本书导读

Visual FoxPro 是由 Microsoft 公司推出的用于中小型企业的一个优秀的小型数据库开发系统。它从单机数据库 FoxBASE 发展而来，现已发展成为具有很强的网络功能的小型数据库系统。它可以用作小型的单机、网络数据库，也可以作为大型数据库的前端开发工具来开发客户机/服务器应用程序。

在目前众多的开发工具中，Visual Basic 、Delphi 等都具有访问数据库的功能，PowerBuilder 更是内置了数据库并设计了自己专用的大型数据库接口。由于 Fox 系列数据库在人们印象中一向是功能简单的单机数据库，所以有些程序员认为 Visual FoxPro 不是一个好的数据库开发工具，甚至传言，Microsoft 将放弃对 Visual FoxPro 的继续升级。事实证明，Microsoft 没有放弃 Visual FoxPro，Visual FoxPro 仍然广泛使用。Visual FoxPro 在数据库开发工具中有着不可替代的地位。与 Visual FoxPro 相比较，用 Visual Basic 可以通过 Jet 和 ODBC 访问数据库，但是使用起来很不方便，在 Visual Basic 6.0 中首次引入了数据环境的概念以加强对数据库的访问，而这在 Visual FoxPro 中早已广泛使用；Delphi 号称是最好的客户机/服务器开发工具，但是在数据库方面还是通过 ODBC，速度上并没有优势，况且帮助太少了，几乎没有多少例子可以参考；PowerBuilder 使用的是自己的 ODBC 驱动连接大型数据库，据说可以比平常的 ODBC 快 10 倍，并且使用自己的数据库。但在易用性方面，笔者认为它和 Visual FoxPro 是无法比拟的。让我们再来看看 Visual FoxPro。第一，它有广泛的用户群；第二，其内置的数据库已经相当完善，并且具有网络功能，可以独立作为网络数据库；第三，能够使用 ODBC 连接大型数据库，符合国际标准；第四，完善的联机帮助，易于学习、使用；第五，由于 Microsoft 的霸主地位是无人能及的，通过 Microsoft 把最新的技术不断地加入进来，相信 Visual FoxPro 的明天会更好。

第一部分 基础篇

目录

第二部分 提高篇

第 1 章　基础知识

什么是数据库 很简单,就是数据存放的地方。在我们的周围有许多数据库的例子,比如您的通讯录就是一个小型数据库,图书馆则一个典型的大型数据库。通过数据库能干些什么呢 请想想您的通讯录能干什么 :您可以查找朋友的电话号码,可以向其中添加一个新朋友的通讯地址,可以修改已发生变化的记录,还可以删除没有用的记录。这些都是数据库的基本功能。它管理数据,而您通过它查询、修改、添加、删除其中的记录。

数据库可以分为两种 :小型数据库和大型数据库。大型数据库主要用于网络,如 Oracle、SQL Server、Sybase 等。小型数据库则在小型企业和个人应用中有广泛的市场,如 Visual FoxPro、Access 等。大型数据库无论从安全性还是稳定性来说都比小型数据库要好得多。那为什么不用大型数据库却还要使用像 VisualFoxPro 这样的小型数据库呢 这是由于 VisualFoxPro 具有大型数据库所没有的优点 :Visual FoxPro 是由 dBase、FoxBASE + 、FoxPro 一步步发展起来的,有很广泛的用户和群和一大批使用 FoxPro 的程序员 ; 它对计算机的配置要求很低, 您的计算机只要能够运行 Windows95 就可以使用 Visual FoxPro ;相对于大型数据库来说,它很容易使用,我们只需花费很少的时间就可以熟练地掌握它 ;它支持大型数据库通用的 SQL 语言,因此可以作为大型数据库的前端开发工具 ;而且它现在具有很强的网络功能,您也可以将它作为网络上的数据库服务器。相反,大型数据库以牺牲硬件为代价实现很强的网络功能、安生性和稳定性, 用户通常需要单独购买一个很高配置的计算机专门作为数据库服务器, 同时大型数据库中繁琐的操作, 复杂的功能使一些初学者望而却步。对于一般用户我认为是没有必要的。当然,如果条件允许的话,还是使用大型数据库好, 因为在功能和安全性上小型数据库是无法和大型数据库的优势相比的, 但是在小型数据库已经足够的情况下就不一定要使用大型数据库了。不要认为小型数据库已经没有市场了,微软之所以没放弃继续升级 Visual FoxPro,原因是小型数据库的使用还很广泛。

那么 Visual FoxPro 是什么呢 VisualFoxPro 是由微软开发的可视化的数据库管理系统,它全面采用面向对象的编程思想, 成为在 Windows 环境下开发小型数据库应用系统的有力工具, 目前版本是 6. 0。在客户 / 服务器应用程序中,Visual FoxPro 是一个理想的前端开发工具。它为您提供组织信息、运行查询、创建集成的关系型数据库系统以及为最终用户编写功能全面的数据管理应用程序的所有工具。

1. 1　了解数据库基本术语

用 VisualFoxPro 创建数据库,需要先了解一些数据库的基本术语。

表 1 - 1 是一个简单的表。在这个表中每一列是一个数据项, 代表一个属性, 被称为字段 Field ;每一行表述一个学生的信息,被称为记录 Record 。记录是有顺序号的,先输入的排在前面,后输入的排在后面,这叫做物理顺序号。

字段 :字段是指一个记录的某个特定区间或一组信息中的某个特定部分,赋于特定的名称。它一般是记录中最小的可识别部分,一个字段包括特定的数据,例如客户名称、公司名等,表 1 - 1 有客户、公司名称、联系人姓名、地址、城市邮政代码字段。

表：表是关系型数据库的基本结构，以记录（行）和字段（列）的形式存储数据，数据常是关于某一类事物的信息，例如姓名、学号、品名、价格、数量等。

表 1-1 客户通讯录

客户ID	公司名称	联系人姓名	地址	城市	邮政编码
ALFKI	三川实业有限公司	刘小姐	大康明路 50 号	天津	343567
ANATR	东南实业	王先生	承德西路 80 号	天津	234575
ANTON	超森行贸易	王炫皓	黄台北路 780 号	石家庄	985060
ABOUT	国顶有限公司	方先生	天府东街 30 号	深圳	826879
BERGS	通恒机械	黄小姐	东园西甲 30 号	南京	799099
BLAUS	森通	王先生	常保路东 80 号	天津	787045
BLONP	国际	黄雪玲	广发北路 10 号	大连	565479
BOLID	迈多贸易	陈先生	福星大街 60 号	西安	907987
BONAP	祥通	张先生	花园东街 90 号	重庆	557890

记录

字段

记录：表中的每一行称为记录，每一条记录代表一个信息，在表中不允许出现相同的记录。

数据库：是指存放在计算机存储设备中的相互关联的数据文件的集合。在 VisualFoxPro 中以表的形式记录数据内容。单独使用表，可以为我们存储和查看信息提供很多帮助。但是如果把若干表组织到一个数据库中，您就可以充分利用 VisualFoxPro 提供的强大功能来使用和管理它们。例如在学生数据库中可以用一个表保存学生基本情况，一个表保存学生个人表现，这样您在打印一张学生获奖情况报表时（表 1-2），报表中的信息来自“基本情况”和“个人表现”两个独立的表。把表放入数据库中，可以减少冗余数据的存储，保护数据的完整性。

表 1-2 学生获奖情况

学号	姓名	性别	年龄	获奖学金
20010918	王杰	男	18	一等
20010919	李春生	男	19	二等
20010920	刘英	女	18	三等

基本情况

个人表现

关系：是指表之间的一种链接，它允许您不仅能从当前选定表中访问数据，而且可以访问其他表中的数据。这种链接指的是联接条件。

关系型数据库：是指根据表、记录和字段之间的关系进行组织和访问的一种数据库。这些表可以用两个字段的公用值链接在一起。例如表 1-1 和表 1-2 就可以用学号链接在一起。

索引：是指由设计者建立的按一定顺序编辑的文件内容的目录表，包括标识或定位这些内容的关键字（Key）或引用标志。人们建立索引的目的是使用它进行查询，当您收集了许多信息，查看它们时最好使用索引，比如您想知道学生根据数据学分数的排名顺序，应该按“数学”索引表；想了解根据英语成绩排名的情况，应该按“英语”索引表。记录在表中是以物理顺序排列存放的，物理顺序反映了向表内输入记录的先后顺序。索引不改变表内记录的物理排列顺序，索引是以关键字排序，按“数学”字段索引表（表 1-3），数学字段就是关键字，按“英语”字段索引表，英语字段就是关键字。

表 1-3 按数学分数排序表

学号	姓名	数学	化学	英语
20010918	王杰	98	78	99
20010919	李春生	95	96	84
20010920	刘英	89	99	78
20010921	包玉杰	88	78	99
20010922	张明生	81	96	84
20010923	李虹英	77	99	78

查询：查询是指搜索那些满足指定条件的记录，可以根据需要对这些记录排序和分组，还可以将查询结果创建为报表、表及图形。查询的来源是表，查询结果可以是新的表、报表或图形。表 1-4 是对健康状况为“良好”的学生的查询结果。

表 1-4 学生身体健康者

姓名	性别	年龄	健康状况
王杰	男	21	良好
李春生	男	17	良好
刘英	女	18	良好
包玉杰	女	22	良好
张明生	男	22	良好
李虹英	女	17	良好

视图：视图是用来查询或更新数据的一种方法，它具备了表和查询的一些特点。视图是一个虚表，它所对应的数据库存储在表中，当运行视图时，数据才填充到视图中。表 1-5 对视图与表、查询进行了比较。

表 1-5 视图与查询的比较

视图与表		视图与查询	
相同点	不同点	相同点	不同点
具有浏览、更改、使用的功能	信息来源于表，表内记录变动视图也变动，视图不能独立存在、只能保存在数据库中	从一个或多个相关联的表中提取有用信息	查询的程序是完全独立的，不依附于任何数据库和表；而视图则是查询程序和表的组合，你只能按照操作表的方法使用它

表单：表单是用户界面。建立数据库后，数据库中的信息在不断变化。比如对于一个学校，每年都要有新生入校、老生离校，为了维护数据库，就需要对数据库执行添加、删除编辑等操作，您可以使用数据库命令在命令窗口 CommandWindow 中直接对它操作，但更为简单的办法是使用表单。表单为数据库信息的显示、输入和编辑提供了直观、便捷的窗口。窗口中使用的命令按钮、文本框、下拉式列表框、复选框、选项卡、选项按钮、微调钮等均可以在 VisualFoxPro6.0 中找到，并将它们放到表单中使用。使用 VisualFoxPro 进行程序设计，您将不再是单纯地从代码的第一行一直编到最后

一行，而可以把精力放在考虑如何创建表单中的对象，使用户尽可能方便和直观地完成信息管理工作，而您又能利用对象来简化程序设计。

下面我们就花十分钟时间用 VisualFoxPro6.0 来建立一个功能齐全的数据库应用程序。

1.2 VisualFoxPro 程序设计举例

以下的介绍需要您对 Windows95 或 Windows98 的操作比较熟悉，知道如何选择菜单、使用鼠标和鼠标用法的术语 单击、双击、拖动 等，掌握了操作 Windows 的方法，会打开、关闭文件、会移动、缩放和关闭窗口……如果您对 Windows95 或 Windows98 还需要了解，请先查阅相关书籍。

如果您的系统内已经安装了 VisualFoxPro6.0 请执行以下步骤：

1 选择开始◇程序◇Microsoft Studio6.0◇MicrosoftVisualFoxPro6.0 进入图 1-1 VisualFoxPro6.0 的主画面。

图 1-1 VisualFoxPro6.0 主画面

2 选择文件◇新建，打开图 1-2“新建”对话框，对话框中列出了 VisualFoxPro6.0 可以生成的全部文件类型。

图 1-2 新建对话框

3 单击“表”旁边的选择按钮，选择“新建文件”，出现“创建”对话框 图 1-3 。在该对话框，系统询问您要将文件保存在哪个磁盘，哪个文件夹及被保存的文件的名字。

图 1 - 3　创建对话框

4 选择要保存文件的磁盘、文件夹，输入表文件的名字“人事 1”。

5 单击“保存”按钮，出现表设计器，如图 1 - 4 所示。

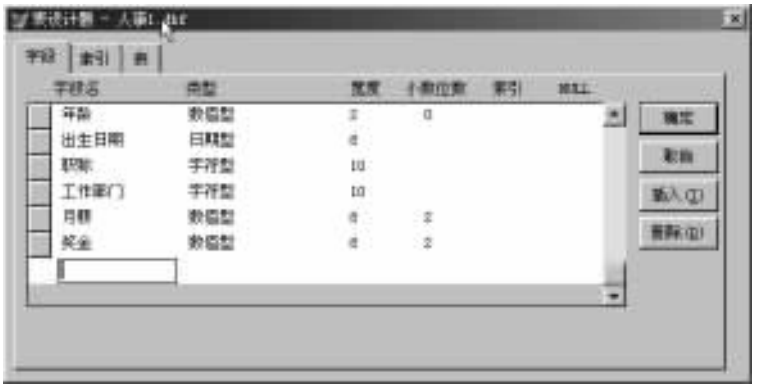


图 1 - 4　表设计器

在这里我们通过表设计器创建学号、姓名、成绩等字段，并通过类型、宽度设置各字段的特性，其结果将如图 1 - 4 所示。

6 设计完字段后，单击“确定”按钮，退出“表设计器”，系统会询问“现在输入记录吗　” 图 1 - 5 请单击“否 N ”。

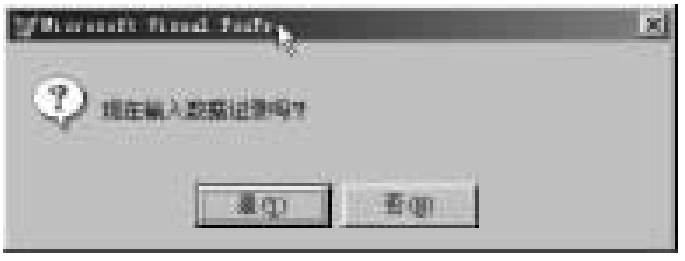


图 1 - 5　是否立即输入记录

7 选择文件➡新建，打开图 1 - 6“新建”对话框。

图 1-6　新建对话框

8 单击“表单”旁边的圆按钮,选择“向导”,启动了图 1-7 表单“向导选取”对话框。

图 1-7　表单向导选取

图 1-8　表单向导步骤 1

9 选择“表单向导”,单击“确定”按钮,,进入表单向导步骤 1 图 1-8 “选择字段”,在“数据库和样表”栏中选择“成绩 1”,“可用字段”就列出了成绩 1 表中的所有字段,单击“►►”按钮,将“可用字段”中所有字段都移到“选定字段”中。

10 单击“下一步”按钮,进入表单向导步骤 2 图 1-9 ,在这里选择表单样式,请选择“新奇式”。

图 1 - 9　表单向导步骤 2

11 单击“下一步”按钮,进入表单向导步骤 3 图 1 - 10 。

图 1 - 10　表单向导步骤 3

12 单击“下一步”按钮,进入表单向导步骤 4 图 1 - 11 。

13 单击“完成”按钮。

14 单击“保存”按钮。

进入图 1 - 12“另存为”对话框。在这里选择磁盘和文件夹,输入文件名。请记住保存文件的磁盘、文件夹和文件的名字,以后运行时要查找它们。

至此我们用向导建立了一个表单,下面看看结果是否满意。

1.3　数据库应用系统的组成

数据库应用系统的目的是把数据库、表、表单和报表汇集起来,把用人工管理的数据实现电脑化管理。例如对学生的管理、对职工的管理、对商品的管理等。数据库应用系统主要由用户界面、信息处理、数据库管理、数据库、辅助功能等模块组成,如图 1 - 11 所示,图中各模块解释如下:

用户界面:由于数据库应用系统是一种面向最终用户的应用系统,用户界面应该具有友好、简单、易操作等特点,它是一个系统能否被用户接受的非常重要的因素之一,使用 VisualFoxPro6. 0 提供的工具能够制作出多窗口系统、菜单驱动的用户界面,并可以将图形、图像、动画、声音等多媒体对象添加到界面中。

信息处理：信息处理是建立数据库应用系统的目的，其基本功能包括各类信息的查询、统计、报表打印等。

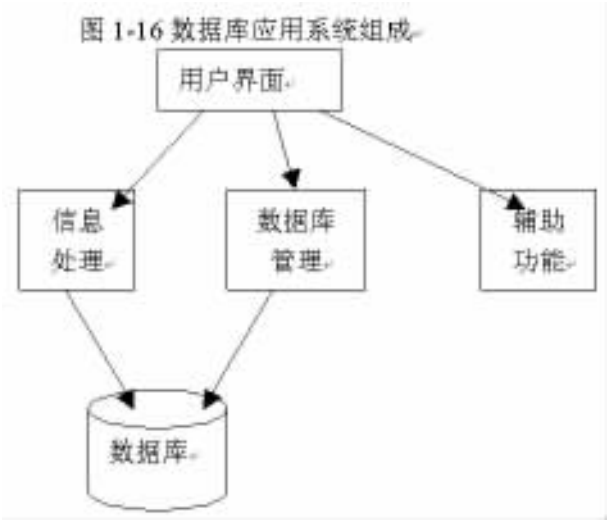


图 1 - 11　数据库应用系统的组成

数据库管理：指数据库表的添加、修改及删除等。

数据库：即数据库应用系统的操作对象，包括表及视图等。

辅助功能：是指数据库应用系统中的菜单、帮助文本、系统安装及版本信息等。

1. 4　用 VisualFoxPro 建立数据库应用系统

VisualFoxPro 应用系统通常由以下几部分组成：数据库　用来存放数据　、主程序　用来设置应用程序的系统环境、执行第一用户界面和读取事件程序的开始点　、用户界面　表单、工具栏和菜单等　、查询和报表　允许用户检索或输出自己的数据　。在设计数据库的时候，首先要分离那些需要作为单个主题而独立保存的信息，然后告诉 VisualFoxPro 这些主题之间有何关系，以便在需要时把正确的信息组合在一起。通过将不同的信息分散在不同的表中，可以使数据的组织工作和维护工作更简单，同时也能够保证建立的应用程序，其性能具有较高的可塑性。

本节以一个大家较容易理解的学生学籍管理为例，来讨论用 VisualFoxPro6. 0 设计数据库应用系统各主要模块的方法和技术，介绍 VisualFoxPro 的编程技巧。书中大部分范例都是围绕这个设计编写的。

1. 4. 1　分析数据

1. 分析数据需求

VisualFoxPro 数据库设计的第一步是明确数据库的目的和如何使用，也就说您需要从数据库中得到哪些信息。明确目的之后，就可以确定您需要保存哪些主题　表　，以及每个主题需要保存哪些信息　表中的字段　。在我们的范例中要实现如下功能并相应地建立如表 1 - 6 所示。

表 1 - 6　功能与所需表

要实现的功能	需建立的表
新生入校时添加	基本情况、社会关系
日常考试成绩、奖惩的处理查询学生的各种信息	成绩、个人表现

2. 将需求分类

通过将信息拆分入表，来增加数据库的灵活性。本系统数据的设计有基本情况、社会关系、成绩、个人表现 4 类。

3. 确定所需字段及关系

确定所需字段，是要确定在每个表中要保存哪些信息。在表中，每类信息称作一个字段，浏览表时同一字段的数据在表中显示为一列。由于各表都需要姓名，为了减少冗余，只有基本情况表保存姓名，而其他表用学号与基本情况表关联，以获取姓名。分析每个表，确定一个表中的数据和其他表中的数据有何关系。必要时可以在表中加入字段或创建一个新表来明确关系。

在最初的设计中，不要担心发生错误或遗漏东西。这只是一个初步方案，您可以在以后对设计方案进一步完善。VisualFoxPro 很容易在创建数据库时对原设计方案进行修改。当然在数据库中输入了数据或连编表单和报表之后，再要修改这些表就困难得多。因此，在连编应用程序之前，应确保设计方案已经考虑得比较全面。

1.4.2　程序设计的过程

程序设计大体要经过下面几个步骤：

1　创建数据库、表，利用 VisualFoxPro 的工具创建数据库、表，并设置表的索引和表间的关系。

2 创建查询、视图和报表，根据需求，创建对数据的查询、视图和报表。

3 创建适合的类，利用 VisualFoxPro 的基类，创建适合的类。

4 创建表单，通过表单将数据库、表、视图、报表集成起来，用类对它们进行操作。

5 创建程序，编制程序将表单连接成一个系统。

6 调试、连编，利用调试工具检查、修改程序错误，最终编译成应用程序文件 app 或者可执行文件 .exe 。

1.5　项目管理器 ProjectManager

项目是数据库应用系统要处理的文件、数据、文档和 VisualFoxPro 对象的集合，而项目管理器则是 VisualFoxPro6.0 的控制中心，处理数据和对象的主要组织工具。

当您将数据库、表、表单等放入项目管理器后，需要时，只要打开项目管理器，就可以直接从项目管理器中读取，而不必再去磁盘中搜索需要的文件。您可以用项目管理器建立数据库、表、表单、报表等，也可以把已有的 DBF 文件添加到一个新的项目中。下面通过建立一个“STUDENT”的项目，了解项目管理器的组成。

1.5.1 新建项目管理器

1 在 VisualFoxPro 中选择文件 ◊ 新建,打开图 1 - 12 所示的“新建”对话框。

图 1 - 12 新建项目

2 在对话框中选择“项目”,单击“新建文件”按钮,接着打开图 1 - 13 所示的“创建”对话框,请在对话框中选择磁盘、文件夹,输入项目名“STUDENT”。

图 1 - 13 创建项目对话框

3 单击“保存”按钮,窗口上出现如图 1 - 14 的“项目管理器”界面,它是一个具有多个选项卡的对话框,其中列出了项目可以管理的文件类型。

“项目管理器”为数据提供了一个组织良好的分层结构视图。若要处理项目中某一特定类型的文件或对象,可选择相应的选项卡中的内容。

“数据”选项卡:包含了一个项目中的所有数据,其中有数据库、自由表、查询和视图。

“文档”选项卡:包含了处理数据时所用的全部文档,输入和查看数据所用的表单,以及打印和查询结果所用的报表及标签。

图 1 - 14　展开后的项目管理

“类”选项卡:包含用户创建的类。

“代码”选项卡:包含用户编制的程序。

“其他”选项卡:包含用户编制的菜单、文本文件和其他文件。

“项目管理器”中的项是以类似于大纲的结构来组织的,我们可以将其展开或折叠,以便查看不同层次中的详细内容。图 1 - 15 是展开后的项目管理器。

图 1 - 15　展开后的项目管理

如果项目中具有一个以上某一类型的项,其类型符号旁边会出现一个“ + ”号。单击类型符号旁边的“ + ”号可以显示项目中该类型项的名称,单击类型项名旁边的“ + ”号可以看到该项的组件。若要折叠已展开的列表,可单击列表旁边的“ - ”号。

1. 5. 2　改变显示外观

“项目管理器”显示为一个独立的窗口。您可以移动它的位置、改变其尺寸或者将它折叠起来,只显示选项卡。

一、移动“项目管理器”

将鼠标指针指向标题栏并按下鼠标左键,可以将“项目管理器”拖到屏幕上的其他位置。

二、折叠“项目管理器”

单击图 1 - 15 中项目管理器右上角的上箭头，就可以折叠项目管理器，此时只显示选项卡的标题部分 图 1 - 16 。

图 1 - 16　折叠后的项目管理器

三、还原“项目管理器”

单击图 1 - 16 中右上角的下箭头，就可以恢复显示完整的项目管理器。

四、拖开某一选项卡

折叠“项目管理器”后，可以拖开选项卡，并根据需要重新安排它们的位置。

1 折叠“项目管理器”；

2 选定一个选项卡，将它拖离“项目管理器”。

当选项卡处于浮动状态时，通过在选项卡中单击鼠标右键可以访问“项目”菜单中的选项。如果您希望选项卡始终显示在屏幕的最顶层，可以单击选项卡上的图钉图标，将选项卡设置成“顶层显示”，这样，该选项卡就会一直保留在其他 VisualFoxPro 窗口的上面。再次单击图钉图标可以取消选项卡的“顶层显示”设置 图 1 - 17 。

图 1 - 17　脱离项目器的选项卡

五、还原选项卡

单击图 1 - 17 中选项卡的“ × ” 关闭 按钮，或将选项卡拖回到“项目管理器”，就可以还原选项卡的显示。

六、停放“项目管理器”

将“项目管理器”拖到 VisualFoxPro 主窗口的顶部停放。这样它就变成窗口工具栏区域的一部分。“项目管理器”处于停放状态时，不能将其展开，但是可以单击每个选项卡来进行相应的操作。对于停放的“项目管理器”，您同样可以从中拖开选项卡。

当您不小心将“项目管理器”拖得不知那里去，在屏幕上找不到它时，请将常用工具栏拖开，看

看是不是隐藏在常用工具栏下面。

1.6　VisualFoxPro6.0 的用户界面

因为您所掌握的 Windows 窗口、下拉式菜单、工具栏及对话框的操作在这里全部适用，在此只介绍了 VisualFoxPro6.0 与其他 Windows 应用程序在用户界面上的不同点。

1.6.1　动态菜单

几乎所有的应用程序都具有菜单或工具栏。菜单向用户展示应用程序的各种功能，同时占用很少的屏幕空间，而动态菜单则是当程序执行某项功能时，系统会动态地增加或修改一些菜单项。图 1 - 18 是没有打开项目管理器的主菜单，而图 1 - 19 则是打开项目管理器后的主菜单，可以看出，打开项目管理器后，在主菜单中增加了“项目”没有了“格式”。同样在执行其他操作时也会出现这种情况，请留意。

图 1 - 18　没有打开项目管理器的主菜单

图 1 - 19　打开项目管理器的主菜单

1.6.2　快捷菜单

VisualFoxPro6.0 中众多工具条、对话框、设计器、窗口及生成器等都具有快捷菜单。快捷菜单是指当用户处于某个特定区域时单击鼠标右键而弹出的菜单项。快捷菜单的选项大多数都可以在相应的主菜单中找到，VisualFoxPro6.0 提供它是给您一个快捷方式。图 1 - 20 是在项目管理器顺的表中单击右键得到的快捷菜单。

1.6.3　工具栏

在 VisualFoxPro 中每种设计器都有一个或多个工具栏，您可以很方便地使用大多数常用的工具。例如，表单设计器就有用于控件、控件布局以及调色板的工具栏 图 1 - 21 。

在工作时，您可以根据需要在屏幕上放置多个工具栏。通过把工具栏停放在屏幕的上部、底部或两边，可以定制工作环境。系统能够记住工具栏的位置，再次进入 VisualFoxPro 时，工具栏将处在最近一次关闭时所在的位置上。

图 1 - 20　快捷菜单

图 1 - 21　众多的工具栏

1.6.4 “命令窗口 Command Window

近几年，开发程序的趋势已经从编写大量代码逐渐转向使用“可视化”语言，这种语言往往只需编写少数代码就能完成工作。但有些编程人员可能会一时不适应这种环境，因此 VisualFoxPro6.0 和早期的版本提供了一个“命令”窗口，它取代了 FoxBASE 中圆点提示符的位置，所有的 VisualFoxPro6.0 命令、函数等都可以在“命令”窗口输入、执行。

一、命令窗口的功能

命令窗口能够将在 VisualFoxPro 开发环境中所进行的每一步操作都以代码形式显示出来，利用这一特点，您可以很快掌握 VisualFoxPro 的一些常用代码，而且这些代码基本上在编程时都可以直接使用。

在命令窗口直接输入命令时，如果命令中有 VisualFoxPro 关键字，系统将会以显著颜色表示，通过这一特点，可以帮助您记住一些常用命令。如果在输入命令时出现拼写或语法错误，VisualFoxPro6.0 会显示错误提示框，提醒您修改。

