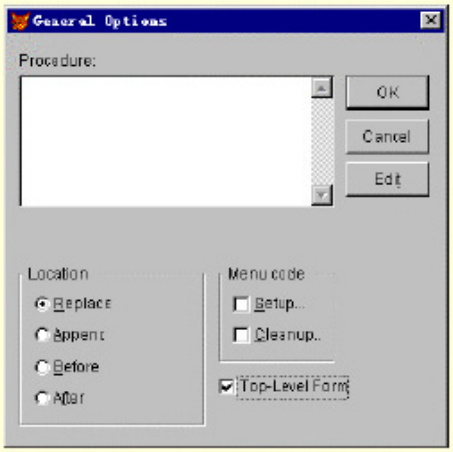


图 7-27 把菜单设置为 SDI 菜单

然后重新编译菜单文件，这样在生成的 .MPR 文件最前面才会有传递参数的语句，例如：

```
LPARAMETERS  oFormRef， getMenuName， lUniquePopups， parm4，parm5
```

（3）在表单的 Init 事件中添加调用菜单的代码：

```
DO 菜单程序文件名（.MPR） WITH THIS
```

以上三条都必须要做到，否则将产生错误。图 7-28 是一个把 SDI 菜单加到表单的例子。

图 7-28 把菜单加入到表单中

7.6.3 将快捷菜单附加到表单控件中

快捷菜单都是单击鼠标右键弹出的，因此可以在表单控件的 RightClick 事件中添加执行菜单的命令代码，这样就能达到目的。

在图 7-29 的例子中，在编辑框的 RightClick 事件中添加了下列代码：

```
DO  d：\Visual FoxPro\shortcut1.mpr
```

其中“d：\Visual FoxPro\shortcut1.mpr”是前面图 7-26 中设计的快捷菜单编译后生成的程序文件。在运行表单时，在编辑框中单击右键，就弹出了这个快捷菜单，如图 7-29 所示。

图 7-29 把快捷菜单附加到控件中

7.7 创建自定义工具栏

如果应用程序中包含一些用户经常重复执行的任务，那么可以添加相应的自定义工具栏，用一个有意义的图标来简化操作，以加速任务的执行。

7.7.1 设计自定义工具栏类

如果用户想向自己开发的应用程序添加工具栏，就必须先设计这个工具栏类，但这个类必须派生于 Visual FoxPro 6.0 的基类 ToolBar。设计自定义工具栏的步骤如下：

（1）在“新建（New）”对话框中选择“Class”，以新建一个可视类。

（2）在出现的“New Class”对话框中，给“Class Name”框输入自定义类的类名，下拉“Base On”组合框，选择“Toolbar”，并在“Store in”中输入或选择可视类库的文件名，如图 7-30 所示。

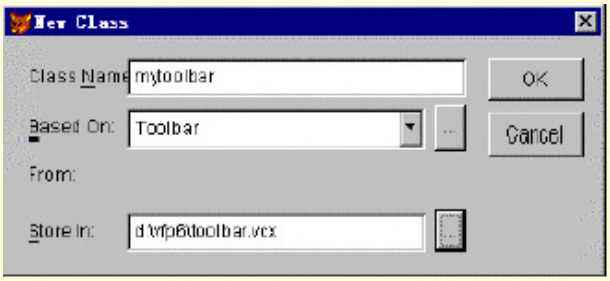


图 7-30 新建一个工具栏类

（3）在类设计器中，向工具栏类添加对象，一般只给工具栏类添加命令按钮，但也能够给自定义工具栏类添加注入标签、文本框等任何表单控件，也可以添加另一个用户自定义类。在图 7-31 中，给一个自定义工具栏类添加了五个命令按钮。

有时还需要给工具栏分组，这就要在工具栏的控件间加入分隔符控件，即“表单控件”工具栏中的“  ”按钮。图 7-31 中给第三、四个按钮间加入了分隔符。

（4）设置工具栏类中对象的属性。

先调整好各个的大小，一般让它们大小相同；不需要调整它们的相对位置，它可以根据情况自动变化，其实这正是工具栏的优点所在。

通常要指定工具栏中对象的 Picture 属性，让它显示一个生动形象的图片。在图 7-31 中，分别给五个按钮指定了一个图片，用来表示“新建”、“打开”、“保存”、“打印”和“预览”。

（5）给各对象添加代码。通常工具栏中对象的 Click 事件代码是不可少的，并且一般只使用它。添加代码的方法与给表单中对象添加代码一样，这里不再赘述。

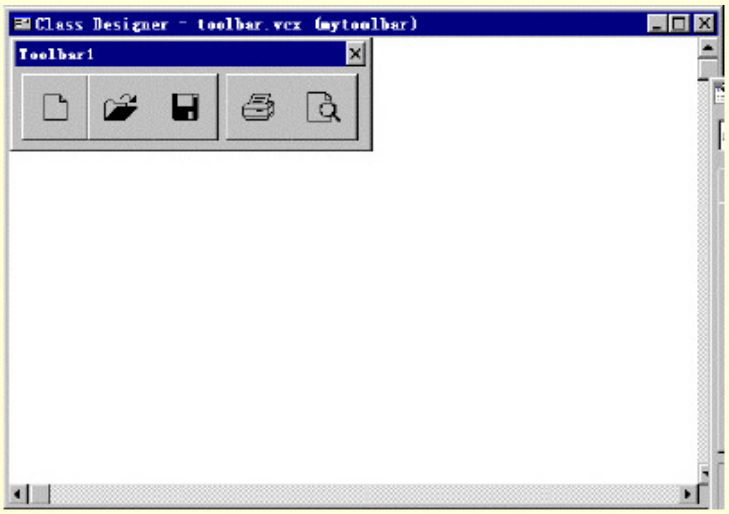


图 7-31 设计自定义工具栏类

7.7.2 向表单集中添加自定义工具栏

先注册自定义的工具栏类，然后在“表单控件（Form Controls）”工具栏中单击“View Classes”，并选取自定义工具栏类所在的类库，把自定义工具栏类显示在“表单控件”工具栏中。

在表单设计器中，系统把工具栏作为一个表单来处理，当用户试图向一个表单中添加工具栏对象时，系统会提示建立表单集。

把工具栏添加到表单集后，就能够改变工具栏及其中对象的属性，或者给这些对象添加事件代码。

7.8 菜单和工具栏的配合使用

在开发的应用程序中，工具栏所完成的通常是菜单中使用较频繁的命令，因此有必要使菜单命令与对应的工具栏按钮同步工作。

这就要求做到：

（1）无论用户使用工具栏按钮，还是使用与按钮相关联的菜单项，都执行同样的操作。

（2）相关联的工具栏按钮与菜单项具有相同的启用或废止状态。

创建协调的菜单和工具栏按钮的步骤如下：

（1）先通过定义工具栏类来创建工具栏，添加命令按钮，并将要执行的代码包括在对应于此命令按钮的 Click 事件的方法程序中，并把工具栏添加到一个表单集中。

（2）创建与之协调的菜单。

在“菜单设计器”中，根据工具栏上的每个按钮对应地创建子菜单。在每个子菜单项的“Result”栏，选择“Command”,用命令来调用相关工具栏按钮的 Click 事件对应的代码：表单集名.工具栏对象名.按钮名.Click 。

打开“Prompt”对话框，在“Skip for”栏中输入表达式，指出当工具栏命令按钮失效时，菜单功能应该“跳过”。例如，使用下面表达式：

NOT 表单集名.工具栏对象名.按钮名.Enabled

（3）生成菜单程序文件（.Mpr）。

（4）在表单集中 Load 或 Init 事件中添加执行菜单程序的代码。

当用户打开菜单时，Visual FoxPro 系统计算“Skip for”条件的值，如果相关的工具栏命令不可用，则菜单也不可用。当用户选择菜单项时，则执行相关工具栏命令的 Click 事件代码。

7.9 本章小结

菜单和工具栏是用户和应用程序交互的重要界面，在目前大多数应用程序中，几乎都要用到菜单和工具栏。熟练掌握菜单和工具栏的设计，才能创建便于交互的应用程序。

本章主要讲述了以下内容：

- **菜单设计器的使用**
- **对菜单的各种控制**
- **在应用程序中使用菜单**
- **创建工具栏**
- **协调菜单和工具栏在应用程序中使用**

第八章　对象的链接与嵌入

OLE 是 Object Linking and Embedding（对象的链接与嵌入）的简写。也就是把一个对象（如文本数据、声音数据、图片数据或视频数据等）以链接或者嵌入的方式包含在其他程序中。通过利用其他应用程序（可以使用自动的应用程序）和 ActiveX 控件的优势，可以扩展 Visual FoxPro 的功能。例如在应用程序的表单或通用型字段中，可以包含从其他应用程序中得来的特殊的功能或数据，例如文本数据、声音数据、图片数据或视频数据。

链接与嵌入的区别是，当应用程序中以链接方式包含 OLE 对象时，应用程序与该 OLE 对象只是一个引用关系，当在应用程序中修改此 OLE 对象时，修改结果将在包含该 OLE 对象的应用程序中反映出来；而嵌入方式包含 OLE 对象时，应用程序保留了修改对象的一个副本，与原来的对象没有关系。

8.1　创建 OLE 应用程序

具有自动功能的应用程序和 COM 组件可以是自动服务程序 、客户应用程序 (Client)，或者两者。作为服务程序的组件可以向其他应用程序提供对象；作为客户应用程序的组件可以创建对象。

可以很容易地在 Visual FoxPro 应用程序中合并 Microsoft Excel 和 Word 等应用程序的功能和灵活性。因为 Visual FoxPro 也可以作为服务程序,所以也可以把它的功能集成到基于 Microsoft Office 或其他 COM 组件的解决方案组中。

可插入的 OLE 对象来自于支持 OLE 的应用程序，例如 Microsoft Excel 和 Word。这样的对象包括 Word 文档和 Excel 工作表。在表单上可以使用 OLE 容器控件链接或嵌入这些对象，并且可以在表的通用型字段中保存这些对象，使用 OLE 绑定型控件在表单上进行显示。

在 Visual FoxPro 应用程序中，可以有很多方法来使用 OLE 和 ActiveX 技术。

8.1.1　链接或嵌入 OLE 对象

可以将其他 Windows 应用程序的文件嵌入或链接到表或表单中。例如，可以将一个 Word 文档嵌入或链接到表的通用型字段中，也可以把 Excel 电子表格嵌入或链接到表单上。

嵌入和链接的不同在于数据的存储地点。嵌入将数据存储到表或表单中，而链接却不是这样。例如，把一个 Excel 电子表格嵌入到一个表单上时，表单将包含该电子表格的副本。然而对于链接，表单仅包含对一个电子表格的引用，而不是电子表格本身。

嵌入和链接的数据一开始包含的都是服务程序文件中的原始内容。但当初始文件改变时，链接的数据将自动更新以反映这些变化，而嵌入的数据则不能更新。但是嵌入的数据并不是静止的。嵌入和链接的数据都可以通过交互方式和编程方式来显示、更改和操作。

8.1.2　在表中添加 OLE 对象

为应用程序设计表时，应考虑表中是否需要 OLE 对象。假设有一个产品介绍表，并想在此表中包含一些 Excel 图表，其中含有对产品性能的说明以便向顾客介绍。要包含这些 Excel 图表，则必须在表中定义一个通用型字段。然后，通过将图表链接或嵌入到通用型字段使其添加到表中。

若要在表中添加 OLE 对象，先使用表设计器创建一个带有通用型字段的表。浏览表并双击通用型字段，或者使用 MODIFY GENERAL 命令，打开通用型字段的窗口。在“编辑”菜单中，选择“插入对象”命令，或者使用 APPEND GENERAL 命令，将显示“Insert Object（插入对象）”对话框。如图 8-1 所示。

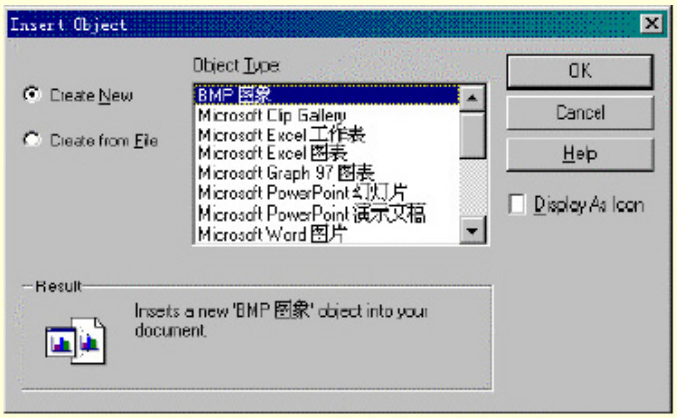


图 8-1 插入对象对话框

当选择“New”选项时此对话框中的对象类型列表框列出了可以插入的对象类型。选择对象类型以后，即可创建新对象；当选择“Create from File（由文件创建）”选项时，对话框如图 8-2 所示。

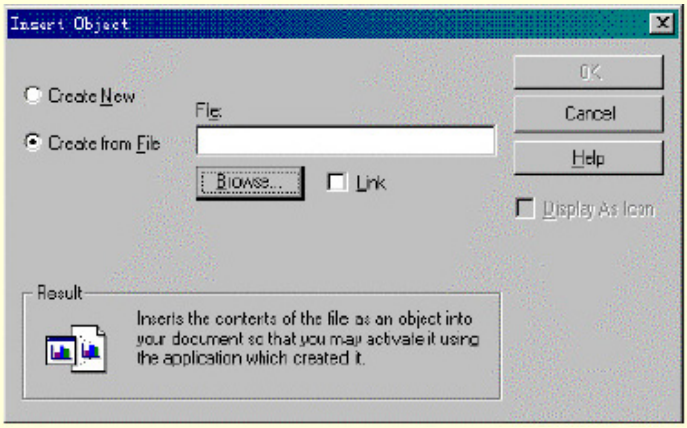


图 8-2 插入对象对话框

如果选中了“链接”复选框，则表示使用了链接方式。在通用字段存在 OLE 对象时，双击该字段可以打开 OLE 对象的编辑器，在 OLE 对象编辑器中双击该对象即可打开编辑该对象的应用程序，对该对象进行编辑。

使用 APPEND GENERAL 命令，可以通过编程方式将 OLE 对象添加到表中。使用该命令，可以从一个文件中导入 OLE 对象，并将其放到通用型字段中。如果该字段已包含对象，则新对象将替换旧对象。

与 APPEND 和 APPEND BLANK 命令不同，APPEND GENERAL 命令并不向表中添加新记录。

可以使用 APPEND GENERAL 命令嵌入或链接 OLE 对象，如 Excel 和 Word 等应用程序创建的 OLE 对象。这些应用程序支持链接和嵌入。但是某些应用程序，如 Microsoft Graph，仅支持嵌入。

例如，将 Microsoft Word 文件存储到一个 Visual FoxPro 表中，而且该表有一个名为 WordDoc 的通用型字段，那么可以使用下列代码将文档嵌入：

```
CREATE TABLE oletable (name c(24), worddoc g)
CD GETDIR()

nFiles = ADIR(aWordFiles, "*.doc")
IF nFiles > 0
    FOR i = 1 to nFiles
        APPEND BLANK
        REPLACE Oletable.Name WITH aWordFiles(i,1)
        APPEND GENERAL WordDoc FROM aWordFiles(i,1)
    ENDFOR
ELSE
    MESSAGEBOX("没有 Word 文件！")
```

```
ENDIF
```

8.1.3 在表单中添加 OLE 对象

使用表单设计器，可以使用“ ActiveX 控件 ”向表单中添加可插入的 OLE 对象。另外，可以使用 ActiveX 绑定控件，显示通用型字段中的 OLE 对象。

若要在表单中添加 OLE 控件，可以采用如下步骤：

在“ 表单设计器 ”中，向表单添加一个 ActiveX 控件。此时“ 插入对象 ”对话框打开。如图 8-3 所示。

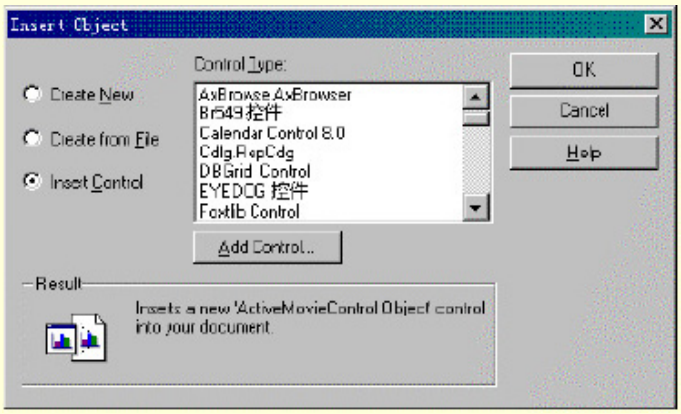


图 8-3 插入对象对话框

在“ 插入对象 ”对话框中，选择“ 新建 ”或“ 由文件创建 ”。从“ 对象类型 ”列表中选择合适的 OLE 对象。

也可以自定义“ 表单控件 ”工具栏，这样就可以直接添加特定的 OLE 对象。

若要显示通用型字段中的 OLE 对象，则可以在“ 表单设计器 ”中，将一个“ ActiveX 绑定控件 ”添加到表单中。

通过设置对象的 ControlSource 属性指定包含数据的通用型字段。

如果表名为 Product，通用型字段名为 Picture，那么将 ControlSource 属性设置成 Product.Picture。

也可以通过编程方式显示通用型字段中的 OLE 对象。

```
frm1 = CREATEOBJECT("form")                    &&创建表单。
frm1.ADDOBJECT("olb1","oleboundcontrol")       &&  添加控件。
frm1.olb1.ControlSource = "Inventory.Current"  &&  将数据与控件绑定。
frm1.olb1.Visible = .T.
frm1.Visible = .T.                             &&  使控件和表单可见。
```