在命令窗口输入 FoxPro 表达式并检测其运行结果。这在编写或调试程序时显得特别有用。

当选择菜单命令时，相应 VisualFoxPro6.0 语句自动反映在“命令”窗口中。

将光标移到以前命令行的任意位置按 Enter 系统将重新执行该命令。

若要分割很长的命令,可以在所需位置的空格后键入分号,然后按 Enter 键。

二、输入和编辑命令

“命令”窗口是一个可编辑的窗口,我们可以在“命令”窗口中执行插入、删除、复制、移动等操作,还可以用光标键或滚动条在窗口中上下移动。如果要输入一条和上次相差不多的命令时,可以:

1 将光标移到该命令处;

2 修改该命令;

3 按回车键。

系统将执行该命令,看上去原来的命令被修改了,但当新命令执行完后,会看到,您的上一条命令还在“命令”窗口中,而窗口底部增加了刚执行的那条命令。

三、“命令”窗口的快捷菜单

在“命令”窗口中单击鼠标右键,可以显示一个具有下列选项的快捷菜单 图 1 - 22 :

图 1 - 22　“命令”窗口的快捷菜单

剪切、复制、粘贴 :在“命令”窗口中移动或删除字符。

生成表达式 :显示表达式生成器窗口,在该窗口中可以使用命令、字段等定义一个表达式。当您单击“确定”时,所生成的表达式就粘贴到“命令”窗口中。

运行所选区域 :将“命令”窗口中选定的文本作为新命令执行。

清除 :从“命令”窗口中移去以前执行命令的列表。

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、字体和语法及着色等选项。

1.6.5　其他

设计器 :设计器为我们提供一个界面,通过它可以迅速地创建表、表单、类、报表、菜单、查询以及视图。用设计器创建新文件的方法是:

在“项目管理器”中选择待创建文件的类型 ,单击新建就可以启动相应的设计器,有关设计器种类请参看附录。

生成器 :生成器是选项卡对话框,用于简化创建和修改表单、复杂控件等的操作。每个生成器显示一系列选项卡,用于设置选中对象的属性。从表单上选择控件,接着快捷菜单中选择生成器命令,就会启动相应的生成器。有关生成器种类请参看附录。

向导 :VisualFoxPro6. 0 提供了大量的向导,可以为您创建功能齐全的数据库、表、表单、报表及查询,而您无需编写任何代码。有关向导种类参请看附录。

第 2 章　数据库与表

与传统的小型数据库相比，Visual FoxPro 的数据库不仅用于存储数据，而且用于存储数据库表、本地视图、远程视图、连接和存储过程等。

有关数据库和表的详细内容，请参阅本书第七章及联机文档《用户指南》中的第二章“创建表和索引”、第三章“将表加入数据库”。

2.1　数据库文件包含什么

数据库文件的扩展名 DBC Database ContainerFile ，实际上它是一个二维表，数据库中的每个元素为该表中的一条记录，即它记载着数据库表、本地视图、远程视图、连接和存储过程的引用，我们可以用数据库设计器来创建它。

表：是用于存放数据的地方，分数据库表和自由表两种，可以用表设计器来创建。

数据库表：指包含在数据库之内的表，借助于 Visual FoxPro 新增的数据字典，数据库表具有很多特点，如：长表名、长字段名、默认值、字段级规则等，其扩展名为 DBF。

自由表：不属于任何数据库，与 FoxBase、FoxPro 等传统的微机上的小型数据库相同，其扩展名为 . DBF。

您可以将自由表转换为数据库表，也可以将数据库表转换为自由表。当数据库表转换为自由表后，数据库表的长表名、长字段名、默认值、字段级规则等特点将自动丢失。

视图：是从一个或几个表 或视图 导出的表，视图是一个虚表，它所对应数据存储在表中，当运行视图时，数据才填充到视图中。

连接：是指在程序设计中，为程序的各模块之间传递控制命令和参数，把他们组成一个可执行的整体的过程。在创建远程视图之前，必须先建立远程数据源的连接。在激活连接之前，连接只是一个定义，它作为一条记录保存在数据库文件中。当您使用远程视图时，Visual FoxPro 创建一个活动连接，与远程数据源相连，然后把这个活动连接作为管道与远程数据源交换数据。

存储过程：是保存在数据库中的过程，该过程包含用户自定义函数中的任何命令和函数。

2.2　创建数据库

当对数据库进行了设计之后，就可以通过“项目管理器”来建立数据库了。

请打开在第一章建立的 STUDENT 项目，我们将数据库建立在该项目中，让 STUDENT“项目管理器。来管理下面要建立的数据库文件。

1 在“项目管理器”中选择“数据”选项卡 图 2 - 1 ；

2 在列表项中选择“数据库” 图 2 - 1 ；

3 单击“项目管理器”右侧的“新建”按钮 图 2 - 1 ；

图 2 - 1　新建数据库对话框

出现“新建数据库”对话框。该对话框有两个选项，一个是“新建数据库”，使用它来创建数据库，另一个是“数据库向导”即在向导的帮助下来创建数据库，在这里，我们自己来创建数据库，所以选择“新建数据库”。

4 在对话框中选择“新建数据库” 图 2 - 1 。

出现图 2 - 2 的“创建”对话框。单击“保存在”下拉式列表，选择要保存数据库的磁盘和文件夹，然后在“数据库名”的右边输入“人事工资”，单击“保存”按钮。

Visual FoxPro 是自动增加 DBC 扩展名。请不要改变扩展名 DBC，因为 Visual FoxPro 默认此扩展名的文件作为数据库文件。系统在指定的磁盘、文件夹中建立数据库“学生”后进入“数据库设计器” 图 2 - 3 ，在此窗口，我们可以向数据库添加表。

图 2 - 2　创建数据库对话框

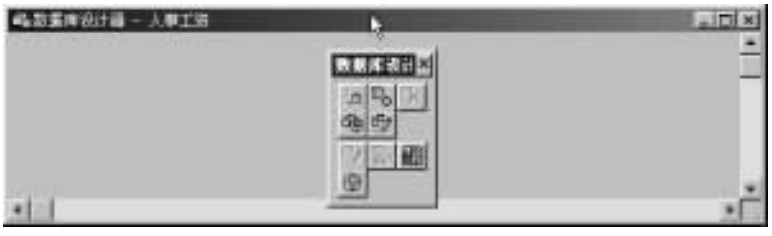


图 2 - 3　数据库设计器

2.3 创建表

在“数据库设计器”中中添加的表是数据库表 ;而我们在“1.2Visual FoxPro 程序设计举例”一节中创建的自由表，使用图 2-3 的“添加表”按钮可以将自由表添加到数据库中，使其成为数据库表。而该图的“新建表”按钮，则可以创建数据库表。

如果要了解“数据库工具栏”中按钮的功能，只要将鼠标移动到按钮上，停留几秒钟，说明性文本就会被显示。

2.3.1 在数据库中建立新表—“基本情况”表

下面我们在新建的数据库中创建一个新表，方法为：

1 在“项目管理器”中选择“数据”选项卡；

2 单击“数据库”；

3 选择数据库名“人事工资”；

4 单击“表”；

5 单击“新建”按钮，出现“新建表”对话框 图 2-4 ，在对话框中有两个按钮“表向导”和“新表”。如果选择“表向导”，就要按照向导的提示，一步一步建立新表。选择“新表则自己建立表。下面我们自己来建立表。

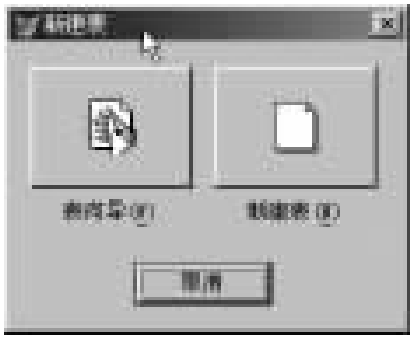


图 2-4 新建表对话框

6 单击“新表”按钮，出现类似图 2-2 的创建表对话框。单击“保存在”下拉式列表，选择要保存数据库的磁盘和文件夹，然后在“输入表名”的右边输入“人事记录”，然后单击“保存”按钮。

Visual FoxPro 自动在表名后增加 DBF 扩展名。系统在指定的磁盘、文件夹中建立数据库表“基本情况”后进入“表设计器” 图 2-5 。

2.3.2 表字段的操作

字段是生成表后数据所占的位置。在“表设计器”中设置新表时，请注意以下几项要求：

第一，字段的数据类型应与将要存储在其中的信息类型相匹配 有关 Visual FoxPro6.0 的数据类型请参阅附录 。第二，使字段的宽度足够容纳将要显示的信息内容。第三，为“数值型”或“浮点型”字段设置正确的小数位数。

图 2 - 5　表设计器

在图 2 - 6 的表设计器中,我们可以:

1 输入字段名,最长可达 128 个字符;

2 选择字段类型;

3 设置字段宽度;

4 规定数值字段的小数位数;

5 输入字段的标题;

6 建立索引。

本书为了解释的清楚,我们在文件名和字段名中使用了汉字,但这并不是一种好方法。这会给代码编写带来麻烦,因此实际开发时最好使用字母。

您可以在字段名中使用字母而在其标题中输入汉字,这样程序在运行时使用字母表示的字段名,而显示汉字表示的字段名的标题,使表容易被理解,同时也避免了代码编写的麻烦。

字段的命令规则

"字段名"说明字段的名字,最大长度为 128 个字符。必须以字母或下划线打头,可以包含任何字符 包括数字 ,但不允许出现空格。要避免使用过长的字段名称,防止带来许多不必要的麻烦;另外在同一表中不允许出现相同的字段名称。

若您输入的字段名是数字打头,系统将拒绝接受,并提示"无效或重复的字段名"。

1 单击"字段名"下的第一个文本框,在其中输入第一个字段的名字。

2 按 TAB 键,使光标移到"类型"栏。"类型"说明字段内所存数据的种类。

Visual FoxPro 的字段类型有 13 种 请参阅附录 ,常用的是字符型、数值型和日期型,缺省类型为字符型。单击"类型"旁的下拉式列表框,可以选择数据类型。

3 按 TAB 键,将光标移到"宽度"栏。若 Visual FoxPro 允许改变这个值,您才能改变它。当宽度值的右边出现一个微调钮,说明该宽度值可以改变。这时可以调整微调钮的数值,也可以直接输入数字来改变它。如果字段宽度太小,不能容纳全部数据,就会造成数据的丢失。例如姓名字段若设置为 6 一个汉字占两个字符位置 ,就不能容纳"上官清明"这样的名字;若字段宽度太大,就会加大表的尺寸,浪费空间。因此必须确定合适的字段宽度,保证既能容纳全部信息,又不造成浪费。

4 按 TAB 键,将光标移到下一列。若数据类型含有小数位,系统会让您输入"小数位数"。"小

数位数”也有一个微调钮，您可以通过它设置字段的小数位。若数据类型不含小数位。则会跳到“索引”列。

在设置“小数位数”时要注意小数点也要占用 1 位字段的宽度。例如您设置字段宽度为 5，小数位数是 2，则整数位数只留下 2 位。因此在设置有小数位的字段时，请仔细规划字段的“宽度”。

5 设置排序。设置排序的方法是单击“索引”的下拉式箭头，选择“⇑”或选择“⇓”。本例将“编号”字段设置为“升序”，升序是指记录按字母 A - Z 的顺序排列；降序则是按字母 Z - A 的顺序排列。有关索引的介绍在 2.7 节。

6 设置 NULL 值。

null 值既不等于 0 也不是空格，null 值是为了说明这样一种情况：在字段或记录里的信息目前还无法得到。例如，某学生的分数在填写记录时还不清楚，那么，与其存储一个有歧义的零，还不如在字段中先存储 null，直到以后再存入有实际意义的信息为止。

7 重复 1 - 4 步骤，输入另一字段。表可以包含多个字段，为了添加下一字段，只需将光标移到下一字段即可。图 2 - 6 是正在定义字段中的表设计器。

图 2 - 6　表设计器

图 2 - 7　数据库设计器

8 定义完字段,单击“确定”按钮。保存表设计结果后,系统会询问“现在输入记录吗 ”您可以回答“Yes”,现在就输入,也可以选择“No”,以后再输入。请单击“No”,返回数据库设计器 图 2 - 7 。在图 2 - 7 中数据库设计器中多了一个表,从表中可以看到表字段名。

除了使用数据库设计器创建表外,我们还可以利用“项目管理器”、菜单或命令等多种方法创建表。

2.3.3　用命令和菜单建立“个人表现”表

表 2 - 1　个人表现表的字段内容

字段名	类型	宽度	索引	标题
编号	C	4	升序	
学号	C	7		
类别	C	2		奖惩类别
时间	D	8		
地点	C	10		

1 在命令窗口键入“OPENDATABASE 人事工资”;

2 选择菜单 :文件 ◊ 新建 ;

3 在“新建”对话框中选择文件类型—“表”;

4 单击“新建文件”;

5 在“创建”对话框中输入数据库表的名字“人事记录”;

6 单击“保存”按钮 ;

7 在“表设计器”中输入表 2 - 1 的内容 ;

8 单击“确定”按钮。

不管使用哪种方法,系统均会询问 图 2 - 4 使用“表向导”还是“新表”。选择“新表”,将使用表设计器设计表 ;若选择“表向导”,则要按照向导的提示,一步一步建立新表。

2.3.4　自由表

自由表是独立于数据库之外的表,它与 FoxBASE 、FoxPro 中的 DBF 文件完全兼容,我们可以创建它, 也可以将您以前在 FoxBASE 或 FoxPro 中建立的 . DBF 数据库文件添加到 Visual FoxPro 中。

一、创建自由表

如果不想使表属于数据库,可以直接将其设计为自由表。我们在第 1. 2 节“Visual FoxPro 程序设计举例”中建立的“成绩 1”表就是一个自由表。在“项目管理器”中创建自由表的方法是:

1 在“项目管理器”中选择“数据”选项卡;

2 选择“自由表”;

3 单击“新建”按钮;

4 在“新建表”对话框中单击“新表”;

5 在“创建”对话框中输入数据库表的名字;

6 单击“保存”按钮；

7 在“表设计器” 图 2-8 中输入字段内容；

8 单击“确定”按钮。

您会看到自由表设计器 图 2-8 和数据库表设计器 图 2-6 相比，少了许多东西。这些将在后面介绍。

图 2-8 自由表设计器

二、将数据库表变为自由表

下面我们将数据库表“基本情况”变为自由表。

1 在“项目管理器”中选择“数据”选项卡；

2 单击“数据库”；

3 选择数据库名“学生”；

4 选择“表”；

5 选择表名“人事记录”；

6 单击“移去”按钮，出现图 2-9 的“移去表提示框 1”，在该提示框中有三个选项；

移去：将数据库从数据库中移去，但不删除它。

删除：将数据库表从数据库中移去，并从磁盘中删去。

取消：取消当前操作。

7 单击“移去”按钮，出现图 2-10 的“移去表提示框 2”，在该提示框中，系统提示您如果将表从数据库中移去，长表名和长字段名就不能被用于索引或者程序，并询问您是否继续；

8 单击“是 Y ”。

图 2-9 移去表提示框

图 2 - 10　移去表提示框 2

数据库表变为自由表，从数据库列表中您已经找不到它了。有关长表名和长字段名在第七章介绍。

三、将自由表变为数据库表

下面我们将自由表“成绩 1”添加到数据库中。

1 在“项目管理器”中选择“数据”选项卡；

2 单击“数据库”；

3 选择数据库名“人事工资”；

4 选择“表”；

5 单击“添加”按钮；

6 在“打开”对话框 图 2 - 11 中选择表名“成绩 1”；

7 单击“确定”按钮。

图 2 - 11　打开对话框

这样您会在数据库列表中找到它。

2.4　使用数据库操作命令

通过前面的学习，我们知道了如何在 VisualFoxPro6.0 的设计环境下创建、修改数据库和表，但您还必须了解如何使用命令对数据库和表进行操作。

命令可以在程序中编写，也可以在命令窗口中键入并执行。

2.4.1 打开数据库

格式：OPENDATABASE〔FileName｜ 〕

功能：打开一个数据库。

参数："｜"：表示"或"的意思，即任选一种，在命令中您可以使用 FileName 文件名 也可以使用" "。

"〔〕"：表示可有可无的选项。

"FileName"：指定欲打开数据库的文件名，如果省略 FileName，系统将显示 Open 打开 对话框。

" "：显示 Open 对话框，从中选择现有的数据库或输入所要创建的新数据库名。说明：在 OPEN DATABASE 后面，您可以输入数据库名，也可以用" "代替数据库名。当数据库被打开，它所包含的所有表均可以使用，但是这些表并没有实际上被打开，要打开它们必须使用 USE 命令。

例：打开数据库"学生"，可将下列语句写入程序中，或输入命令窗口执行：

OPENDATABASE 学生

2.4.2 打开表

格式：USE TableName

功能：打开指定表。

参数：TableName，指定要打开的表名字。您要对表进行操作，必须先将其打开。

例：打开数据库表"基本情况"，执行下面命令：

USE 基本情况

当执行 USE 命令时，VisualFoxPro 会寻找在当前数据库中的表，如果没有找到表，VisualFoxPro 会在数据库的外面寻找此表，这意味着如果数据库表和自由表有同样的名称时，首先打开的是数据库表。

2.4.3 显示表记录

打开表后，使用 DISPLAY 或 LIST 命令，可以显示它的记录。LIST 命令与 DISPLAY 命令很相似，不同的是 LIST 连续显示 类似于 DOS 中的 DIR 命令 ；而 DISPLAY 在满屏时暂停，并有提示 类似于 DOS 的 DIR /P 命令 。

例：打开"基本情况"表，并显示它的记录信息。

USE 基本情况

LIST

或

USE 基本情况

DISPLAY ALL

使用 LIST 和 DISPLAY 命令，记录从主窗口的左上角开始显示，如果"项目管理器"挡住显示，请用鼠标将它移开。

2.4.4　关闭表

当 USE 命令不带表名时,就关闭当前表。例:使用 USE 命令关闭当前打开的表:USE

2.4.5　关闭数据库

关闭数据库的命令是:CLOSEDATABASE 该命令关闭当前数据库和表。

例:关闭数据库“学生”,执行如下命令:

CLOSEDATABASE 学生

2.5　表记录的操作

2.5.1　浏览记录

当您的表中有数据记录时,可以浏览它。

方法一:用“项目管理器”

1　在“项目管理器”中选择要编辑记录的表

2　单击“浏览”按钮

方法二:用命令 BROWSE

1　用 USE 命令打开表

2　输入 BROWSE 命令

例:浏览“基本情况”表,执行下面的命令:

USE 基本情况

BROWSE

2.5.2　定位记录

表文件一旦被打开,记录指针就指向表内的第一条记录。但这里所说的第一条记录对于有无索引文件打开的情况是不相同的。无索引文件打开,各记录按其物理顺序排列,即按记录号从小到大的顺序排列,有索引文件打开,各记录按其逻辑顺序进行一个“虚”的排队,当用户要求逐个记录列表输出时,各记录按此逻辑顺序出现。

一、指针绝对移动命令

格式 1:GORecordNumber 记录号

功能:移动记录指针指定的记录号按记录号移动,而不是按逻辑顺序号移动。

例:下面 2 种方法都是将指针定位在第 5 条记录;

GO5

或

5

格式 2:GO TOP | BOTTOM

功能:移动记录指针定位在表文件头 TOP 或文件尾 BOTTOM,按逻辑顺序排列,而不是按记录排列。

例：GO TOP　　&&移到文件头

GO BOTTOM　　&&移到文件尾

二、指针相对移动命令

格式：SKIP [nRecords]

功能：在表内按照指定的位移量向前或向后移动记录指针。

例：

SKIP　　&&往后移动 1 条记录，不带参数。

SKIP3　　&&往后移动 3 条记录，带参数。

SKIP - 6　　&&向前移动 6 条记录，带参数。

2.5.3 添加记录

当您在表设计器中设置了表结构后，系统会询问“现在输入记录吗”选择“是 Y ”，立即输入记录，选择“否 N ”，以后采用其他方式向表中添加记录。下面我们用两种方法向表中添加记录：

方法一：在“项目管理器”中

1 选择“人事记录”表；

2 单击“浏览”按钮，进入浏览窗口 图 2 - 12 ；

图 2 - 12　在浏览窗口中输入新记录

3 选择菜单：显示 ▷ 追加方式；

4 在图 2 - 12 窗口输入新记录；

5 按 Crtl + W 存盘 或 Ctrl + Q 放弃存盘 。

方法二：用命令 APPEND

1 在命令窗口输入“USE 基本情况”，打开“基本情况”表；

2 接着在命令窗口输入“APPEND”，进入图 2 - 13 编辑窗口；

图 2 - 13　在编辑窗口输入新记录

3 在图 2 - 13 窗口输入新记录；

4 按 Ctrl + W 存盘　K Ctrl + Q 放弃存盘 。

无论您是在浏览窗口还是在编辑窗口添加记录，结束操作都应该：按 Ctrl + W：将输入的信息保存在磁盘，然后返回。

按 Ctrl + Q 不保存输入信息，返回。

2.5.4　编辑记录

当您输入记录后，想修改它，可以采用下面两种方法：

方法一 在“项目管理器”中

1 选择要编辑记录的表；

2 单击“浏览”按钮；

3 选择菜单 显示 ▷ 编辑；

4 在编辑窗口 图 2 - 13 编辑记录；

5 按 Ctrl + W 存盘 或 Ctrl + Q 放弃存盘 。

方法二 用命令 USE 和 EDIT

1 在命令窗口用“USE”命令打开表；

2 接着输入命令“EDIT”；

3 在图 2 - 13 窗口中编辑记录；

4 按 Ctrl + W 存盘 或 Ctrl + Q 放弃存盘 。

2.5.5　删除记录

删除表中的任何记录，需要经过下面两个步骤换段：1　对想删除的记录作标记，称作删除操作。这时被作了标记的记录并没有真正地从表中清除掉，列表时仍然能看到它。

2 对表进行彻底操作，这个操作将作了标记的记录真正从表中删除。

一、删除

方法一 在“项目管理器”中

若您要删除个别记录，可以按以下步骤进行；

1 在“项目管理器”中选择要删除记录的表；

2 单击“浏览”按钮，进入表的浏览窗口，用垂直滚动条查找要删除的记录；

3 单击要删除记录的左边框，使其变黑 图 2 - 14 。

图 2 - 14　删除记录

4 按 Ctrl + W 存盘。

方法二 在“项目管理器”中使用菜单命令

若表中有上千条记录，要删除一批，就不可能一条一条的单击记录的左边框，此时可以用下面

的方法：

1 在“项目管理器”中选择要删除记录的表；

2 单击“浏览”按钮，进入表的浏览窗口；

3 选择菜单“表”→“删除记录”，进入删除对话框（图 2-15）；

图 2-15 删除对话框

作用范围：表示该命令执行操作的范围，只有落在这个范围内的记录才进行删除，范围可以是：

RECORDN：仅作用于第 N 条记录。

NEXTN：作用于当前记录及其后续 N－1 条记录，共对 N 个记录进行操作

ALL：作用于表内所有记录。

FOR 条件是一个表达式，当使用“FOR”时，表示该命令对所有那些能使“FOR”后的条件返回值为真（.T.）的记录进行删除。

WHILE 条件是一个表达式，使用“WHILE”时，表示该命令从当前记录开始在顺序删除的过程中遇到第一个不能使“WHILE”后面的条件为真（.T.）的记录便停止操作。

图 2-15 中 FOR 文本框内表达式的含义是删除所有的数学成绩小于 60 分的记录。

4 在对话框中选择删除范围，输入删除条件；

5 单击“删除”按钮，浏览窗口中被删除记录的左边框变为黑色；

6 按 Ctrl + W 存盘。

方法三：用 DELETE

删除记录用命令书写是很简单的，上例可以写为：

DELETE ALL FOR 数学＜60。

上面的命令中“All”表示范围，“FOR 数学＜60”表示条件。

二、撤消删除标记

使用 LIST 命令显示，可以看到被删除的记录前面有了星号“＊”（图 2-16）。

对于误删除的记录您可以将它们恢复，即将删除标记“＊”去掉。如果要恢复记录，可以用下面三种方法。

方法一：用“项目管理器”逐条恢复

1 在“项目管理器”中选择要删除记录的表；

公司编号	姓名	性别	年龄	出生年月	部门	职称
200017875	刘军	男	21	02/07/75	多媒体开发	设计员
200001784	李英	女	23	02/02/72	多媒体开发	程序员
2000011234	刘杰	女	19	11/12/78	多媒体开发	程序员

图 2-16 被删除的记录带“＊”标记

2 单击“浏览”按钮；

3 单击记录变黑的左边框，使其变为白色。

这样就将删除的记录恢复了，用 LIST 查看，删除标记“*”没有了。

方法二：用“项目管理器”和菜单命令恢复一批记录

1 在“项目管理器”中选择要删除记录的表；

2 单击“浏览”按钮；

3 选择菜单“表”|“恢复记录”，系统将出现类似图 2 - 15 的恢复对话框；

4 添写好恢复记录的范围和条件；

5 单击“恢复”按钮。

方法三：用命令 RECALL

恢复命令同样很好使用，下面的命令将当前记录开始的 10 条记录内数学成绩小于 60 分的记录恢复。

RECALL NEXT 10 FOR 数学<60

三、隐藏或显示有删除标记的记录

如果不想在记录清单上显示带有删除标记的记录，可以在“命令窗口”输入命令隐藏它们，隐藏后使用 LIST 命令将看不到它们，在“项目管理器”中的浏览窗口也看不到，当您想看到带有删除标记的记录时，通过命令的设置，可以再将它们显示出来。该命令是：

SET DELETED ON　隐藏带删除标记的记录

SET DELETED OFF　显示带删除标记的记录

四、彻底删除

要将有删除标记的记录真正从表中删除，可以采用以下两种方法之一：

方法一：用“项目管理器”和菜单命令

1 在“项目管理器”中选择要删除记录的表；

2 单击“浏览”按钮；

3 选择菜单“表”|“彻底删除”，出现图 2 - 17 的对话框，询问“是否将表中被删除的记录移去”；

4 在对话框中选择“是(Y)”，带删除标记的记录将被真正删除。

方法二：用命令 PACK

在命令窗口输入“PACK”，能够将所有带删除标记的记录从表中真正删除，此时系统没有任何提示，请谨慎使用。

ZAP = DELETE ALL + PACK。命令 ZAP 可以删除表中的所有记录，相当于命令 DELETE ALL 后接着执行命令 PACK。

图 2 - 17　提示是否真正删除记录

2.6 表结构的操作

在表设计器中创建表把字段名称、字段宽度等属性告诉 Visual FoxPro 的同时，也就生成了表结构。在生成表时，是根据将来存入的数据量结构设计表的，如果表的数据量增加了，您必须性修改表结构，将增加的数据容纳到表中。使用表设计器可以对表结构进行修改、增加字段或删除字段。

2.6.1 修改表结构

修改表结构要使用表设计器，启动表设计器用下面的方法：

1 在“项目管理器”中选择要修改结构的表；

2 单击“修改”按钮。

进入图 2-6 的表设计器。在表设计器中您可以更改字段的定义。当表中已有数据，更改表结构有可能会危及数据的安全，请谨慎操作。

一、更改字段名称

1 在表设计器内单击要修改的字段名，VisualFoxPro 将该字段名加亮显示；

2 输入字段的新名称，VisualFoxPro 用新字段名代替原有字段名。请注意新名称必须符合字段命令规则 参看本章 2.3.2 节 ，否则系统将不接受；

3 单击“确定”按钮，系统会提示结构更改是永久性更改，询问是否进行 我们可以选择“否”，不修改，也可以选择“是”，修改结构；

4 单击“是”按钮，字段名称被改变。

二、改变字段数据类型

假定为某字段设计了整型，当您想保存小数时，就必须将该字段类型改变为数值型，因为整型字段内没有设小数位。

当您要将数值型字段改变为整型字段时，需要注意是否会将小数位内的数据丢掉。

1 在表设计器内单击要修改字段的类型，该类型反白显示。

2 在类型下拉式列表中选择需要的数据类型。

3 单击“确定”按钮。

3 单击“是”按钮，数据类型被改变。

三、改变字段宽度

字段宽度大，浪费空间；字段宽度小，则不能容纳全部数据。因此应该给字段宽度设置合适的值。按照下述步骤，可以增大或减小字段的宽度。

1 在表设计器内单击要修改字段的宽度，该宽度出现微调按钮；

2 单击微调按钮选择数值或输入宽度值；

3 单击“确定”按钮；

4 单击“是”按钮，字段宽度被改变。

增大字段宽度，不会影响字段内的数据，但减小字段宽度，有可能丢失数据，丢失的数据是不可能恢复的。

改变小数位数的方法与改变字段宽度类似，不再赘述。

2.6.2　添加字段

假如您想向表中添加新字段,请用下面的方法：

1 在“项目管理器”中选择要添加字段的表；
2 单击“修改”按钮；
3 在表设计器内,添加新字段之处单击字段左边的按钮,按钮上出现双箭头；
4 单击“插入”按钮,系统用“新字段”取代选定字段,选定字段及其下方的字段依次被下移。
5 在“新字段”内输入新字段的名称,输入的新字段名称,取代了“新字段”。
6 建立字段的“类型”、“宽度”等属性；
7 单击“确定”按钮；
8 单击“是”按钮。

2.6.3　删除字段

对于多余字段可以将它删除,方法为：

1 在“项目管理器”中选择表名；
2 单击“修改”按钮；
3 在表设计器内,单击要删除的字段名,按钮上出现双箭头；
4 单击“删除”按钮,该字段被删除；
5 单击“确定”按钮,出现确认对话框；
6 在对话框中单击“是”按钮。

当您删除字段时,请留意是否会造成用数据的丢失,丢失的数据是不可能恢复的。

2.7　排序与索引

当我们创建一个表时,首先建立的是 .DBF 文件,如果结构中有备注或通用型字段,还将建立 FPT 文件　表备注文件　。当您把记录输入到表中时,这些记录间存在着一定的顺序关系　输入顺序　,我们叫它为物理顺序。然而使用表中记录时,经常按照工作的需要调整表中记录的顺序,例如您希望“成绩 1”表按数学分数的降序排列　高分在前,低分在后　,排序和索引可以为您解决这类问题。在数据库中使用索引可以提高对数据的查询速度。当数据量很小时,这种速度的提高是不明显的,但是当数据量很大时,索引就显示出它的优越性来。我们在改造某个银行的一个软件中深有体会。由于银行的数据量非常大,在一个保存日数据的表中有近 80 多万条记录要处理,由于原先的程序没有使用索引,在数据的累计、查询等计算中花费很长时间。后来通过对数据库添加索引使原先需要三天才能完成的计算现在在十分钟内就可以完成。

通过编写符合 Rushmore 技术的索引,可以使您的查询速度进一步提高。

Rushmore :微软提出的在数据库中用来加速查询的技术,它是类似二分法的一种查询算法,但比二分法复杂得多,使用 Rushmore 技术可以使查询速度提高近一倍。

2.7.1 数据的排序

SORT 命令用于排序。排序是根据当前表中指定字段的值进行升序或降序的排列,通过排序调整表中记录的实际位置,并产生一个新表。

SORT 命令不仅能根据当前表中指定字段的值进行升序或降序的排列,同时还可以确定表中排序的记录范围和条件。产生的新表不仅可以包括原表中的全部字段,也可以只包括原表中的部分字段,即通过排序可以创造一个简化后的新表。具体使用方法请参考联机帮助。

例 :将"成绩 1"表按数学分数排列,并显示它:

```
CLOSEDATABASE                          &&关闭数据库
OPENDATABASE 学生                      &&打开"学生"数据库
USE 成绩 1                             &&打开"成绩 1"表
SORTTO 成绩 11ON 数学                  &&该命令生成一个按照"数学成绩"从小到大排
                                       列的新表,其表名叫"成绩 11"。
SORT TO 成绩 12 ON 数字 DESCENDING     &&该命令生成一个按照"数学成绩"从大到
                                       小排列的新表,其表名叫"成绩 12"。
USE 成绩 11                            &&打开"成绩 11"表
LIST                                   &&显示记录
USE 成绩 12                            &&打开"成绩 12"表,同时关闭"成绩 11"表
DISP ALL                               &&显示记录
USE                                    &&关闭表
CLOSEDATABASE                          &&关闭数据库
```

我们看到用 SORT 排序,虽然能满足排列顺序的目的,但要在磁盘上生成新的表文件,如果您还希望按照英语成绩、物理成绩等排序的话,就要生成若干新的表文件,这样将要占用许多磁盘空间 ;如果记录很多时,运行速度也成问题,而且如果源表中信息有变动,先前排序的表将毫无意义,为了解决这些问题,在 VisualFoxPro 中使用更多的是索引。

2.7.2 使用索引

我们经常可以在一些书籍上看到索引,索引提取书中重要的词句,并附有与该词句的相对应的页码,这些词句按一定的顺序排列以便于查找。检索索引就能找到您要的词句,然后再翻看相应的而,便可以找到所要的文字内容。

从包含数百万条记录的数据库表中查找满足条件的记录,更需要使用索引,不过在使用索引之前,您必须先建立它。表 2 - 2 是表与其索引的关系示例。

表 2 - 2 "成绩 1"表与其数学成绩索引表的关系

成绩 1 表

记录号	学号	姓名	数学	英语
1	9401101	马天明	98	88
2	9401102	江　山	94	77

索引表

记录号	数学
4	79
3	88

续表

3	9401103	李　黎	88	95	5	88
4	9401104	田　园	79	74	2	94
5	9401105	赵明山	88	96	6	96
6	9401106	赵　晶	96	90	1	98

表 2 - 3　按数学成绩排序的成绩 1 表

记录号	学号	姓名	数学	化学	英语
1	9401101	马天明	98	92	88
6	9401107	赵　晶	96	85	90
2	9401102	江　山	94	89	77
3	9401103	李　黎	88	86	95
5	9401106	赵明山	88	74	96
4	9401105	田　园	79	90	74

当您为表建立索引时，必须指定其关键字，即按照哪个字段进行排序，建立索引后，表中记录存放的物理位置没有变，只是增加了一个按关键字排序的索引文件，索引文件中只保留关键字值和与其对应的记录号。例如按数学成绩排序，“数学”就是关键字，索引文件中只保留记录号和数学成绩。如果想浏览数学成绩，同时打开表文件和索引文件，将会看到如表 2 - 4 的清单。若您想查找“数学成绩 = 96”的学生，系统是这样操作的：

1　在索引文件中查找“96”；

2　知道与其对应的记录号是“6”；

3　将记录指针定位到表的 6 号记录。

怎么样，有了索引文件检索信息是否很方便呢

一、索引的四种类型

VisualFoxPro6. 0 为数据库表提供了四种类型的索引，您可以使用表设计器的“索引”选项卡来建立索引。

1. 主索引

一个表只能有一个主索引，在主索引中绝对不允许有重复值。当出现重复值时，系统将产生一个出错信息。例如在“成绩 1”表中学号是不允许重复的，我们可以将它设置为主索引，您在操作时输入了重复的学号系统会有出错信息提示。

2. 候选索引

候选索引也不允许在指定的字段或表达式中有重复值。候选这个名词是指索引的状态。因为候选索引禁止重复值，因此它们在表中有资格被选作主索引，即主索引的“候选”。一个表可以有多个候选索引。

3. 唯一索引

唯一索引允许存在重复值。但是,唯一索引只存储索引文件中第一次出现的重复值。在这种意义上,“唯一”指的是索引文件中入口值是唯一的,因为它对每一个特定的关键字只存储一次,而忽略了其重复值的第二次或以后的出现。例如用数学作唯一索引,有两个“88”表 2-2 ,但索引文件只保存第一个 88 的记录号是“3”而不保存第二个 88 的记录号,尽管第二个 88 存在于表文件中,但唯一索引文件却不包括它。惟一索引是为了向下兼容 兼容 FoxPro 或 FoxBASE 而提供的索引方法。

4. 普通索引

可以使用普通索引排序和查找记录,在这些记录中并不要求数据的唯一性。例如表 2-2 的索引就是将数学设为普通索引而建立的。

二、索引文件的两种结构、三种类型

在 VisualFoxPro6.0 中,索引文件有两种结构,即传统索引和复合索引。

传统的索引文件 IDX,只有一个索引关键字表达式。复合索引文件 .CDX,包含多个索引关键字表达式,这些索引关键字表达式称为索引标记。复合索引文件如同是将多个 .IDX 文件合成在一个文件中一样。复合索引文件也有两种,因此索引文件有三种类型:

1. 结构复合索引文件 .CDX 简称结构索引文件

当创建表或修改表结构时,可以从表结构中挑选用于创建索引标记的字段,然后 VisualFoxPro6.0 创建一个 .CDX 复合索引文件。这个复合索引文件具有与表名相同的文件名,并且当打开与它同名的表时也自动打开该 .CDX 复合索引文件;关闭表的同时自动关闭它。此外,当在表中进行记录的添加、修改和删除时,系统会自动对 CDX 复合索引文件中的全部索引标记进行维护。因此它被称作结构复合索引文件。

2. 独立复合索引文件 CDX 也称非结构索引文件

独立复合索引文件是另行建立的,它必须用显式命令打开。只有在该文件打开时,系统才能维护其索引标记。换句话说,当您只打开表,而没有打开其独立复合索引文件,而进行记录的添加、修改和删除时,系统不会对独立复合索引文件中的索引标记进行维护,除非您在打开表的同时,用命令将相应的独立复合索引文件也同时打开。

3. 独立单项索引文件的扩展名为 IDX,它的主文件名不能与相关表同名,而且该文件不会随着表的打开而打开。

三、索引文件的创建

结构复合索引文件 CDX 是在创建表结构的同时建立的,而另外两种索引文件需要用命令建立。

1. 建立结构复合索引文件 CDX

1 在“项目管理器”中打开表设计器 图 2-6 ,不知如何打开,请参考 2.2 节;

2 选择“字段”选项卡,为字段设置索引;

3 选择字段名,单击“索引”下拉式列表,在其中选择“⇧”升序 或“⇩”降序 ;

4 选择“索引”选项卡,出现图 2-18;

5 在类型列选择索引类型;

6 单击“确定”按钮。

2. 建立独立复合索引文件 .CDX

使用 INDEX 命令中的 TAG 和 OF 子句。

例 1 建立独立复合索引文件,排序数学成绩和英语成绩,可执行如下命令:

```
CLOSE DATABASE
OPEN DATABASE 学生
USE 成绩 1EXCLU
INDEXON 英语 TAG 英语 OF 成绩 11
INDEXON 数学 TAG 数学 OF 成绩 11
LIST
```

上述命令建立了一个独立复合索引文件"成绩 11. CDX",该文件有两个索引标记"数学"和"英语",用 DISPLAYALL 命令显示后,您会看到记录按数学成绩升序排列,如果数学成绩相同,则按英语成绩升序排列。

3. 建立独立单项索引文件 . IDX

向下兼容的 . IDX 文件也需要以命令的形式建立。由于一个 . IDX 文件只能有一个索引顺序它只能接受一个关键字表达式 ,所以为了获得多个索引顺序,就必须建立多个 . IDX 文件。

建立 . IDX 文件,使用 INDEX 命令中的 TO 子句。

例 2 建立独立单项索引文件,排序数学成绩和英语成绩。可执行下面的命令:

```
CLOSEDATABASE
OPENDATABASE 学生
USE 成绩 1DXCLU
INDEXON 英语 TO 成绩 12
INDEXON 数学 TO 成绩 13
LIST
```

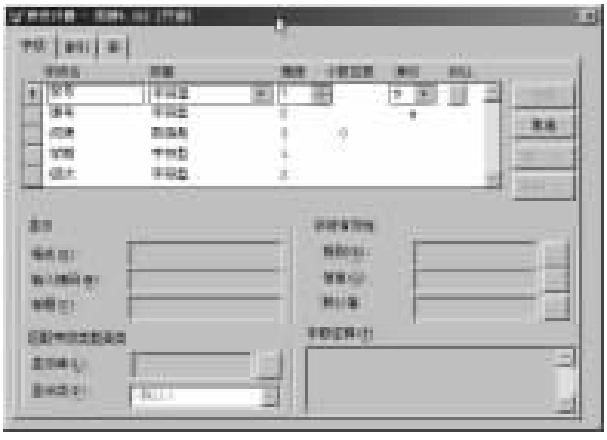


图 2 - 18　索引选项卡

例 1 和例 2 实现同样的功能,但例 1 只需一个索引文件而例 2 需要建立两个索引文件:"成绩 12"和"成绩 13"。

四、索引文件的引用

结构复合索引文件是随表文件 DBF 自动打开的,而另外两种独立索引文件都需要用命令打开。

当您用 INDEX 命令建立索引文件时,索引文件自动打开。建立它们后立即用 LIST 命令浏览记

录,就会看到索引文件已经起作用了。但是您在下一次用 USE 命令打开一个表文件后,先前建立的索引文件并没有同时打开,这时可以用命令 SET INDEX TO 打开它,用 SETORDERTO 命令使表按索引关键字排序。

例 3 打开"成绩 1"表及其索引文件"成绩 11",并显示其内容:

```
CLOSE DATABASE
OPEN DATABASE 学生
USE 成绩 1 EXCLU
SET INDEX TO 成绩 11
SET ORDER TO 英语
LIST
```

五、索引的维护

在表中建立索引后,就将当前表中的记录放到了索引文件中,以便从索引中存取这些记录。但当您下一次打开表而没有将相关的独立索引文件打开时,表记录的增、删、改是不会反映在索引文件中的,这样索引文件就会过时。过时的索引文件是无法获得正确结果的,甚至会导致系统的崩溃。在表记录发生变化后,索引文件没有被修改的情况下,我们应该及时 维护表中所有索引标记和索引文件。维护索引的简单办法是重建索引。

VisualFoxPro6.0 提供了两种重建索引的方法:

方法一:选择菜单"表"→重建索引

方法二:使用 REINDEX 命令

重建索引很费时间,特别是对很大的表重建索引时更是如此,因此最好在程序的初始部分和结束部分来进行重建索引,而不要在应用程序的主体部分进行索引维护。

2.8 表的关联

在数据库表中建立索引后,就能将它们关联起来,以便达到减少冗余,共享数据的目的。例如在"学生"数据库中学生的信息分为"基本情况"、"成绩"、"个人表现"和"社会关系"几个表,每一个表都需要有姓名,为了便于检索,我们将"学号"设为主索引,即只在"基本情况"表中存放"姓名"字段,其他表通过"学号"和它关联 如图 2-19 。从图中可以看到几个表之间有线连着,这是它们之间的关联关系。索引用打开的书标识,索引下方是索引关键字前有钥匙标记的是主索引,无钥匙的是普通引。

在数据库中建立的这处关系被称作永久关系。

图 2-19 建立关联的数据库表

永久关系 :是数据库表间的关系,它们存储在数据库文件中,有以下特点:

1 永久关系不必在每次使用表时重新创建。

2 在“数据库设计器”中,显示为联系表索引的线。

3 在“查询设计器”和“视图设计器”中,自动作为默认联接条件。

4 在“数据环境设计器”中,作为表单和报表的默认关系。

5 用来存储参照完整性信息。

1. 创建表间的永久关系

1 在“数据库设计器”中,选择想要关联的索引名。

2 把它拖到相关表的索引名上

通过查看“数据库设计器”中该数据库的分层结构,您就会看到两个表间有一条连线,这表示建立了新的永久关系。

在一对多关系中,“一方”必须用主索引关键字,或者用候选索引关键字 ;而“多方”则使用普通索引关键字。如“基本情况”是“一方”,每个学生在表中只能有一条记录,而“社会关系”是“多方”,一个学生可以有若干个社会关系。

2. 删除表间的永久关系

1 在“数据库设计器”中,单击两表间的关系线,关系线变粗,表明已选择了该关系。

2 按下 DELETE 键,关系线没有了,表明您已经删除了该关系。

由于永久关系并不控制各表内记录指针间的关系　使用永久关系不能实现在一个表中移动记录指针而改变另一个表中的指针 ,因此在开发 VisualFoxPro 应用程序时,您可能既要使用临时的 SETRELATION 关系,又要使用永久关系。

2.9　同时使用多个表

当用 USE 命令打开一个表时,同时也就关闭了先前打开的那个表。为了能够同时使用多个表,VisualFoxPro 引入了“工作区”的概念,您还可以用“数据工作期”来操作工作区。

2.9.1　工作区

工作区 :是一个编号区域,它标识一个已打开的表,在 VisualFoxPro6. 0 中我们可以在 32767 个工作区中打开和操作表,即最多可以打开 32767 个表,但这些表要在不同的工作区,也就是说一个工作区只能有一个打开的表。

在应用程序中工作区通常通过使用该工作区的表的别名来标识。表别名是一个名称,它可以引用在工作区中打开的表。前 10 个工作区指定别名是字母 A 到 J,在工作区 11 到 32767 中指定的别名是 W11 到 W32767。您可以使用这些 VisualFoxPro 指定的别名来引用在一个工作区中打开的表。

如果要在最低可用工作区中打开表,可以在 USE 命令的 IN 子句后面加工作区“O”用此方法是让系统选择一个最合适的工作区,而选择工作区则使用 SELECT 命令。下面的命令在不同的工作区分别打开表,并使用它们。

```
USE 成绩 1INO                    &&打开成绩 1 表
LIST                             &&显示成绩 1
```

```
USE 基本情况 INO                    &&打开基本情况表
LIST                                &&显示基本情况
SELE 成绩 1 INO                     &&选择成绩 1 表
LIST                                &&显示成绩 1 表
```

以后我们就可以使用别名“成绩 1”和“基本情况”在命令或函数中标识被打开的表。

表别名：是 VisualFoxPro 用来指定在一个工作区中打开的表的名称。打开一个表时，VisualFoxPro 自动使用文件名作为默认表别名。上例的“成绩 1”和“基本情况”就作为别名被使用，您也可以为表创建自己的别名。

2.9.2 数据工作期

在不同的工作区中打开表除了使用命令外，还可以使用数据工作期。

数据工作期是指表单、表单集或报表使用的当前动态工作环境（图 2-21）。每个数据工作期包含了它自己的一组工作区，并显示在工作区中打开的表、表索引以及表之间的关系。

使用数据工作期打开表的方法是：

1 打开“数据工作期窗口”；

2 单击常用工具栏中“数据工作期”按钮。

图 2-20 是尚未打开表时的“数据工作期”窗口，在这个窗口中可以打开和显示表或视图，在表或视图间建立临时关系等。

图 2-20 未打开表的“数据工作期”窗口

1.“数据工作期”窗口选项意义：

1 当前工作期：显示当前工作期名称。工作期是启动 VisualFoxPro 时所创建的一个 VisualFoxPro 实例，当退出 VisualFoxPro 时工作期结束。

2 别名：显示不带扩展名的视图名或表名。

3 关系：指明“别名”框中的表或视图之间的临时关系。

4 属性：显示工作区属性对话框，可以用此对话框改变表的结构，选择索引文件和字段，定义数据筛选条件等。

5 浏览：在浏览窗口中显示“别名”框中选定的表或视图，可以在此窗口中检查、编辑或添加数据（就像在“项目管理器”中选择表名后单击“浏览”按钮一样）。

6 打开：显示打开对话框，用此对话框选择要打开的表或视图。

7 关闭 :从“别名 ”框中删除选定的表或视图和任何相关文件。

8 关系 :使用表达式生成器定义表和视图之间的关系。

9 一对多 :显示一对多对话框,从而在子表和父表之间建立一对多的临时关系。

2. 在数据库表间建立临时关系

用“数据库设计器 ”在表间建立的是永久关系,而在“数据工作期 ”建立的关系是临时关系。所谓临时关系是当“数据工作期 ”关闭时,这种关系就结束。

1 在“数据工作期 ”窗口中选定要关联的表 ;

2 使用“关系 ”按钮创建关系。

图 2 - 21 就是在“数据工作期 ”窗口要经常被使用,在这个关系中,所有的表都和“基本情况 ”表建立了关联,而“社会关系 ”表的连线是双线,表示“一对多 ”关系。

在调试程序时 “数据工作期 ”窗口要经常被使用, 通过您可以查看当前打开的表和视图是否合适 ,必要时您可能要通过它将不 打开的表关闭。

图 2 - 21　打开表后的“数据工作期 ”窗口

在调试程序时 “数据工作期 ”窗口要经常被使用, 通过它您可以查看当前打开的表和视图是否合适,必要时您可能要通过它将不应打开的表关闭。

第 3 章　查询与视图

数据库最常用的操作就是查询，由于查询操作非常频繁，因此查询效率的高低将在很大程度上影响程序执行的效率。使用视图能从表中迅速查询需要的数据，以提高数据查询的效率。视图是 VisualFoxPro 非常实用的特性。

3.1　查询

我们在为报表组织信息、即时回答问题或者查看数据中的相关子集等操作时，都需要建立查询。查询是指搜索那些满足指定条件的记录，同时也可以根据需要对这些记录排序和分组，我们可以将查询结果创建为报表、表及图形，并将它保存在磁盘上。无论目的是什么，建立查询的基本过程是相同的。

本章使用“查询设计器”建立一个单表查询。关于远程数据的查询请参阅第十六章。有关查询的详细内容，请参阅联机文档《用户指南》中的第四章“检索数据”、第六章“查询和更新多表”。

3.1.1　创建查询

当确定了要查找的信息，以及这些信息存贮在哪些表和视图中后，就可以通过以下几个步骤来建立查询：

1 使用“查询向导”或“查询设计器”；

2 选择将要出现在查询结果中的字段；

3 设置选择条件，以满足所需结果的记录；

4 设置排序或分组选项，用以组织查询结果；

5 定向查询结果，如浏览、报表、表、标签等；

如果要保存创建的查询，可以指定一个名称，将查询文件保存为带 QPR 扩展名的文件。

6 运行查询。

一、使用“查询向导”

要想快速地创建查询，可以使用“查询向导”。“查询向导”将询问从哪些表或视图中搜索信息，并根据您对一系列提问的回答来建立查询。操作方法为：

1 在“项目管理器”中选择“数据”选项卡，然后选择“查询”；

2 选择“新建”；

3 单击“查询向导”按钮；

4 选择要使用的向导类型；

遇到困难时，只要在使用向导时按 F1 键即可获得联机帮助。

二、使用“查询设计器”

我们也可以用“查询设计器”建立更为灵活的查询，方法为：

1 在“项目管理器”中选择“数据”选项卡；

2 选择“查询”；

3 单击“新建”按钮，启动“新建查询”对话框；

4 在对话框中单击“新建查询”按钮。

图 3 - 1　在查询中添加表或视图

进入“查询设计器”，系统将提示从当前数据库或自由表中选择表或视图 图 3 - 1 。您至少要为查询选定一个表，若查找的信息在多个表中，也可多选几个表。在这里我们从“数据库”中选择“学生”，从“数据库中的表”选择“成绩 1”。

5 单击“添加”按钮，若选择多个表，每选择一个表，就要单击一次添加；

6 单击“关闭”按钮，进入图 3 - 2 的“查询设计器”，您会看到，我们选择的表已经在“查询设计器”中。

图 3 - 2　查询设计器

三、“查询设计器”

在图 3 - 2 中的“查询设计器”窗口中，有上下两个部分。上部的左边放置着查询的数据源——表；右边是“查询设计器工具”，有这样一些工具：

“添加”表：向查询中添加表。

“移去”表 :将表从查询中移走。

“添加联接”:在表间添加联接。

“查询去向”:将查询定向到浏览、报表、表或标签等。

“SQL”:显示 SQL 窗口。我们在“查询设计器”中做的所有操作 VisualFoxPro 都把它解释成 SQL 语句，通过查看 SQL 窗口，可以学习 SQL 语言，当您熟练地掌握了 SQL 语言后，就可以直接使用 SQL 语句设置查询,而不必再借助于“查询设计器”了。

下部的页框可以对查询结果进行各种设置。

SQL 是结构化查询语言 StructuredQueryLanguage 的缩写，是美国国家标准局 ANSI 确认的关系数据库语言的标准,一条 SQL 语言就可以替代许多条 VisualFoxPro 命令。

3.1.2 定义输出结果

选择了包含需要信息的表或视图后，接下来就应该定义要输出的信息了。首先选择要查询的字段,然后您还可以设置要显示的记录和显示顺序。

一、选择所需字段

使用“查询设计器” 图 3-2 下部页框中的“字段”选项卡来选定需要包含在查询结果中的字段。如果不选定字段则不能进行查询。

1. 在查询中添加字段

1 在“可用字段名”列,选定字段名；

2 单击“添加”按钮,您也可以不选择字段名而单击“全部添加”,将所有字段都添加到“选定字段”列。

2. 设置输出字段的次序

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。如果要改变显示顺序,上、下拖动位于字段名左侧的移动框即可。

我们要查看学生的数学成绩,设置字段如图 3-3。

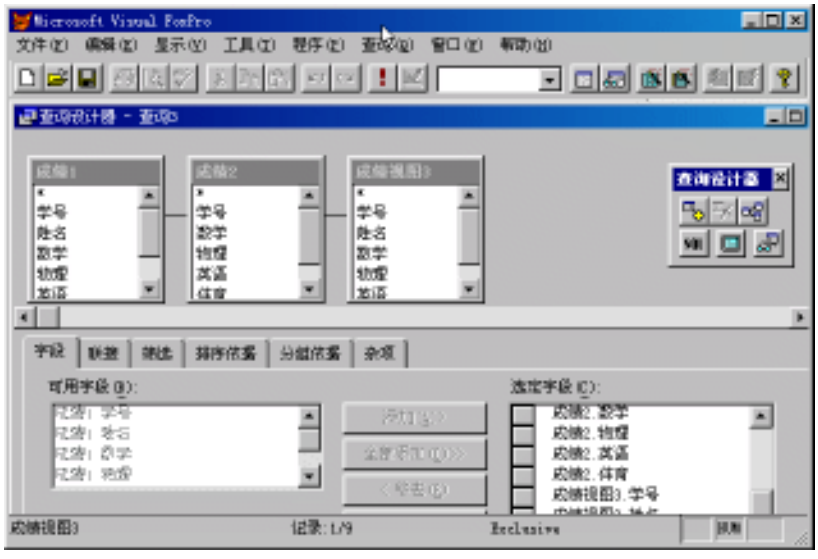


图 3-3 设置字段

二、设置排序结果 可选项

利用“排序依据”选项卡 图 3 - 4 可以设置查询的排序次序，排序次序决定了查询输出中记录的排列顺序，如果按照记录的物理顺序查询，则不必设置它。

图 3 - 4　“排序依据”选项卡

1. 设置排序条件

1 在“选定字段”列中选定字段名；

2 单击“添加”按钮。

该字段被移到“排序条件”框中，作为查询结果中排序的依据。字段在“排序条件”框中的次序决定了查询结果排序时的次序，第一个字段决定主排序次序。

例如，在“排序条件”框中的第一个字段是“数学”，第二个字段是“学号”，查询结果将首先以“数学”进行排序，如果遇到相同的分数，则按“学号”排序。

2. 用“排序依据”选项卡设置排序次序

为了调整排序字段的次序，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，可以确定结果是按升序还是按降序排序。在“排序条件”列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。

按照图 3 - 3 设置的字段，图 3 - 4 设置的排序，查询结果在图 3 - 5 中。数学为主排序，学号为次排序，当数学分数一样时，按学号排序 实例中的“初级查询成绩 1”。

学号	姓名	数学	物理	英语	体育
9401101	马天明	93	92	85	90
9401102	江　山	90	89	96	78
9401103	王　黎	87	86	95	70
9401105	田　园	91	90	94	80
9401106	赵明山	88	80	90	80
9401107	赵　晶	90	85	88	85
9401108	王晓晓	100	100	100	100

图 3 - 5　查询结果

该查询对应的 SQL 语句如下：

```
SELECT 成绩 1. 学号,成绩 1. 姓名,成绩 1. 数学；　　&&字段选择
FROM 学生! 成绩 1；　　&&表
ORDERBY 成绩 1. 数学 DESC,成绩 1. 学号；　　&&以数学升序、学号降序排列
```