8.2 应用自动服务管理对象

表单或程序中的 OLE 对象，或者 OLE 容器控件中的 ActiveX 控件，都可以通过代码来管理，管理的方式与管理 Visual FoxPro 内部对象的方式相同。

8.2.1 管理外部对象的属性

在代码中，可以通过对象的属性来管理对象。引用属性的方式取决于该对象是单独的，还是容器的一部分，例如 ActiveX 控件或 ActiveX 绑定控件。在这里，ActiveX 控件总是 OLE 容器控件的一部分。

容器中的对象具有两部分：对象本身和包含对象的容器。对象和容器都具有属性，并且有时它们具有相同的属性名。为了确保引用的是对象的属性，请在对象名后面紧跟该对象的属性。例如，下面的代码引用的是对象的 Left 属性。

```
frm1.olecontrol1.Object.Left = 25   &&  对象的 Left 属性
```

如果略去对象名称，则引用的将是容器的 Left 属性。

```
frm1.olecontrol1.Left= 25   &&  容器的 Left 属性
```

例如，假设有一个应用程序，当用户单击一个组合的命令按钮时发送邮件。如果在表单中已加入了一个 Microsoft MAPI 消息控件 olecontrol1，与命令按钮相关的 Click 事件的代码可以如下：

```
THISFORM.olecontrol1.Object.Compose
THISFORM.olecontrol1.Object.Send(.T.)
```

除了使用 Object 属性来引用容器包含对象的属性，也可以使用容器控件的其他属性。例如，可以引用只读的 OLEClass 属性来辨别容器中对象的类型；引用 Sizable 属性来防止用户更改对象的大小。

在“表单设计器”和“类设计器”中，ActiveX 控件的属性显示在“属性”窗口中，但是大多数 ActiveX 控件也具有自己的界面用于设置公共属性。通过从 ActiveX 控件的快捷菜单中选择特定对象的属性，则可以看到这一属性界面。

例如，要想打开一个 rich text 控件的属性对话框，在快捷菜单中选择“RichTextCtrl Properties”对话框。如图 8-4 所示。

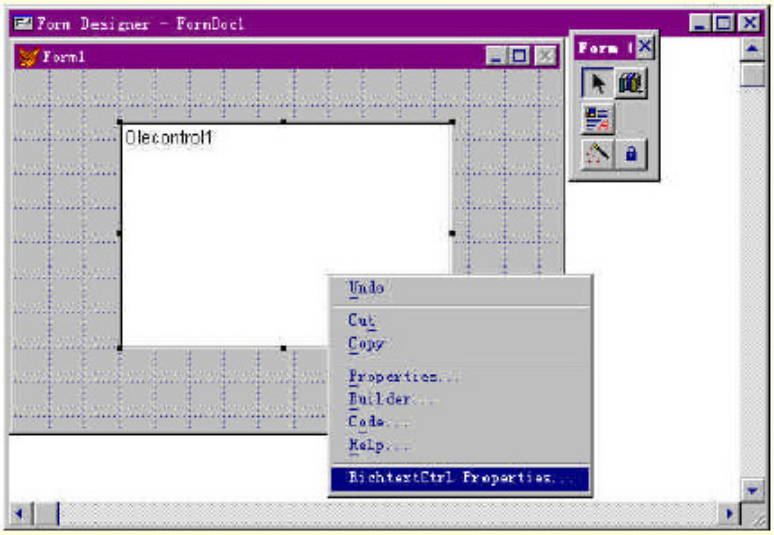


图 8-4 在快捷菜单中选择 RechtextCtrl 属性对话框

选择后将弹出 RechtextCtrl 属性对话框，如图 8-5 所示。可以看到其属性对话框与 Visual FoxPro 的标准属性对话框不同。

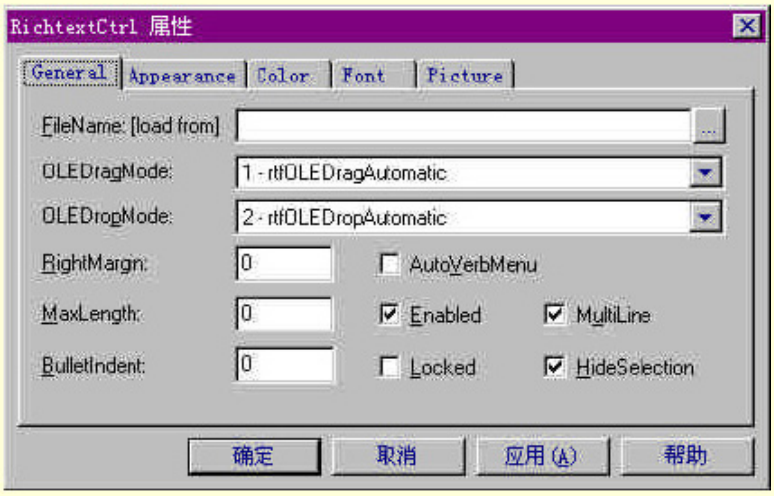


图 8-5 RechtextCtrl 属性对话框

8.2.2 使用外部对象方法程序

除了设置和检索对象的属性，还可以使用对象支持的方法程序来管理对象。例如，可以使用 Microsoft Excel 集合对象的 Add 方法程序新建一个 Microsoft Excel 工作簿。

下面的自动服务示例使用 Add 方法程序来创建一个 Excel 工作簿，Save 方法程序用来保存工作簿，Quit

方法程序用来结束 Excel：

```
oleApp = CREATEOBJECT("Excel.Application")          &&启动 Excel。
OleApp.Visible=.T.                                  &&显示 Excel。
OleApp.Workbooks.Add                                &&创建一个工作簿。
OleApp.Cells(1,1).Value=7                           &&设置单元格的值。
OleApp.ActiveWorkbook.SaveAs("C:\TEMP.XLS")         &&保存工作簿。
OleApp.Quit                                         &&退出 Excel。
```

如果使用 OLE 容器控件或 OLE 绑定型控件创建了一个对象，可以使用该控件的 DoVerb 方法程序对该对象执行一个谓词。例如，使用 DoVerb(0) 执行默认谓词，使用 DoVerb(-1) 激活对象以进行可视编辑，使用 DoVerb(- 2) 在独立的窗口中打开对象。

8.2.3 设置时间期限

向一个 OLE 对象发出请求时，则自动服务程序处理它。无法过多地控制服务程序的进程，但是可以设置 OLERequestPendingTimeout 和 OLEServerBusyTimeout 属性来指定等待进程直到结束的时间。设置 OLEServerBusyRaiseError 属性，可以决定当这个时间用完后将发生的进程。

8.2.4 访问对象集合

一种对象类型可以代表单个的对象或者相关对象的集合。例如，一个 Microsoft Excel Workbook 对象代表单个的工作簿，然而 Workbooks 对象代表所有当前加载的工作簿。由于 Workbooks 对象代表了一个对象集合，所以就称它为一个集合对象。

在代码中，集合是一个无序的列表，无论从集合中添加或移去对象时，其中对象的位置都会改变。可以使用集合的 Count 属性，通过在集合中循环遍历来访问该集合中的对象。Count 属性返回集合中各项的编号。也可以使用 Item 方法程序返回集合中的一项。

例如，使用下列代码可以显示 Microsoft Excel 工作簿中工作表的名称：

```
oleApp = CREATEOBJECT("Excel.Application")
oleApp.Workbooks.Add
FOR EACH x IN oleApp.Workbooks
 ? x.Name
ENDFOR
```

也可以访问集合中的子集合。例如，使用下列代码可以访问一定范围的单元格集合：

```
oleApp = CREATEOBJECT("Excel.sheet")
oleApp.Workbooks.Add
oleApp.Range(oleApp.Cells(1,1),oleApp.Cells(10,10)).Value=100
oleApp.Visible=.T.
```

8.2.5 使用对象数组

可以向方法程序传递数组，也可以传递回数组。但是必须在数组名的前面加 @ 符号作为前缀，通过引用传递数组。

例如，若要向 Microsoft Excel 传递一个 Visual FoxPro 数组，可使用下列代码。该段代码在 Visual FoxPro 中创建一个数组，并为数组赋了值，启动 Microsoft Excel，创建一个工作表，设置工作表中第一个单元格的值，然后将该值复制到数组包含的其他工作表中：

```
DIMENSION aV(3)
aV(1) = "Sheet1"
aV(2) = "Sheet2"
aV(3) = "Sheet3"
oleApp=CREATEOBJECT("Excel.Application")
oleApp.Workbooks.Add
```

```
oleI=oleApp.Workbooks.Item(1)
oleI.Sheets.Item(1).Cells(1,1).Value = 83
oleI.Sheets(@aV).;
  FillAcrossSheets(oleI.Worksheets("Sheet1").Cells(1,1))
oleApp.Visible = .T.
```

反过来，下面的代码向 Visual FoxPro 返回一个数组，然后显示数组中的内容：

```
oleApp = CREATEOBJECT("Excel.Application")
aOleArray = oleApp.GetCustomListContents(3)
FOR nIndex = 1 to ALEN(aOleArray)
? aOleArray(nIndex)
ENDFOR
```

需要注意的是，使用 Visual FoxPro，不能向 OLE 对象传递超过二维的数组。

8.2.6 释放外部对象

当自动服务程序不可见或者在引用对象的作用域中没有变量，则自动释放自动服务程序。可以使用 RELEASE 命令释放与对象相关的变量。如果服务程序是可见的，则使用 Quit 方法程序释放对象。

8.3 派生对象的子类

通过派生 Visual FoxPro 提供的基类或 OLE 控件，可以创建自定义的 OLE 对象。例如，下面的代码派生了 Visual FoxPro 提供的 Outline 控件子类：

```
PUBLIC frmMyForm, cFilename
SET SAFETY OFF                          && 声明变量并初始化。
frmMyForm = CREATEOBJECT("form")
frmMyForm.Width = 100
frmMyForm.ADDOBJECT("oleOutl","myoutline")
DIMENSION aSection(3)
aSection(1) = "Table"                   &&创建表单，将自定义的 Outline
aSection(2) = "Field"                   &&控件添加到表单中，然后为控件
aSection(3) = "Index"                   &&列出的的各项创建一个数组。
cFilename = GETFILE("dbc","Select a DBC")
USE (cFilename)
INDEX ON objecttype FOR (objecttype = "Table" ;
    OR objecttype = "Field" ;           &&提示输入一个数据库名，
    OR objecttype = "Index" ) ;         &&此数据库中包含需要
    TAG fname                           &&控件列出的信息。
FOR nIndex = 1 TO 3 STEP 1
    frmMyForm.oleOutl.AddItem(aSection(nIndex))
    frmMyForm.oleOutl.Indent;
     ((frmMyForm.oleOutl.ListCount-1)) = 1
    SCAN
        IF objecttype = aSection(nIndex)
            frmMyForm.oleOutl.Additem(objectname)
            frmMyForm.oleOutl.Indent;
            ((frmMyForm.oleOutl.ListCount-1)) = 2
```

```
        ENDIF
    ENDSCAN
    GO TOP
ENDFOR                              && 从数据库中收集信息，然后将其添加到控件中。
frmMyForm.oleOutl.Visible = .T.
frmMyForm.Show                      && 使控件可见，然后显示表单。
DEFINE CLASS myoutline AS olecontrol
    OleClass = "msoutl.outline"
    Top = 5                         &&定义 OLE 容器控件的一个子类，
    Left = 5                        &&并通过设置该容器的 OleClass 属性
    Height = 10                     &&来添加一个 Outline 控件，
    Width = 60                      &&并定义其他的可自定义设置
ENDDEFINE
```

8.4 从其他应用程序中控制 Visual FoxPro

因为 Visual FoxPro 可以作为一个服务程序（具有二级兼容性），也可以作为一个客户应用程序，所以支持自动服务的应用程序可以创建 Visual FoxPro 的实例，运行 Visual FoxPro 命令，访问 Visual FoxPro 的对象。

使用 Fpole.dll，甚至可以在不支持自动服务的应用程序中管理 Visual FoxPro。

通过使用 Visual FoxPro 的 Application 对象可以从其他应用程序中控制 Visual FoxPro。只要启动 Visual FoxPro，就直接通过 DDE 或自动服务自动创建了一个 Application 对象。

例如，下列位于 Visual Basic 中或 Microsoft Excel 模块中的代码创建了一个对 Visual FoxPro Application 对象的引用：

```
Dim oFox as Object
Set oFox = CreateObject("VisualFoxPro.Application")
```

引用一个 Visual FoxPro 的 Application 对象后，就可以调用与该应用程序对象相关的方法程序，并且可以通过此 Application 对象的属性集合来访问其他对象。

下例在一个 Excel 模块中使用 Visual Basic for Applications 代码创建一个 Visual FoxPro Application 对象，然后打开 Visual FoxPro 表，将查询结果添加到活动的电子表格中：

```
Sub FoxTest()
Dim oFox as Object
Set oFox = CreateObject("VisualFoxPro.Application")

oFox.DoCmd "USE customer"
oFox.DoCmd "SELECT contact, phone FROM customer
    WHERE country = " + Chr$(39) + USA+ Chr$(39) + " INTO CURSOR cust"
oFox.DataToClip "cust",,3
Range("A1:B1").Select
ActiveSheet.Paste
End Sub
```

8.4.1 Visual FoxPro 的 Application 对象模型

只要启动 Visual FoxPro，就直接通过自动服务或者是 DDE 自动创建了一个 Application 对象。这个 Application 对象通过集合属性提供了对所有在 Visual FoxPro 工作期内创建的其他对象的访问。Visual FoxPro 的 Application 对象模型如图 8-6 所示。

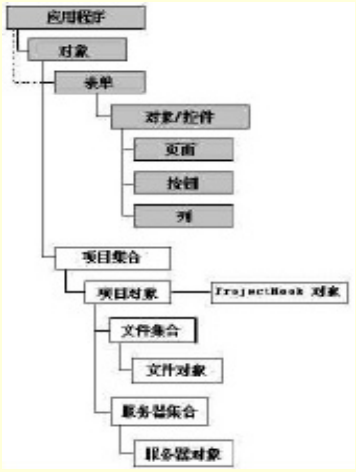


图 8-6 Visual FoxPro 的 Application 对象模型

8.4.2 通过集合属性访问对象

Visual FoxPro Application 对象和 Visual FoxPro 中所有容器对象都具有与之相关的一个计数属性和一个集合属性。该集合属性是一个引用集合所包含对象的数组。计数属性是一个数值属性，它表明了所包含对象的数目。

下表列出了对象以及相应的集合属性和计数属性。

表 8-1 对象以及相应的集合属性和计数属性

对 象	集合属性	计数属性
Application	Objects Forms	Count FormCount
FormSet	Forms	FormCount
Form	Objects Controls	Count ControlCount
PageFrame	Pages	PageCount
Page	Controls	ControlCount
Grid	Columns	ColumnCount
CommandGroup	Buttons	ButtonCount
OptionGroup	Buttons	ButtonCount
Column	Controls	ControlCount
ToolBar	Controls	ControlCount
Container	Controls	ControlCount
Control	Controls	ControlCount

这些属性允许通过编程方式使用循环语句来处理所有的或特定的对象，这些对象都包含在例如 Application 的容器对象中。例如，下面的代码将所有表单的 Visible 属性设置为“真”(.T.)：

```
FOR EACH Form IN Application.Forms
    Form.Visible = .T.
ENDFOR
```

8.5 本章小结

对象的链接与嵌入是 Windows 应用程序经常要用到的一项技术，例如在 Mircrosoft Word 中常常会插入 Excel 表格，这在 Visual FoxPro 应用程序中也是常常用到。Visual FoxPro 为用户提供了非常方便的 OLE 开发环境，使

用户快捷地开发出 OLE 应用程序。

本章主要讲述了以下内容：

- 创建 OLE 应用程序
- 使用 ActiveX 控件
- 应用自动服务管理对象
- 派生对象的子类
- 从其他应用程序中控制 Visual FoxPro

第九章　客户／服务器解决方案

客户／服务器应用程序将本地计算机上 Visual FoxPro 的功能与远程服务器所提供的存储和安全性等优点结合在一起。可以在本地建立应用程序的原型，然后使用升迁向导把应用程序迁移到客户／服务器环境。

9.1　设计客户／服务器应用程序

要创建一个成功的客户／服务器应用程序最重要的是一个好的设计方案。本节讨论开发客户／服务器应用程序的方法。

9.1.1　客户／服务器应用程序的设计目标

在设计一个客户／服务器应用程序时，必须首先了解各方面的需求，权衡利弊。如果既想为自己的用户建立一个速度很快、效率极高的应用程序，又想保证应用程序数据的完整性，则应尽可能地利用现有的硬件设备，并考虑程序将来的可扩展性。同时，做为一个 Visual FoxPro 开发人员，还应使开发过程尽可能得简单，并降低开发费用。