有关 SQL 语言的详细描述,请参阅本书第七章。

三、选定所需的记录 可选项

查询的大多数情况是要选定满足某种条件的记录。用“查询设计器”中的“筛选”选项卡 图 3 - 6 ,可以构造一个选择语句,用来通知 VisualFoxPro 想要搜索并检索的记录,如果要看全部记录,则不必设置它。

图 3 - 6 “筛选”选项卡

例设,我们要查询学号小于“950000”的学生,可以按着下列步骤操作:

1 从“字段名”列表中选定用于选择记录的字段 成绩 1. 学号;

2 从“条件”列表中选择比较的类型 =、〈〉等 ,选择〈 小于 ;

3 在“实例”文本框中,输入比较条件“950000”;

4 在搜索字符型数据时,如果想忽略大小写匹配,请选择“大小写”下面的按钮;

5 设置“逻辑”值。

如果您设置了两个条件,就要在“逻辑”下面决定它们之间的关系。“AND”是“与、并且”的意思,它使查询必须完全符合两个条件。“OR”是“或者”的意思,使查询符合两个条件中的任意一个。本例采用的是“AND”关系,图 3 - 6 查询条件的含义是“学号小于 9500000,并且数学成绩大于 85”。图 3 - 7 是查询结果 实例中的“初级查询成绩 2”。

成绩视图3

学号	姓名	数学	物理	英语
NULL.	NULL.	90	85	88
NULL.	NULL.	92	87	85
9401101	马天明	86	92	88
9401102	江 山	88	89	66
9401103	李 黎	77	86	95
9401105	田 园	92	90	54
9401106	赵明山	65	60	96
9401107	赵 晶	78	85	88
9401111	王丽黎	92	94	70

图 3 - 7 设置筛选后的查询结果

该查询对应的 SQL 语如下:

```
SELECT 成绩 1. 学号,成绩 1. 姓名,成绩 1. 数学;     &&字段
FROM 学生! 成绩 1;                                  &&表
```

WHERE 成绩 1. 学号＜“9500000”；　　&&条件

AND 成绩 1. 数学＞85；

ORDERBY 成绩 1. 数学 DESC,成绩 1. 学号　　&&按数学降序、学号升序排列

1 备注字段和通用字段不能用于设置查询条件。

2 逻辑位的前后必须使用句点号,如 .T. 。

3 如果输入查询中表的字段名,系统就将它识别为一个字段。

4 只有当字符串与查询的表中字段名相同时,要用引号将字符串括起来,否则不需要用引号将字符串括起来。

5 日期不必用花括号括起来。

四、分组查询 可选项

所谓分组就是将一组类似的记录压缩成一个结果记录。例如,“成绩 1”表记录着学生两个学期的成绩,按学号分组,计算每个学生本学年数学的总分,这样就可以将一个学生的两条记录合成一条记录,如果您不想压缩结果记录,则不必设置它。

1. 设置分组选项

如果要控制记录的分组,可使用“查询设计器”中的“分组依据”选项卡。分组在与某些累计函数联合使用时效果最好,诸如 SUM 求和 、COUNT 记数 、AVG 求平均值 等等。方法为：

1 在图 3 - 3 的“字段”选项卡中,单击“函数和表达式”右边的三点按钮,出现图 3 - 8 的“表达式”生成器。

图 3 - 8　表达式生成器

2 在“数学”下拉式列表中双击 SUM expN ,在这里我们建立一个表达式：“SUM 成绩 1. 数学 ”它的含义是计算成绩表中数学成绩的和；

3 单击“确定”按钮,表达式出现在图 3 - 9 的“函数和表达式”框中；

图 3-9　生成的表达式

图 3-10　表达式成为选定字段

4 选择生成的表达式,使其反白显示;

5 单击“添加”按钮,该表达式被添加到“选定字段”列中 图 3-10 。将来查询中就会有一列数据求和的数学成绩;

6 单击“分组依据”选项卡,进入“分组依据”窗口 图 3-11 ;

图 3-11　“分组依据”选项卡

7 在可用字段选择“成绩 1. 学号”,再单击“添加”按钮,在这个查询中我们设置了字段、排序依和分组依据,按学号求学生数学的总分,并按成绩和学号排序,它的运行结果如图 3-12 实例中的“初级查询成绩 3”。

学号	姓名	数学	物理	英语
[illegible]	[illegible]	80	85	88
[illegible]	[illegible]	92	87	85
04101101	马天明	86	92	88
04101102	江　山	88	89	86
04101103	李　黎	77	80	85
04101105	田　晶	92	90	94
04101106	延明山	85	80	86
04101107	赵　晶	78	85	88
04101111	王丽黎	90	94	78

图 3-12　分组后的查询结果

该查询对应的 SQL 语句如下:

SELECT 成绩 1. 学号,成绩 1. 姓名,SUM 成绩 1. 数学 ;

FROM 学生! 成绩 1；

GROUP BY 成绩 1. 学号　　&&按学号分组

ORDER BY 3 DESC,成绩 1. 学号　&&按 SUM 成绩 1. 数学 升序、成绩 1. 学号降序排列

设置分组的要点是 在“字段”选项卡中的“函数和表达式”框中输入表达式。

选择“添加”按钮,将表达式添加到“选定字段”列中。

在“分组依据”选项卡中,加入分组结果依据表达式。

2. 选择分组

我们也可以对已进行过分组的记录设置过滤器，例如查询数学总分大于 175 分的同学。在分组依据选项卡中利用“满足条件”按钮,就可以限制查询的结果。

1 在图 3 - 11 中的“分组依据”选项卡上,单击“满足条件”按钮,进入图 3 - 13 分组过滤器 ；

图 3 - 13　对分组记录设置过滤

2 从“字段名”列表中选定用于选择记录的字段 :SUM 成绩 1. 数学 ；

3 从“条件”列表中选择比较的类型 =、〈〉选择〉大于 ；

4 在“实例”文本框中,输入比较条件 :175；

5 单击“确定”按钮,图 3 - 14 是运行结果 实例中的“初级查询成绩 4”。

学号	姓名	数学
[illegible]	[illegible]	90
[illegible]	[illegible]	92
04011101	[illegible]	86
04011102	江　山	88
04011103	王　黎	77
04011105	田　晶	92
04011106	起明山	85
04011107	赵　晶	78
04011111	王丽黎	92

图 3 - 14　对分组设置过滤器后的查询结果

该查询对应的 SQL 语句如下：

SELECT 成绩 1. 学号,成绩 . 1 姓名,SUM 成绩 1. 数学 ；

FROM 学生! 成绩 1；

GROUP BY 成绩 1. 学号；　　&&按学号分组

HAVING SUM 成绩 1. 数学 〉175；　&&分组结果满足的条件

ORDER BY 3 DESC 成绩 1. 学号　　&&按 SUM 成绩 1. 数学 升序、成绩 1. 学号降序排列

五、使用“杂项”选项卡 可选项

在“查询设计器”中有一个“杂项”选项卡,下面简单介绍它的用途。

1. 在查询中删除重复记录

重复记录是指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,只要选择“杂项”选项卡中的“无重复记录”即可。如图 3 - 15 所示。

图 3 - 15 “杂项”选项卡的使用

图 3 - 16 设置百分比

2. 查询最靠前一定数目的记录

利用“杂项”选项卡的“列在最前面的记录”的设置,可以将查询设置到一定数目。例如显示前 10 个最高的记录。方法为:

1 在“排序依据”选项卡中,设置排序选项为“降序”;

2 在“杂项”选项卡中取消“全部”选择,此时“记录个数”被激活;

3 在“记录个数”内输入数字 图 3 - 15 。

3. 查询一定百分比的记录

我们还可以设置查询最高或最低值百分数,例如查询后面 20% 的记录 实例中的“初级查询成绩 5”。方法为:

1 在“排序依据”选项卡中,设置排序选项为“升序”;

2 在“杂项”选项卡中取消“全部”选择;

3 选择“百分比”,此时“记录个数”变为“百分比”;

4 在“百分比”内输入数字 图 3 - 16 。

该查询对应的 SQL 语句如下:

```
SELECT TOP 20.00 PERCENT 成绩 1 学号,成绩 1. 姓,名 SUM 成绩 1. 数学 ;     &&选择百分之 20
FROM 学生! 成绩 1;
GROUP BY 成绩 1. 学号;
ORDER BY 3 DESC,成绩 1. 学号,                    &&按 SUM 成绩 1. 数学 升序、成绩 1. 学号降序排列 O
```

3.1.3 查询的备注

当您在设计查询时,很清楚自己的设计目的,但过一段时间后有可能就忘记了。如果能以某种方式标识查询,对它作一些注释说明,对以后确认查询及其目的是很有帮助的。Visal FoxPro 为您想到了这一点。

一、为查询添加备注

1 选择菜单“查询”→“备注”，进入“备注”文本窗口；

2 在“备注”窗口输入文字，在输入的文字前面有一个“*”号表明其为注释；

3 单击“确定”按钮。

二、查看备注

为查询添加了备注后，可按以下方法查看它：

单击“查询设计器工具”中的“SQL”按钮（或在“查询设计器”中选择菜单“查询”→“查看 SQL”）。

在 SQL 窗口中有“*”号标记的就是查询的注释（一般是绿色）。而其他语句是用“查询设计器”产生的 SQL Select 语句。

3.1.4　其他

一、在设计中运行查询

在设计查询的过程中，我们可以随时运行查询，查看设计是否合理，是否达到预期效果。方法是：

在查询设计器中选择菜单“查询”→“运行查询”（或按 Ctrl + Q）。

系统将把查询结果显示在“浏览”窗口中。如图 3 - 5、图 3 - 7、图 3 - 12、图 3 - 14 那样。若没有达到预期效果，可以修改查询的设计。

二、确定查询的目的

查询检索的信息，可以做各种不同的用途。前面我们曾经使用浏览方式显示查询结果，您也可以在表、图、屏幕、报表、标签中使用从查询收集到的信息。方法为：

单击“查询设计器工具”中的“查询去向”按钮；或在“查询设计器”中选择菜单“查询”→“查询去向”。

此时将显示一个“查询去向”对话框（图 3 - 17），您可以在其中选择将查询结果送往何处。

图 3 - 17　查询去向对话框

对话框中按钮的含义如下：

“浏览”：在“浏览”窗口中显示查询结果，这是查询缺省设置。如果不设置查询去向时，系统自动将查询结果定义为浏览。

“临时表”：将查询结果存储在一个临时只读表中。

“表”：使查询结果保存到一个表中。

“图形”：将查询结果用于 Microsoft Graph（Graph 是包含在 VisualFoxPro 中的一个独立的应用程序。

“屏幕”:在 VisualFoxPro 主窗口或当前活动输出窗口中显示查询结果。

“报表”:将输出送到一个报表文件 .FRX 中。

“标签”:将输出送到一个标签文件 .LBX 中。

选定一个去向,按步骤设置一些属性,然后单击“确定”按钮,VisualFoxPro 就按您的意图去放置查询的结果。

在设定查询去向时,对于一些选择不知如何操作时,请按 F1 键求得在线帮助。

三、退出查询设计器

设计完查询后,单击“查询设计器”右边的关闭按钮,屏幕上会出现“另存为”对话框,询问您将查询保存在何处。此时需要选择保存文件的路径、输入文件名、单击“保存”按钮,系统就会按您的设置将文件保存,同时在“项目管理器”的“查询”列表中出现了您命名的文件。

四、运行查询

在“项目管理器”中选择查询文件名,单击“运行”按钮。

怎么样,和您在设计查询时看到的一样吧

3.2 视图

视图是数据库中的一个特有功能,只有当包含视图的数据库被打开时,才能使用视图。利用视图,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。当您不但要检索数据,还想更新它,就需要使用视图。从多表中选取字段也是视图的一个重要用途。

假如希望一个数据能包含学生的全部背景资料,每个人必须有多种数据:学号、姓名、出生年月、籍贯、班级、成绩、社会关系、家庭住址、获奖记录等等。如果把这些信息存入数据库的一个表内,数据的处理速度会大大减慢。我们应该按照 1.4 节的介绍,把全部数据分存在多个表中,需要访问存储在不同表中的相关信息时,可以创建一个视图,将需要的多个表或视图添加进去,VisualFoxPro 借用公用字段把各表联接起来。

在 VisualFoxPro 中可以创建两种类型的视图:本地视图和远程视图,远程视图的内容请参阅本书第十六章,有关视图的详细内容,请参阅联机文档《用户指南》中的第五章“使用视图更新数据”、第六章“查询和更新多表”。

3.2.1 创建视图

由于视图和查询有很多相似处,因此创建视图与创建查询的步骤类似,创建视图时要选择包含在视图中的表和字段、指定用来联接表的联接条件、指定过滤器选择特定的记录。与查询不同的是,视图可以把视图中做的数据修改传给原始表。

若要创建本地表的视图,可以使用“视图向导”和“视图设计器”。本地表包括本地 VisualFoxPro 表和使用 DBF 格式的表以及存储在本地服务器上的表。

下面我们使用“视图设计器”创建一个包含两个表的视图。

1 从“项目管理器”选定一个数据库。

2 单击“数据库”符号旁的加“+”。

3 在“数据库”下,选定“本地视图”并单击“新建”按钮,启动“新的本地视图”对话框。

4 在“新的本地视图”对话框中,单击“新视图”按钮,进入“视图设计器”,系统将提示从当前

数据库或自由表中选择表或视图 图 3 - 1 。我们在“数据库”中选择“学生”,在“数据库中的表”列选择“基本情况”。

5 单击“添加”按钮,再选择“成绩 2”,再单击添加。由于在我们使用的数据库中,表之间具有永久关系，VisualFoxPro 就利用这些已有的关系作为默认的联接，所以屏幕出现图 3 - 18“联接条件”对话框。对话框中显示两表的默认联接。

图 3 - 18　“联接”条件对话框

6 单击“确定”按钮。

7 单击“关闭”按钮,进入图 3 - 19 的“视图设计器”,您就会看到,我们选择的表已经显示在“视图设计器”中,并且在“基本情况”表和“成绩 2”表之间有一条连线,这个连线就是两表之间的联接关系。“基本情况”表是左表,“成绩 2”是右表。

图 3 - 19　“视图设计器”

“视图设计器”的“字段”选项卡、“联接”选项卡、“筛选”选项卡、“排序依据”选项卡、“分组依据”选项卡、“杂项”选项卡的作用与“查询设计器”中各选项卡的作用是完全相同的。所不同的是“视图设计器”中多了一个“更新条件”选项卡。由于在查询中我们创建的是单表,没有涉及联接的问题,因此先介绍“联接”选项卡的使用,然后再介绍“更新条件”选项卡。

3.2.2 联接

如果往视图 查询 中添加多个表，VisualFoxPro 就会询问如何把表与您添加的第一个表联接起来，联接是用公用字段链接两个表的方法。如果正确地规划了数据库 见 1.4 节 ，所有的表都应有一个公用字段。这些公用字段都赋予一个相同的名字，VisualFoxPro 在图 3-18 的“联接”条件对话框中把这些字段作为联接的条件。

一、建立联接

添加表时系统会自动显示联接 图 3-18 。但是，如果相关的字段名不匹配，VisualFoxPro 便不会进行联接，这时您就必须自己创建表间的联接。

创建联接的方法是：

1 从“视图设计器”工具栏上选择“添加联接”按钮，显示“联接条件“对话框 图 3-18 ；

2 在“联接条件”对话框中，从两个表中选择相关的字段名，要注意只有当字段的大小相等、数据类型相同时才能联接；

3 选择联接类型。

在“联接条件”对话框中有四种连接类型，其含义如下：

内部联接 Inner Join ：两个表中仅联接满足条件的记录，即只联接两表中都有的记录。这是最普通的联接类型 图 3-20 。

图 3-20 内联接

实例中“初级成绩视图 1”内联接 的 SQL 语句如下：

```
SELECT 基本情况.学号,基本情况.姓名,成绩2.数学,成绩2.物理,成绩2.英语;
                                                    &&选择的字段
FROM 学生!基本情况 INNER JOIN 学生!成绩2;          &&基本情况成绩2内联接
ON 基本情况.学号=成绩2.学号;                        &&联接条件
ORDER BY 基本情况.学号                              &&按学号的升序排列
```

左联接 Left Outer Join ：联接左表的所有记录和右表的满足联接条件的记录 图 3-21 ，左表有，而右没有的记录用 .NULL 填充。

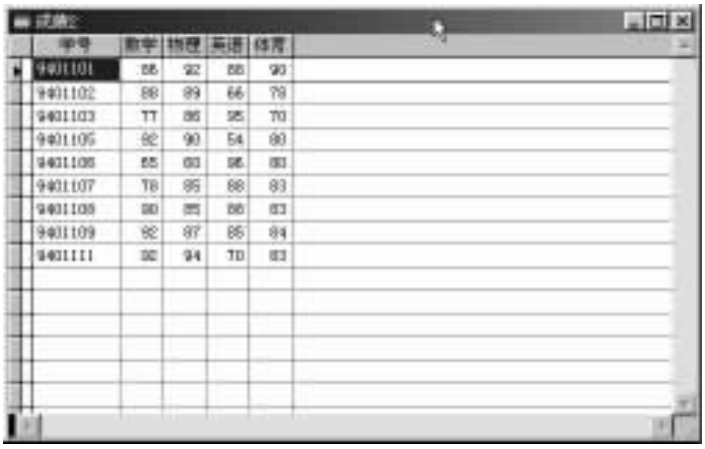


图 3 - 21　左联接

下面是实例中“初级成绩视图 2”内联接 的 SQL 语句：

```
SELECT 基本情况 . 学号,基本情况 . 姓名,成绩 2. 数学,成绩 2. 物理,成绩 2. 英语；
FROM 学生! 基本情况 LEFT OUTER JOIN 学生! 成绩 2；
ON 基本情况 . 学号 = 成绩 2. 学号；
ORDER BY 基本情况 . 学号
```

右联接 Right Outer Join ：联接右表的所有记录和左表的满足联接条件的记录 图 3 - 22 ，左表有，而右表没有的记录用 . NULL. 填充。

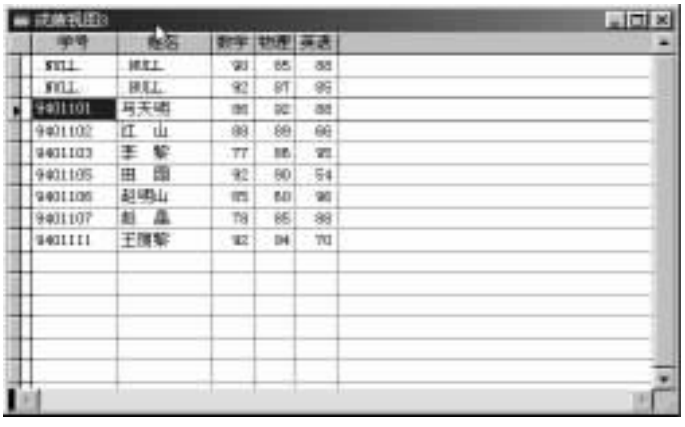


图 3 - 22　右联接

实例中“初级成绩视图 3”右联接 的 SQL 语句如下：

```
SELECT 基本情况 . 学号,基本情况 . 姓名,成绩 2. 数学,成绩 2. 物理,成绩 2. 英语；
FROM 学生! 基本情况 RIGHT OUTER JOIN 学生! 成绩 2；
ON 基本情况 . 学号 = 成绩 2. 学号；
ORDER BY 基本情况 . 学号
```

完全联接 Full Join ：联接所有记录 图 3 - 23 ，不论记录是否一致，没有数据的部分用 . NULL. 填充。其 SQL 语句 实例中“初级成绩视图 4”为：

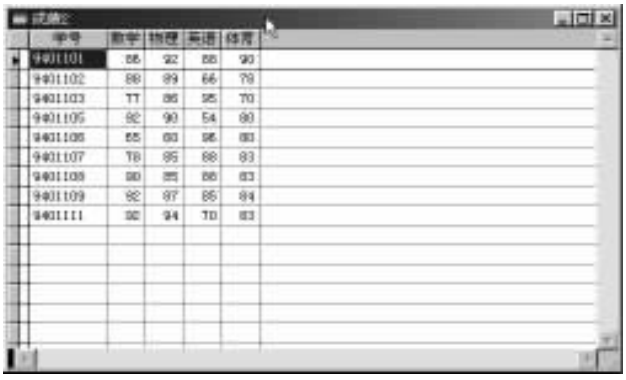


图 3-23　完全联接

SELECT 基本情况.学号,基本情况.姓名,成绩 2.数学,成绩 2.物理,成绩 2.英语;
FROM 学生!基本情况 FULL JOIN 学生!成绩 2;
ON 基本情况.学号;
ORDER BY 基本情况.学号

二、修改联接

对于不合适的联接可以进行修改,方法是:

1 在"视图设计器"中选择"联接"选项卡 图 3-24 ,选择要修改的联接;

2 按需要改变联接条件。

图 3-24　"联接"选项卡

三、删除联接

对于不必要的联接可以将它删除,方法是:

1 在"视图设计器"中选择"联接"选项卡 图 3-24 ;

2 选择要删除的联接单击"移去"按钮。

3.2.3　更新数据

一、使表可更新

更新:把在视图中做的修改保存到相应的表里。

同查询一样,使用视图可以从一个或多个表中查看数据,在缺省状态下视图是不可以更新的,这加强了数据库的安全性。但是我们又常常希望对视图进行的操作能够保存下来,而不仅仅是查看它,这就需要通过修改视图的更新状态,使表可更新。

能够更新表,这也是视图有别于查询的主要特征。查询设计器和视图设计器看起来极为相似,不同的是视图有"更新条件"选项卡,而查询没有。这个区别使视图的功能大大加强。您可以像使用

表一样使用视图，系统则会把所做的修改存放到相应的表中。

对于远程视图，如果想使视图中的修改能回送到源表中，就需要设置“发送 SQL 更新”选项 图 3 - 25 ，对于本地视图它不是必选的。但您必须至少为视图中使用的每个表设置一个关键字段，使系统确定需要更新数据的位置。如果选择的表中有一个关键字段，并且已在“字段”选项卡中选中，则“视图设计器”自动使用表中的该关键字段，作为视图的关键字段。如果在使用中有可以修改关键字，就必须将关键字设为可更新的。

图 3 - 25　“更新条件”选项卡

关键字必须是唯一的，如果使用的关键字不惟一，系统将修改所有关键字相同的记录，比如使用班级号作为关键字，那么对 9502 班中的一个同学的修改会使整个表中所有 9502 班的同学发生同样的修改。

如果没有一个字段是唯一的，建议使用多个关键字，甚至使所有的字段全部为关键字，但是关键字过多会使系统性能下降，有时还会使系统发生错误。

由于系统是通过对关键字构造 SQL 语句来完成更新的，所以过多的关键字会诱发“SQL”语句过长的错误。

二、设置关键字段

当在“视图设计器”中首次打开一个表，“更新条件”选项卡会显示表中哪些字段被定义为关键字段。

如果要设置关键字段，单击“字段名”旁的关键字段列 钥匙图标

如果已经改变了关键字段，又想把它恢复到源表中的初始位置，选择“重置关键字”按钮。

三、使表中的指定字段可更新

如果字段没有标注为可更新，您可以在视图的浏览窗口中修改这些字段，但修改值不会送到源表中。您可以指定表中只有某些字段可以更新，方法是单击“字段名”旁的可更新列 笔形图标 。

四、使所有字段可更新

如果想使表中的所有字段可更新，表中必须有已定义的关键字段，然后选择“全部更新”。

“全部更新”不影响关键字段。

第 4 章　报表与标签

查询和视图能够显示或存储数据，而报表和标签则是数据打印输出的工具。在 VisualFoxPro 中打印数据，并不是直接送到打印机，而是先建立一个报表或标签文件，从数据表中提取内容，并且设计报表或标签格式，然后再打印报表或标签。本章我们将向您介绍如何创建报表和标签。

有关报表和标签的详细内容，请参阅联机文档《用户指南》中的后面“设计报表和标签”。

4.1　使用“报表向导”

使用“报表向导”可以快速创建报表。下面将“成绩视图 1”用“报表向导”设计成报表。

4.1.1　创建报表

1 在“项目管理器”的“文档”选项卡中，选择“报表”。

2 单击“新建”按钮，出现“新报表”对话框。

3 在对话框中单击“报表向导”按钮，出现“向导选取”对话框。该对话框有两个选项：

“报表向导”：用一个单一的表创建一个带格式的报表；

“一对多报表向导”：创建一个内容包含了一组父表的记录及相关子表的记录的报表。

4 选择“报表向导”后，单击“确定”按钮，进入“报表向导”步骤 1“字段选取” 类似图 1 - 8 表单向导步骤 1 。

5 在“数据库和表”中选择“成绩视图 1”，按照向导屏幕上的指令完成后面的操作。最后要保存报表文件 . frx ，在保存对话框中输入报表文件名 本例中保存为“初级成绩表 1”。

4.1.2　预览已有报表

用报表向导 或使用报表设计器 创建报表后，可以通过预览查看报表的设置是否符合要求。

1 在“项目管理器”的“文档”选项卡中，选择“报表”；

2 在“报表”中选择报表名；

3 单击“预览”按钮，屏幕显示如图 4 - 1，用垂直滚动条和水平滚动条可以看到屏幕外的内容。

图 4 - 1　报表预览窗口

您可以单击“打印预览”工具上的“◀”、“▶”等按钮 如果有多页时 ,以查看报表的内容 如果要将结果打印出来,请单击“打印机”按钮 如果要退出预览,请单击右上角“ × ”按钮。

4.1.3　修改已有报表

如果想对已有的报表的外观进行修改,可以按下步骤操作：

1 在“项目管理器”的“文档”选项卡中,选择“报表”。

2 在“报表”中选择报表名。

3 单击“修改”按钮,进入图 4 - 2 的“报表设计器”窗口。

在“报表设计器”中,有报表区和工具栏,我们可以使用工具栏中的“工具”对已有的报表区进行修改,包括标题字体等。

4 单击“确定”按钮。

下面介绍标题和字体的修改方法。

图 4 - 2　设置字体对话框

1. 将标题居中

在图 4 - 1 的报表预览中看到标题“成绩”和日期在页面的左边,如何将它们显示在页面的中间呢

1 在“报表设计器”中,单击“报表控件”工具栏的“选定对象”按钮,该按钮被按下,表示被选中；

2 选择文字“成绩”,文字旁边出现四个点,表示被选择；

3 单击“布局”工具栏的“水平居中”按钮,文字被移到页面中间。

如果您的屏幕上没有“报表控件”工具栏和“布局”工具栏,请选择菜单 :显示⇨报表控件工具栏 或显示⇨布局工具栏 将它们打开。

2. 改变字体

如果改变字体的属性,可以按以下步骤操作：

1 单击“报表控件”工具栏的“选定对象”按钮,该按钮被按下,表示为被选中；

2 选择文字“成绩”,文字周围出现四个点,表示被选择；

3 选择菜单 :格式⇨字体,出现“字体”对话框 图 4 - 3 ,在对话框中可以选择报表中的字符的“字体”、“字体样式”、“字体大小”,还可以设置它的颜色、下划线等。

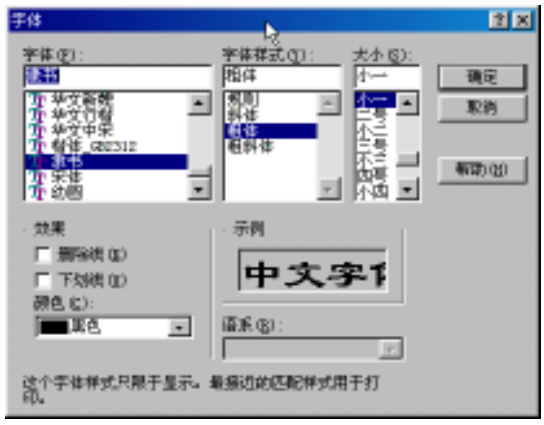


图 4-3　设置字体对话框

修改后可选择菜单：显示⇨预览，进入图 4-1 的报表预览窗口，预览报表。如果您认为还有不满意的地方，可以返回报表设计器，继续进行修改。

4.2　“报表设计器”的使用

如果您想自己设计报表中的所有格式和内容，就要学会使用“报表设计器”。下面我们用“报表设计器”设计一张“学生社会关系”的报表。

4.2.1　创建报表

一．启动“报表设计器”