达到上述要求的最佳方法是在设计应用程序时把这些要求作为设计目标。下面就给出一些基本的开发技巧和相关技术，以帮助用户设计高性能的客户／服务器应用程序。

9.1.2　高性能的设计

用 Visual FoxPro 创建一个快速、高性能的客户／服务器应用程序涉及到如何利用 Visual FoxPro 引擎的惊人速度的问题。可以考虑使用一些新技术来实现这一目的，例如，使用基于集合的数据访问技术而不是传统的本地定位技术；创建参数化查询仅下载符合需求的数据；将表安排在优化的平台上；平衡使用 Visual FoxPro 和远程存储过程等。

在使用这些新技术之前，必须首先对将要使用的系统进行分析。在设计一个本地或文件服务器应用程序时，必须确定应用程序将要使用或创建的查询、表单、菜单和报表。在设计一个客户／服务器应用程序时，除了这些常规的系统分析之外，还需要针对客户／服务器应用程序的特点进行分析，考虑查询、表单、菜单和报表所使用的数据存放在何处，如何访问这些信息等。此外，还需考虑下列一些问题：

- 应用程序完成后，哪些表存储在远程服务器上？
- 将哪些表作为本地查询表存储可提高效率？
- 需要哪些视图来访问远程数据？
- 在服务器上实施何种商业规则，应用程序怎样和这些规则相互作用？

在确定了客户／服务器应用程序的基本组件之后，就可以开始设计应用程序访问和更新数据的方式了。

1．仅下载所需要的数据

要使客户／服务器应用程序运行速度快并且效率高，应考虑的一个最重要的因素就是尽量减少从服务器上下载的数据。由于客户／服务器应用程序可能要访问远程服务器上潜在的大量数据，使用传统的本地定位技术会导致客户／服务器应用程序运行速度的降低。为了加快执行速度，可以使用基于集合的数据访问技术来过滤将要下载的数据。

（1）有效访问基于集合的数据

远程数据是基于集合的数据：要访问远程数据，必须使用 SQL SELECT 语句从一个大的数据存储地点中选择一个数据集合。建立一个传统的本地应用程序和创建一个客户／服务器应用程序的最重要区别在于传统的 Visual FoxPro 定位技术和基于集合的服务器数据访问技术的区别。

（2）使用传统的定位技术

在传统的本地数据库程序设计中，可以使用 GO BOTTOM 命令查询访问离散的而且往往是大量的数据。通过执行 SET RELATION 命令在两个表之间创建一个临时关系，可以在数据中定位，然后通过 SKIP 命令在相关的记录中移动。

当这种定位记录的方法应用于远程数据时，如果远程数据量巨大，将会导致访问效率低下。例如，假定为了访问远程数据源中的一个大表，首先创建了一个远程视图，然后执行 GO BOTTOM 命令。这时，视图开始从数据源获取所有数据，然后通过网络传送数据，最后加载到本地系统的视图临时表中，这段检索操作需要经历漫长的等待过程。

（3）使用参数化查询

访问远程数据的一种更为有效的方法是每次仅下载所需要的数据，需要取得其他额外的记录或新记录时，再重新执行查询。可以使用参数化的 SELECT 语句下载特定的小数据集合，通过使用 REQUERY（ ）函数可以请求新的数据集以访问新记录。

不能对远程服务器数据发出 GO BOTTOM 命令，因为这样做具有以下缺点：

■ 下载的数据量巨大将对网络资源造成不必要的负担。

■ 处理不需要的数据会使应用程序的性能降低。

■ 潜在地降低了本地临时表中数据的准确性，因为远程数据的改变只有在重新查询的时候才能反映到本地临时表中。

例如，若要建立一个客户／服务器应用程序来查阅特定客户的订单。可创建一个远程视图来访问 Customer 表，然后创建另一个远程视图来访问 Orders 表，并基于 cust_id 字段将该视图参数化。这样，就可以把当前的顾客记录做为 Orders 表的视图的查询参数。

可以使用参数给下载的数据集规定一个范围，使其下载的数据量比较适中。如果一次请求的数据量过少，会导致对远程服务器的查询工作过于频繁，因而降低了程序的性能。而如果请求过多的数据，则会因为下载不需要的数据而浪费时间。

（4）选择最佳的客户／服务器设计方案

下面将举例介绍获得客户／服务器技术优势的方法，以及如何避免因编程方法不当而造成的隐患。第一种方法使用传统的编程方法检索数据，从一个远程数据源中获取所有的数据，下载到本地临时表中，然后用 SET RELATION 命令将这些临时表联系起来。而第二、三、四种方法逐步应用更有效的获取数据的技术，使用“just-in-time”方法有效地限制下载的数据量，从而提高网络的响应时间并获取最新的数据。

（5）使用一种未经优化的客户／服务器策略

对远程数据使用本地数据的定位技术是一种直接的未经优化的客户／服务器解决方式。例如，假定某一远程数据源中存有一千万条客户记录和一亿条订货记录，可以创建一个不考虑效率的应用程序，把所有客户和订货记录下载到本地临时表中。然后对一亿条订货记录索引，在本地临时表中对 Customer 和 Orders 表建立一个临时关系，使用 SKIP 命令来定位记录。这种方法未考虑性能的优化，不过，如果“一”方在本地而“多”方在远程，这种方法可能还是可行的。

（6）筛选多方

对上面这种方法稍加改进，可以得到另一种客户／服务器解决方案。这种方案就是减少关系中“多”方数据的下载，但仍然下载“一”方的所有记录，所以可以跳过一些记录。在这种方案下，需要创建一个远程视图来显示关系中的“多”方，也就是 Orders 表，该视图可用客户的 ID 值作为参数。但这时，仍然需要下载整个 Customer 表。

虽然对 Orders 表创建一个参数化视图相对于下载所有的订单信息来说是一个改进，但是，在下载整个 Customer 表的同时，仍然带来了一些不必要的信息。Customer 表也同样存在过时的危险，因为在系统上的其他用户会对其进行修改。这种方案在关系的“一”方仅包含少量数据的时候还是可行的。

（7）筛选一方

更好一些的客户／服务器解决方案是为所有的远程数据创建远程视图。在 Customer 表的远程视图中，使用 SELECT 语句选择某个地区的客户，这样就可以限制下载到远程视图中的客户记录的数量。然后再对关系中

“多”方 Orders 表创建一个远程视图，并以客户 ID 值为参数。

这种方案检索更少量的记录。使用 SKIP 命令可以在关系的“一”方（即 Customer 视图）中移动。使用 REQUERY（）函数可以访问“多”方（即 Orders）中的新数据。

在这种方案中，既限制（或筛选）了关系中的“一”方又筛选了关系中的“多”方，但仍可以使用 SKIP 命令在筛选过的数据之中定位。如果关系中的“一”方经过筛选以后，仍然能够提供足够的信息满足后续的一些查询，而不需要再次向远程服务器发出请求，那么推荐使用这种方案。

（8）使用主关键字访问一对多关系

效率最高的客户／服务器解决方案是放弃使用开销大的 SKIP 命令，而创建一个表单，让用户输入或选择一个客户 ID，然后把它既做为 Customer 表的远程视图参数，又做为 Orders 表的远程视图参数。

例如，可以创建一个一对多表单，其中“一”方显示客户信息，而表格控件显示关系中的“多”方。该表格可以与表单“一”方中选定的客户 ID 绑定。然后可以把 CURSORSETPROP（）的 MaxRow 属性设置为 1，使用下列代码填充表单的“一”方：

```
SELECT * FROM customer WHERE customer.cust_id = ?cCust_id
```

当用户想查看另一个客户的记录时，必须输入或选择一个新客户 ID。表单用新客户的 ID 值重新查询订单中的数据资源，然后用新的订单数据刷新表格控件。

使用这种技术，应用程序仅仅下载需要的数据。由于减少了下载的数据量，从而提高了网络的响应速度，而且，在显示所请求的信息之前能够重新查询数据，这样总能提供最新的信息。

想用任意主关键字值随机访问一对多关系时推荐使用这种方法。在打开一个表单时，还可以把一些主关键字下载到控件中（比如一个下拉式列表），并且提供一个控件，让用户在需要的时候进行选择，以达到刷新主关键字值列表的目的。

（9）在客户／服务器应用程序中使用数据环境

在表单或表单集中使用远程数据时，请将视图加入到表单或表单集的数据环境中。可以把数据环境的 AutoOpenTables 属性设置为“假”（.F.），这样便可以指定什么时候应用程序用远程数据对视图进行刷新。在调用数据环境的 OpenTables 方法程序之后，可为文本框或其他数据绑定型控件设置 ControlSource 属性，一般是在与表单的 Init 事件相关联的代码之中进行设置。

2．在最佳平台上放置数据

把数据库的数据和其他属性保存在最佳平台上，可以获得最佳的性能。某一特定元素的最佳平台取决于访问和更新该元素的方法。例如，可以将邮政编码一类的服务器表保留一个本地副本，用于一个查阅表，只有当后台的表更改时，才对本地副本进行刷新。

表 9-1 列出一些常用的应用程序元素，以及在何处放置这些元素可以获得最佳性能。

表 9-1　根据平台放置元素

元　素	位　置	类　型	注　释
表	本地	服务器的查阅表的本地副本；或小型的、不经常更改的表	如果远程服务器支持，可以使用时间戳来比较并且选择性地刷新本地表，以使得对本地表的更改与后台源表的更改同步。
	远程	大型的或经常更改的表	
规则	本地	在远程视图中的规则	可以使用 DBSETPROP（）把字段和记录级规则保存在远程视图中。在应用程序中可以使用这些本地规则来检查数据的有效性，然后再把这些数据送到后台做为对远程表的更新。
	远程	远程基表的行级别和列级别的规则	
存储过程	本地	Visual	FoxPro 存储过程
	远程	后台服务器的存储过程	使用 SQLpass-throughh 函数 SQLEXEC（）可调用服务器的存储过程。
事务	本地	Visual	FoxPro 事务

	远程	服务器事务	
触发器	本地视图	在视图上无触发器	
	远程	服务器触发器	

为了减少查找时的网络传输，不仅可以把不常更改的表保存在本地，而且可以把经常更改的查阅表也保存在本地。例如，可以下载公司的客户列表，只有当客户信息更改时才对其进行刷新。

为了做到这一点，可以编写应用程序来比较表的本地副本时间戳和后台数据时间戳（如果远程服务器支持时间戳）。只有当服务器的表更改时，才更新本地副本。也可以在表单中增加一个命令按钮，用于强制实时对表的立刻下载，这样用户一旦需要，便可刷新他们的本地表副本。

3．选择正确的方法

可以使用远程视图或 SQL pass-through，或者联合使用二者，来创建客户／服务器应用程序。可以把两者结合起来获得强大的功能：使用视图来满足数据管理的大部分需要，使用 SQL pass-through 来增强应用程序的功能。

（1）使用视图

可以使用视图做为开发强大的客户／服务器应用程序的核心方法。远程视图是一个强有力的技术，用它可以从一个远程服务器中选择所需的数据并下载到一个本地 Visual FoxPro 临时表中，同时可以查看并更新远程数据。一个视图基本上是使用 SQL SELECT 语句所得到的一个结果集合。

视图具有长期性，因为视图定义保存在数据库中。视图定义具有可以设置的属性，借助这些属性可对活动视图的临时表做进一步定制。视图是创建可更新的结果集合的最佳数据定义工具。

可以使用本地视图建立一个本地原型，然后使用“ Upsizing Wizard ”把本地视图转换为远程视图。

如果应用程序用户想在动态的工作中使用数据，可应用脱机视图。脱机视图便于随身携带数据，允许膝上型或其他便携式计算机用户操作源数据的存储副本，在旅途上他们可以更新副本。当用户与服务器相连时，应用程序可以很方便地将脱机数据的更改合并到源表中。

也可以使用脱机视图技术，以便本地用户“ 脱机 ”操作数据，过一段时间后再合并它们的更新。

（2）使用 SQL pass-through

SQL pass-through 技术使用 Visual FoxPro SQL pass-through 函数对远程服务器进行直接访问。这些函数提供了视图所不能提供的服务器访问和控制能力。例如，可以在远程服务器上完成数据定义，设置服务器属性，并且访问服务器存储过程等。

表 9-2　SQLpass-throught 函数

SQLCANCEL（）	SQLCOLUMNS（）	SQLCOMMIT（）
SQLCONNECT（）	SQLDISCONNECT（）	SQLEXEC（）
SQLGETPROP（）	SQLMORERESULTS（）	SQLPREPARE（）
SQLROLLBACK（）	SQLSETPROP（）	SQLSTRINGCONNECT（）
SQLTABLES（）		

SQL pass-through 是创建只读结果集合和使用其他服务器当地 SQL 语法的最好工具。视图是 SQL SELECT 语句的结果集合，SQL pass-through 与此不同，它允许使用 SQLEXEC（ ）函数向服务器发送符合服务器语法的任何命令。表 9-2 列出了 Visual FoxPro 的 SQL pass-through 函数。

使用 SQL pass-through 技术，可以自己创建临时表。尽管 SQL pass-through 提供了更直接的服务器访问方式，但是相对视图来说，这种访问方式缺乏持久性。视图定义永久地保存于数据库中，而 SQL pass-through 与此不同，它所创建的临时表只存在于当前的工作期内。

（3）视图和 SQL pass-through 的结合

建立一个 Visual FoxPro 客户／服务器应用程序的最有效办法是结合使用视图和 SQL pass-through 技术。由于视图比较容易创建，而且能够提供自动缓冲和更新功能，因此可以使用视图完成大部分的数据管理任务。同时，还可以使用 SQL pass-through 执行远程服务器上的特定任务，例如数据定义和服务器存储过程的创建和执行等。

9.1.3 快速开发应用程序

无论采用何种编程方法，总是需要一种策略使得开发客户／服务器应用程序快速而有效。由于 Visual FoxPro 可以快速地创建程序原型和建立应用程序，因此可以首先为应用程序建立一个本地原型，然后根据远程数据源再升迁程序，分阶段完成程序。如果想在开发过程中就能访问远程数据源，那么可以考虑使用远程视图来建立应用程序原形，以便访问远程数据。

1．使用视图生成原型

开发 Visual FoxPro 客户／服务器应用程序的第一步就是建立一个原型。通过以模块方式建立应用程序原型，可以在开发过程中尽早发现应用程序的哪些方面应该修改和增强。使用这种方法，可以只用小批量的数据精心调整程序，从而避免处理远程异构大数据集时与其复杂规则打交道的麻烦。

（1）使用本地视图创建一个本地原型

客户／服务器应用程序的本地原型也就是一个有效的使用本地视图访问本地表的 Visual FoxPro 应用程序。由于最终的客户／服务器应用程序将使用远程视图访问远程数据，因此可在应用程序原型中也使用视图。通过使用本地视图建立应用程序的原型，又距离最终的应用程序近了一步。

如果在开发阶段不必总访问远程数据，或不想使用远程数据来建立应用程序原型，那么生成一个本地原型是更实际的办法。本地视图访问本地 Visual FoxPro 表，而不是远程数据源表。不过，可以模拟服务器上的数据结构来创建本地数据。使用本地数据代替远程数据是一种快速开发和测试应用程序基本设计方案的理想方法。可以限制选入视图的数据量来加快开发过程。

（2）准备升迁

所谓升迁就是使用与原来的 Visual FoxPro 数据库相同的表结构、数据和内在的许多其他属性，在远程服务器上创建一个新的数据库。利用升迁可以把现有的 Visual FoxPro 应用程序转变为一个客户／服务器应用程序。

如果设计的应用程序准备在最后升迁，那么在设计应用程序的结构和编写模块时就需要仔细筹划，为如何发挥远程数据源的最佳性能做考虑。

（3）使用远程视图建立程序原型

在开发客户／服务器应用程序的时候，如果能够访问一个远程数据源并且希望直接使用远程数据，可以使用远程视图建立程序原型。当使用远程视图来生成原型时，便可跳过升迁这个步骤，因为数据已经存放在远程服务器上，并且已经可以使用远程视图来访问这些数据。

2．实现客户／服务器应用程序

通过分阶段地完成应用程序原型，可以简化对应用程序的测试和调试工作。在完善一个应用程序原型过程中，可以分阶段、有系统地加入多用户功能，并把数据转移到远程数据源，然后逐个模块地测试和调试应用程序。

在实现应用程序时，可以借助 SQL pass-through 技术，使用服务器本地语法，访问服务器的特有功能，例如，服务器存储过程等。

3．优化应用程序

如果程序对远程数据的操作功能已经完成，并且已经通过了测试和调试阶段，那么就可以开始考虑如何对整个应用程序的速度和性能进行优化。

9.1.4 确保开发的准确性和数据的完整性

可以把 Visual FoxPro 的数据有效性规则和存储过程以及数据源的数据有效性规则和存储过程结合起来，使生成的客户／服务器应用程序能够有效地维护数据完整性。

1、维护数据完整性

可以创建远程服务器有效性规则的本地版本，为用户提供一些信息。例如，有可能因为输入的数据破坏了某些服务器关系的完整性，或者破坏了数据有效性规则，导致不允许将某些更新数据发送给后台。

（1）在远程视图和脱机视图中使用 Visual FoxPro 规则

可以在远程视图和脱机视图中创建字段级和记录级规则，在数据发送给远程数据源之前，在本地检查输入数据的有效性。由于这些视图规则的目的在于防止发送的数据违反服务器的数据完整性规则，由此更希望将数据源的规则直接复制到远程视图的规则中。使用 DBSETPROP（ ）函数可以为视图创建规则。

可以在远程视图中创建一个本地有效性规则，该规则调用远程服务器存储过程，并且把需要检查其有效性的值做为参数发送给服务器。但是，使用远程存储过程会增加数据输入时的处理时间。

（2）使用服务器规则

可以选择服务器上的规则来检查数据的有效性。出现错误时，出错处理程序可以调用 AERROR（ ）函数来获取信息，包括错误信息号、远程出错信息的文本以及有关此错误的连接句柄等。

（3）使用服务器触发器

虽然可以在本地表中创建 Visual FoxPro 触发器，但在视图中却不能。不过可以在远程数据源中使用触发器。服务器触发器可以用来处理二级数据更新，比如，级联更新或删除。使用服务器触发器处理二级更新比从 Visual FoxPro 应用程序向远程服务器发送多个命令更有效。

2、防止数据丢失

Visual FoxPro 和大部分的远程数据源都提供了事务处理日志功能，用来防止数据丢失。

可以将 Visual FoxPro 事务用于本地原型和本地数据处理。而服务器事务可用于远程数据的更新、插入和删除。

9.2 升迁 Visual FoxPro 数据库

在完成客户／服务器的设计工作之后，就可以开始构造并升迁一个本地原型。本地原型是应用程序的一种工作模式，它使用 Visual FoxPro 的表、视图和数据库来表示以后将放在远程服务器上的数据。可以借助“Upsizing Wizard”将这些数据库、表和视图从本地系统迁移到远程 SQL Server 或 Oracle 服务器上。

9.2.1 构造原型的目标

在使用 Visual FoxPro 构造应用程序的原型时，可以充分利用可视化编程的各种工具，如表单、向导、生成器、设计器和项目管理器等，快速有效地开发出一个正常工作的应用程序。如果最终目标是使这个应用程序成为一个客户／服务器模式的程序，那么选择一个固定的原型进行程序开发可以取得事半功倍的效果。

1．减少开发时间

通过建立快速的程序原型，不必访问远程服务器以在服务器上创建表和数据库，就可以在本地机上简单快捷地对应用程序的设计方案和本地结构进行精心调整。此外，还可以在一个规模较小的数据集上测试并调试应用程序，并对程序的用户界面做出相应的修改。由于程序的结构总是对系统要求较低，这样就避免了为测试程序原型而对远程数据进行的重建、重索引以及重连接等工作，从而节省了开发时间。

2．降低开发费用的同时满足用户要求

由于本地原型存放在本地 PC 机上，因而在应用程序开发周期的早期就可以向最终用户演示应用程序的工作模式。让用户尽早了解工作的进展有助于增强用户对程序开发的信心，相信设计者交付合格产品的能力。与此同时，还能在远程服务器投入力量进行下一步的工作之前，倾听用户意见，掌握用户对程序用户界面以及报表等程序组件的反馈信息。

用户在看到、使用程序原型之后，会提出一些修改意见，并对程序中一些有待改进和完善的地方提出建议。设计者可以根据这些意见和建议，做出相应的修改，然后再次提交给用户，就这样不断地重复，直到双方都对设计方案满意为止。这时，原型程序也就达到了为最终的客户／服务器应用程序奠定基础的目的。

9.2.2 构造应用程序的本地原型

在建立应用程序的本地原型时，可以从草图开始，也可以将一个已有的 Visual FoxPro 应用程序转换为客户／服务器应用程序。创建客户／服务器应用程序的本地原型与开发其他 Visual FoxPro 应用程序的主要差别在于使用本地视图和表来表示以后将要升迁的数据。

若要构造并升迁一个本地原型，应注意以下几点：

- 创建应用程序时，使用本地视图和本地表来表示以后将要升迁到远程服务器的数据。
- 在应用程序的表单和报表的数据环境中使用本地视图。
- 利用 Visual FoxPro 6.0“SQL Server Upsizing Wizard（升迁向导）”或“Oracle Upsizing Wizard”升迁本

地视图和表。

在“Set Upsizing Options（设置升迁选项）”步骤中，可在“改变为本地化”区域里，选择“将视图重新定向到远程数据”。

当选择了此选项后，升迁向导将向选定的本地表复制到远程服务器上，然后重新定向本地视图以使用可用的远程数据。

9.2.3 使用升迁向导

Visual FoxPro 提供了两个升迁向导：“SQL Server 升迁向导”和“Oracle 升迁向导”。这两个升迁向导可以创建一个 SQL Server 数据库或 Oracle 数据库，实现 Visual FoxPro 数据库中各表的功能。还可以重定向 Visual FoxPro 视图，使其使用新建的远程数据而不是本地数据。利用“Upsizing Wizard”可以实现以下功能：

- 将本地数据转移到远程服务器上。
- 将本地基表和本地视图转换为远程基表和远程视图。

将本地应用程序移植为客户／服务器应用程序。

尽管“Upsizing Wizard”只访问 SQL Server 或 Oracle 服务器，但是实际上可以为任何远程 ODBC 数据源创建客户／服务器应用程序。对于非 SQL Server 和非 Oracle 服务器，可以使用 SQL pass-through 函数创建远程表，然后使用 Visual FoxPro 来创建远程视图访问这些服务器上的表。

9.2.4 升迁到 SQL Server 上

在运行“Upsizing Wizard”之前，必须在客户和服务器端都做好准备。

1．准备 SQL Server 端

在升迁之前，必须确保对服务器有必要的访问权限，并需要对数据库的大小做出估计，然后检查服务器是否有足够的磁盘空间。此外，升迁到多个磁盘或设备上还需要特殊的准备工作。

（1）检查空闲的磁盘空间

确保在服务器上有足够的磁盘空间。如果“Upsizing Wizard（升迁向导）”遇到服务器上磁盘空间不足的问题，它将停止，并将未完成的数据库及所创建的所有设备遗留在服务器上。可以利用 SQL Server 管理工具清除这些设备、数据库和表。

（2）设置 SQL Server 数据库的权限

为了运行“Upsizing Wizard”，必须在准备接收升迁的 SQL Server 上拥有足够的权限。拥有哪种权限取决于需要完成的任务。

（3）若要升迁到一个现有数据库上，需要具有 CREATE TABLE 和 CREATE DEFAULT 权限。

若要创建一个新数据库，需要在主数据库的系统表上拥有 CREATE DATABASE 和 SELECT 权限。

若要创建新设备，必须是一位系统管理员。

（4）估计 SQL Server 数据库和设备的大小

在创建一个新数据库时，“Upsizing Wizard”会要求为数据库和日志选择设备，并要求给出数据库和设备的大小。

（5）估计 SQL Server 服务器的大小

创建一个数据库时，可在一个或多个设备上预留固定大小的磁盘空间。数据库并不会用到全部的空间，指定数据库大小仅仅用来限制数据库的可增长规模，避免遇到磁盘空间不足的情况。

可以在创建一个 SQL Server 数据库之后，可以使用 SQL Server 中 ALTER DATABASE 命令

若要估计数据库所需空间的大小，可将要升迁的 Visual FoxPro .dbf 文件的大小，加上将增加的新的 SQL Server 数据库的大小，其和即为数据库空间的估算值。一般来讲，每兆字节的 Visual FoxPro 数据在 SQL Server 上需要至少 1.3 到 1.5 兆字节。

如果服务器上有足够的磁盘空间，不妨预留两倍于 Visual FoxPro 表的空间。这样就能确保“Upsizing Wizard”有足够的空间升迁数据库，并为未来的增长预留空间。如果希望以后添加大量数据到此数据库中，可以再多留一些空间。

（6）估计 SQL Server 设备的大小

SQL Server 的所有数据库和日志都放在设备上。设备是一个物理文件，同时又是一个存放数据库和日志的逻辑位置。在创建设备时，SQL Server 实际上创建了一个文件，因此需要保留一定数量的磁盘空间供其使用。

“Upsizing Wizard”可以显示现有 SQL Server 设备的自由空间大小，升迁时所选择设备的自由空间大小应大于或等于估计的数据库大小。

如果现有的设备都没有足够的自由空间，可以用“Upsizing Wizard”创建新设备。新设备必须大于或等于估计的数据库大小。如果可能，应使设备占用的空间大于数据库占用的空间，这样以后可以继续扩展此数据库，或者将其他数据库或日志放在此设备上。

应当注意，设备大小不能修改，因此必须一开始就留出足够的空间。

（7）使用多个 SQL Server 磁盘或设备

在大多数情况下，“Upsizing Wizard”对 SQL Server 设备提供足够的控制权。但是，如果服务器有多个磁盘，或者想把数据库或日志放在多台设备上，那么可能需要在运行“Upsizing Wizard”之前创建设备。

（8）具有多个物理磁盘的服务器

如果服务器具有多个物理硬盘，您可能希望将数据库放在一个磁盘上，而数据库日志放在另一个磁盘上。这样，在某一磁盘损坏的情况下，还有可能对数据库进行恢复。

“Upsizing Wizard”允许新建多台设备，但只能在一个物理磁盘上，而且是主数据库设备所在的磁盘上创建。

若要将数据库和日志放在不同的磁盘上，应确保在两个磁盘上的设备都足够大，必要时可创建新设备。然后再运行“Upsizing Wizard”。

（9）将数据库或日志放在多台设备上。

SQL Server 允许数据库和日志分布在多台设备上。但是“Upsizing Wizard”只允许为数据库指定一台设备，为日志指定另一台设备。

若要为数据库或日志指定多台设备，可以设定这些设备（而不是其他设备）为默认设备。然后运行‘SQL Server 升迁向导”，并设定数据库或日志的设备为默认设备。

如果新的 SQL Server 数据库或日志大小不足以占用所有的默认设备，SQL Server 将只使用能容纳数据库或日志的设备。

2．准备客户端

在升迁之前，必须能够通过 ODBC 数据源或命名连接访问 SQL Server。此外，在运行 SQL Server “Upsizing Wizard”之前还必须备份 Visual FoxPro 数据库。

创建 ODBC 数据源或命名连接的方法如下：在新建远程数据库时，要确保 Visual FoxPro 数据库中已有一个 ODBC 数据源或命名连接可以访问 SQL Server，此数据库将升迁到此 SQL Server 上。如果没有选择一个命名连接或数据源，将不能使用“Upsizing Wizard”进行处理，所以在开始升迁过程之前，需要建立一个合适的数据源或命名连接。

3．备份数据库

在升迁之前，最好能备份数据库（.dbc、.dct 和 .dcx 文件）。虽然“Upsizing Wizard”不修改 .dbf，但是它会对 .dbc 进行操作，例如，不时以表的形式直接打开 .dbc，或在创建新的远程视图时，通过重命名表或视图间接地打开 .dbc。如果在升迁前备份了数据库，就可以用原来的备份覆盖升迁之后的 .dbc、.dct 和 .dcx 文件，从而将数据库恢复到原来状态。

4．关闭表

“Upsizing Wizard（升迁向导）”会尝试以独占方式打开待升迁数据库中的所有表，如果其中有任何表已经以共享方式打开，“Upsizing Wizard”会将其关闭并以独占方式重新打开。以独占方式打开表可以避免其他用户修改正在导出的数据。如果有某一表不能以独占方式打开，“Upsizing Wizard”会显示相应的信息，并且此表不能被升迁。

5．启动升迁向导

在创建一个 ODBC 数据源并完成了客户端和服务器端两方面的准备工作之后，就可以开始升迁工作了。

若要启动 SQL Server 升迁向导，从“Tools”菜单中选择“Wizards（向导）”，然后选择“Upsizing（升迁）”。从“Wizard Selection（向导选取）”对话框中，选择“SQL Server Upsizing Wizard”。如图 9-1 所示。

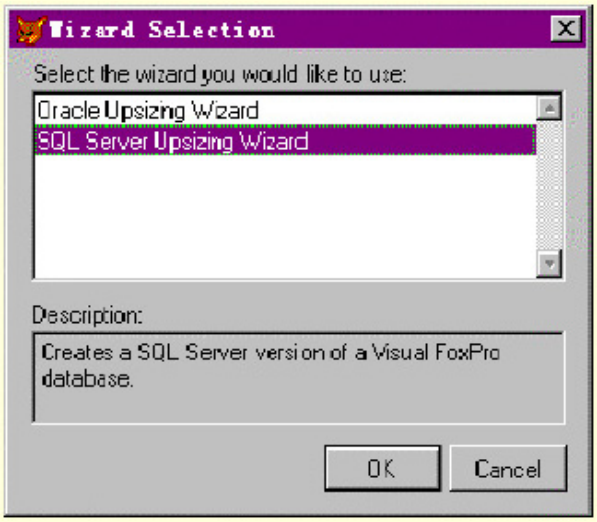


图 9-1 升迁向导选取对话框

根据向导的提示，逐步进行各种工作。

可以在任何时候选择“取消”按钮，退出“Upsizing Wizard”。在选择“完成”按钮之前，向导不会对服务器执行任何操作。

在升迁工作准备就绪后，选择“完成”按钮。

选择“完成”按钮后，“Upsizing Wizard”开始将数据库迁移到服务器上。

“完成”按钮在提供了升迁所需的足够信息后即可使用。如果在所有向导指示步骤完成之前就选择了“完成”按钮，“Upsizing Wizard”将采用剩下步骤的默认设置。

9.2.5 升迁到 Oracle 上

Oracle 升迁向导和 SQL Server 升迁向导的操作过程类似。

建立与 Oracle 服务器相联的命名连接或者 ODBC 数据源，同时完成了在客户机和服务器上一些必要的准备以后，就可以开始升迁。

若要启动 Oracle 升迁向导，从“Tools”菜单上，选择“Wizards”，然后再选择“Upsizing”。在“Wizard Selection（向导选取）”对话框中，选择“Oracle Upsizing Wizard”。按照向导屏幕中的提示进行。

任意时候都可以选择“取消”来退出向导，在没有选择“完成”之前，向导不会对服务器上进行任何操作。

当准备好升迁时，请选择“完成”。

当提供了升迁所需要的基本信息之后，就可以选择“完成”这一步。在没有逐步走完所有的向导步骤之前选择“完成”，“Oracle 升迁向导”会采用剩余步骤的默认值。

9.3 实现客户／服务器应用程序

无论通过创建然后升迁本地原型，还是使用远程视图根据远程数据开发客户／服务器应用程序，都可以访问远程服务器数据库中存储的大量可用数据。还可以使用远程服务器上性能完善的安全性管理和事务处理能力。在用远程视图执行主要的数据管理任务时，可以使用 SQL pass-through （SPT）技术在服务器上创建对象，运行服务器存储过程，使用当地服务器语法执行各种命令，从而大大增强应用程序的性能。

9.3.1 使用 SQL pass-through 技术

客户／服务器应用程序可以使用下列两种方法访问服务器数据：

- 远程视图
- SQL pass-through

远程视图提供了访问和更新远程数据的最简单、最通用方法。可以利用“Upsizing Wizard”在数据库中自动

创建远程视图，也可以在升迁以后使用 Visual FoxPro 来创建远程视图。

使用 SQL pass-through 技术可以直接把 SQL 语句发送给服务器。由于这些 SQL 语句在后台服务器上运行，因而能够在很大程度上提高客户／服务器应用程序的性能。下页中的表 9-3 是远程视图和 SQL pass-through 的比较情况。

SQL pass-through 技术与远程视图相比，具有如下优点：

■ 可以使用服务器的特有功能，比如存储过程和基于服务器的内部函数等。

■ 不但可以使用服务器所支持的 SQL 扩展功能，而且可以进行数据定义、服务器管理和安全性管理。

■ 对 SQL pass-through 的更新、删除和插入语句拥有更多控制权。

■ 对于远程事务拥有更多控制权。

Visual FoxPro 可以处理返回多个结果集合的 SQL pass-through 查询。

SQL pass-through 查询也有缺点：

■ 一个 SQL pass-through 查询能够快速得到远程数据结果，但是在默认情况下得到的结果不能更新，它保存在一个活动的视图临时表中。要使得此临时表可更新，必须用 CURSORSETPROP（ ）函数设置它的属性。与此相反，在更新远程数据之前，可更新的远程视图不要求设置属性。因为属性设置保存在数据库的视图定义中。

■ 必须在命令窗口或程序中直接输入 SQL 命令，而不能使用图形化的视图设计器。

■ 必须创建并管理对数据源的连接。

表 9-3 远程视图和 SQL pass-through 技术的比较

远程视图	SQLpass-through
基于 SQLSELECT 语句。	基于当地服务器的任何 SQL 语句，允许数据定义语句或者服务器存储过程的执行。
在设计时可以作为控件的数据源。	不能作为控件的数据源。
不能对数据源执行 DLL 命令。	提供了可以对数据源执行 DLL 命令的方法。
取一个结果集合。	取一个或多个结果集合。
提供内部连接管理。	需要明确的连接管理。
为更新、插入和删除提供内部的默认更新信息。	不提供默认更新信息。
隐含地执行 SQL 语句并获取数据。	需要明确地执行 SQL 语句，并控制结果获取方式。
不提供事务处理。	提供明确的事务处理。
在数据库中永久保存属性。	基于工作期属性，为 SQLpass-through 临时表提供临时属性。
在执行 SQL 时采用异步的逐步获取方式。	全面支持编程方式的异步获取。