1 在“项目管理器”的“文档”选项卡中，选择“报表”；

2 单击“新建”按钮，出现“新报表”对话框；

3 单击“新报表”按钮，出现一个空白的“报表设计器”窗口 图 4-4 。

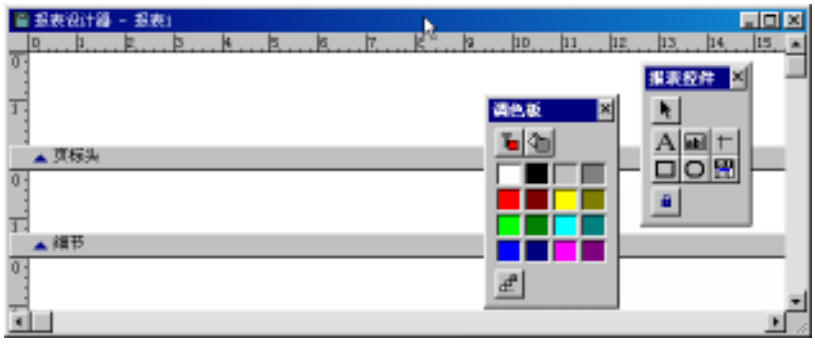


图 4-4　报表设计器

二、“报表设计器”窗口

在“报表设计器”窗口中，有报表区和工具栏。

工具栏包含报表控件工具栏、布局工具栏、调色板工具栏。

1 “报表控件”工具栏：使用它可以向报表中添加对象。在该工具中有：

“选定对象”按钮：当您在报表中要移动、删除对象时，请先将它按下，再用鼠标选择对象。

“标签”按钮：如果要打印标题、公司名称等文本时请使用它添加标签对象。您希望什么地方出现文本，就将标签放在什么地方，但添加的标签字符数不能超过 256 个。

“域控件”按钮：向报表中添加字段要用的域控件。系统日期、页码等变量也使用域控件添加。

控件：由系统提供或由用户预选制作好的类。它的代码是经过严格调试，且没有任何错误的。对用户来说，类如何实现它的功能是完全透明的。有关类的概念在第五章介绍。

“线条”、“矩形”、“圆角矩形”按钮：当您在报表中画水平线、垂直线、矩形或圆角矩形时，就要用到它们。添加形状对象，可以使报表更加美观。

“图片/OLE 绑定型控件”按钮：可添加照片、商标及其他图像对象，使报表更生动形象。

“按钮锁定”按钮：如果要连续使用同一个按钮，将“按钮锁定”按下，就不必每次都去选择同一个按钮了。

2 “布局”工具栏：在调整对象的位置时，“布局”工具可以帮助您高效、快速地完成任务。

3 “调色板”工具栏：用它报表中的对象添加颜色。使用彩色打印机就可打印出彩色的报表。在 VisualFoxPro 中，只要在调色板上单击鼠标即可选择颜色，每一对象可以设置两种颜色：即前景色和背景色。前景色：用于文本和线，背景色：用于除文本和线之外的地方。

4 标尺：报表区的上面和左面分别有水平标尺和垂直标尺，用于标记对象的坐标。

在报表区有如下不同的区域：

1 标题区：不论报表有多少页，每张报表只打印一次标题，当报表多于一页，只有第一页有标题，其他页则没有。

2 页标头区：页面的标头，每个页面都有一个页标头，如果希望每页都打印某一相同的信息，请将该信息放在页标头区。

3 细节区：记录的内容是放在细节区的。在此处放置字段名或表达式，打印次数受记录数和页面分配给细节区的大小而定。

4 页脚注区：打印页码、制表人等信息。

观察标尺，移动栏（如页标头栏），可以调整区（如页标头区）的大小。

图 4 - 5 和图 4 - 6 是报表区与报表对应位置的比较。

图 4 - 5　报表设计器

成绩

03/15/99

学号	姓名	数学	物理	英语	体育	总分
9401101	马天明	86	92	88	90	356
9401102	江 山	88	89	96	78	321
9401103	王 黎	77	86	95	70	328
9401105	田 园	92	90	54	80	316
9401106	赵明山	65	60	96	80	301
9401107	刘 昌	76	85	88	83	334

图 4-6　报表

三．设置数据环境

设计报表的目的是要打印数据，因此应该告诉报表设计器的数据源的位置。

1 在“报表设计器”中单击右键；

2 选择“数据环境”，出现“数据环境设计器”；

3 在“数据环境设计器”中单击右键；

4 选择“添加”按钮，出现“添加表或视图”对话框 类似图 3-1 ；

5 在对话框中，将“基本情况”表和“社会关系”表添加到数据环境；

6 单击“关闭”按钮。

在“数据环境设计器”中 图 4-7 ，我们看到两个表之间有连线，这是由于它们有共同的字段—“学号”，系统自动将其设为永久关系。

图 4-7　“数据环境”设计器

报表布局并不将数据排序，它只是按数据在数据源中存在的顺序处理数据。如果您希望报表的信息是经过排序的，请为表设置索引或将视图、查询作为数据源。

四．将字段添加到细节区

有了数据源，下一步就可以将表中的数据放到报表中，步骤如下：

1 在“数据环境设计器”中选择字段；

2 将其拖住“报表设计器”的“细节”区，如果有多个字段，就要拖动多次；

3 单击“报表控件”工具栏的“选定对象”按钮；

4 选择“细节”区中的字段，被选字段周围出现四个点；

5 选择菜单 格式⇨对齐⇨顶边对齐，细节区的字段排列成一线；

6 选择菜单 显示⇨预览，预览报表，查看字段的安排是否满意，如不满意可返回重新设计。

4.3　报表中对象的操作

在图 4 - 4 中您已经看到“报表设计器”中有三个工具栏：“报表控件工具栏”、“布局工具栏”和“调色板工具栏”。用“报表控件工具栏”可以添加对象，而用另外两上工具栏则可以修饰对象。

4.3.1　填写页标头

将字段从“数据环境设计器”中拖拽到“报表设计器”的“细节”区，系统会根据字段宽度自调整该字段所占空间。因此先添加“细节”区的字段，然后根据字段占用的空间再填写“页标头”的文字，这样便于文字的安排。

1 单击“报表控件”工具栏的“标签”按钮，该按钮被按下，表示被选中；

2 单击“报表控件”工具栏的“按钮锁定”按钮；

3 参照“细节”区字段的位置，在“页标头”区输入文字；

4 单击“报表控件”工具栏的“选定对象”按钮；

5 选择“页标头”文字和与其对应的“细节”区字段名，将它们同时选择，以便对齐，例如在图 4 - 8 中按住 Shift 键同时用鼠标左键选择“页标头”区和“细节”区的“学号”；

6 单击“布局”工具栏的“垂直居中”按钮，被选文字垂直居中对齐；

7 重复第 5 和第 6 步，将所有文字和字段名对齐。

选择多个对象的方法是，按下“报表控件”工具栏的“选项对象”按钮，按住 Shift 键，用鼠标左键单击所有要选的对象。

4.3.2　填写标题

将标题放在“标题”区，打印出的报表只有第一页有标题。如果希望在报表的每一页面上方都出现标题，可以将标题放在“页标头”区。

1 将鼠标放在“页标头”栏，按住左键往下拖，使“页标头”区的高度满足要求。“报表设计器”的左边有垂直标尺，您可以通过它观察移动的距离；

2 单击“报表控件”工具栏的“标签”按钮；

3 将鼠标放在“页标头”区，输入文字“学生社会关系表”；

4 选择菜单“格式⇨字体”，在“字体”对话框中设置“字体”、“字体样式”、“字体大小”；

5 单击“确定”按钮，返回到“报表设计器”；

6 单击“布局”工具栏的“水平居中”按钮，使标题居中放置；

7 选择菜单“显示⇨预览”，看看设置是否满意，如果不满意还可以用上面的方法再修改。

4.3.3　绘制线条

使用“线条”控件，可以在报表中添加垂直和水平直线，以增强报表布局的视觉效果。

1 单击“报表控件”工具栏的“线条”按钮；

2 在报表页面中拖动线条。

绘制线条后，可以移动或调整其位置及长短，或者更改它的粗细和颜色。

单个对象的对齐方式用“水平居中”和“垂直居中”。对于多个对象，处于不同行的用“垂直居中对齐”、“水平居中对齐”、“左对齐”或“右对齐”；处于同一行的用“顶边对齐”或“底边对齐”。

当您执行某操作后想撤消此操作,请选择菜单 编辑⇨撤消。但它只能撤消最近的一次操作。

4.3.4 更改线条粗细或样式

我们可以更改线条的粗细 从细线到六磅粗的线 ,也可以更改线条的样式 从直线、点线到点线和虚线的组合 。

1 单击“报表控件”工具栏的“选定对象”按钮,选定希望更改的线条;

2 选择菜单 格式⇨绘图笔;

3 从子菜单中选择适当的粗细或样式。

4.3.5 添加制表日期

您是否想把制作报表的日期打印在报表上呢 下面我们将制表日期添加在报表的右上角。

1 单击“报表控件”工具栏的“域控件”按钮;

2 在报表页面的右上角拖拽出一个矩形,当您拖拽完毕弹起左键时,就出现一个“报表表达式”对话框 图 4-8 ;

3 单击“表达式”右边的“…”按钮,会出现图 4-9 所示的“表达式生成器”;

4 单击图 4-9 中“日期”下拉式箭头,双击“DATE ”,“DATE ”就出现在图 4-9 的“报表字段的表达式”框内。“DATE ”是日期函数,它返回当前系统日期;

5 单击“确定”按钮,回到“报表表达式”对话框;

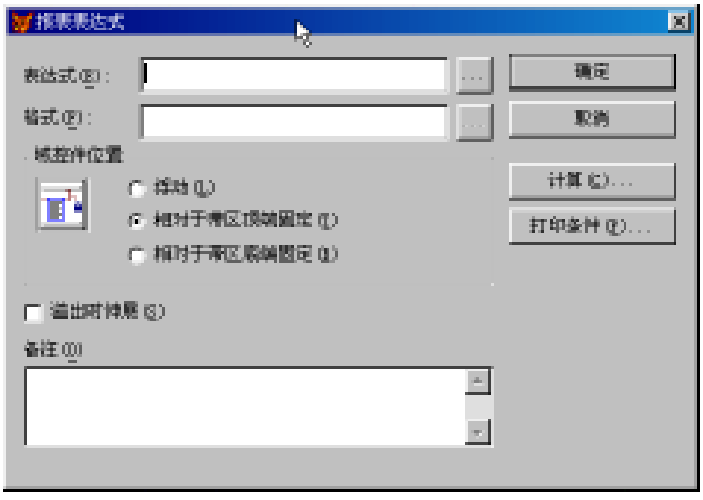


图 4-8 报表表达式

6 单击“确定”按钮,返回“报表设计器”。

通过预览,您会看到在报表的右上角有了系统日期。

4.3.6 添加页码

当报表页数多于一页时,您一定希望能在报表上反映出当前页码。

1 单击“报表控件”工具栏的“域控件”按钮;

2 在报表“页注脚”区拖拽出一个矩形,出现图 4-8 所示的“报表表达式”对话框;

3 单击“表达式”右边的“…”按钮,出现图 4-9 所示的“表达式生成器”;

图 4 - 9　表达式生成器

4　选择“字符串”下拉式箭头，双击“文本”，双引号出现在图 4 - 9 所示的“报表字段的表达式”框内；

5　在双引号内输入“第”，您也可以直接输入“第”，但一定要用引号将它括起来；

6　将光标移到行末，输入“ + ”；

7　选择“字符串”下拉式箭头，双击“ALLTRLM expC ”，该函数出现在“报表字段的表达式”框内，光标停在 expC 内，ALLTRIM 函数从字符串表达式中删除前导空格和后继空格，并返回一个整齐的字符串表达式；

8　选择“字符串”下拉式箭头，双击“STR expN ”，该函数出现在“报表字段的表达式”框内，光标停在 expN 内，STR 函数将一个数值型表达式转换成一个字符串表达式；

9　在“变量”框内双击“ - pageno”，使 - pageno 代替“报表字段的表达式”框内的“expN”，- pageno 是系统内存变量，决定当前的页号；

10　将“ - pageno”后面的逗号删掉；

11　将光标移到行末，输入“ + ”；

12　输入汉字“页”，表达式结果为：“第 + ALLTRIM STR - pageno + 页 ，系统内存变量“ - pageno”是数值型的，用“STR ”函数将它转换为字符型，再用“ALLTRIM ”函数将前、后空格去掉，“第”与“页”采用字符串照原样打印，用“ + ”号将三部分字符串连接。其含义是打印出“第 n 页”，n 取自系统内存变量；

13　单击“确定”按钮，回到“报表表达式”对话框；

14　单击“确定”按钮，返回“报表设计器”。

这样设置后在报表的每一页都会有对应的页码。图 4 - 10 是添加了线条、日期和页码的报表。

如果您要查看或修改“域控件”内的表达式，可以用鼠标右键单击“域控件”，在快捷菜单 图 4 - 11 中选择“属性”，将会打开图 4 - 8 的“报表表达式”对话框，使用上面的方法在“报表表达式”或“表达式生成器”中查看或修改表达式。

图 4-10　加线条、日期和页码的报表

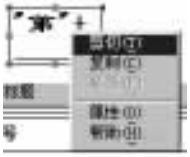


图 4-11　对“域控件”单击右键

4.3.7　为报表添加颜色

1 选择要添加颜色的对象；

2 在“调色板”工具栏中，选择“前景色”或“背景色”。如果屏幕上没有“调色板”工具栏，请选择菜单：显示⇨调色板工具栏，将工具栏打开；

3 单击选中的颜色。

4.3.8　在报表中加图片

在 VisualFoxPro5.0 中只能加入 BMP 图片，而 VisualFoxPro6.0 可以加入几乎所有格式的图片，如 JPG、GIF、ANI、CUR、DIB、ICON 等。当磁盘中有图形文件时，在报表中加图片是一个调用过程，具体步骤为：

1 单击“报表控件”工具的“图片/OLE 绑定型控件”按钮；

2 在报表中想放置图片的位置上拖动鼠标，出现图 4-12 所示的“报表图片”对话框。在该对话框中选择“图片来源”的“文件”；

图 4-12　报表图片对话框

3 单击“文件”右边的“…”按钮,出现打开文件对话框;

4 在对话框中选择所要的图形文件名,如果需要,设置大小、位置或打印选项;

5 单击“确定”按钮,回到“报表图片”对话框;

6 单击“确定”按钮,图片显示在报表的相应位置中。

4.3.9　用网络线调整对象的位置

网络线可以帮助您在布局上放置对象。在缺省情况下,对象根据网格对齐位置 网格线不会被打印在纸上 。

一. 将对象放置在特定的位置

1 选择菜单 :显示⇨显示位置;

2 选择一个控件,状态栏上就显示该对象的位置信息;

3 将对象拖拽到特定位置。

二. 显示网格线

选择菜单 :显示⇨网格线,网格线将出现在报表区。

三、更改网格的度量单位

1 选择格式⇨设置网格刻度,出现“设置网格刻度”对话框;

2 在“网格”框内,输入每一网格水平宽度和垂直高度的像素数目;

3 单击“确定”按钮。

4.3.10　在报表中添加字段值

“域控件”是个工具,利用它可以显示数据库字段内容,还可以显示多个字段的运算结果 如字符串相加、累计求和、求平均值等 。实例中名为“成绩表 2”的报表,其数据环境是“基本情况”表和“成绩 2”表。用域控件可以在报表中求总分,如图表 4 - 13 所示。

成绩

学号	姓名	数学	物理	英语
9401101	马大明	86	92	88
9401102	江　山	88	89	66
9401103	李　莉	77	86	95
9401105	田　园	92	90	54
9401106	赵明山	65	60	96
9401107	赵　磊	76	85	88

图 4 - 13　成绩表 2

1 单击“报表控件”工具的“域控件”按钮;

2 在“细节”区拖拽出一个矩形,当您拖拽完毕,弹起左键时,就出现一个“报表表达式”对话框 图 4 - 8 ;

3 单击“表达式”右边的“…”按钮,将会出现图 4 - 14 的“表达式生成器”;

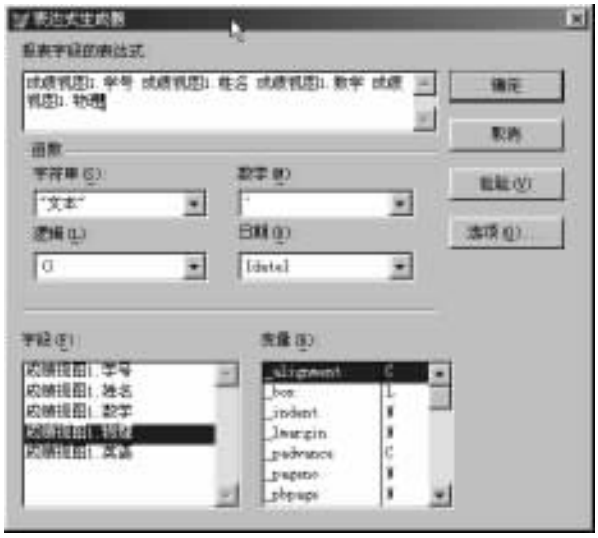


图 4 - 14　表达式生成器

4 单击图 4 - 14 中“字段”下拉式箭头，双击“成绩 2. 数学”，就出现在图 4 - 14 的“报表字段的表达式”框内；

5 在“成绩 2. 数学”后面输入“ + ”号，并添加各科成绩字段，如图 4 - 14 所示；

6 单击“确定”按钮，回到“报表表达式”对话框；

7 单击“确定”按钮，返回“报表设计器”；

8 单击“报表控件”工具栏的“标签”按钮；

9 在域控件的上方，“页标头”区输入汉字“总分”；

10 选择菜单 格式⇨字体，在“字体”对话框中设置“字体”、“字体样式”、“字体大小”；

11 单击“确定”按钮，返回“报表设计器”；

12 选择菜单 显示⇨预览，将会看到图 4 - 13 的结果。

通过上面的练习，相信您已经初步掌握了“表达式生成器”的使用。“表达式生成器”的主要目的是提供选择报表内容，使创建表达式更容易。我们可以从各种设计器、窗口、生成器和向导中启动它。要创建表达式，可直接在表达式框中键入，也可以从下拉列表中选取，双击它使其粘贴到表达式框中。

4.4　将数据分组

通过报表的字段值，可以在表或视图中将信息进行分组，这样会使报表更易于阅读。例如，可以把组设在“年级”字段上来打印所有记录，同一年级的记录在一起，数据源必须按“年级”字段排序。

如果数据源是表，记录的顺序可能不适合分组。这时就应该为表设置索引，或者在数据环境中使用视图、查询作为数据源，把数据适当排序来分组显示记录。数据分组的步骤为：

1 选择菜单 报表⇨数据分组，出现“数据分组”对话框 图 4 - 15 。

图 4 - 15　“数据分组”对话框

2 选择第一个“分级表达式”框右边的“…”按钮,出现“表达式生成器” 图 4 - 14 。

3 在“表达式生成器”中创建表达式。

此处的表达式为“SUBSTR 基本情况 . 学号,1,2 ”意思是按学号的前两位分组 在本系统中学号的前两位决定年级,94 表示 94 级,95 表示 95 级 。

SUBSTR 函数从给定字符表达式中返回一个字符串,字符表达式后面的第一数字是返回字符串的起始位置,第二个数字是返回字符串的长度。本例从“基本情况”表的“学号”字段返回加字符串,起始位置是第 1 位,长度共 2 位。

4 单击“确定按钮,返回“数据分组”对话框。

5 在“组属性”区域,选定想要的属性。

“组属性”有如下选项:

每组从新的一页上开始 :选择它,一页纸上不会混有多个组。若不选此项,同一页纸上组与组是顺序排列的。

每组的页号重新从 1 开始 :使每一组作为独立的报表,页号重新开始。若不选此项,报表的页号是顺序排列的。

每页都打印组表头 :在组出现的每页上都打印组标头,若不选此项,组标头只在组第一次出现时打印。

6 单击“确定”按钮。

图 4 - 16　组标头和组注脚

分组之后，有两个新的组域会自动插入报表中，一个是组首（组标头），另一个则是组尾（组脚注）。有关组标题方面的信息可以放在组首，而概括性信息及计算结果可以放在组尾。

一开始，组的首尾两部分（图 4-16）是没有空间存放对象的，要想在组首和组尾配置空间，只需用鼠标在该区域中点击并往下拖。如果空间做得太大，用鼠标在区域中点击并往上拖。

可以在报表内最多定义 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。

4.5 标题和总结带区

4.5.1 添加标题和总结带区

“标题”带区包含在报表开始时要打印的信息，“总结”带区包含报表结束时要打印的信息。它们都可以单独占用一页。带有总计表达式的域控件，放置在“总结”带区后，将对表达式涉及的所有数据求和。

选择有单（报表⇨标题/总结，进入“标题/总结”对话（图 4-17），在对话框内做如下选择：

标题带区：若选择它，将产生一个标题带区。

图 4-17 “标题总结”对话框

图 4-18 总结带区

新页：如果希望标题成为独立的一页，请选择它。

总结带区：如果要对报表的数据求总，选择它将产生一个总结带区。

新页：选择它，总结将占用独立的一页。

经过上述选择后，单击“确定”按钮返回“报表设计器”，您将看到报表区增加了新带区（图 4-18）。

4.5.2 在总结带区添加对象

1 在总结带区放入标签对象，输入汉字“总计”；

2 在总结带区放入域控件，进入“报表表达式”对话框；

3 单击“表达式”右边的“…”按钮进入“表达式生成器”；

4 在“字段”列双击“成绩 2. 数学”，该字段出现在“报表字段的表达式”区；

5 单击“确定”按钮，回到“报表表达式”对话框；

6 单击“计算”按钮，出现“计算字段”对话框（图 4-19）；

图 4 - 19　“计算字段”对话框

7 单击“总和”按钮,表示在“总结”带区打印数据的总和;

8 单击“确定”按钮,回到“报表表达式”对话框;

9 单击“确定”按钮,返回“报表设计器”。

重复第 2 步到第 9 步,将物理、英语、体育、总分成绩都用域控件设置,设置结果如图 4 - 18。选择菜单:显示⇨预览,预览结果如图 4 - 20。

图 4 - 20　“总结带区”的“总和”数据

4.6　定义报表的页面

规划报表时,通常会考虑页面的式样。例如页边距,纸张类型等。如果要对页面进行设置,可以进行如下操作:

1 选择菜单:文件⇨页面设置进入图 4 - 21“页面设置”对话框。

在该对话框中设置下列参数:

列:页面横向打印的记录的数目,而不是单条记录的字段数目。

列数:默认值为 1,如果希望在页面上打印两列数据,请在“列数”区输入“2”,其他以此类推。

宽度:指列在页面的宽度。

间隔:列与列之间的距离,如果“列数”为 1,则没有间隔。

左页边距:您可以根据信息在页面的分布,调整数据与页左边的距离。

2 选择“打印设置”,进入图 4 - 22 的“打印设置”对话框。

图 4 - 21　页面设置

图 4 - 22　打印设置对话框

3 从“大小”列表中选定纸张大小。

4 从“方向”区选择一种方向,即“纵向”还是“横向”。

5 单击“确定”按钮,回到“页面设置”对话框。

6 单击“确定”按钮,返回“报表设计器”。

4.7　创建标签

标签是多列报表布局,为适应特定标签纸而具有对列的特殊设置。在 VisuaalFoxPro 中,可以使用“标签向志”或“标签设计器”迅速创建标签。当您的数据库中有学生的姓名、邮编和通讯地址等通讯信息,就可以用标签打印邮件标签,用于发入学通知书等。下面我们用“标签向导”将“基本情况”表的相关数据打印出来。

4.7.1　启动“标签向导”创建标签

利用“标签向导”是创建标签的简单方法。用向导创建标签文件后,可以再用“标签设计器”修

改标签文件。

1 在“项目管理器”中选择“标签”；

2 单击“新建”按钮，进入“新标签”对话框；

3 选择“标签向导”，进入“标签向导”步骤 1 选取表，在“数据库和表”区域选择“基本情况”表；

4 单击“下一步”按钮，进入“标签向导”步骤 2 图 4 - 23 选择标签类型；

图 4 - 23　“标签向导”步骤 2

5 单击“下一步”按钮，进入“标签向导”步骤 3 图 4 - 24 定义标签的布局，将左边“有效字段”内的字段放入右边的“已选字段”中。

图 4 - 24　“标签向导”步骤 3

如果您在选取字段时不单击“♩”按钮，字段将在“已选字段”区同一行依次排列，反之则分行排列。单击“空格”按钮，则在字段中添加空格 如图 4 - 24 。

6 单击“下一步”按钮，进入“标签向导”步骤 4 排序记录；

7 单击“下一步”按钮，进入“标签向导”步骤 5；

8 单击“完成”按钮，在“另存为”对话框中为标签文件选择存放的磁盘、文件夹，输入文件名；

9 单击“保存”按钮，返回系统主画面。

4.7.2 预览已有标签

在打印标签之前，我们可先预览它，看看设计是否合理。

1 在“项目管理器”中，选择标签名；

2 单击“预览”按钮，进入预览窗口，显示标签如图 4 - 25 所示。此标签的文字设置太紧密，必须修改。

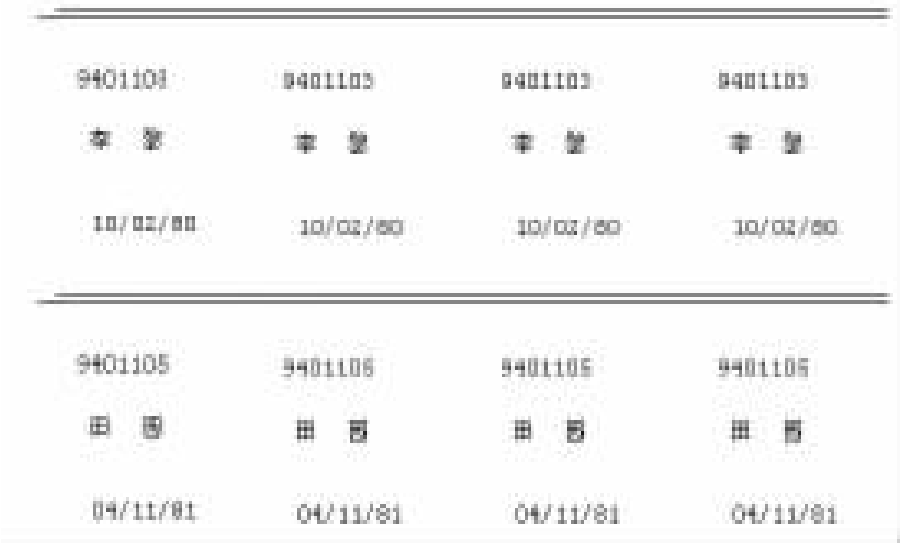


图 4 - 25 标签预览

4.7.3 修改已有标签

1 在“项目管理器”中，选择标签名；

2 单击“修改”按钮。

进入“标签设计器”。“标签设计器”是“报表设计器”的一部分，两种设计器使用相同的菜单和工具栏，但使用不同的默认页面和纸张。您可以使用修改报表的方法对标签进行修改。对于图 4 - 25 的标签，可以用添加空格和回车键增大间距。

4.8 报表 或标签 创建过程小结

通过上面的练习，我们可以得出创建表或标签的过程：

1. 决定布局

在创建报表 或标签 之前,应该先确定其格式。报表可以简单的象电话号码列表一样,也可以由多表数据得出企业经济效益指标。为了使界面更加美观,如果数据的字段多,页面列数取缺省值1,如果数据字段少,就可以将列数据设置为 2 或者 3。