无论远程视图还是 SQL pass-through 都可以查询和更新远程数据。事实上，在许多应用程序中，远程视图和 SQL pass-through 两种技术常常同时使用。

1．使用 SQL pass-through 函数

在使用 SQL pass-through 连接到一个远程 ODBC 数据源时，需要首先调用 Visual FoxPro 函数 SQLCONNECT（ ）创建一个连接。然后，使用 SQL pass-through 函数把命令发送给远程数据源以执行命令。

使用 Visual FoxPro SQL pass-through 函数时要注意以下几点：

（1）确保系统能与数据源相连。可以使用像 ODBC Test 这样的 ODBC 工具进行检查。

（2）用 SQLCONNECT（ ）或 SQLSTRINGCONNECT（ ）函数建立与数据源的连接。

例如：如果要把 Visual FoxPro 连接到 SQL Server 数据源 sqlremote 上，那么必须以系统管理员（用户标识 sa）身份注册，并给出密码 secret，即是用如下命令：

```
nConnectionHandle = SQLCONNECT（'sqlremote','sa','secret'）
```

也可以使用 SQLCONNECT（ ）函数实现一个命名连接。

（3）使用 Visual FoxPro SQL pass-through 函数将数据检索到 Visual FoxPro 临时表中，然后使用标准的 Visual FoxPro 命令和函数处理这些检索到的数据。

例如，可以使用如下命令查询 authors 表并且浏览查询结果的临时表：

```
? SQLEXEC（nConnectionHandle,"select * from authors","mycursorname"）
BROWSE
```

（4）使用 SQLDISCONNECT（ ）函数切断与数据源的连接。

2．Visual FoxPro 中 SQL pass-through 函数

下表给出了 Visual FoxPro 中与远程数据源协同工作的 SQL 函数，并按所完成的任务对其分组。

表 9-4 Visual FoxPro SQL pass-through 函数

任 务	函 数	目 的
连接管理	SQLCONNECT（）	为 SQLpass-through 操作连接一个数据源
	SQLSTRINGCONNECT（）	使用 ODBC 连接字符串语法连接数据源
	SQLDISCONNECT（）	断开一个对 ODBC 数据源的连接，使指定的连接句柄失效
SQL 语句的执行和控制	SQLCANCEL（）	在一个活动连接上，取消异步执行的 SQL 查询
	SQLEXEC（）	在一个活动连接上执行 SQLpass-through 查询；或者返回已生成的结果集合的数目，如果 SQLEXEC（）仍在执行（异步处理），则返回 0 值
	SQLMORERESULTS（）	把另一个结果集合放到临时表中。如果创建结果集合的语句仍在执行，则返回 0 值
	SQLPREPARE（）	预编译数据源上的 SQL 语句,同时带有 Visual FoxPro 参数；也就是说，在 SQL 命令中，将所有参数保存为实际的参数表达式
	SQLCOMMIT（）	申请提交一个事务
	SQLROLLBACK（）	申请回滚一个事务
数据源信息	SQLCOLUMNS（）	将列名以及每列的信息存入一个临时表，如果成功，则返回 1；如果仍在执行，则返回 0
	SQLTABLES（）	将数据源中的表名存入一个临时表。如果成功，则返回 1。如果仍在执行，则返回 0
其他控制	SQLGETPROP（）	获得活动连接的属性
	SQLSETPROP（）	设置活动连接的属性

如果 SET ESCAPE 设置为 ON，则 SQLEXEC（ ）、SQLMORERESULTS（ ）、SQLTABLES（ ）和 SQLCOLUMNS（ ）语句可以通过按 ESC 键以同步方式取消。此外，也可以通过发出 SQLCANCEL（ ）命令以异步方式取消这些语句。其他 SQL pass-through 语句都以同步方式工作，不能被中断。

3．使用 SQL pass-through 函数访问服务器存储过程

使用 Visual FoxPro SQL 的传递技术，可以建立和执行在远程服务器上的存储过程。存储过程可以极大地增强 SQL 的有效性、灵活性和功能性，也能够大大提高 SQL 语句和批处理的性能。许多服务器都提供了存储过程，以便对服务器的数据库对象进行定义和操作，而且实施服务器系统和用户管理。

若要调用一个服务器存储过程，可使用带有该存储过程名称的 SQLEXEC（ ）函数。

例如，下面的代码将显示 SQL Server 上名为 sp_who 的存储过程的调用结果。该 SQL Server 使用一个活动的连接与名字为 sqlremote 的数据源相连。

```
nConnectionHandle = SQLCONNECT（'sqlremote'）
? SQLEXEC（nConnectionHandle, 'use pubs'）
? SQLEXEC（nConnectionHandle, 'sp_who'）
BROWSE
```

如果执行了一个存储过程，该过程包含了本地服务器的语法：即 SELECT 语句，则每个结果集将分别返回到一个独立的 Visual FoxPro 临时表中。使用这些临时表，可以从一个服务器存储过程向 Visual FoxPro 客户机上返回值或者参数。

若要返回多个结果集，用 SQLEXEC（ ）函数来选择使用服务器本地语法的多个结果集。

例如，下面的代码建立和执行了一个 SQL 服务器存储过程 my_procedure，该过程返回三个 Visual FoxPro 的临时表：sqlresult、 sqlresult1 和 sqlresult2：

```
=SQLEXEC（nConnectionHandle,'create procedure my_procedure as ;
```

```
        select * from sales; select * from authors;
        select * from titles')
=SQLEXEC(nConnectionHandle,'execute my_procedure')
```

由于在建立存储过程时，服务器对每个存储过程进行编译，所以在建立存储过程阶段，可以看到任一服务器语法错误。当执行存储过程时，服务器将依次执行编译过的 SQL 语句（就像 Visual FoxPro 中的程序一样），然后 Visual FoxPro 按照执行的顺序，在存储过程内从每个 SQL 语句中取出每个结果集。

结果集和错误按照接收的顺序返回。当遇到一个错误时，处理过程就会停止。例如，假设服务器正在执行一个包含四条语句的存储过程，当在执行第三条语句时发生了“运行时刻错误”，那么，只能接收前两个结果集，以及处理第三个结果集时的错误。在返回错误信息之后，处理过程中断，不会继续处理第四个结果集。可以使用 AERROR（）函数得到最近发生的错误信息。

只有通过 Visual FoxPro SQL pass-through 函数，才能在 Visual FoxPro 中执行服务器存储过程。视图不支持服务器存储过程，因为每个视图在它的 SQL 定义中包含一个显式的 SQL SELECT 语句。

4. 向数据源传递 SQL 语句

可以使用 SQLEXEC（）函数将一条 SQL 语句不经任何解释发送到数据源。在最简单的情况下，SQLEXEC（）函数第二个参数中包含的所有字符串都不经过任何解释便传递给数据源。这样，就可以执行数据源当地的任何 SQL 语句。

SQLEXEC（）函数还可以用来创建参数化查询，或将 SQL 的 ODBC 扩展传递到数据源。

5. 建立参数化查询

如同使用视图设计器或以编程方式创建参数化视图一样，也可以创建 SQL pass-through 的参数化查询。

若要用 SQL pass-through 创建一个参数化查询，用 SQLEXEC（）函数发送一个 SQL 字符串，在字符串中包含 Visual FoxPro 参数，在参数之前需要加上问号（?）。提供的参数作为 Visual FoxPro 表达式进行求值，求得的值将作为该视图 SQL 语句的一部分被发送。如果求值失败，Visual FoxPro 会提示输入参数的值。

如果参数是一个表达式，请用括号将该参数括起来。这样可以保证整个表达式做为参数的一部分进行求值。

例如，如果在远程服务器上的 Testdata 数据库中有一个 customer 表，下列代码创建一个参数化查询，只查看那些国家/地区名与 ?cCountry 参数值相匹配的客户：

```
? SQLEXEC(1,'SELECT * FROM customer WHERE customer.country = ?cCountry')
```

如果想提示用户输入参数值，可把参数表达式用引号括起来。

ODBC 数据源不接受下列位置中的参数：

- 在 SELECT 字段或表清单中
- 作为比较谓词的两个表达式。
- 作为二元操作符的两个操作数。

在一个 SELECT 语句的 WHERE 或 HAVING 子句中，ODBC 数据源不接受下列情况的参数：

- 作为 BETWEEN 谓词的第一和第二个操作数。
- 作为 BETWEEN 谓词的第一和第三个操作数。
- 作为 IN 谓词的表达式和第一个值。
- 一元操作符 + 或 - 的操作数。
- SET 函数的参数。

6. 使用 SQL Server 的输入/输出参数

可以使用输入/输出参数在 Visual FoxPro 和 SQL Server 之间传递参数。只有在使用 SQL pass-through 功能时才可以使用输入/输出参数，在视图中则不能使用它们。

表 9-5 提供了一个示例，介绍如何使用输入/输出参数来实现从 Visual FoxPro 向一个 SQL Server 存储过程传递值，并把结果返回到一个 Visual FoxPro 变量中。

表 9-5 在一个 SQL Server 存储过程中使用输入/输出参数

代 码	注 释
resultCode = SQLExec（connHand,; "CREATE PROCEDURE sp_test; @mult1 int, @mult2 int, @result int; OUTPUT AS SELECT ; @result = @mult1 * @mult2"）	建立一个名字为 sp_test 的存储过程，该过程将两个变量相乘（mult1 和 mult2），然后将结果保存在变量 result 中
outParam = 0	建立一个 Visual FoxPro 变量，接收从 SQL Server 向 Visual FoxPro 传递的输出参数值
ResultCode = SQLExec（connHand, ; "{CALL sp_test （2, 4, ?@outParam）}"）	执行该 SQL Server 存储过程，传递参数值‘2’和‘4’，准备在存储过程中进行乘法运算
? "outParam =", outParam && the value is 8	显示结果参数的值

输出参数的语法是：

?@parameter_name

在执行输入/输出参数时，需要定义包含在 SQL pass-through 命令中的 Visual FoxPro 变量，才能在 SQL 语句中使用此变量。要想成功地使用输入/输出参数发送和接收信息，必须定义一个带有输出类型的存储过程参数，该参数返回一个值。

例如，如果存储过程参数为 @result，则必须给 @result 分配一个输出类型（如 INT）并为它赋值。

一个输出参数 （@parameter_name）表达式，该表达式对一个已有的 Visual FoxPro 变量进行计算。

例如，如果输出参数表达式是 ?@outParam，则应用程序必须已经定义了 Visual FoxPro 变量 outParam。

如果在 Visual FoxPro 或存储过程中没有使用输出参数，或者没有定义一个接收返回值的 Visual FoxPro 变量，则该 Visual FoxPro 参数的值将不变。

Visual FoxPro 使用下面的规则来转换返回的变量值：

- 浮点数据类型（N、F、B）变量转换为数值型 N。
- 显示的长度设置为 9。
- 小数点格式设为当前工作期的设置。小数点格式只影响默认的显示格式，并不影响小数的精度。
- 日期和时间变量（D、T）转换为时间变量（T）。

在输入/输出参数中，不能使用备注、通用、图片或者 NULL 数据类型。

如果应用程序使用临时表字段作为参数，那么 Visual FoxPro 会试图将结果转换回原始字段的数据类型。

只有在获取了语句的最后一个结果集之后，输入/输出参数才可用。这意味着，只有在发生下列情况之后，输入/输出值才返回给 Visual FoxPro： SQLEXEC（ ）以批处理方式返回 （1）或者 SQLMORERESULTS（ ）以非批处理方式返回 （2）。

如果 SQLEXEC（ ）语句需要多个结果集，那么只有从数据源中获取最后一个结果集之后，输出参数才能保证有效。

7．建立与远程数据的外部连接

如果服务器支持外部连接的话，就可以使用服务器本地语法，使用 SQL pass-through 实现和远程数据的外部连接。无论是否找到满足条件的记录行，外部连接都将一个或多个表的信息合并起来。

若要在服务器上实现外部连接，按照服务器的外部连接语法，使用 SQLEXEC（ ）函数。

例如，下面的代码使用 Visual FoxPro SQL 的传递函数 SQLEXEC（ ），显示与 SQL Server 的外部连接结果，该 SQL Server 使用了活动的命名连接 sqlremote：

```
? SQLEXEC（sqlremote, 'select au_fname, au_lname, pub_name ;
from authors, publishers ;
where authors.city *= publishers.city'）
BROWSE
```

8．对 SQL 实施 ODBC 扩展

可以使用 SQLEXEC（ ）函数来实现 SQL 的 ODBC 扩展，方法是把 SQL Access Group 的标准语法或扩展的 escape 语法包含到 SQL 语句之中。

如果服务器支持外部连接，可以使用 ODBC 的 escape 语法，用 SQL pass-through 功能实现和远程数据的

外部连接。无论是否找到满足条件的记录行，外部连接都将一个或多个表的信息结合起来。

使用 ODBC escape 子句的外部连接的语法是：

```
{oj outer-join expression}
```

下面的例子中，将建立一个 544 号项目的雇员名字和部门的结果集：

```
SELECT employee.name, dept.deptname;
    FROM {oj employee LEFT OUTER JOIN dept;
            ON employee.deptid = dept.deptid};
    WHERE employee.projid = 544
```

9. 用 SQL pass-through 管理连接

在创建一个远程视图时，需要选择一个 ODBC 数据源名称或者一个连接名称，作为活动视图与远程服务器之间的数据通道。要用 SQL pass-through 直接访问远程数据，必须拥有活动连接的句柄。句柄实际上是一个指向某一对象的值。在这里，句柄指向的是一个数据源连接。为了获得一个句柄，可以用 SQLCONNECT（ ）或 SQLSTRINGCONNECT（ ）函数来请求一个与数据源的连接。如果连接成功，则应用程序得到连接句柄，在以后的 Visual FoxPro 调用中就可以使用此句柄。

应用程序可以对同一数据源请求多个连接。也可以通过对每一数据源都请求一个连接，同时处理多个 ODBC 数据源。如果想减少使用的连接数目，可以设置远程视图共享相同的连接。使用 SQLDISCONNECT（ ）函数可以切断与数据源的连接。

Visual FoxPro 依赖于对 ODBC 数据源的定义来连接一个数据源，这个定义存储在 Windows ODBC.INI 文件或者 Windows NT 的注册表中。如果更改一个数据源的名称或注册信息，请注意这些更改可能影响到使用该数据源的应用程序与所需远程服务器的连接。

客户／服务器环境是在每次打开 Visual FoxPro 时建立起来的，此环境的生存期是 Visual FoxPro 的工作期，每次关闭 Visual FoxPro 时，此环境随之消失。客户／服务器环境包括：

- 可供新连接作为参考原型的全局属性。
- 在特定连接之外出错所产生的错误值。

可以使用第 0 句柄，也就是“环境”句柄来进行全局属性的设置。要设置属性，可以使用 SQLSETPROP（ ）函数，此函数不但能够控制连接环境的默认属性设置，而且可以控制单个连接的属性。SQLSETPROP（ ）函数设置属性的方法，对于环境连接和单个连接来说都是一致的：

- 对于只有两种设置的属性，可以在 eExpression 中使用一个逻辑值 （.F. 或 .T.）。
- 属性名可以用尽可能短的缩写形式，只要保证无畸义就行。例如，可以用“Asynchronous”、“Asynch”或“A”来表示“Asynchronous”属性。属性名不区分大小写。

在创建一个新连接时，该连接将继承默认连接的属性值。如果不想使用这些默认值，可以使用 SQLSETPROP（ ）函数来更改。

可使用 SQLGETPROP（ ）函数以及相应的连接句柄来查看一个连接的当前属性设置。表 9-6 列出了可用 SQLGETPROP（ ）函数访问的连接属性。

表 9-6 Visual FoxPro 连接属性

目 的	所需使用的属性	用 途
显示用于创建活动连接的信息	ConnectString	注册连接字符串
	DataSource	由 ODBC 定义的数据源名称
	Password	连接密码
	UserID	用户标识符
在共享连接上工作	ConnectBusy	一个共享连接忙时为“真”（.T.），否则为“假”（.F.）
控制界面显示	DispLogin	控制何时显示 ODBC 注册对话框
	DispWarnings	控制是否显示非致命错误信息

续 表

目 的	所需使用的属性	用 途
控制时间间隔	ConnectTimeout	指定在返回一个连接超时错误之前的等待时间（以秒为单位）
	IdleTimeout	指定空闲超时设定（以秒为单位），指定活动连接在指定的时间间隔之后变成不活动
	WaitTime	控制以毫秒计的时间量，在经过这段时间之后，VisualFoxPro 检查 SQL 语句是否已执行完毕
	QueryTimeout	控制返回一个一般超时错误之前的等待时间（以秒为单位）
管理事务处理	Transactions	决定连接如何管理远程表上的事务处理
控制将结果集合放入到视图临时表中的方式	Asynchronous	指定结果集合是同步返回（默认值）还是异步返回
	BatchMode	指定 SQLEXEC（）是一次返回所有结果集（默认值），还是由 SQLMORERESULTS（）逐个返回
	PacketSize	指定连接使用的网络包的大小
显示内部 ODBC 句柄	ODBChdbc2	内部 ODBC 连接句柄，可被外部库文件（.fll 文件）用来调用 ODBCAPI 函数
	ODBChstmt2	内部 ODBC 语句句柄，可被外部库文件（.fll 文件）用来调用 ODBCAPI 函数

在人工事务处理模式，连接将保持活动状态。

如果连接不活动，ODBChdbc 和 ODBChstmt 的值不再有效。在用户库中不能自由改变这些值。

可以使用 SQLSETPROP（ ）设置连接属性及其默认设置。用 0 句柄在 Visval FoxPro 环境中设置的值在每一个后续的连接或附件中被作为原型或默认值。

要查看当前环境属性设置，可使用 SQLGETPROP（ ），并将 0 作为句柄值。

下面的例子将在屏幕上显示当前环境的 WaitTime 属性设置：

```
? SQLGETPROP（0, "WaitTime"）
```

如果把 DispWarnings 属性设置为“真”(.T.)，Visual FoxPro 将从设置时刻开始把所有环境错误显示出来，而且对于以后创建的连接也都设置 DispWarnings 属性为“真”(.T.)。

除了使用句柄 0 设置的值作为原型值用于每个连接以外，还可以为一个单独的连接设置自定义的属性，只要以该连接句柄为参数使用 SQLSETPROP（ ）即可。属性 ConnectTimeout 例外，它在连接时刻便确定了。如果改变 ConnectTimeout 属性设置，新的设置只有重新连接时才能使用。

通过对连接和视图对象的属性进行设置，以控制连接和视图。那些对数据库、表、表字段、视图定义、视图字段、命名连接、活动连接以及活动的视图临时表进行控制的属性，我们称之为引擎属性。使用表 9-7 中的 Visual FoxPro 函数，可以显示或者设置引擎属性：

表 9-7 显示和设置引擎的函数

使用如下函数显示引擎属性	使用如下函数设置引擎属性
CURSORGETPROP（）	CURSORSETPROP（）
DBGETPROP（）	DBSETPROP（）
SQLGETPROP（）	SQLSETPROP（）

要使用什么样的函数，主要依赖于是否想为对象 0（连接 0 和 临时表 0）、数据库中的对象定义（命名连接定义或视图定义）以及活动对象（活动连接或活动的视图临时表）设置属性。表 9-8 中，列出了对象以及要为该对象设置属性的函数：

表 9-8 与对象相适应的连结函数

为以下对象设置属性	连 接	视 图
对象 0	SQLSETPROP（）	CURSORSETPROP（）
数据库中的对象定义	DBSETPROP（）	DBSETPROP（）
活动对象	SQLSETPROP（）	CURSORSETPROP（）

表 9-9 中，按照字母的顺序列出了引擎属性，同时注明每个属性所适用的对象。

表 9-9 引擎属性及所适应的对象

引擎属性	适用于
Asynchronous	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
Batchmode	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
BatchUpdateCount1	视图定义，请参阅函数 DBSETPROP（）。活动视图临时表，请参阅函数 CURSORSETPROP（）
Buffering	活动视图临时表，请参阅函数 CURSORSETPROP（）
Caption	表和视图定义中的字段，请参阅函数 DBSETPROP（）
Comment	数据库、表、表中字段、视图定义、视图定义中的字段以及连接定义，请参阅函数 DBSETPROP（）
CompareMemo	视图定义，请参阅函数 DBSETPROP（）。活动视图临时表，请参阅函数 CURSORSETPROP（）
ConnectBusy	活动连接，请参阅函数 SQLGETPROP（）
ConnectHandle	活动的视图临时表，请参阅函数 CURSORGETPROP（）
ConnectName1	视图定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLGETPROP（）。活动视图临时表，请参阅函数 CURSORGETPROP（）
ConnectString	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLGETPROP（）
ConnectTimeout	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
Database	活动视图临时表，请参阅函数 CURSORGETPROP（）
DataSource	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLGETPROP（）
DataType	在视图定义中的字段，请参阅函数 DBSETPROP（）
DefaultValue	在表和视图定义中的字段，请参阅函数 DBSETPROP（）
DeleteTrigger	表，请参阅函数 DBGETPROP（）
DispLogin	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
DispWarnings	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
FetchAsNeeded	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
FetchMemo（＊）	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
FetchSize（＊）	视图定义，请参阅函数 DBSETPROP（）。活动视图临时表，请参阅函数 CURSORSETPROP（）
IdleTimeout	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
InsertTrigger	表，请参阅函数 DBGETPROP（）
KeyField	在视图定义中的字段，请参阅函数 DBSETPROP（）
KeyFieldList（＊＊）	活动的视图临时表，请参阅函数 CURSORSETPROP（）
MaxRecords（＊）	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）
ODBCHdbc	活动连接，请参阅函数 SQLGETPROP（）
ODBCHstmt	活动连接，请参阅函数 SQLGETPROP（）
Offline	视图定义，请参阅函数 DBGETPROP（）

PacketSize	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
ParameterList	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）
Password	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLGETPROP（）
Path	表，请参阅函数 DBGETPROP（）
Prepared	视图定义，请参阅函数 DBSETPROP（）
PrimaryKey	表，请参阅函数 DBGETPROP（）
QueryTimeOut	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
RuleExpression	表、表中字段、视图定义、视图定义中的字段，请参阅函数 DBSETPROP（）
RuleText	表、表中字段、视图定义、视图定义中的字段，请参阅函数 DBSETPROP（）
SendUpdates（＊＊）	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）
ShareConnection	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
SourceName	活动的视图临时表，请参阅函数 CURSORGETPROP（）。
SourceType	视图定义，请参阅函数 DBGETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
SQL	视图定义，请参阅函数 DBGETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
Tables（＊＊）	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）
Transactions	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
Updatable	视图定义中的字段，请参阅函数 DBSETPROP（）
UpdatableFieldlist（＊＊）	活动的视图临时表，请参阅函数 CURSORSETPROP（）
UpdateName	视图定义中的字段，请参阅函数 DBSETPROP（）
UpdateNameList（＊＊）	活动视图临时表，请参阅函数 CURSORSETPROP（）
UpdateTrigger	表，请参阅函数 DBGETPROP（）
UpdateType	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）
UseMemoSize（＊）	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORGETPROP（）
UserID	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLGETPROP（）
Version	数据库，请参阅函数 DBGETPROP（）
WaitTime	连接定义，请参阅函数 DBSETPROP（）。活动连接，请参阅函数 SQLSETPROP（）
WhereType	视图定义，请参阅函数 DBSETPROP（）。活动的视图临时表，请参阅函数 CURSORSETPROP（）

＊属性主要用于远程视图；对本地视图设置此属性不起作用。如果想在本地视图中预先设置这个属性，可以在本地视图中设置此属性，升迁后再创建远程视图。

＊＊为了将更新传送到远程数据源必须设置引擎属性。

可用下列两种方法中的一种，通过使用事务处理方式来进行远程数据的更新、删除和插入等操作。

■ 自动事务处理模式

■ 人工事务处理模式

所选择的事务处理模式将决定 Visual FoxPro 在本地机器上处理事务的方式。在默认情况下，Visual FoxPro 自动为发送给远程服务器的每一个可事务化的命令包装一个事务处理的外壳。这个默认的自动事务处理只有在 Transactions 属性设置为 1（或 DB_TRANSAUTO）时才产生。若要使用自动事务处理模式，可使用 DBSETPROP（）函数将连接的 Transactions 属性设置为 1 或者 DB_TRANSAUTO。或者使用 SQLSETPROP（）函数将活动连接的 Transactions 属性设置为 1 或者 DB_TRANSAUTO。远程表上的事务处理将自动进行。

Visual FoxPro 的 BEGIN TRANSACTION 和 END TRANSACTION 命令只为本地 Visual FoxPro 临时表创建事务处理。它们不能把事务处理扩展到远程服务器上。

如果想人工控制事务处理，可以设置 Transactions 属性为 2 或 DB_TRANSMANUAL。如果采用人工事务处理模式，Visual FoxPro 将在发出第一个可事务化的 SQL 语句时，自动开始一个事务处理，但必须使用 Visual FoxPro 的 SQLCOMMIT（）或 SQLROLLBACK（）函数来结束这个事务处理。

若要使用人工事务处理模式，可使用 DBSETPROP（）函数将连接的 Transactions 属性设置为 1 或者 DB_TRANSMANUAL。 或者使用 SQLSETPROP（）函数将活动连接的 Transactions 属性设置为 2 或者 DB_TRANSMANUAL。通过 SQLCOMMIT（）和 SQLROLLBACK（）函数将进行人工事务处理。

在提交或回滚前一个事务处理之后，Visual FoxPro 在接收到下一个可进行事务处理的 SQL 语句时，自动开始一个新的事务处理。

Visual FoxPro 可以嵌套事务处理，支持最多五层的本地数据事务处理嵌套。单层事务处理的支持已内置到 SQL pass-through 中。

如果服务器支持多层事务处理，那么可用 SQL pass-through 来明确地管理事务处理的层数。但是这种明确的事务处理管理方式很复杂，因为很难控制内置事务处理和远程服务器事务处理之间的相互作用。有关明确的事务处理管理的详细内容，请参阅 ODBC 文档。

9.3.2 用 SQL pass-through 处理远程数据

用 SQL pass-through 检索一个结果集合之后，可用 Visual FoxPro 函数 CURSOR GETPROP（）和 CURSORSETPROP（）来查看和控制结果集合的临时表属性。在对一个活动视图临时表的属性进行设置时，所用的函数与这些函数相同。

临时表不是对象，因而与对象模式无关。但是，可用 CURSORGETPROP（）查看它们的属性，用 CURSORSETPROP（）设置它们的属性。

1. 设置远程数据的临时表属性

表 9-10 列出 Visual FoxPro 临时表的属性，这些属性适用于视图和连接的结果集合，并按任务类型分组。

表 9-10 Visual FoxPro 临时表属性

任 务	属 性	用 途
视图临时表定义	SQL	存放用以生成临时表的 SQL 语句
控制 Visual FoxPro 和 ODBC 之间的相互作用	ConnectHandle	临时表所使用的远程连接的句柄
	ConnectName	临时表所使用的连接名
	Prepare	指定在执行视图查询之前，是否进行准备
	FetchAsNeeded	在空闲循环期间或者仅在必要时，指定是否自动对行存取
	CompareMemo	无论 UpdateType 属性如何设置，指定在 UPDATE 语句的 WHERE 子句中，是否包含备注和通用字段
	FetchMemo	指定备注和通用字段是随结果集合一起自动获取，还是在备注和通用字段打开时，根据要求来获取
	UseMemoSize	将结果返回到备注字段的列的最小宽度（1 到 255）

	FetchSize	指定每次从远程表中所取得的记录数
	MaxRecords	返回结果集合时获取的最大行数
更新数据	SendUpdates*	指定对临时表的更新是否发送到临时表所基于的表中
	BatchUpdateCount	指定发送到后台缓冲表的 update 语句数量
	Tables（*）	以逗号分隔的数据源上的表名列表，用于定义 UpdateNameList 和 UpdatableFieldsList 属性的范围
	KeyFieldList*	以逗号分隔的 VisualFoxPro 字段的列表，字段中包括用于更新的结果集合的主关键字
	UpdateNameList（*）	以逗号分隔的、与临时表中的 VisualFoxPro 字段相对应的、接收更新的表名及列名
	UpdatableFieldList（*）	以逗号分隔的、要更新的 Visual FoxPro 字段的列表
	Buffering	指定在临时表上使用的缓冲类型
	UpdateType	指定是使用 UPDATE 命令还是联合 DELETE 和 INSERT 命令进行更新
	WhereType	指定在对表数据进行更新时，应该在 WHERE 子句中包含什么内容

注：标*的是必须在更新数据之前设置的属性。

可用这些属性来控制应用程序与远程数据交互作用的方式，如控制逐步获取时检索的行数、控制对远程数据的缓冲以及对远程数据的更新等。

用 SQL pass-through 设置属性时，在新建一个临时表时，临时表从环境临时表或者当前工作期的 0 号临时表中继承属性设置，如 UpdateType 和 UseMemoSize。可用 CURSOR SETPROP（ ）函数并以 0 做为临时表编号，来更改这些默认属性设置。

用 SQL pass-through 创建一个视图临时表之后，可用 CURSORSETPROP（ ）函数更改活动视图临时表的属性设置。用 CURSORSETPROP（ ）进行的更改是临时的；在关闭视图时，活动视图的临时设置也将消失；在关闭 Visual FoxPro 工作期时，第 0 号临时表的临时设置也随之消失。

连接继承属性的方式与此类似，在数据库中创建并保存一个命名连接时，此连接将默认地继承第 0 号连接的属性。可用 SQLSETPROP（ ）函数改变第 0 号连接的这些默认属性设置。在新建了一个连接并且将其存入数据库以后，可用 DBSETPROP（ ）函数更改连接属性。使用连接时，活动连接将继承存储在数据库中的该连接的属性设置。可用 SQLSETPROP（ ）函数以该活动连接的句柄为参数，更改这些属性。

SQL pass-through 的视图临时表和命名连接都能使用命名的 ODBC 数据源。如果在一个 SQL pass-through 的视图临时表中使用 ODBC 数据源，该连接将从工作期默认值中继承属性。

2．用 SQL pass-through 更新远程数据

使用 SQL pass-through 函数来更新一个远程服务器上的数据时，通过设置结果集合临时表的属性，不但可以决定是否进行更新，而且可以控制有关更新的具体细节。这样，在请求一个更新时，Visual FoxPro 会在提交更新之前检查这些属性。

若要更新远程数据必须设置五个属性：Tables、KeyFieldList、UpdateNameList、UpdatableFieldList 和 SendUpdates。还可以指定其他属性，如 Buffering、UpdateType 和 WhereType ，以更好地满足应用程序的要求。

若要使活动视图临时表可更新，可用 CURSORSETPROP（ ）函数指定视图临时表的更新属性：Tables、KeyFieldList、UpdateNameList、UpdatableFieldList 和 SendUpdates。

但是 SQL pass-through 视图临时表只有在指定更新属性之后，才是可更新的。如果要永久存储更新属性设置，可以创建一个视图定义。在用视图设计器或以编程方式创建一个视图时，Visual FoxPro 提供一些默认值，使得视图可以更新。此外，可用 CURSORSETPROP（ ）函数设置其他信息，来补充或者定制这些默认值。

对活动视图临时表设置的更新属性与 DBSETPROP（ ）设置的更新属性稍有不同。表 9-11 列出视图定义和活动临时表分别使用的属性名。

表 9-11 视图和临时表更新属性

用 途	视图定义属性*	活动临时表属性**
使远程表可更新	Tables	Tables
指定视图字段的远程名称	UpdateName（字段级属性）	UpdateNameList
指定要做为关键字的视图字段	KeyField（字段级属性）	KeyFieldList
指定可更新的视图字段	Updatable（字段级属性）	UpdatableFieldList
允许更新	SendUpdates	SendUpdates

通过设置临时表的 Buffering 属性，可以控制如何缓冲对远程数据的更新。在五个可能的缓冲属性设置中，有两个对远程视图有效：

■ DB_BUFOPTROW 或者 3，为默认值，开放式行锁定。

■ DB_BUFOPTTABLE 或者 5，开放式表锁定。

Visual FoxPro 在远程临时表上只支持开放式锁定。而保守式的行缓冲和表缓冲设置（2 和 4）不能应用于远程视图，因为 Visual FoxPro 不能锁定服务器上的数据。而 Buffering 属性设置为 1 也不能应用于远程视图，因为视图经常放在缓存里。

默认的 Buffering 设置是 DB_BUFOPTROW，对远程数据进行开放式逐行锁定。例如，如果要逐行提交对 titles 表的更新，比如使用 SKIP 命令时，可把 Buffering 属性设置为 3：

```
CURSORSETPROP（'buffering', 3, 'titles'）
```

当 Buffering 设置成行缓冲时，可用两种方法向远程服务器发送更新：

■ 调用 TABLEUPDATE（ ）函数。

■ 用命令把记录指针从该行移开，例如 SKIP 或者 GO BOTTOM 命令。

TABLEUPDATE（ ）函数更新服务器但并不移动记录指针。移动记录指针的命令向远程服务器发送更新，做为指针移开该记录的附带操作。

若用行缓冲并且希望能还原对行的更改，必须用 SQL pass-through 的事务处理函数将这些更改放入一个事务中。

若要每次成批提交对一个表的更改，例如用户在一个表单中单击“保存”或“确定”按钮时执行一个批处理操作，这时，必须把 Buffering 属性设置为 5 或 DB_BUFOPTTABLE。必须调用 TABLEUPDATE（ ）函数来向服务器发送更新。

下面的例子在表单的初始化代码中设置 Buffering 属性，然后在进行保存的代码中提交更改。

```
CURSORSETPROP（'buffering', 5, 'sqltitles'）    在初始化代码中设置
* 成批更新所做的修改；
* 不管其他人所做的修改
TABLEUPDATE（.T., .T., 'titles'）      在进行保存的代码中设置
```