2. 确定数据源

确定在报表 或标签 中打印的数据来自哪个表或视图。

3. 设置布局

使用“报表向导”、“报表设计器”或“标签向导”、“标签设计器”设置报表 或标签 的布局。

第 5 章　表单

在 Visual FoxPro 中表单是面向对象编程的主要工具，在一般应用中，面向对象编程的大多数工作将在表单中进行。设计表单的过程就是设计程序界面的过程。通过表单的设计，设计出用户界面，然后运行它，使用户能够与系统是行交互操作。

就像报表可以简化表和查询的打印工作一样，表单为数据库信息的显示、输入和编辑提供了非常方便的方法。您能够用过去常用的纸面表单的形式来创建表单，以此提供一个人们所熟悉的数据操作环境。

有关表单的详细内容，请参阅本书第九章和联机文档《用户指南》中的第五章“用表单管理数据”

5.1　创建表单

表单显示了表和视图中的字段和记录，而且包含定位控件，以帮助用户从一条记录移到另一条记录。在 Visual FoxPro 中可以使用“表单向导”或“表单设计器”创建表单。

5.1.1　用“表单向导”创建一对多表单

要创建新表单，可以用表单向导开始工作。向导会根据您对一系列问题的回答生成一个表单。您可以在几种不同的类型选项中进行选择，并在创建之前预览它。

Visual FoxPro 提供了两个不同的表单向导帮助我们创建表单：

表单向导 :创建基于一个表的基本表单，我们在第一章曾使用过它。

一对多表单向导：包含两个表中按一对多关系链接的数据的表单。一对多是指一条记录对应多条记录，因此需要两个表，一方称为父表，多方称为子表。例如，“基本情况”表保存着每个学生的学号、姓名、籍贯等基本数据，一个学生只对应一条记录，而“社会关系”保存着学生的父母及兄弟姐妹的信息，有的学生会有若干条记录，而有的学生则可能一条记录也没有。两张表用“学号”链接，“基本情况”表是“一方”，“社会关系”表是“多方”。

下面我们就用“一对多表单向导”为这两个表创建一个表单 实例中“初级社会关系”表单 。

一、创建表单

下面是向导建立一对多表单的步骤：

1 在“项目管理器”中选择“文档”选项卡，然后选择“表单”；

2 单击“新建”按钮，出现“新建表单”对话框；

3 选择“表单向导”，出现“向导选取”对话框；

4 选择“一对多表单向导”，单击“确定”按钮，进入表单向导步骤 1；

5 选择“基本情况”表 父表 ，将需要的字段放入“可用字段”列；

6 单击“下一步”按钮，进入表单向导步骤 2；

7 选择“社会关系”表 子表 ，将需要的字段放入“可用字段”列；

8 单击“下一步”按钮，进入表单向导步骤 3，为两个表设置关联；

9 单击“下一步”按钮，进入表单向导步骤 4，选择表单样式和按类型；

10 单击“下一步”按钮，进入表单向导步骤 5，您可以选择“预览”看看表单样式是否满意，如果不满意，单击“上一步”重新选择样式；

11 单击“完成”按钮，在“存为”对话框中为表单选择磁盘和文件夹，并命名表单为“社会关系”；

12 单击“保存”按钮，回到“项目管理器”。

二、运行已有表单

创建表单后，可以随时运行它。

1 在“项目管理器”中选择表单文件名，单击“运行”按钮。

您就会看到如图 5 - 1 所示的表单。表单的上部是“父表”数据，中部是“子表”数据，“父表”的一条记录，对应“子表”若干条记录。下部是控制按钮，选择它们，可以前一个、下一个地查看记录 这里的“前一个”、“下一个”等按钮是对父表而言，如果要查看子表信息，请使用滚动条 ，还可以执行添加、编辑、删除等操作。

2 单击“添加”按钮，屏幕出现图 5 - 2 对话框，询问您向哪个表添加记录。

根据对话框揭示，选择是向“父表”添加记录，还是向子表添加记录，或是向两者添加记录。

3 单击表单 图 5 - 1 右下角的“退出”按钮，回到“项目管理器”。

图 5 - 1　一对多表单

图 5 - 2　一对多表单“添加记录”对话框

三、修改已有表单

使用“表单向导”创建的表单可以对它进行修改。

在“项目管理器”中选择表单文件名，单击“修改”按钮进入“表单设计器”图 5-3 。在这里您可以按照自己的意愿修改已创建的表单。例如将表头居中，字体字号重置等。

5.1.2 用“表单设计器”创建表单

Visual FoxPro 6.0 **提供的“表单设计器”功能很强，我们在前面修改表单用的就是它。如果您想创建一个灵活的表单，请直接使用它。下面用“表单设计器”创建一个学生“基本情况”表单 实例中的“初级基本情况 1”，步骤如下：**

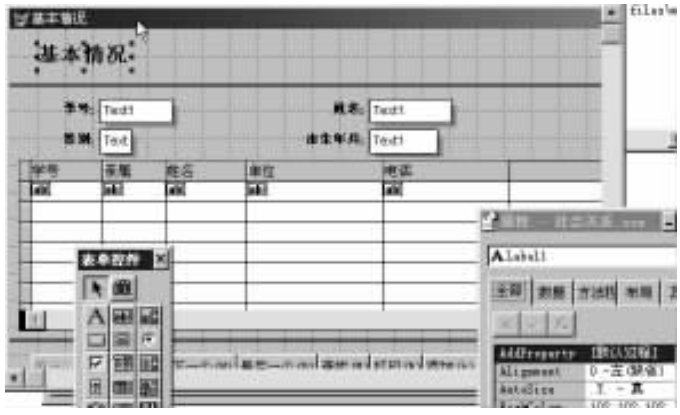


图 5-3 “表单设计器”

一、启动“表单设计器”

1 **在“项目管理器”中选择“文档”选项卡，然后选择“表单”；**

2 **单击“新建”按钮，出现“新建表单”对话框；**

3 **单击“新建表单”按钮，进入“表单设计器”如图** 5-4。

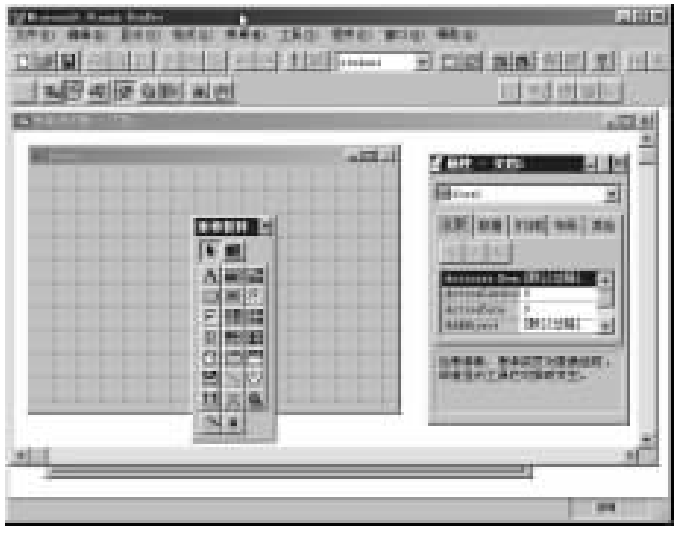


图 5-4 “表单设计器”及其部分工具

二、“表单设计器”窗口

1.“表单设计器”窗口介绍

图 5 - 4 是一个空白表单，上面只有网格线　表单运行时网格线是不会被显示的　。在“表单设计　”中有表单和九种工具栏，图 5 - 4 只放了三个　所有工具按钮均在常用工具栏下方放置，您需要哪一个，单击其按即可　。其中　：

1　“布局”工具栏　：与“报表设计器”中的“布局”工具栏完全一样。

2　“表单控件”工具栏　：与“报表设计器”中的“报表控件”类似，可以向表单添加各种对象，但它比“报表控件”工具强大的多。

3　“属性”工具栏　：设置、编辑表单对象的各种属性值。

在表单中　包括表单　各种对象都有若干属性，例如名字、标题、前景色、背景色等。“属性”工具栏把对象的属性分为五类，用选项卡将它们分别放置。这五类分别是：

数据　：这类属性只影响显示表单时所用的数据。例如指定与对象建立联系的数据源、存储对象的有关信息等。

方法　：列举表单事件属性。事件是针对某个对象所进行的各种操作。例如显示、击键、关闭表单等。

面局　：这类属性包括对象的颜色、字型及字体等。

其他　：前三栏不包括的属性都放在这类。它列举表单命名的属性、表单所用的窗口等。

“全部”选项卡，列出了上面四个选项卡的全部内容。在设置属性时，可以使用“全部”选项卡，也可以单击其他卡。

2.　用表单设计器修改对象属性

在表单上添加一个对象，系统会自动在“属性”工具中增加该对象的相应属性值。如果你想修改某对象的某属性值，方法为：

图 5 - 5　“属性”工具

1　在表单中选择对象，此时“属性”工具就定痊在该对象名上　您也可以在“属性”工具上方的下拉式列表框中选择对象　。

2　单击“属性”工具的属性名，该属性值就显示在属性文本框中。如图 5 - 5 对“Form1”，选择了属性“Caption”，其属性值马上对应为“Form1”。

3　在文本框修改属性值　若该值是灰色，表示不能被修改

4　单击文本框左边的“√”，表示确认修改，如果选择“X”则表示取消修改。

用鼠标右键单击“属性”工具左上角的“控制菜单框”,在下拉式菜单中选择“属性说明”。当您选择某一属性时,“属性”窗口的下方就显示该属性的功能,如图 5 - 5 选择“Caption”属性,窗口下方解释为:“指定对象标题文本”。

三、将表 或视图 与表单结合起来

与“报表设计器”一样,需要使用“数据环境设计篇”定义表单要处理的数据对象,具体操作方法如下:

1 在表单上单击鼠标右键;

2 选择“数据环境”,出现“数据环境设计器”;

3 在“数据环境设计器”中单击右键;

4 选择“添加”按钮,出现“添加表或视图”对话框 类似图 3 - 1 ;

5 在对话框中双击“基本情况”表,将它添加到数据环境中。双击的每张表均会弹入“数据环境设计器”中;

6 单击“关闭”按钮,“添加表或视图”对话框消失,在“数据环境设计器”中有了“基本情况”表。

7 单击“ 表单设计器”,“数据环境设计器”消失,进入“表单设计器”。

四、将字段添加到表单

有了数据源,下一步就可以将表中的数据放到表单中,具体操作方法如下:

在“数据环境设计器”中选择字段,将其拖往表单,您希望数据出现在表单的什么位置,就将其相应的字段拖拽到表单的相应位置。

把一个字段拖曳到表单后,该字段的内容变为两部分 图 5 - 6 :

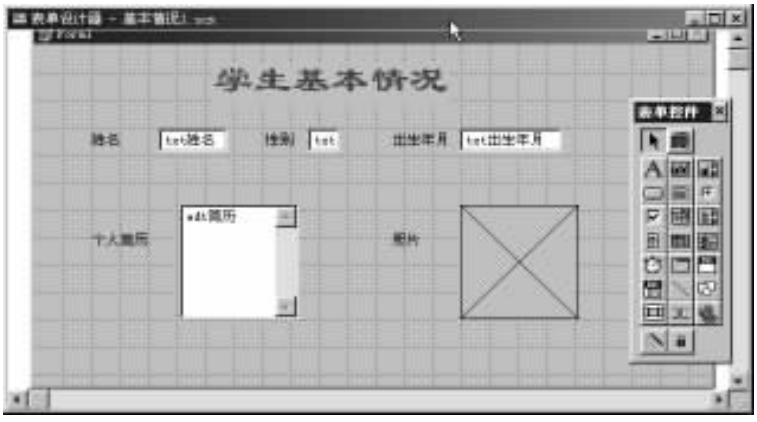


图 5 - 6 添加字段和标签的“表单设计器”

用标签表示字段名 如果定义字段时,给字段名赋予了标题,标签将显示标题 ;

用文本框表示字段的内容 对大多数字段而言 。

您可以使用“布局”工具栏调整它们的位置,如果屏幕上没有该工具,请选择菜单 :显示 ◊ 布局工具栏或单击“表单工具栏”上相应的按钮,将它打开。

重复操作将需要的其他字段拖拽到表单进行位置调整。

五、保存表单

在创建表单过程中请随时将修改保存,以防电源掉电造成数据丢失。保存方法为:

选择菜单 :文件 ◊ 保存,在“另存为”对话框中选择磁盘、文件夹,输入文件名,单击“保存”按

钮,执行保存操作后,可以继续工作。

六、给表单添加标题

为查找方便,应给表单添加标题,添加标题的方法为:

1 单击“表单控件”工具栏,如果屏幕上没有该工具,请选择菜单:显示 ▷ 表单控件工具栏或单击“表单工具栏”上相应的按钮,将它打开;

2 选择“标签”按钮,将鼠标移到表单上,此时鼠标箭头变为十字型;

3 拖拽鼠标指针,在表单上拖出一个矩形,该矩形用来放置标题文字,矩形内的文字 Labell 是标签的缺省标题 图 5 - 6 ;

4 单击“属性”工具,如果屏幕上没有该工具,请选择菜单:显示 ▷ 属性或单击“表单工具栏”上相应的按钮,将它打开;

5 选择 Caption 标题 属性,在文本框输入“学生基本情况”取代 Labell;

6 选择 FontName 字体名 属性,在文本框下拉式列表中选择“隶书”;

7 选择 FontSize 字体大小 属性,在文本框下拉式列表中选择“24”;

8 选择 FontBold 黑体 属性,在文本框下拉式列表中选择“T”;

9 选择 ForeColor 前景色 属性,单击文本框右边的“…”按钮选择您喜欢的颜色。

可以给标签对象继续设置其他属性。

选中对象周围的 8 个点 图 5 - 6 的标签 Labell 组成缩放框,用它可以更改对象的大小,方法是:

对准缩放框,按下鼠标左键不放,鼠标指针变为定向箭头拖拽缩放框,用上下左右键和 Shift 键组合可以微调。

七、给表单添加向导控件

与用向导创建的“个人”表单相比,我们正在创建的“基本情况 1”表单缺少“前一个”、“下一个”、“添加”、“编辑”、“删除”等操作记录的控件。但是从图 5 - 6 中可以看到该表单的对象已经安排得很满,要想继续再添加对象必须把表单尺寸加大。增加表单尺寸的方法为:

1 选择表单,鼠标变为定向箭头后,用鼠标左键拖拽表单的边框;

表单尺寸加大后,我们将向导类添加到“基本情况”表单中,以便控制表单的操作。有关类的详细内容请参考本书第六章。

2 在“表单控件”工具上单击“查看类”按钮,出现快捷菜单;

3 单击“添加”按钮,在“打开”对话框中选择“… / VFP98 / WIZARDS / WIZBTNS. VCX”文件;

4 单击“打开”按钮,该向导类代替原来的常用类,出现在“表单控件”工具栏上。

5 选择“Picbtns”按钮,在表单上拖拽出一个矩形,直到将该类的控件全部显示出来为止,如果表单的长度不够,可设置它的 WIDTH 属性或用鼠标进行拖拽;

6 用“布局”工具栏调整表单对象位置,设计结果如图 5 - 7 所示。设置完毕运行一下,看看是否满意;

7 选择菜单:表单 ▷ 执行表单,或在表单上单击右键,选择“执行表单”,系统会询问您是否要将所做修改保存,请回答“是”。运行结果如图 5 - 8 所示。

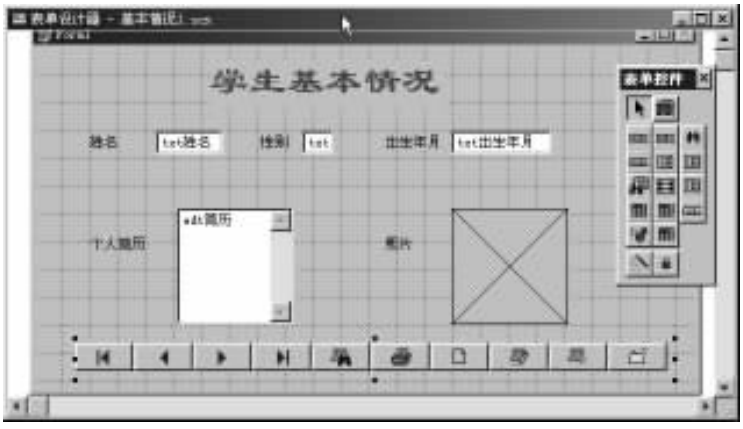


图 5-7　设计完毕的表单

图 5-8　运行“基本情况”表单

在图 5-8 中我们选择的向导类是图片按钮，如果您不知道按钮的含义，请将鼠箭头放在按钮上，系统会显示出该按钮的功能。

5.2　表单常用控件

表单设计过程中最重要的一环，就是向表单添加控件并设置其各种属性、方法和事件。“表单控件”工具栏为表单的设计提供了大量的“零件”，我们可以将它们拖放到要设计的表单上，经过适当的调整后，即可完成一个精美的表单。

打开“表单设计器”后，如果没有“表单控件”工具栏，可以单击“表单控件”工具栏上的“表单控件工具栏”按钮，将它打开。“表单控件”工具栏被打开后，最初显示的只有常用控件，共 25 个按钮。它们是一个“选择”按钮和一个“锁定”按钮 这两个按的功能与“报表设计器”中的功能一样 ；一个“查看类”按钮 ；21 个标准控件和一个“ActiveX”控件。有关 ActiveX 控件在本书第十七章介绍。单击“查看类”按钮，可以增加可视类库 增加方法参阅本书 5.1.2 的第七节 。

向表单中添加控件，只需在工具栏中单击需要的控件，然后在表单相应位置拖拽鼠标或单击鼠标左键，控件对象就会显示在表单中，我们可以在“属性”窗口设置它的各种属性。通过下面的介

绍，希望您能常用控件和常用属性有所了解。

可以利用“复制”和“粘贴”，将添加到表单中的控件生成相同属性的一组控件。

5.2.1　标签

“表单控件”工具栏的大写字母“A”按钮表示标签。标签是一种存放文本的控件，用来对表单上的区域或其他控件加以说明，例如说明表单的名称、字段的名称等。标签有许多属性，包括标签文本的字体、字体颜色、大小等（参阅本书 5.1.2 第六节）。下面您会看到它的另外一些常用属性（参考实例的“初级标签”表单），而对于其余属性使用默主值即可。借助 Visual FoxPro 6.0 的帮助系统，可以进一步了解这些属性。

一、字体

把标签放入表单后的第一件事情是改变标签的默认值文本，以反映您希望表达的意思。改变文本的方法是：

1　在“全部”选项卡或“布局”选项卡中选择 Caption 属性，在文本框输入文本；

2　双击 FontName，选择文本的字体名（默认为宋体）；

3　双击 FontSize，选择文本字体的大小（默认 9）；

4　双击 FontBold，设置文本字体是否为黑体（F 为否，T 为是，默认 F）；

5　双击 FontItalic，设置文本字体是否为斜体（F 为否，T 为是，默认 F）。

如果对系统默认值认可，就不必再设置它们了。

二、颜色

标签有两种颜色：

前景色（ForeColor）：文本的颜色（默认为黑色）。

背景色（BackColor）：除文本外其余部分的颜色（默认为浅灰）

标签的颜色在“全部”选项卡或“布局”选项卡中可以找到，改变标签颜色的步骤是：

1　单击标签，该标签被激活；

2　单击“表单控件工具栏”的“属性”工具，“属性”窗口被反白显示；

3　双击 ForeColor 或 BackColor 属性，显示出一张调色板；

4　单击你想要的颜色。

三、边框样式

创建一个标签后，不选择它，是不会知道该标签的尺寸的，因为它没有边界，利用 BorderStyle 属性，可以设置或取消标签的边框，该属性可以从“全部”选项卡或“布局”选项卡中找到。BorderStyle 有两种可能值：0—无（默认值），1—固定单线。

双击 BorderStyle，以确定是否需要标签边框。每次双击该属性，您都会看到属性窗口的值和表单内该标签的边框在发生变化。

四、背景

标签的默认值为深灰色。当标签的 BackStyle 属性为透明时，您能看到表单，当其为非透明时，可以很明显地看到档签的范围，即 Visual FoxPro 会显示出标签的背景色，并遮盖住表单网格线。

从“全部”选项卡或“布局”选项卡中可以找到 BackStyle 属性，它也有两种可能的值：1—不透明（默认值），0—透明。双击 BackStyle，就可以确定背景的透明与否。每次双击该属性，都会看标签透明与不透明的效果。

五、自动调节标签尺寸

当往标签内输入文字时，如果文字内容多，有些文字就显示不出来，您只有将文本框尺寸拖的大一些才行。有没有好一些的办法呢 答案是肯定的，可以使用 AutoSize 属性来自动调节标签的大小。

AutoSize 属性有两个值：F—否，T—是，缺省值 F。双击 Autosize 就可以设置标签的尺寸是否根据文本自动调节。

六、使用文字竖排

如果想显示竖排的文字，选择 WordWrop 属性。

WordWrop 属性也有两个值：F—否，F 是缺省值为 F。双击 WordWrop 就可以设置标签的文字为竖直排列 前提必须使用 Autosize 属性为 .T. 。

七、显示立体文字

将表单的标题显示成立体字，将会使您的表单增色不少，实现的方法如下：

1 在“表单控件”工具栏选择“标签”按钮；
2 在表单上合适的位置单击；
3 在“属性”窗口输入文本；
4 设置 AutoSize 属性为 .T. ；
5 设置文本的字体名、字体号、字体颜色等属性；
6 复制该标签；
7 执行粘贴操作
8 修改复制的标签文本的字体颜色；
9 移动二个标签的位置，使两个标签文字呈现立体状。

设置结果见实例的“初级标签”表单。

5.2.2 文本框

文本框是一类基本控件，它允许用户添加或编辑保存在表中非备注字段中的数据。文本框包含一行单独的文本，这行文本可以由使用表单的人输入或改写，所以您可以使用文本框来显示、输入或修改数据库所存的信息。

“表单控件”工具栏的“ab”按钮表示文本框。单击该按钮，再把文本框放入表单中，系统就会自动显示出文本框。文本框围绕着“Text 1”，这是文本框的缺省标题。

一、文本框的字体、颜色属性

标签设置的属性，只要能在文本框中找到，对文本框也同样适用。字体、颜色、边框样式、背景等属性的使用方法与标签一样。

有关文本框的练习在实例的“初级文本框”表单中。

二、将文本框与字段相连

前面我们说过，在文本框中出现的文本通常是数据库某表中一个字段的内容。如何将文本框与字段相联系呢 ControlSource 属性可以指定与文本框相联系的数据源。将文本框与字段建立联系的方法是：

1 表单的数据环境中必须有表或视图，如果没有，请添加；
2 选择“表单控件”工具栏的“文本框”按钮；
3 在表单上合适的位置拖拽出一个矩形；

4 单击“属性”窗口的“全部”选项卡或“数据”选项卡；

5 选择 ControlSource 属性；

6 单击“属性”窗口选项卡上方的下拉列表，您将会看到在数据环境中添加的表字段清单；

7 选择与文本框相联系的字段。

三、设定输入字符的最大个数

文本框可以接受键盘上键入的全部字符。当光标停留在文本框内，只要有键盘输入，文本框就显示出字符。通过设置 MaxLength 属性，可以对键入字符的个数加以限定。

1 选择表单上的文本框，该文本框的周边出现 8 个点，表示被选中；

2 单击“属性”窗口的“全部”选项卡或“数据”选项卡；

3 选择 MaxLength 属性，该属笥的默认值为 0，表示无限制；

4 在位于“属性”窗口选项卡下方的文本框中输入最大字符数。

四、使文本框只读

如果只希望文本框显示字段数据，而不希望用户修改它，ReadOnly 属性可以帮您这个忙，通过它能够将文本框设置为只读型。

1 选择表单上的文本框；

2 单击“属性”窗口的“全部”选项卡或“数据”选项卡；

3 选择 ReadOnly 属性，该属性有两个值：.F. 假 读写 ，.T. 为真 只读 ，缺省值为 .F. ；

4 设定属性为 .T. 。

五、在状态条显示提示信息

在 Windows 环境下的软件均使用状态条，显示系统的提示信息。您是否希望在您的数据库应用系统中也给用户一些提示信息呢 例如当用户在输入某信息时，状态条中显示出您给他的提示信息。下面介绍如何在状态条中显示提示信息。

1 选择表单上的文本框；

2 选择“属性”窗口的“全部”选项卡或“布局”选项卡；

3 单击 StatuBarText 属性，该属性的默认值为无；

4 在“属性”窗口选项卡下方的文本框中输入希望在状态条中显示的字符。

5.2.3　编辑框

编辑框允许用户编辑长字段或备注字段文本，允许自动换行并能用方向键、PgUp 和 PgDn 键以及滚动条来浏览文本。

在表单中放入编辑框，Visual FoxPro 就在“Edit 1”中显示一个垂直滚动条 Edit 1 是编辑框的缺省标题 。实际上编辑框和文本框是一样的，只是用户输入信息的行数不同而已。文本框中含有一行文本，而编辑框可以包含多行文本，当需要较多行 例如输入一个备注性字段 时，就需要使用编辑框。编辑框和文本框的属性差不多，参照本书 5.2.2 节，就可以设置编辑框比较常用的属性。

5.2.4　列表框

列表框给用户提供一个可滚动的列表，在列表框中，可以有多项选择，但它不接受输入的文本。

在表单中放入列表框，就出现类似编辑框一样的控件，只是标题为“List 1”。列表框的使用与组合框类似，如何设置它的属性，请参阅下面组合框的介绍。如何用列表框显示字段数据信息，请参

阅实例的“初级文本列表组合框”。

5.2.5 组合框

组合框类似列表框和文本框的组合，使用它既可以在其中输入文本，也可以从列表中选择条目。在表单中放入组合框时，其标题显示为“Combol”。

设置组合框的 Style 属性为 2，它就变为下拉式列表框，此时您只能在列表中选择，不能进行编辑。但与列表框相比，只能看到一个项，单击下箭头按钮显示可滚动的下拉列表框。但列表框则是固定的，任何时候都能看到多个项。

下面我们将“成绩表 1”的字段内容显示在组合框中，并且可以让您任意选择或编辑。要使用组合框对字段内容进行显示、选择或编辑，首先要在表单中设置数据环境，然后再设置组合框的属性，具体步骤为：

1 将表与表单建立联系；

2 把组全框放入表单；

3 选择组合框；

4 打开“属性”窗口，选择“全部”选项卡或“数据”选项卡；

5 选择“ControlSource”属性，该属性指定与对象建立联系的数据源；

6 请在下拉式列表中选择字段“成绩 1. 姓名”；

7 选择“RowSource”属性，指定组合框控件中数据值的数据源；

8 请在“属性”窗口选项卡下方的文本框中输入字段名：“成绩 1. 姓名”；

9 选择“RowSourceType”属性，指定控件中数据值的数据源类型；

10 请在下位式列表中选择“6—字段”；

11 选择“全部”选项卡或“布局”选项卡单击“Style”属性，指定控件的样式；

该属性有两个值：

0—下拉组合框 此值可以修改数据 ，

2—下拉列表框只显示 此值不能修改数据 。

12 选择“下拉组合框”。

设置结果在实例的“初级文本列表组合框”表单中。

5.2.6 命令按钮

命令按生成我们常用的“确定”、“取消”和其他在 Windows 程序中出现的按钮，在“表单控件”工具栏中它以矩形按钮表示。

在表单中添加命令按钮后，就可以重新设置能影响命令按钮外观和功能的各种属性。Caption 属性建立按上出现的文本，通过字体、颜色、背景等可以改变该按钮的外观，您可以用与标签、文本框相同的方法设置它们。

一、代码窗口

命令按钮可以装饰表单，但是把命令按钮放入表单的真实目的是在表单上单击该按钮时产生某一操作。例如您的表单上有一个“退出”按钮，用来结束表单的操作。

如果要命令按钮做某件事，必须给按下达明确的指令，以便当您单击该命令按钮时，它能去执行命令。给按钮下达的指令是由 Visual FoxPro 程序能够识别的命令组成的。这些命令必须放入命令按钮的代码窗口中。“代码”窗口如图 5 - 9 所示。