若要恢复一个表的原始值，并且阻止更新发送给远程服务器，可以调用 TableRevert（ ）。通过设置临时表的 Buffering 属性并发出 TableRevert（ ） 命令，可以控制是恢复单独一行还是所有的行。下面的例子只恢复当前行。可在代码中加入下面的语句，当用户单击表单中的“取消”按钮时执行此代码：

```
= TABLEREVERT（.F., 'titles'）        && 恢复当前的行
```

若要恢复所有的行，当用户按 ESC 键离开一个表单时，可以使用与此相同的代码。但需要更改 Buffering 属性的设置，同样用 TABLEREVERT（ ）命令，根据整个缓冲表来恢复所有的行：

```
= TABLEREVERT（.T., 'titles'）        && 恢复所有的行
```

在多用户应用程序中，与其他用户的更新冲突由 SQL Update 查询检测，此查询在试图进行一个本地的写操作时生成。检测的级别依赖于 CURSORSETPROP（ ）中 WhereType 属性的设置

可用 TABLEUPDATE（ ）函数控制在发送对表或临时表的更新时，是否要覆盖网络上其他用户对它的更新。若把 TABLEUPDATE（ ）的 Force 参数设置为“真”(.T.)，并用 CURSORSETPROP（ ）将 UpdateType 属性设置为默认值 1，那么只要远程表上的记录的关键字段值没有被更改，旧的数据就将被最近发送的新数据所更新。如果远程表的关键字段值已被更改，或者 UpdateType 属性设置为 2，那么 Visual FoxPro 先向远程表发送一条 DELETE 语句，然后再发送一条 INSERT 语句。

表 9-12 用于远程更新的错误信息

错误信息	解 释	操 作
没有更新指定的表。请使用临时表的 Tables 属性	临时表的 Tables 属性没有包含远程表名。要能对远程服务器进行更新，最少需要一个表	使用 Tables 属性为该临时表指定最少一个表
没有指定更新表 table_name 的关键字的列(有可能是多个列)。请使用临时表的 KeyFieldList 属性	在错误信息中指出的远程表的主关键字没有包括到临时表的 KeyFieldList 属性中，每个更新的表都需要一个主关键字	用 KeyFieldList 属性指定远程表的主关键字
列 column_name 中没有指定合法的更新表，用临时表的 UpdateNameList 和 Tables 属性	列 column_name 中的 UpdateName 属性包含一个非法的表名	用 UpdateNameList 属性设置表名，或把表名添加到 Tables 属性设置中，或者两者都做
临时表的 KeyFieldList 属性没有定义一个唯一的关键字	多个远程记录具有相同的关键字。	用 KeyFieldList 属性为远程表定义一个唯一的关键字
来自于 ODBC：ODBC 非法对象	ODBC 不能找到远程表或者列，因为指定名称的表或列并不存在。Visual FoxPro 字段名由 VisualFoxPro 检查，远程表和列名只由远程服务器检查	检查对象的名称

表 9-12 列出了专门应用于远程更新的 Visual FoxPro 和 ODBC 错误信息。“操作”栏包含了解除错误所应采取的操作。

3. 选择有效的 SQL pass-through 处理方式

Visual FoxPro 提供了用 SQL pass-through 来检索和更新远程数据的两种处理方式：同步和异步。使用 SQL pass-through 函数时，可以选择喜欢的方式。对于远程视图，不需要选择方式；Visual FoxPro 对远程视图自动采用逐步获取技术，并且管理这种远程视图的处理方式。

默认情况下，以同步方式处理 Visual FoxPro SQL 函数：Visual FoxPro 直到一个函数调用完成时，才把控制权返回给应用程序。与 Visual FoxPro 交互工作时，同步处理方式是很有用。

异步处理比同步处理具有更大的灵活性。例如，当应用程序异步处理一个函数时，应用程序可以建立一个进度指示器来显示正在执行的语句的进展情况，显示鼠标指针的运动，创建循环并且设置时钟使得占用时间过长的处理过程自动中断。

异步使用 SQL pass-through 时，应用程序可为四个用以向数据源发出请求、检索数据的函数申请异步处理方式。它们是：SQLEXEC（）、SQLMORERESULTS（）、SQLTABLES（）和 SQLCOLUMNS（）。可以使用 SQLSETPROP（）函数设置连接的 Asynchronous 属性，以启用异步处理方式；一旦建立了该连接的异步通信方式，所有这四个函数都将以异步方式进行操作。

若要检查 Asynchronous 属性的设置，可使用 SQLGETPROP（）函数查看 Asynchronous 属性设置。在下面例子中，nConnectionHandle 代表活动连接的句柄编号：

```
? SQLGETPROP（nConnectionHandle,'Asynchronous'）
```

若要允许异步处理，可使用 SQLSETPROP（）函数指定 Asynchronous 属性：

```
? SQLSETPROP（nConnectionHandle,'Asynchronous', .T.）
```

在异步方式下，必须重复地调用每个函数，直到返回一个非 0 值(0 表示仍在运行)。如果 SET ESCAPE 设置为“真”(.T.)，则在函数运行过程中，可以按 ESC 键取消对函数的处理。

只有在原来就与连接句柄相关的 SQLCANCEL（）或者异步函数（SQLEXEC（）、SQLMORERESULTS（）、SQLTABLES（）或 SQLCOLUMNS（））结束处理时，应用程序才能使用该连接句柄。在一个函数结束之前，不能把该函数使用的连接句柄用于其他三个异步函数或者 SQLDISCONNECT（）函数。

处理多个结果集合时，用 SQLEXEC（）函数发出多个 SQL SELECT 语句，或者执行一个用来发出多个 SELECT 语句的存储过程时，应用程序都会检索多个结果集合。每个 SQL SELECT 语句的结果都返回在一个独立的 Visual FoxPro 临时表中。

第一个临时表的默认名称是 SQLRESULT，在该默认名的基础上加上数字就得到后续的临时表的名称。例

如，由 SQLEXEC（ ）语句返回的三个结果集合的临时表默认名是：Sqlresult、Sqlresult1 和 Sqlresult2。

在批处理方式下，如果一个函数返回多个结果集合，在 Visual FoxPro 中各个临时表名都具有唯一的后缀，并且能达到 255 个字符。例如，下例把 BatchMode 属性设置为批处理方式，然后发出一个 SQLEXEC（ ）语句，其中包含有四个 SQL SELECT 语句，得到四个结果集合。

```
? SQLSETPROP（nConnectionHandle,'BatchMode', .T.）
? SQLEXEC（nConnectionHandle,'select * from authors ;
                    select * from titles ;
                    select * from roysched ;
                    select * from titleauthor','ITEM'）
```

当上述函数处理完成以后，Visual FoxPro 以临时表 Item、Item1、Item2 和 Item3 的形式返回四个结果集合。

可以用 SQLEXEC（ ）或 SQLMORERESULTS（ ）函数的 cCursorname 参数来更改这个默认名。如果指定的结果集合名已被使用，则新的结果集合将覆盖原来的临时表。

在检索多个结果集合时，应用程序可在同步处理或异步处理、批处理或非批处理方式之间做出选择。

如果使用批处理方式，由 SQLSETPROP（ ）函数设置的 BatchMode 属性可以控制 SQLEXEC（ ）返回多个结果集合的方式。此属性的默认值为 1，即批处理方式。批处理方式意味着 Visual FoxPro 的 SQLEXEC（ ）调用只在检索完了所有的单个结果集合后，才返回结果。

如果使用非批处理方式，用 SQLSETPROP（ ）函数把 BatchMode 属性设置为 0，那么每个结果集合将单独地返回。第一个结果集合由 SQLEXEC（ ）函数调用返回。其他结果集合必须由应用程序重复调用 SQLMORERESULTS（ ）函数得到。若函数返回值为 2，则表明已没有其他结果。

在非批处理方式中，临时表名可在每一个后继的 SQLMORERESULTS（ ）调用中更改；因此，如果上例中第一个临时表为 Item，并且第二个 SQLMORERESULTS（ ）调用把 cCursorName 参数更改为 Otheritem，那么结果临时表名就为 Item、Item1、Otheritem 和 Otheritem1。

下面将详细说明批处理与非批处理、同步与异步之间的区别。

在同步方式下，函数在执行完毕以后，才将控制权返回给应用程序。

（1）同步批处理方式

在以批处理方式同步执行一条 SQL pass-through 语句时，直到所有结果集合都检索完毕以后，才返回控制权。在最初的函数中，用 cCursorname 参数指定第一个临时表的名称。如果指定的临时表已经存在，结果集合会覆盖原来的临时表。在用同步批处理方式请求多个结果集合时，Visual FoxPro 通过在第一个临时表名的基础上加数字来创建其他的临时表名。

（2）同步非批处理方式

在以非批处理方式同步执行一条 SQL pass-through 语句时，第一条语句检索第一个结果集合，并且返回 1。然后，必须重复地调用 SQLMORERESULTS（ ）函数得到其他结果集合，并且可以指定一个新的临时表名。如果不为临时表指定一个新名称，那么会以原来的名称为基础并分别加上数字来创建多个结果集合的多个名称。当 SQLMORERESULTS（ ）返回值为 2 时，说明已没有其他结果。

在异步方式下，应用程序必须连续地调用同一个 SQL pass-through 函数，直到它返回一个非 0 值（0 表示仍在执行）。默认的结果集合名 Sqlresult，可以在第一次调用函数时通过 cCursorname 参数明确地更改。如果指定的结果集合名已被使用，那么新的结果集合会覆盖原来的临时表中的信息。

（3）异步批处理方式

以批处理方式异步执行时，在所有结果集合都返回给指定的临时表之前，对原函数的每一次重复调用都将返回 0（表示仍在执行）。在所有的结果都检索完毕之后，所得到的返回值可能是临时表的编号，也可能是一个负值：表示出错。

（4）异步非批处理方式

以非批处理方式异步执行时，SQLEXEC（ ）每完成一个结果集合的检索，就返回 1。应用程序必须重复调用 SQLMORERESULTS（ ）函数，直到返回值为 2，表明已经没有其他结果。

应当注意，远程结果集合必须经历两个步骤完成检索：首先，在服务器上得到结果集合。然后，结果集合放在一个本地 Visual FoxPro 临时表中。在异步方式下，可调用 USED（ ）函数来查看 Visual FoxPro 是否已开始获取所请求的临时表。

4、控制数据类型的转换

在远程服务器和 Visual FoxPro 之间转移数据时，多少会遇到服务器上的数据类型与 Visual FoxPro 中的数据类型有差别的情况，远程数据源和 Visual FoxPro 的数据类型完全对应的情况是极少的。为了处理这些差别，Visual FoxPro 借助 ODBC 数据类型将远程数据类型映射为本地 Visual FoxPro 数据类型。通过掌握如何在 ODBC 和 Visual FoxPro 之间映射数据类型，可以对 Visual FoxPro 应用程序如何处理服务器上的远程数据做到心中有数。

如果需要，还可以调整在服务器上或者应用程序中使用的数据类型。可以通过为远程数据集合创建一个视图，然后在数据库中设置 DataType 视图字段属性来忽略默认的 Visual FoxPro 字段数据类型。DataType 属性是一个字符属性，指出一个远程视图中每个字段所需的数据类型。

表 9-13 ODBC 远程字段与 Visual FoxPro 字段类型的比较

远程字段的 ODBC 数据类型	Visual FoxPro 临时表中字段数据类型
SQL_CHAR SQL_VARCHAR SQL_LONGVARCHAR	字符型或备注型（*）
SQL_BINARY SQL_VARBINARY SQL_LONGVARBINARY	备注型
SQL_DECIMAL SQL_NUMERIC	数值型或货币型（**）
SQL_BIT	逻辑型
SQL_TINYINT SQL_SMALLINT SQL_INTEGER	整型
SQL_BIGINT	字符型
SQL_REAL SQL_FLOAT SQL_DOUBLE	双精度型；小数位数由 Visual FoxPro 中 SET DECIMAL 的值决定
SQL_DATE	日期型
SQL_TIME	日期时间型（***）
SQL_TIMESTAMP	日期时间型（****）

注意：

- 如果 ODBC 字段宽度小于临时表属性 UseMemoSize 的值，在 Visual FoxPro 临时表中它将变成一个字符型字段；否则，它成为一个备注型字段。如果服务器字段为 money 数据类型，则在 Visual FoxPro 中它变成货币数据类型。日期默认为 1/1/1900。如果 SQL_TIMESTAMP 字段的值包含秒的小数部分，则在转换为 Visual FoxPro 的日期时间型数据时，该小数被截断。

从一个远程 ODBC 数据源检索数据时，Visual FoxPro 把每个 ODBC 字段的数据类型转换成结果集合临时表中等价的 Visual FoxPro 数据类型。表 9-14 列出 ODBC 数据源上提供的数据类型，以及对应的 Visual FoxPro 的数据类型。

注意：

- 在 ODBC 数据源字段中的 null 值在将变成 Visual FoxPro 临时表中的 null 值，应用程序检索远程数据时将忽略 Visual FoxPro 中的 SET NULL 设置。

如果存于临时表中的 Visual FoxPro 数据来源于远程数据，当这些数据发送给远程服务器时，会转换成它们原来的 ODBC 类型。如果通过 SQL pass-through 把来源于 Visual FoxPro 的数据发送给远程服务器，将进行下面的转换。

表 9-14 ODBC 与 Visual FoxPro 数据类型的比较

VisualFoxPro 数据类型	ODBC 数据类型
字符型	SQL_CHAR 或 SQL_LONGVARCHAR1
货币型	SQL_DECIMAL
日期型	SQL_DATE 或 SQL_TIMESTAMP2
日期时间型	SQL_TIMESTAMP
双精度型	SQL_DOUBLE
整型	SQL_INTEGER
通用型	SQL_LONGVARBINARY
逻辑型	SQL_BIT
备注型	SQL_LONGVARCHAR
数值型	SQL_DOUBLE

注意：

- 如果映射为一个参数的 Visual FoxPro 内存变量创建了一个宽度小于 255 的表达式，那么在 ODBC 数据源中它成为 SQL_CHAR 类型；否则，成为 SQL_LONGVARCHAR 类型。
- 对于除 SQL Server 以外的所有其他 ODBC 数据源，Visual FoxPro 日期型数据都被转换为 SQL_DATE 类型，对于 SQL Server，它被转换为 SQL_TIMESTAMP 类型。

可把一个 Visual FoxPro 参数值映射为一个特定的远程数据类型，方法是对该参数设置格式，使其成为一个字符表达式，该字符表达式符合对应的远程数据类型的语法要求。例如，如果服务器提供日期时间型数据类型，可以以该服务器所支持的日期时间型格式为 Visual FoxPro 参数创建一个字符表达式，在服务器接收到该参数值的时候，它会尝试把这个经过设置格式的数据映射为日期时间型数据。

在发送一个参数给远程服务器时，必须确保 WHERE 子句中的数据类型与参数表达式中所使用的数据类型相匹配。

9.3.3 处理 SQL pass-through 错误

如果 SQL pass-through 函数返回错误，Visual FoxPro 将把错误信息存储在一个数组中。AERROR（）函数能够提供从各个级别组件中检测到的错误信息：Visual FoxPro、ODBC 数据源或者远程服务器。通过检查 AERROR（）的返回值，用户可以确定所发生的服务器错误以及错误信息的文本。

特别要注意：要获得错误信息，必须在发生错误后立即调用 AERROR（）。若在调用 AERROR（）之前又发生其他错误，那么前一个错误信息便会丢失。

9.4 本章小结

Visual FoxPro 为创建功能强大的客户／服务器应用程序提供了一些专用工具。

Visual FoxPro 客户／服务器应用程序将 Visual FoxPro 功能强、速度快、图形化的用户界面以及高级的查询、报表和处理等优点与严密的多用户访问、海量数据存储、内置安全性、可靠的事务处理和日志以及 ODBC 数据源或服务器的本地语法等功能紧密地结合在一起。

Visual FoxPro 和强有力的服务器之间的协作可以为用户提供功能强大的客户／服务器解决方案。

本章主要讲述了以下内容：

- 设计客户／服务器应用程序
- 升迁 Visual FoxPro 数据库
- 实现客户／服务器应用程序

附录 Visual FoxPro 6.0 的语言基础

Visual FoxPro 6.0 是一个数据库管理和开发系统，它的语言继承了以往版本的许多优点，它的语言简单、功能强大，与 C 语言有许多相同之处，同时它又吸收了 Visual Basic 等可视化语言的特点，支持面向对象的程序设计。本章介绍的关于编程语言的基础知识将有助于读者对 Visual FoxPro 6.0 的深入学习。

数据类型

和其他许多程序设计语言一样，Visual FoxPro 6.0 也给用户提供了多种数据类型，不同的数据类型具有各自的取值范围和特点，计算机根据不同的数据类型分配不同的存储空间，并进行不同的操作。

Visual FoxPro 6.0 中的程序设计语言的常量、变量、数组所具有的数据类型有以下几类：

逻辑型（Logical）

逻辑型数据具有两个布尔值.T.和.F.。如果保存、处理的信息只有两个值，例如“是”与“非”、“正”与“负”，就应当尽量选用逻辑型数据来保存它们，因为逻辑型数据只占有 1 个字节，能节省存储空间、提高程序执行效率。在 Visual FoxPro 6.0 中，用户定义的变量、数组在默认情况时都是逻辑性的值.F. 。