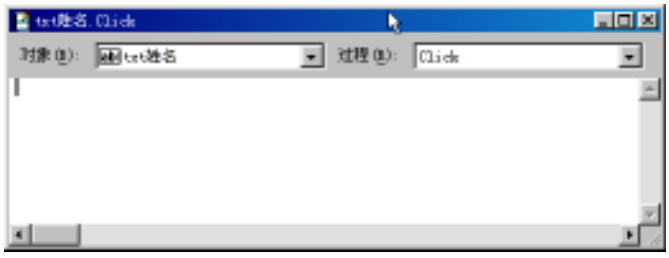


图 5 - 9　代码窗口

代码窗口的顶部有两个组合框，左边组合框的名称是“对象”，右边组合框的名称为“过程”。如果右边的单词是“Click”，您在代码窗口中书写的命令就变成 Click 过程。

对象指数据库、表、表单、控件、查询、报表等，表单上所有对象的名字都在代码窗口左边的列表中 图 5 - 10 。

过程是一组能被 Visual FoxPro 识别指令，在本例中过程的名称是 Click，该名称是和一个事件相联系的。

事件是在屏幕上显示表单、数据库、表和 Visual FoxPro 的任一部分所发生的事件。单击命令按钮是一个事件 Click ，右击命令按钮也是一个事件 RightClick ，滚动代码窗口右边的过程列表，可以看到其左边相应对象可能发生的所有事件 图 5 - 11 。

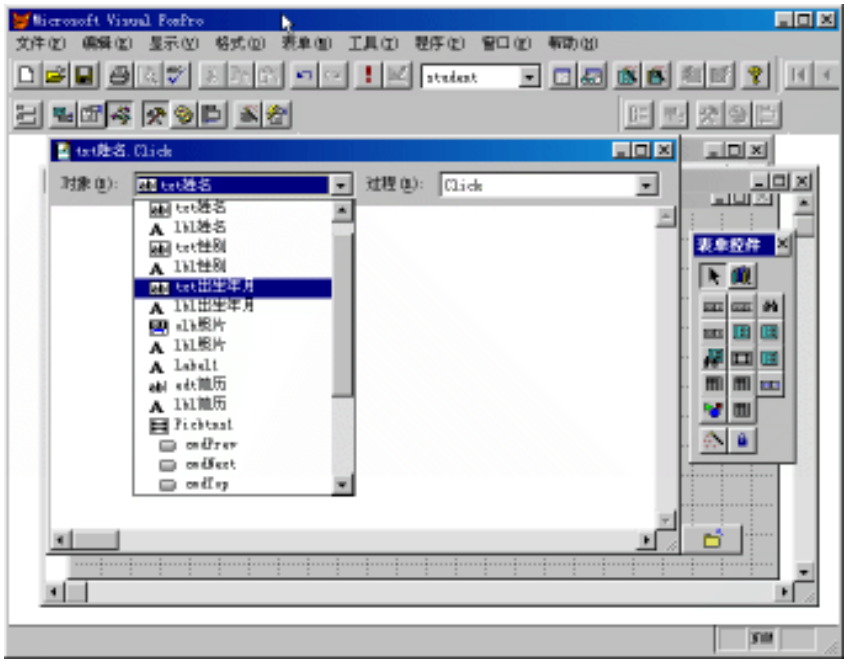


图 5 - 10　代码窗口的对象列表

二、代码的书写

下面我们在表单中添加一个“退出”按钮，使您的表单能够结束操作。

1 在表单中放入命令按钮；

2 设置按钮的 Caption 属性为“退出”；

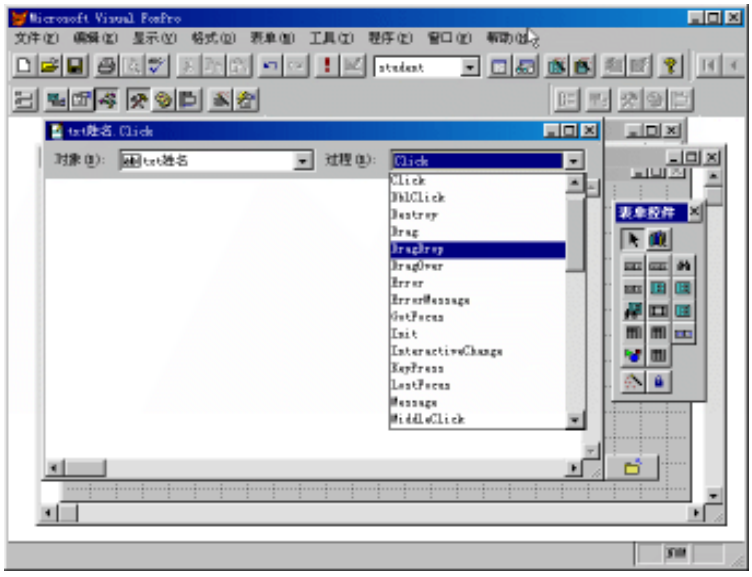


图 5 - 11　代码窗口的过程列表

3 设置字体名、字体号、颜色等属性；

4 双击该按钮，出现“代码”窗口；

5 在代码窗口中输入“ELEASE THISFORM”，该命令的含义是释放表单 输入命令时，右边的单词必须是“Click”，如果不是“Click”，请在右边的列表中选择它 ；

6 单击“代码”窗口右上角的“X”，退出命令窗口，返回表单。

三、“缺省”和“取消”属性

在 Windows 程序每次显示对话框时，对话框至少包括两个命令按钮“确定”和“取消”。“确定”按钮用来接受输入对话框的数据，“取消”按钮则用来取消输入操作，您可以不用鼠标而用键盘来选择它们 用回车键选择“确定”按钮，用 Esc 键选择“取消”按钮 。在命令按钮的“其他”选项卡中也有两个属性 :Default 缺省 和 Cance 取消 。如何使用它们呢

1 表单中放入两个命令按钮；

2 选择其中一个；

3 在“属性”窗口选择“全部”或“其他”选项卡；

4 设置“Default”为“. T. ”，被设置按钮周围会有暗色边界，按回车键会自动选择它；

5 选择另一个命令按钮；

6 在“属性”窗口选择“全部”或“其他”选项卡。

7 设置“Cancel”为“. T. ”，这样按 ESC 键时，会自动被选择。

设置结果在实例的“初级命令按钮”表单中。

四、用图形代替文本

我们还可以像 Windows 的常用工具栏一样，在命令按钮上用图标来表示某种操作。例如用一扇门代替“退出”二字。选择“布局”选项卡的“Picture”属性，就可以用图标代替文本出现在命令按钮中。

1 选择命令按钮；

2 在“属性”窗口选择“全部”或“布局”选项卡，单击“Picture”属性；

3 单击“属性”窗口选项卡下方文本框右边的“…”按钮，出现“打开”对话框，在对话框中选择图形；

4 单击“确定”按钮。

该图形的路径出现在选项卡下方文本框内。以后装载表单时，Visual FoxPro 6.0 自动到指定路径读取图形文件。

5.2.7　复选框

复选框指明一个选项是选定还是不选定，即“真”、“假”或“是”、“否”。选定时在框中出现复选标记。如果复选框与某字段相关联。则该字段也应是逻辑型，逻辑型字段有两个可能值：真和假。

5.2.8　选项按钮组

选项按钮组是包含选项的容器。通常，选项按组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，选项按钮组可以指定是将查询结果显示在屏幕上，还是打印在纸上。

关于容器的详细论述请参考本书第六章。

一、设置按钮数目

缺省的选项按钮组按钮个数是两个，但是我们可以按照需要，设置任意数量的按钮。该属性是“布局”选项卡的“ButtonCount”。

1 在表单中放入选项按钮，并选择它；

2 在“属性”窗口的“全部”或“布局”选项卡中选择“ButtonCount”；

3 在“属性”窗口选项卡下方的文本框中输入按钮数量，并单击“✓”。

表单中选项按钮组按照您输入的按钮数量发生了改变。如果控件的尺寸不足以显示按钮，请拖拽其边框，将尺寸放大。

二、选择控件

在选项按组中包含若干个独立的控件，这些控件与选项按钮组的关系可以从图 5-12“属性”窗口的列表中看出。表单 Form 包容选项按钮组 OptionGroup1，选项按钮组又包容选项按钮 Option1、Option2 等，如果您想修改某一个控件的属性，必须先选择它，然后才能在“属性”窗口中进行设置。选择单个控件的方法有两种。

第一种方法：在图 5-12 的“属性”窗口的列表中选择。

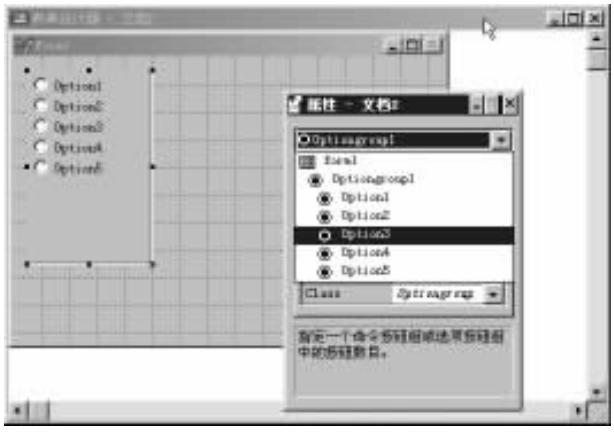


图 5-12　“属性”窗口的列表

第二种方法：

1 在选项按钮组上单击右键，出现图 5 - 13 的快捷菜单；

图 5 - 13 快捷菜单

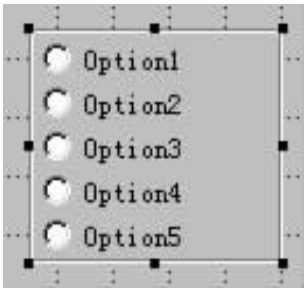


图 5 - 14 被选择的容器控件

2 选择“编辑”，容器选项按钮组四周的边框变宽；

3 单击容器内的控件，被选控件四周出现八个点 图 5 - 14 ，表示被选中。

控件选中后您就可以修改其属性了。

程序运行时，选项按组的 Value 属性值记录着当前的选择序号。

5.2.9 表格

如果要在表单中显示多列数据，请使用表格控件。用表格显示数据。其效果与浏览是相似的。

表格是一个容器对象，表格可以包含列。这些列除了包含标头的控件外，每一个列还拥有自己的一组属性、事件和方法程序，从而为表格单元提供了大量的控件。

在“表单控件”工具栏选择表格控件，单击表单就会创建一个表格。创建表格后的第一个事是设置表格列数目 缺省的表格列值是 1 。

一、设置表格列数目

1 选择表格；

2 在“属性”窗口的“全部”或“布局”选项卡中选择“ColumnCount”；

3 在“属性”窗口选项卡下方的文本框中输入表格列数目；

4 单击“√”。

图 5 - 15 是表格在表单及表单“属性”窗口的显示，从中可以看出：表格是一个容器，在这个容器中可以容纳多列，列用 Column 表示。Column 后面的数字是列序号，每列又由两部分组成：Header 和 text。Header 可以放置标题，例如表字段名，Text 放置内容如表记录。

二、填写表格列标题

表格列标题的缺省文本是“Header”修改 Header 的 Caption 属性，就可以改变表格列标题。选择“Alignment”属性还可以设置标题是否居中显示。当然，象标签和文本框一样，您还可以改变表格列标题的字体名、字体号和颜色等属性。

三、建立表与表格的关联

若要用表格显示表，必须在表与表格之间建立关联。但前提是已经在数据环境中建立了表与表单的关联 参照 5.1.2 ，并且表单中已经创建了表格。可以用一个表格显示多个表的数据，因此建立表与表格的关联确切地说应该是表与表格列的关联，其步骤为：

图 5 - 15　表格控件

1 在图 5 - 15 的“属性”窗口的列表中选择“Text”,您希望字段数据出现在第几列就选择第几个 Column 的 Text；

2 在“全部”或“数据”选项卡中选择“ControalSource”属性；

3 单击“属性”窗口选项卡下方文本框右边的“…”按钮,出现表或视图字段名；

4 选择需要的字段名；

5 重复 1 — 4 步,设置其他列的数据源。

四、变动列顺序

如果想改变表格列数据显示的顺序,不必改变标题和文本的数据源,直接拖拽表格列即可,方法为：

1 在表格上单击右键,出列快捷菜单；

2 选择“编辑”,表格四周的边框变宽；

3 用鼠标按住 Header 栏,拖拽到指定的地方。

关于表格的应用,请参考实例“初级表格与形状”表单。

五、改变列宽

如果表格列的宽度不合适,可以改变它,方法为：

1 在表格上单击右键,出现快捷菜单；

2 选择“编辑”,表格四周出现蓝绿色边框；

3 将鼠标放在列标题与标题交界处的竖线处,鼠标指针变为双箭头；

4 按住左键,拖拽竖线到合适宽度。

六、设置数据为只读

有时我们可能只希望显示表格中的数据,而不允许用户修改它。只在您将 Text 的 ReadOnly 属性设为 . T. 即可。

1 在图 5 - 15 的“属性”窗口的列表中选择相应的 Text；

2 在“全部”或“数据”选项卡中选择“ReadOnly”属性；

3 双击“ReadOnly”,使其值为 . T. 。

5.2.10 微调控件

在微调控件中单击向上按钮，可减少微调控件值，单击向下按钮，可增加微调控件值，用户也可以直接在微调框中键入数值。

在“表单控件”工具栏中表示为上下箭头的控件就是微调控件，它在表单上显示的标题是“Spinner”。微调控件的最大输入值是 2147483647，最小输入值是 - 2147483647，我们可以根据需要设定它。比如要用一个微调控件表示分钟，此时就应该设置最大值为 60，最小值为 1。设置输入值的方法为：

将 KeyboardHighValue 键入值 和 SpoinnerHighValue 单击向上按钮的值 属性设置为 60。

将 KeyboardLowValue 键入值 和 SpinnerLowValue 单击向下按钮的值 属性设置为 1。

运行表单时，如果输入值大于最大值或小于最小值，系统会提示“范围 :60 到 1”。

5.2.11 页框

选项卡在 Windows 应用程序中是很常见的，比如 Visual FoxPro 的“项目管理器”和“属性”工具栏就属于选项卡。使用页框可创建带选项卡的表单，在表单上一个页框可以有多个页面，页框定义了页面的位置和页面的数目。

页框是包含页面的容器对象有，页面又可包含控件。我们可以在页框、页面或控件级上设置属性。

下面建立一个页框，设置四个页面，分别放置四个成绩视图。

1 新建一个表单；

2 将这四个视图放入数据环境中；

3 在“表单控件”工具栏中选择“页框”按钮，并在“表单”窗口拖动成想要的尺寸。此时页框在表单中的标题为 PageFramel，您还会看到 Page1 Page2 文本，那页面 1 和页面 2 缺省页面是 2 个 ；

4 指定页框中包含的页面数，设置 PageCount 属性为 4 属性在布局选项卡中 ，您会看到，此时页框中增加了标题为 Page3 和 Page4 的页面；

5 激活页框，在页框上单击右键，在快捷菜单中选择“编辑”命令；

6 更改页面标题，分别修改 PageX 的 Caption 属性 X 可为 1，2，3，4 ；

7 分别将成绩视图放入页面；

8 在代码窗口 PageX 的 Click 事件分别添加“SELE 成绩视图 X” X 为 1，2，3，4 ；

9 添加说明表单功能的标签和退出的命令按钮。

运行结果如图 5 - 16 实例的“初级页框”表单

将控件添加到页面上必须先使页面成为活动的，然后再将控制件添加到面。要想使页面活动，可以单击鼠标右键，将页面作为容器激活，然后选择“编辑”，再选择要使用的页面选项卡。或者在“属性”窗口的“对象”框中选择这一页面。

和其他容器控件一样，必须选择页面，并从用鼠标右键弹出的快捷菜单中选择“编辑命令，或在“属性”窗口的“对象”下拉列表中选择容器。这样，才能选择这个页面 具有宽边 ，再朝正设计的页面中添加控件。在添加控件前，如果没有将页面作为容器激活，控件将添加到表单中而不是页面中，即使看上去好像是在页面中。

图 5 - 16　页框练习

5.2.12　线条和形状

在表单的适当位置加入线条，可以丰富表单的外观。单击“表单控件”中的“线条”按钮，就可以在表单中添加线条，还可以使用“形状”工具为表单添加线条，因为“形状”工具有 3 维效果的属性，可以使线条更加生动 参考实例“初级表格与形状”表单 。使用形状工具给表单加线条的方法为：

1 在“表单控件”工具栏形状控件；

2 在表单中拖拽出一条直线；

3 在“属性”窗口的“全部”或“布局”选项卡中选择“SpeciaEffct”；

4 在“属性”窗口选项卡下方的下拉式列表选择“0 - 3 维”。

改变形状控件的“Curvature”属性的数值，可以获得正方形或正圆，您还可以使用边界样式 BorderStyle 、边界宽度 BorderWidth 、填充样式 FillStyle 、填充颜色 FillColour 等属性，对形状进行进一步设置。

5.3　给表单添加对象

5.3.1　将视图放入表单

可以将表、视图、控件等对象放到表单中，制作出功能各异的表单。下面的练习是将在第三章中建立的成绩视图放到表单中，我们要将四张成绩视图放在一个表单中，并且命令按钮选择它们的显示。方法是：

1 将视图放入数据环境；

2 将数据环境中的视图名称拖拽到表单中，该视图成为一个表格；

3 调整表格在表单中的大小及位置；

4 在“属性”窗口将表格列标题居中；

5 重复 2 到 5 步，将其他三个视图放入表单；

6 在“属性”窗口,将后三个表格的“Visible”属性设置为 .F. 对象的 Visible 为“T”,是可见的,若其 Visible 为“F”则不可见。我们先将它隐藏起来,因为在这个表单中要放置四个表格 ;

7 在表单中放置四个命令按钮,这四个按钮的 Caption 分别显示四个视图的名称;

8 在“代码”窗口分别给四个按钮键入命令,使一个表格可见,而其他表格隐藏;例如命令按钮 1:

ThisForm.grd 成绩视图 1.VISIBLE = .T.

ThisForm.grd 成绩视图 2.VISIBLE = .F.

ThisForm.grd 成绩视图 3.VISIBLE = .F.

ThisForm.grd 成绩视图 4.VISIBLE = .F.

9 在表单中放置一个结束操作的命令按钮。

结果在实例的“初级成绩视图”表单中。

5.3.2 改变对象的名字

我们在“成绩视图”表单中一共放了 5 个命令按钮,这些按在“代码”窗口的“对象”列表和“属性”窗口的列表中分别被显示为“Commandl”、“Command2”、……“Command5”等,如果对象再多些,将不容易分清楚哪个按钮对应什么操作通过改变对象的名字 Name ,可以解决这个问题。

1 选择对象;

2 选择“属性”窗口的“全部”或“其他”选项卡;

3 单击“Name”属性;

4 在“属性”窗口选项卡下方的文本框输入对象的名字。

您可以将控件的 Caption 属性和 Name 属性设置为同样的文本,使它们一致,有利于填写代码和调试程序时的操作。

5.3.3 将报表放入表单

在第四章中,我们曾经建立了三张报表,下面把它们也放入表单。

1 在表单中放置了三个命令按钮,使这三个按钮的 Caption 分别标明三张报表;

2 在“代码”窗口分别给三个命令按钮键入命令,启动其中的一个报表。例命令按钮 1 为:“REPORT FORM 成绩表 1 PREVIEW”;

3 在表单中放置一个结束操作的命令按钮。

结果在实例的“初级成绩报表”表单中。

5.3.4 放入移动记录指针的命令按钮

5.1.2 节中的“基本情况 1”表单是使用向导控件对记录进行操作,我们还可以自己填写命令代码移动记录指针。下面建立一个“基本情况 2”表单,数据环境、控件等已经放置妥当,只需在“代码”窗口填写代码。

一、记录指针移到和第一条记录

在表单启动时,就应该让表指针定位在第一条记录,

REFRESH 过程用于刷新屏幕,CLICK 是当您单击按钮时要处理的过程。

在 CLICK 过程中填写:

```
GO TOP                          &&指针到第一条记录
```

THISFORM. REFRESH　　&&刷新表单

在 REFRESH 过程中填写：

IF BOF　　&&如果记录号为 1 或已经到了文件头

THIS. ENABLED = . F.　　&&使该控件失效

ELSE　　&&否则

THIS. ENABLED = . T.　　&&控件有效

ENDIF

ENABLED = . F. 称为控件失效。当控件失效时，不接受用户引发的事件，比如您单击该按钮，它不引发 CLICK 事件。

二、记录指针移到上一条记录

在 CLICK 过程中填写：

SKIP - 1　　&&指针向上移动一条记录

THISFORM. REFRESH　　&&刷新表单

在 REFRESH 过程中填写：

IF BOF　　&&如果记录号等于 1 或已经到了文件头

THIS. ENABLED = . F.　　&&使该控件失效

ELSE　　&&否则

THIS. ENABLED = . T.　　&&控件有效

ENDIF

三、记录指针移到下一条记录

在 CLICK 过程中填写：

SKIP　　&&指针向下移动一条记录

THISFORM. REFRESH　　&&刷新表单

在 REFRESH 过程中填写：

IF EOF　　&&如果当前记录号等于文件尾

THIS. ENABLED = . F.　　&&使该控件失效

ELSE　　&&否则

RHIS. ENABLED = . T.　　&&该控件有效

ENDIF

四、记录批针移到最后一条记录

在 CLICK 过程中填写：

GO BOTTOM　　&&批针移到最后一条记录

THISFORM. REFRESH　　&&刷新表单

在 REFRESH 过程中填写：

IF DOF　　&&如果批针到了文件尾

THIS. ENABLED = . F.　　&&时该控件失效

ELSE　　&&否则

THIS. ENABLED = . T.　　&&该控件有效

ENDIF

5.3.5 设置访问键

访问键能在表单中的任何地方通过按 Alt 和访问键来选择一个控件，例如“编辑”操作可以用 Alt + E。

设置访问键的方法是在控件的 Caption 属性中，在想作为访问键的字母前键入一个反斜杠和一个小于符号 h < 。

下面是对“编辑”命令按钮的 Caption 属性的设置，将 E 作为它的访问键。

h <Edit

我们在表单中任何地方按 Alt + E，均可以选择这个命令按钮。

5.3.6 设置控件的 Tab 次序

在 Windows 程序中可以用 Tab 键在控件中移动，Visual FoxPro 的“Tab 键次序”也能提供这个功能。表单控件的默认 Tab 键次序是控件添加到表单时的次序。您也可以通过设置改变默认的控件 Tab 键次序。

一、显示 Tab 键次序

选择菜单 :显示 ◊ Tab 键次序。

二、改变控件的 Tab 键次序

图 5 - 17 TAB 键次序

1 选择菜单 :显示 ◊ Tab 键次；

2 双击控件旁边的框，这个控件将在表单打开时具有最初焦点 图 5 - 17 ；

3 按需要的 Tab 键次序依次单击框；

4 单击控件外的任何地方，完成设置；

5.4 表单属性

前面我们谈了控件的许多属性，其实表单也有属性。表单的缺省名字是 From，您可以改变它的名字 Name 、标题 Caption 、位置 Left、Top 和大小 Height、Width 等。

5.4.1　将图片作背景

我们可以通过 Picture 属性，给表单添加一个图片，使它美化。

1 单击表单，“属性”窗口下拉式列表指向“From1”；

2 选择“全部”或“布局”选项卡；

3 单击“Picture”；

4 在“属性”窗口选项卡下方文本框右边单击“…”按钮，打开对话框；

5 在对话框中选择图片文件的格式和文件名；

6 单击“确定”按钮。

当您运行表单时，您选择的图片就会代替灰色的表单底色。

5.4.2　给表单赋予图标

当表单最小化后，Visual FoxPro 就在屏幕下方以图标形式显示它，狐狸头是缺省的图标，通过 Icon 属性将它改变为您喜欢的样子。给表单赋予图标和添加图片的方法基本相同，不同的是图标的属性是：Icon，图标文件的扩展名是 . Ico。

5.4.3　使边框固定

Windows 程序的窗口界面化大多数是允许用户改变大小的，Visual FoxPro 的表单同样给用户提供这一功能，但也许在某些场合。您希望将表单的大小固定，改变 BorderStyle 边框样式 为“2—固定样式”就可以达到此目的 默认值 3—可调边框 。

5.4.4　最大化、最小化的和关闭按钮

Windows 程序的窗口右上角有最大化按钮、最小化按钮和关闭按钮，如果您希望当表单运行时，不出现它们，可以通过改变属性 MaxButton、MinButton 和 Closable 来设置它们。

5.4.5　表单的显示属性

表单可以显示在系统桌面上、窗口中、顶层表单中，这就是表单的显示属性。通过表单的 Show Widnow 属性可以用来设置表单的显示属性。

在默认状态下，表单是显示在窗口中的。即该表单显示在 Visual FoxPro 的系统窗口或自定义窗口中。最显著的特征是当窗口最小化时，该表单随之从屏幕上消失。

为了使表单脱离窗口的束缚，可以设置表单的显示属性为顶层表单，顶层表单可以脱离窗口，直接显示在系统桌面上。表单还可以显示在顶层表单中，这样，我们就可以建立一个抛开窗口，完全用表单构成的项目。

第 6 章　类

6.1　Visual FoxPro 中的类

类的特性和优点：

1. **代码封装，隐藏复杂性；**
2. **继承性，减少重复开发；**
3. **独立性，便于修改、维护和升级。**

在 Visual FoxPro 中提供了许多的基本类供大家使用。现在您可以通过使用这些类来生成自己的类库，使编程工作变得简化多了。在 Visual FoxPro 中，您的类必须从基本类派生，当然基本类已经足够丰富，可以满足您的任何需要。下面类的分层结构会对您在制作自己的类时有所帮助，如图 6 - 1 所示。

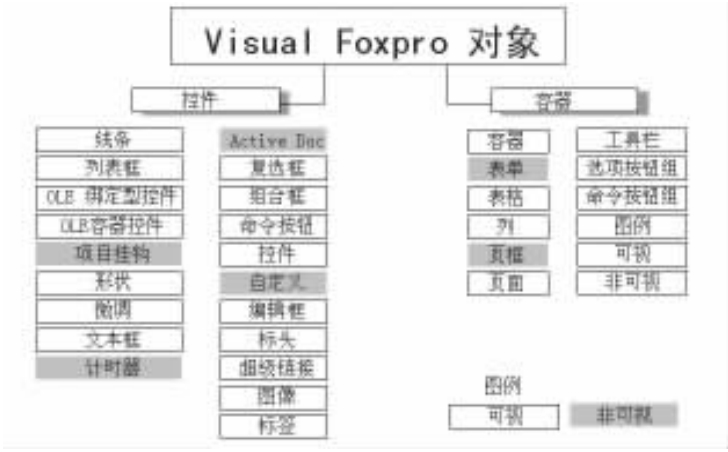


图 6 - 1　Visual FoxPro 类的分层结构

从图 6 - 1 中可以看出，基本类分为控件和容器两种。控件就是一些进具有某种功能的元件，控件是类的基本单位。容器就是用来存放其他控件的一种特殊的控件，您可以将任何控件放到容器中。容器的出现是为了使控件更容易管理和使用，如您可以将列表框、命令按钮等放在一个容器中，统一管理。容器也是一个类，有自己的属性和事件。容器中的控件都是它的子类。在调用容器中的子类时必须指明容器的名称。如您要调用表单上的某个容器 MTRQ 中的一个按钮 COMDI 的 CLICK 事件，应该在程序中使用："THISFORM. MYRQ. COMD1. CLICK0"这样的语句。