计算机只能识别简单的数据 0 和 1，这其实就是逻辑型数据的两个布尔值.F.和.T.，因而在 Visual FoxPro 6.0 中条件控制语句中的条件表达式返回逻辑型值，当它为.T.时，条件成立，执行下面的语句，当它为.F.时结束条件语句。

数值型（Numeric）

它表示一个整数或实数，在内存中占有 8 个字节，它的大小从-0.9999999999E+19 到 0.9999999999E+20。

只有这种数据类型才能作为参数用于数值型函数中。

字符型（Character）

字符型数据是由字母、数字、空格、符号、标点和汉字等组成的。在定义时常用引号，如“中国”、“307”等都是字符型数据，即便是一个完全由数字组成的字符型数据也不能进行数学运算。字符型数据的长度是可变的，最多可以占据 254 个字节，一个汉字在内存中占 2 个字节，其他字符占 1 个字节。

日期型（Date）

日期数据类型存储有关年、月、日的信息，通常用大扩号扩起来，如{12/31/99}，它在内存中的存储格式是“YYYYMMDD”，年份占 4 个字节，月份和日子各占 2 个字节，{12/31/99}与{12/31/1999}表达的信息完全相同，由此可见，Visual FoxPro 6.0 和由它开发的应用系统不存在计算机的 2000 年问题。

日期型数据的输入输出格式可以用 SET DATE、SET MARK、SET CENTURY 等系统设置命令来设置。

日期时间型（DateTime）

日期时间型数据同时包括日期、时间，也可以只含两者之一，如{01/01/99 12：00am}。日期时间型数据占有 8 个字节，前 4 个字节用整数型用来存储日期，后 4 个字节用整数型来存储时间。

日期时间型数据也有多种存储格式，对日期部分的格式，仍用 SET DATE、SET MARK、SET CENTURY 设定，对时间部分的格式，用 SET HOURS、SET SECONDS 来设置。

货币型（Currency）

当需要保存有关货币计量的数据时，最好不要使用数值型，而使用货币型数据，也是因为货币型数据具有自动控制小数位数的功能。当货币型数据被赋予的数值的小数位数超过 4 位时，Visual FoxPro 6.0 将在运算之前自动将其作舍入处理。

在 Visual FoxPro 6.0 中，货币型的数据在存储时占用 8 个字节的空间，取值范围介于正负 922337203685477.5808 之间。

在货币型数据的数字前，必须使用一个字符$来表示出该货币类型的单位。例如下面的货币型数据：

$100000

$65454.75778978

第二个货币型数据在存储时被保存为$65454.7578。

对象型（Object）

它是一个对象的名字，唯一地表示这个对象。它用一些特殊的命令来创建，如下面的语句创建一个对象型变量 oMyform：

oMyform=CREATEOBJECT（"FORM"）

用户可以使用 TYPE（）函数来测试一个数据的类型，如果用户想测试变量 oMyform 是否是一个对象型变量，可以在命令窗口键入上面的语句创建这个变量后，再键入命令：

？TYPE（"oMyform"）

按回车后在 Visual FoxPro 6.0 主窗口中显示了一个 O，它是 Object 的缩写，表示变量 oMyform 是一个对象型变量。

测试其他类型的变量类型也可以应用这个函数。

不定型（Varient）

不定型变量能存储任意类型的数据，一旦把数据赋值给一个不定型变量，该变量的类型即为该数据的类型。

操作符

操作符是处理各种数据的符号，它最重要的功能是进行数据运算。

数据操作符

数据操作包括数值操作、字符操作、日期操作等，数据操作符主要有 + 、- 、* 、/ 、**或^ 等，并且它们可以和括号组合。这些操作符的优先级顺序为：()，**或^，*和/，%，+和-，见表 1。

表 1 数据操作符

操作符	说明	实例	结果
+	加法运算	"Visual "+"Foxpro"	"Visual Foxpro"
*	乘法运算	10*10	100
**	乘方运算	10**3	1000
-	减法运算	{13:31:01}-{13:30:59}	2
/	减法运算	15/7	2.14
%	取余	10%8	2

关系操作符

关系操作用于对两个或多个数据进行比较，它返回一个逻辑值，主要有：=、 < 、<= 、> 、>= 、<>或!=或# 、== 等，见表 2。

表 2 关系操作符

操作符	说明	实例	结果
=	等于	1=2	.F.
<	小于	{10/15/98}<{10/14/98}	.F.
<=	小于或等于	3.14<=3.14	.T.
>=	大于或等于	13>=18	.F.
<>|!=|#	不等于	Time（）!={01/01/00}	.T.
==	等同于	1.0==1.0	.T.
$	包含	"Fox"$"Foxpro"	.T.

逻辑操作符

Visual FoxPro 6.0 中逻辑操作符有（）、NOT、OR、AND，它们的优先权顺序为：（），NOT 或 !，AND，OR，见下表：

表 3 逻辑操作符

操作符	说明	实例	结果
AND	逻辑与	.T. AND .F.	.F.
OR	逻辑或	.T. AND .F.	.T.
NOT 或！	逻辑非	NOT .F.	.T.
（）	括 号	NOT（.T. AND .T.）	.F.

存储数据

数据是信息的载体，例如在程序中要执行某一操作，需要借助一些内存变量或变量数组才能完成。用户开发的管理系统中也需要存储各种各样的数据，在应用程序的开发过程中，要使用许多常量、变量和数组。本节讲述的就是怎样创建和使用常量、变量和数组，怎样存储和读入数据。

常量

常量在运行时固定不变，最常用的常量是字母、汉字和数字，如字母"A"是一个字符型常量，数字 0 时一个数值型常量，"中国"是一个字符型常量。在 Visual FoxPro 6.0 中使用常量时不一定需要先定义它。当常量较复杂且常用时，为减少程序开发的工作量，常常定义一个常量来替代它。Visual FoxPro 6.0 中定义常量的方法与 C 语言类似，即使用 DEFINE 语句。如：

```
#Define  e  2.718281828
```

这个语句定义了一个数值型常量，定义以后，再用到 2.718281828 的地方都可以用 e 来替代。也可以定义一些字符型、日期型或逻辑型常量。

常量还常常应用在程序开发调试阶段，开发者只要在定义常量的语句中改变常量的值，在程序中所有用到这个常量的地方都无需改变，程序调试的速度将大大提高，并且用户也可以在程序中使用 UNDEFINE 来中止常量的使用。

变量

变量在运行时其值可以改变，它在程序开始运行时在内存中创建，在程序结束时从内存中释放。变量名必须以字母（汉字也可）或下划线开头，变量名中只能使用字母、汉字、下划线或数字，在 Visual FoxPro 6.0.0 版本中，其长度不受任何限制，但不能与系统的内存变量和保留字同名。下面的语句都定义了一个变量整型 I，并给它赋初值为 0：

```
I=0
Store  0  to  I
```

第一种创建语句实际上就是一个赋值语句，因而，变量并不需要有专门的声明语句，可以直接拿来使用，在第一次使用时系统自动创建这个变量。

变量的类型由存入它的数据的类型来确定，并且与 C 语言等高级语言不同，在 Visual FoxPro 6.0 中的变量的类型可以任意改变，它可以时而是一个整型，时而变成逻辑型，时而又变成日期型，这主要看赋值语句给变量所赋的值是什么类型。

可以使用 LOCAL、PRIVATE 和 PUBLIC 关键字来指定变量的作用域：

（1）用 PUBLIC 创建全局变量。这种变量在当前 VISUAL FOXPRO 工作期中，任何运行的程序都能使用和修改它，它不能自动释放。

（2）用 LOCAL 创建局部变量。这种变量只能在创建它的程序中使用和修改，当它所在的程序停止运行时，这种变量自动从内存中释放。

（3）PRIVATE 关键字。PRIVATE 关键字将其他程序中定义的变量在当前程序中隐藏起来，以便在当前程序中使用和这些变量同名的变量，也不会使原有变量的值发生变化。

用户在开发时应当尽力避免变量名与表中的字段重名，但由于在程序运行时可能会访问到许多数据库或表，这种情况也可能发生，这时 Visual FoxPro 6.0 将访问字段而忽略同名的内存变量，如果用户希望访问的是变量，就需要在这个变量名的前面加上“ m. ”或“ m-> ”来强制访问这个变量。

数组

数组是一组有序数据的组合，它能高效地组织数据，存储在其中的数据称作数组元素。数组的命名规则与变量相同，数组要用系统保留字 DIMENSION 或 DECLARE 来声明。如：

```
DIMENSION   A[10]
DECLARE     B（10，10）
```

第一个声名语句定义了一个一维数组，它包含 10 个元素，第二个声明语句定义了一个二维数组，它包含 10×10 个元素，数组后缀中的中括号与小括号的用法相同。用户也可以直接使用 PUBLIC、LOCAL、PRIVATE 等作用域控制命令，在声明数组的同时指定这个数组的作用域，如：

```
PUBLIC   C[5]
LOCAL    D[4]
```

在一个新数组声明后，这个数组元素的值都默认为.F. ，用户不能用一个赋值语句给数组中的所有元素赋值，但可以使用 STORE 来完成。

例如在定义了数组 A[10]后，使用下列语句试图给每个数组元素赋值是不对的：

```
A= .T.
```

用户可以使用下列语句：

```
STORE   .T.   TO   A
```

或者

```
FOR   I=1 TO 10
    A[I]= .T.
ENDFOR
```

在 Visual FoxPro 6.0 中，一个数组中可以储存不同的数据类型，并且数组元素的数据类型也可以任意改变，下面的语句是可以的：

```
DIMENSION   A[5]
A[1]=0
A[2]="FOX"
A[3]="PRO"
A[4]=.T.
A[5]={13：01：59}
A[1]=A[2]+A[3]
```

也可以使用 RELEASE 语句从内存中释放声明过的数组，下面的语句先声明一个数组，然后把它从内存中

释放：

```
DIMENSION  A[10]
RELEASE    A
```

在 Visual FoxPro 6.0 中，数组更多地被用于内存与表中的记录之间实现数据交换，如用数组来保存字段中的纪录，或把通过数组把内存中的数据存储到一个表的字段中。由于这些操作使用了 Visual FoxPro 中的 Rushmore 技术，在与大型数据库实现数据交换时能给应用程序的用户节省许多时间。

对象的属性

Visual FoxPro 6.0 是面向对象的程序设计语言，每个对象都有它的许多特征值，这些值通常存储在对象的属性中，它类似于一个变量，但它的名字是由系统指定的，并且每个属性都有它特定的意义。例如用户要改变一个对象的宽度，就必须改变这个对象的 Width 属性值。用户也可以添加一些新属性来存储额外的数据。

对象的属性多使用在设计类和表单时，在本书第十四章中，用户能深刻体会到它的功用。

函数、命令与系统内存变量

Visual FoxPro 6.0 的程序设计就是正确调用函数和系统内存变量、使用命令的过程。系统给定的函数、命令与系统内存变量都是 Visual FoxPro 6.0 的保留字。它们都有自己特有的功能和语法结构，当然，用户也可以自己定义一些新的函数与命令，现在先简单地介绍一下 Visual FoxPro 6.0 提供的函数、命令与系统内存变量。

函数

根据主要功能和特点，可以把 Visual FoxPro 6.0 的函数分为以下几类：

（1）数据库操作函数：用来处理数据库、表、表中的字段、视图以及表间的关系等。如 BOF（）函数可以用来测试一个表的记录指针是否在表头。

（2）环境设置函数：用来管理 Visual FoxPro 6.0 的系统和环境参数。如 HOME（）函数返回启动 Visual FoxPro 6.0 的目录名。

（3）数值函数：进行数值运算，处理数值型数据。如 SIN（），MAX（）等。

（4）字符函数：对字符型数据进行处理。如 UPPER（）函数能把字符串中的小写字母变成大写。

（5）日期和时间函数：获得或处理日期和时间型数据。如 TIME（）函数返回系统时间，YEAR（）函数从一个日期型数据中获得年份。

（6）数据转换函数：它一种类型的数据转换成另一种类型的数据。如 STR（）函数把一个数值型数据转换成字符型。

（7）文件管理函数：管理和处理磁盘文件。如 GETFILE（）将打开 Windows 中的“ OPEN FILE ”对话框，让用户读取文件。

（8）网络函数：用于网络开发中让多个用户共享数据时。

（9）键盘和鼠标函数：返回键盘、鼠标的一个特征值。如 INKEY（）返回用户按键的键编号。

（10）内存变量处理函数：给内存变量赋值、进行有关内存变量的数据操作。如 AERROR（）把 Visual Foxpro 产生的错误信息存入到一个新创建的内存变量数组中。

（11）SYS（）函数：返回 Visual Foxpro 的系统信息。如 SYS（2020）返回默认磁盘的空间。

在引用函数时，一定要在函数名后加上一对小括号。通常小括号中是函数的参数，参数的数目与数据类型由每个函数的特定格式来指定；有的函数并不接受任何参数，但这对小括号仍然不能省略。

命令

命令执行一项功能，命令与函数的最大区别是命令不返回值，一个命令在使用时后面不能带小括号。同函数一样，也可以根据命令的功能和特点来给它们分类：

（1）面向对象程序设计命令：创建和处理类和对象，控制事件发生时程序执行的动作。如 CREATE CLASS 创建一个新的类定义。

（2）数据格式化命令：格式化地输入输出数据、对窗口或表单的一些特征进行处理，这类命令在设计时已被表单设计器取代。如 @ .BOX 在屏幕上画一个方框，在表单设计器中用形状（SHAPE）来代替。

（3）文件管理命令：创建和管理磁盘文件。如 RENAME 命令给一个文件更改名字。

（4）数据库操作命令：用来创建、处理和监视数据库、表，给表中的纪录或纪录指针定位，也可处理字段、表间的关系等。如 REPLACE 命令用来更新字段中的纪录。

（5）键盘和鼠标命令：使键盘或鼠标执行某一操作。如 MOUSE CLICK 命令让鼠标执行一个单击动作。

（6）系统设置命令：这类命令一般格式为“ SET …… ”，它们在较大范围中管理 Viaual Foxpro 的系统和环境参数。如 SET HOURS 命令将系统时间设置为 12 小时制或 24 小时制。

（7）网络命令：允许多个用户在网络中共享数据。

（8）其他命令：还有一些命令如 SQL 命令、时间命令、窗口命令等。

系统内存变量

系统内存变量是 Visual Foxpro 系统创建并维护的内置内存变量，它们的名字都以下划线开头。在 Visual FoxPro 6.0 版本中，许多内存变量的功能已经被新的功能所替代，在设计时尽量不要再使用它们。但仍有许多系统内存变量能给用户设计程序带来方便，如可以使用_SHELL 来调用外部程序文件。

流程控制语句

Visual FoxPro 6.0 全面支持面向对象编程和事件驱动，因此在总体上它是在响应事件后才产生相应操作，不同于传统的高级语言自上而下逐条执行语句。但是在每个事件代码的小范围内，它的语言结构与流程控制与传统的高级语言基本相同，也给用户提供了三种基本结构语句：顺序语句、条件语句和循环语句。用户可以把各种问题分解成这三种基本结构语句的组合，然后使用相应代码来解决问题。

顺序语句

准确来说，程序整体上都是顺序的，如果没有专门的命令来改变程序的执行顺序，Visual FoxPro 6.0 默认为从上而下逐条顺序执行。下面命令的执行情况能说明这一点。

```
？"开始执行"
WAIT
？"程序结束"
```

用户应先把这些语句存入一个命令 （COMMAND） 文件中，然后编译并执行这个文件。关于命令文件的具体内容，见下一章。

条件语句

Visual FoxPro 6.0 中的条件语句有三种：

① IF..ELSE..ENDIF

② IFF（ ）

③ DO..CASE..ENDCASE

前两种语句的使用方法基本相同，后一种语句主要用于条件较多且类似时，也叫分支语句。下面是几个例子：

```
IF number<20          &&判断条件是否成立
? number              &&成立时执行的语句
ELSE
?number-10            &&不成立时执行的语句
```

```
ENDIF

IIF（score>85）
?"成绩优秀"              &&条件成立时执行

DO   CASE
CASE     score<60
   WAIT WINDOWS "成绩不及格!"
CASE     score>=60 AND score<85
   WAIT WINDOWS "成绩及格!"
CASE     score>=85
   WAIT WINDOWS "成绩优秀!"
ENDCASE
```

循环语句

循环就是反复地执行某一操作，如果需要重复执行某一段代码，就可以把这段代码放到 循环体中，同时添加相应的条件语句来控制退出循环。常用的循环语句有两类：

① FOR..ENDFOR

② DO WHILE ..ENDDO

③ SCAN..ENDSCAN

FOR **循环运行的速度较快，且语句简单；但是如果不知道循环进行的次数时，就需要利用** DO **循环；而** SCAN **循环用于对表中的每一个纪录执行一组操作，由于使用了** Rushmore **技术，它运行的速度很快。看下面的例子：**

```
SUM=0
FOR     I =1    to 100          &&每次循环 I 加 1，直至 I 等于 100
   SUM=SUM+I                    &&循环中的语句
ENDFOR

SUM=0
I=0
DO WHILE    SUM<=5500     &&条件成立时才执行循环
   I=I+1
   SUM=SUM+I
ENDDO

SCAN
   REPLACE   工资   WITH   工资*1.1
ENDSCAN
```

下面的语句将执行无限循环：

```
DO   WHILE   .T.
   .……
ENDDO
```