图 6 - 1 中还标明了各个基本类的可视与不可视。所谓可视就是当您把该控件放到表单上，运行该表单时能看到该类。不可视则正相反。不可视类在编程中有很重要的地位，通常将一些通用的函数模块放到不可视类中。可视类一般用来处理与用户的界面、取得用户的输入、将信息显示给用户等等，不可视类则是用来存放完成实际功能的代码模块。

类的优点虽然很多，但是不通过实际使用，您是体会不到它的好处的。所以最重要的是去使用

它。在下一节中我们将介绍如何在 Visual FoxPro 中构造自己的类和类库。

6.2　构造自己的类库

6.2.1　类的构造举例

下面以一个很简单的例子说明构造一个类的过程。

假设我们要做一个表单上用的返回按钮。

1 在菜单中选择"文件→新建→类。

2 单击"新建文件"出现新建类对话框如图 6-2 所示。

图 6-2　新建类对话框

"类名"就是您自己的类的名字,在这里因为要做返回按钮,我们添入"CMDRETURN"。

"派生于"后的下拉列表框表示您的类是从哪个类派生的。在这个下拉列表框中显示的是 Visual FoxPro 的基本类,您也可以通过后面的"…"按钮选择其他的类库中的类进行派生。

"存储于"表示您要新建的类所在的类库名,您可以随意添入一个名字以创建一个新的类库或者通过后面的"…"按钮选择一个已经存在的类库。

3 单击"确定"按钮,您就进入了类设计器,如图 6-3 所示。

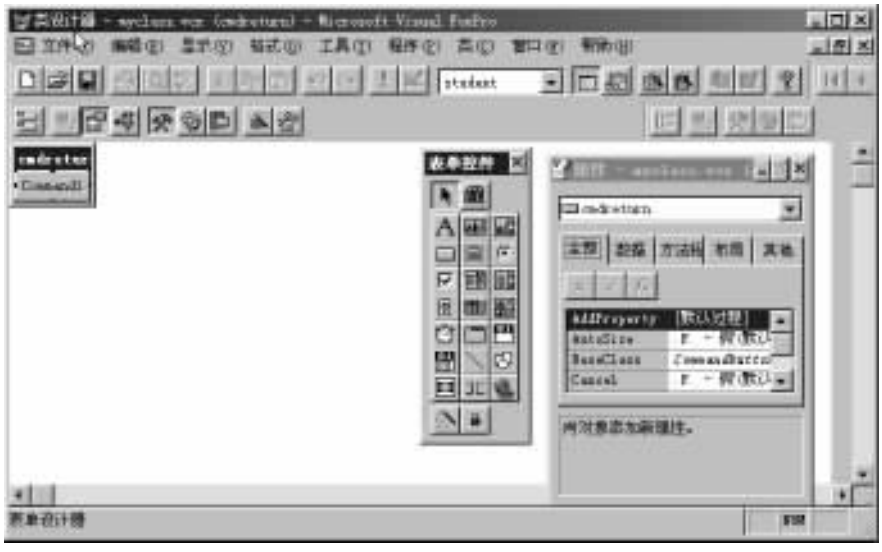


图 6-3　类设计器

4 将 CAPTION 属性修改为“返回”。

5 双击该按钮,在它的 CLICK 事件中添加如下代码:

```
YN = MESSAGEBOX “您要返回吗 ”,4 + 32,“系统提示”
IF YN = 6    &&用户选择“是”
RELEASE THISFORM
ENDIF
```

6 存盘后退出类设计器。

这时我们已经建立了一个只包含一个类的很简单的类库。您可能发现类设计器中和表单设计器中的环境有很多的相似之处,这使得我们很容易在熟悉的环境中建立自己的类。

现在来看看我们的成果。

7 新建一个表单。

图 6 - 4 添加自己的类库

8 在“表单控件”工具栏中选择“添加”。

如图 6 - 4 所示。

9 在“打开”对话框中找到我们刚才建立的类库,把它添加进来。这时,在控件工具栏中看到的就应该是我们刚才做的按钮控件了。

10 把这个按钮拖到界面上,您会看到一个标题为“返回”的按钮出现在表单上。双击它可以发现在它的 CLICK 事件中没有任何代码,这是因为代码被封装到父类中,对用户是透明的。

11 保存并运行该表单。

单击我们添加的按钮类,弹出了确认对话框,单击“是”后便退出了该表单。以后在每个界面中只要把这个按钮拖上去便实现了我们在类中实现的功能,是不是很方便

其实,这个类制作并不完善,使用缓冲区时,如果用户没有保存在缓冲状态下的操作,将使在缓冲区中所做的修改丢失。所以在“返回”按钮中,还应该检测用户是否修改了记录,并提示给用户。这样,我们就必须修改 CLICK 事件中的代码使它更完善,修改后的代码如下:

```
*************************************************
*程序名 :CLICK
*作    用 :返回代码,可以检测用户是否有没有保存的修改,并提示给用户
*
*作者 :王杰
```

```
*2000/04/08
**************************************
  CHR 7
IF THISFORM.FLAG=1
    GO TOP
    LMODI=.F.                    &&修改标志,为..T. 时有修改
    DO WHILE.T.                  &&循环检测
        IF EOF
            EXIT
        ENDIF
        FOR FILNUM=1 TO FCOUNT   &&循环检测所有字段
            IF GEIFLDSTATE FILNUM =2
*调用函数 GETFLDSTATE 检测状态,为 2 表示有修改
                LMODI=.T.
                EXIT
                ENDIF
            ENDFOR
            IF LMODI
                EXIT
            ENDIF
            SKIP
            LOOP
ENDDO      &&循环结束
TUCHU=.F.                    &&退出标志,由用户选择后确定
IF LMODI                     &&如有修改提示给用户,并让用户选择是否保存和退出
   RESULT=MESSAGEBOX "记录已修改,保存并返回吗 ",3+32,"信息窗口"
   DO CASE
      CASE RESULT=6          &&用户选择是
          TUCHU=.T.
          THISFORM.STUAPPEIT1.CMDSAVE1.CLICK
      CASE RESULT=7          &&用户选择否
          TUCHU=.T.
      CASE RESULT=2          &&用户选择取消
          TUCHU=.F.
  ENDXASE
ELSE
  RESULT=MESSAGEBOX "您要返回吗 ",4+32,"信息窗口"
  IF RESULT=6
      TUCHU=.T.
  ENDIF
```

```
ENDIF
IF TUCHU
    THISFORM.RELEASE
ELSE
    GO TOP
    THISFORM.REFRESH
ENDIF
    ELSE
  RESULT = MESSAGEBOX "您要返回吗 ",4 + 32,"信息窗口"
  IF RESULT = 6
    THISFORM.RELEASE
ENDIF
    ENDIF
```

上面的代码是我们在项目制作中，实际使用过的一个类。可以看到，这段检测代码并不很简单，如果不使用类，那在每一个表单的“返回”按钮中都要写一遍这些代码，加大了工作量。从这里我们能看出类的使用确实减少的代码的长度，加快了开发项目的速度。

6.2.2 在项目中类库的设计

在数据库项目的开发过程中，类的使用非常频繁。那么是否需要建立自己的类 应该构造哪些类 如何设计类库呢

对于第一个问题我们的答案是肯定的，当然需要！为了界面的统一，为了将来修改的容易，为了便于调整试，为了减少开发时间，为了……，有这么多的好处，为什么不用类呢

那么，我们需要构造哪些类呢 首先，为了程序有一个统一的外观和风格，我们应该对在界面上可能会使用的每一个类进行派生，如标签、文本框、表单等等。在派生的类中，我们通过使用统一的字体、色彩、图标等来达到整个软件系统界面的统一。其次，是对通用功能的封装。例如，允许用户在表中移动指针的命令按钮、关闭表单的按钮以及帮助按钮等。都可以保存为类，并在需要表单具有这些功能时，把它们添加到表单中。

我们知道，类的存储在类库中的，那么如何设计自己的类库呢 下面将我们在实际使用中总结出的一些经验作个简单介绍，供读者参考。

首先，构造一个基本库，在这个库中对 Visual FoxPro 中的大多数基本类进行派生，至少应该包含常用的控件和表单。在这些派生类中，一般不做代码，只是修改它们的字体、大小等有关界面外观，在派生的表单类中，应该修改它的背景、图标等属性。然后在菜单 :工具→选项对话框中“表单”选项卡的“模板类”中，使默认的“表单”类指向自己的类，这样当新建表单时就会自动调用您自己设计的类。见图 6 - 5。

其次，针对实际使用需要，对通用的代码进行封装，建立一个应用库，这个库是实际要使用的类，是对基本类的进一步细化，如将标签分为用于标题和字段名的等等，分别修改它们的有关属性，使它们成为更接近实际使用中的类，这些类最好从刚才建立的基本库中派生。如果用于界面，用户可以看得到的，一定要从基本库中派生，这样保证了界面的风格一致。可视类可以直接从 Visual FoxPro 的基本库中派生。如果对于某些通用代码您找不到合适的父类，如用于与远程数据库连接的类，可以使用 Visual FoxPro 的 Custom 类，它是一个非可视类，用来提供给用户建立自己的专用类。

在这一步中,还可以对一些有特殊用途的类单独建立库,如对记录定位按钮专门建立一个类库。

图 6 - 5　设置表单模板

最后是对应用库的进一步集成和深化。如上面提到的记录定位按钮，应用中一定是一起使用它们,而不会只使用“前一个”却不使用“后一个”。所以,我们把这些集成起来,改进代码,统一控制。

为什么是改进代码呢　比如在每一个按中,为了判断是否到了记录头或尾进行了许多检测,这其中有许多重复,会影响系统的速度,现在集中起来统一检测,当然是改进了代码。下面就是一个实际使用中的例子。将记录定位按钮放到一个容器中统一管理。给该容器建立三个自定义属性，其中“ISBOF”和“ISEOF”来表示表的状态,“ISBROWS”用来表示当前的操作状态。再建立一自定义函数“GETTABLESTATE”来取得当前表的状态,然后在 REFRESH 事件中完成按钮的设置。原代码如下：

```
*********************************************
* 程序名 :GETTABLESTATE
* 作　用 :取得当前表的记录状态
*
*                          作者 :王杰
*                           2000 /04 /08
*********************************************
IF  BOF
   THIS. ISBOF = . F.
ELSE
```

```
    SKIP - 1
    IF BOF
    THIS. ISBOF = . F.
      ELSE
     THIS. ISBOF = . T.
     SKIP
    ENDIF
   ENDIF

  IF BOF
   THIS. ISEOF = . F.
  ELSE
   SKIP
   IF EOF
    THIS. ISEOF = . F.
   ELSE
   THIS. ISEOF = . T.
  ENDIF
    SKIP - 1
 ENDIF
  RETURN

 ********************************************
 * 程序名 :REFRESH
 * 作   用 :刷新函数
 *
 *                                  作者 :王杰
 *                                   2000 /04 /08
 ********************************************
 IF thisform. opcode = 2
       this. ISBROWS = . F.                  &&非浏览态
 ELSE
       this. ISBROWS = . T.
       this. GETTABLESTATE
 ENDIF
 * * * * * * 开始设置各按钮状态 * * * * * *
 this. CMDPRE1. ENABLED = THIS. ISBROWS. AND. THIS. ISBOF
    * 浏览状态并和记录不在第一条时,按钮"上一个"可用
 this. CMDTOP1. ENABLED = THIS. ISBROWS. AND. THIS. ISBOF
    * 浏览状态并且记录不在第一条时,按钮"第一个"可用
```

```
this. CMDBOTTOM1. ENABLED = THIS. ISBROWS. AND. THIS. ISEOF
    * 浏览状态并且记录不在最后一条时,按钮"最后一个"可用
this. CMDNEXT1. ENABLED = THIS. ISBROWS. AND. THIS. ISEOF
    * 浏览状态并且记录不在最后一条时,按钮"下一个"可用
```

6.3　组件管理器和类浏览器

Visual FoxPro 提供了两个用来管理类的工具：组件管理器和类浏览器。其中，组件管理器是 Visual FoxPro 6.0 中的新工具。在本节，大概介绍一下这两种工具的使用方法，您可以通过工具菜单中的组件管理器和类游览器子菜单来打开它们。

"类浏览器"是用来显示类库或表单中的类，也能够显示 .tlb、.olb 或 .exe 文件中的类型库信息。可用"类浏览器"显示类库或表单中的表，以及查看、使用和管理类及其用户定义成员，如图 6-6 所示。

图 6-6　类浏览器

"类浏览器"的使用很简单，我们不做详细的介绍，读者可以通过 Visual FoxPro 的帮助 CMSDN 得到详细的使用方法。

"组件管理库"是软件对象　例如类库、表单、按钮等　类型的容器，也包含新的 Visual FoxPro 类，如图 6-7 所示。

图 6-7　组件管理器

“组件管理库”有很多的用处，您可以使用它将组件分成对象、项目、应用程序或其他类。这种可视分组方式可进行动态定制，这样可以使用、复制或重新组织“组件管理库”中几种不同分类的组件。在“组件管理库”的任何地方，只要放置了对指定数据项的引用，都可以对其进行访问。对于单独的项目，在不同的目录或文件夹中可以用多个引用。例如，一个按钮可能会出现在一个或几个“组件管理库”的项目目录中 用文件夹代表 ，也可能在“工具”目录中出现，因为在“工具”目录中包含了所使用的所有按钮。

您可以使用“组件管理库”管理由不同的“项目管理器”、“类浏览器”和表单控件工具栏提供的所有功能。在指定的项目文件或类库的环境中，其他每个 Visual FoxPro 组件对项目或类提供了特定的功能。使用 “组件管理库” 可以在一个抽象的设计水平上或在一个有明确的开发目标的水平上，管理各组件间的关系及组件的许多行为。

可以在“组件管理库”中拖放组件，也可以从“组件管理库”中将组件拖放到项目或表单。如图 6 - 8 所示。在“组件管理库”中还可以更改对象或类的属性。

图 6 - 8 从组件管理器向表单添加控件

总之，“组件管理库”的功能非常强大，它可以包含任何 Visual FoxPro 元素，如本地或远程的文档、文件或文件夹、Microsoft Excel 和 Word 等自动服务程序 Automation Server 、HTML 位置和文件，也包括含有代码片段、类向导或生成器的 . prg 文件使用好这两个非常有用的工具会使您事半功倍。

第 7 章　程序

7.1　VFP 程序设计概述

随着面向对象编程技术的发展，程序设计语言也变得越来越高级。VFP 经历了从 DBASE、FoxBASE、FoxPro 到现在的 VisualFoxPro，编程的方式和思路也发生了很大的改变。在 VFP 的程序设计中将过程化程序设计与面向对象程序设计结合在一起，帮助用户创建出功能强大、灵活多变的应用程序。从概念上讲，程序设计就是为了完成某一具体任务而编写一系列指令；从深层次来看，VFP 程序设计涉及到对存储数据的操作。

也许您曾经在 DOS 下编写过一些程序，但随着 Windows 的广泛使用，您必须适应另一种编程思路，即事件驱动。这个看似高深的话题其实很简单，就是把主动权交给用户，让用户能控制程序的流程。在 Windows 中，用户的任何动作都是一个事件，如移动鼠标、按下一下键等，都会触发相应的处理程序。我们要做的就是为这些事件编写相应的代码来完成用户要求。当然，我们不必对任何事件都编写代码，只需对我们要用到的事件编写代码。没有编写代码的事件 Windows 会自己调用缺省的过程进行处理。

在本章中，我们不会浪费过多的篇幅去解释每条命令的用法，也不会像每一本编程书那样详细讨论会支、循环等命令的用法。我们认为，从更高一点的角度来看，无论哪种编程语言，最基本的语法元素都是相同的，每种语言都有分支、循环等语句，只是用不同的关键字来实现罢了。所以，过多的讨论这些基本的语法是没有必要的，正所谓“殊途同归”。如果您已经熟悉某种语言，只要看看 VFP 中用什么命令实现就可以了，其他的基本上都是一样的。那么，我们在本章中还有什么要讲的吗 当然有。即使您懂得了所有的命令，也不一定能作出好的程序，即使您能编写一些程序，也不一定能成为一个优秀的程序员。为什么有的程序员写的代码很乱，功能也很差又难于理解，而另一些程序员写的程序却清晰易读、功能强大呢 这就是好程序员与差程序员的区别。那么，怎样才能作出一个好的程序呢 有哪些技巧呢 怎样才能成为一个优秀的程序员呢 这就是本章着重要讲的东西。

如果您确实需要了解关于语言方面的知识，请参考联机帮助 MSDN，那里有您需要的所有关于 VFP 语言方面的知识。

7.2　良好的编程习惯

从小我们就被学校或家长强迫要养成一些良好的习惯，如早睡早起、好好学习、遵守纪律等等。我们似乎对习惯这个词都有些听腻了。但仔细想想这些习惯虽然使我们变得循规蹈矩，但还是很有些好处的。编程也是如此，总有些使人很厌烦但很有好处的习惯不得不去遵守。在我初学编程时，总是按自己的习惯来写程序。但是，当程序的规模越来越大时，老师教的规矩显示出它的好处来了。请大家记住，好的习惯和规矩虽然看似带来一些不必要的麻烦，其实，从长远看，它会给我们将来减少更多的麻烦。下面让我们来看一看，编程中的良好习惯。

7.2.1 程序头

在编写代码时，最基本的是格式，要求每段程序都有程序头，其中应该注明程序名称、作者、时间、程序功能等信息。程序头使得程序显得美观，同时也给日后修改时带来一些方便，至少不用读代码来确定它的功能，当多人开发时，还可以直接找到是谁编写的，让作者来修改当然比别人改好得多。

7.2.2 注释

在程序编写过程中还要注意的是要添加注释。有些人认为注释是没有必要的，自己写的程序，修改时一看就知道是什么意思。其实不然，即使是自己做的程序，经过一段时间后，也许连自己也想不出当初的用意来了。在写程序时，经过反复考虑或是很有技巧的地方一定要加上注释。其实这是给自己将来修改时提供方便。

7.2.3 美观

写代码时要注意美观，该缩进的地方要缩进，使程序看起来错落有致，也使程序的结构更加清晰，有利于查错和理顺思路。如果您懒得用空格一个个的缩进、对齐，VFP 的编程环境中提供了自动修饰的功能。在编程时通过鼠标右键弹出的菜单中选择“修饰”，里面的设置很简单，然后运行就行了，怎么样，效果不错了吧

7.2.4 保护程序的运行环境

在做程序时还应注意的是程序运行环境的保护。

光讲理论显得很枯燥，让我们从 VFP 编程的实际应用来看一看。在 VFP 中有很多环境变量，如系统默认目录、搜索目录、打开方式、字符比较方式等等许多设置。VFP 提供了一个名为 CONFIG. FPW 的配置文件专门用来设置用户的运行环境。比如说，您在一个程序中为了方便某些操作修改了默认目录，而结束时没有将默认目录改回来，那么，当其他涉及到目录的程序运行时，如调用表单、程序将产生错误。由于这些错误的出现和程序运行的先后次序有很大的关系，使得发现和调试这些错误显得很困难，使一些潜在的不稳定因素存留在了程序中。

通过我们的实践认为一个程序的基本框架应该是这样的：首先，保存当前的环境，当然保存所有的环境也是多余的，只须保存那些您在程序中也许会改变的环境，最常用的如当前的工作区、当前的表名等等，可以通过使用一些命令得到这些环境配置，如 ALLAS　返回当前打开的表的别名，SELECT　返回当前的工作区；其次，设置要用到的环境，如打开一些表等等。为了有较好的通用性和防止错误的发生，通常使用下面一组命令打开表：

```
IF NOT USED“TABLENAME”
    USE TABLENAME IN 0
 ENDIF
```

先测试是否已打开该表，如果没打开才在新的工作区中打开该表。这样的程序就具有很高的可靠性；再次，就是程序体了，在这部分实现您要实现的功能；最后，当程序结束时，还原程序运行前的环境。

7.2.5　不使用绝对路径

还有一点，最好在编写代码时不要使用绝对路径，多使用相对路径。我们的项目将来也许会在许多机器上使用，安装时不可能每次都安装在相同的位置，所以绝对路径会使程序变得很不灵活，路径稍有改变就不能运行了。您一定不想在使用说明中写上“我们的系统只能安装在 C：XXX 目录下，否则它将不能正确运行，所以请不要修改该目录位置的名称”这样的话吧！那显得我们的程序也太死板了吧。

绝对路径：指明盘符、位置等详细信息的路径。如“C：h WINDOWS h COMMAND”就是绝对路径。

相对路径：通过相对的位置指明的路径，通常在不同的环境中的实际位置是不同的。如“… h PROG”表示当前的目录的父目录下的 PROG 一目录。如果当前目录不同，其实际位置也不相同。

使用 SYS　函数可以返回一些路径的信息，如盘符、默认路径等。详见联机帮助 MSDN 中相应命令的解释。

7.2.6　命名规范

命名规范也是一个好的编程习惯。它不像其它了规范那样是必须遵守的，您完全可以自己随意来命名一个变量、表单或是其他什么东西。您甚至可以有自己的一套命名规范，供您的项目使用。当然，如果您希望开发出一个更好一点的程序，如符合国标或符名 Windows 徽标的项目，那就不能乱来了。

在许多命名规范中，匈牙利命名规范也许是使用最广的一种，它主要是针对使用 C + + 在 Windows 中编程时使用的规范。

具体的规范随个人喜好而定，只要易用、易记就行了。但在同一个项目最好使用相同的命名规范。表 7 - 1 是一些我们常用的命名规范。它不很全面，仅供参考。在您实际应用中应该规定出一套供自己使用的规范来。在一个由多人开发的大型项目中，统一的命名规范显得尤为重要。

表 7 - 1　　自定义命名规范示例

类型	前缀	示例
字符型	C	C_Name
逻辑型	L	L_Sex
日期型	D	D_Date
数值型	N	N_Age
表单	Frm	FrmLoad
菜单	Menu	MenuSytem
组合框	Comb	CombName
文本框	Txt	TxtAddress
命令按钮	ComB	ComBExit

7.3 主程序的构造

主程序一般是作为一个应用项目的入口。在 VFP 中,主程序不是必需的,表单、菜单都可以成为一个项目的入口。VFP 中称应用程序的入口点为主文件。您可以从鼠标右键弹出的快捷菜单中选择设置主文件来修改一个应用项目的入口点。但是,使用主程序显得更规范。VFP 也推荐大家使用一个程序作为项目的入口。

VFP 的主程序主要完成初始环境的设置、调用开始界面、进入事件循环、等待用户输入、结束时清理环境,它只完成了一些最基本的功能,一个最简单的主程序如下:

```
*****************************************
* 程序名:MAIN
* 作  用:主程序
*                          作者:王杰
*                               2000/04/08
*****************************************
DOSETTING.PRG            &&执行环境设置程序
DOFORMSTART              &&运行开始界面
READEVENTS               &&进入事件循环
DORESETTING.PRG          &&执行环境还原程序
```

其中的 SETTING 和 RESETTING 都是用户自己建立的程序,分别用来设置环境和恢复环境。一个实际使用中的例子如下所示:

```
*****************************
* 运行环境设置
* 程序名:SETTING.PRG
*****************************

SET SYSMENU OFF
SET SYSMENU TO
SET TALK OFF
SET NOTIFY OFF
SET CLOCK STATUS
SET PALETTE OFF
SET BELL ON
SET SAFETY OFF
SET ESCAPE ON
SET  KEYCOMP TO WINDOWS
SET CARRY ON
SET CONFIRM ON
SET EXACT ON
```

```
SET NEAR ON
SET ANSI OFF
SET LOCK ON
SET EXCLUSIVE OFF
SET MULTILOCKS ON
STE DELETED ON
SET OPTIMIZE ON
SET REFRESH TO 0,5
SET ODOMETER TO 100
SET DATETO YMD
SET RESOURCE ON
SET CENTURY on
SET CURRENCY LEFT
SET CURRENCY TO
SET HOURS TO12
SET DECIMALS TO2
SET FDOW TO1
SET FWEEK TO1
SET MARK TO .
SET SEPARATOR TO
SET POINT TO .

**************************
* 环境设置还原
* 程序名:RESETTING. PRG
**************************
SET SYSMENU TO DEFAULT
SET SYSMENU ON
SET NOTIFY ON
SET EXCLUSIVE ON
SET SAFETY ON
MODIFY WINDOWS SCREEN
```

可以看到,在 SETTING 函数中,作了很多的设置,包括时间、日期、小数点、货币符号等等。如果您对这些设置语句不熟悉,请查阅连机帮助。这样详尽的设置覆盖了几乎所有可能使用的环境,使程序在受系统设置方面的影响减少到了最小。函数 RESETTING 在项目中不是必须的。使用它主要是在调试项目时方便使用。比如,在程序中修改了系统菜单,在程序中修改了系统菜单,在 Vicual-FoxPro 开发环境中运行后,系统菜单并不会恢复,这使我们调试很不方便,所以在 RESETTING 函数中完成一些环境的恢复,而不是恢复所有的设置。

在主程序中 READEVENTS 这条语句是必须有的,它将你的应用程序带入事件循环中去等待

用户的干预。曾经有人问我,为什么做出来的程序连编完运行时一闪而过 其实就是因为少了这条语句。当执行到这条语句时,主程序就停下来进入事件循环中。但是请注意必须有相应的清除循环语句才能正常退出。即使在屏幕上已经没有任何窗口时,您的程序还在运行,只不过是在后台运行罢了。您只能通过按 Ctrl + Alt + Del 来结束应用程序。那么,怎样才能从事件循环中正确退出呢 其实很简单,只要在您的系统的退出事件中 一般会是一个退出按钮的 CLICK 事件,或是主界面的 UNLOAD 事件 加一上条语句:CLEAR EVENTS。这条语句使用程序结束事件循环,返回到使用 READEVENTS 的程序中,继续向下运行。

当然,作为一个实际使用的主程序,上面的例子显得过于简单。在实际应用中,主程序应该完成的功能还有公用变量的声明和初始化、错误的截获和处理程序等。

7.4 修改程序时的注意事项

由于程序不可能一次就编写正确,没有任何错误,加之用户的需求也在不断的改变,修改程序变得不可避免。

想必大家都有过修改程序的经历,经常是越改越乱,反而不如重做一次快。自己修改自己的程序还好些,但有时需要你去修改别人的程序,那就不是那么简单的事了。修改程序时,前面提到的那些良好习惯和命名规范就有很大的用处了。如果大家都用同一种命名规范,程序中注释完整,那修改起来也就事半功倍了。尤其是在修改别人的程序时,要是他完全随心所欲定义自己的变量,而且缺少注释,那这种程序的修改工作真不如重做一次快。我曾修改过的一个程序中,使用了一个表,其字段有 80 个,字段名依次是 A1 ~ A16,B1 ~ B16,C1 ~ C16,D1 ~ D16,E1 ~ E16,F1 ~ F16,每个字段用处都不同,类型也不尽相同,且没有任何注释,在程序中,检测月末时间的地方有四、五处,且分别使用不同的方法。面对这样的程序,您说能好修改吗 连看看也觉得麻烦。所以,写程序时一定要规范一点。免得给自己和别人修改时带来不必要的麻烦。牢骚发够之后,任务还得完成,改程序确是一件费神的事,也有些方法可以使你不至于越改越乱。

在修改程序时首先是注意备份,先把原程序备份了再修改,每当修改到有一定成果时也要做备份。常做备份是一个很好的习惯,不仅仅在修改程序时如此,在做项目时、写文档时也应该如此。

再者就是别轻易删除原代码。如果原代码确实须删除,也最好注释掉,否则到了最后改得一塌糊涂时,连想恢复原来的代码都很困难了。

对所有的改动都要加上注释,最好是用“*”框起来。在注释中要写明修改人和修改的时间,还有修改的原因等信息。

下面是一个修改程序的简单例子:

```
*************************************************
* 程序名:CLICK
* 作  用:* 表单按钮“添加”的单击事件。
*                                  作者:王杰
*                                       2000/4/8
*************************************************
OLDTABLE = ALIAS                  &&取初始环境
```

```
    IF NOT USED“STUDENT”
      USE STUDENT IN0
    ENDIF
    SELE STUDENT
* APPEND BLANK                    &&XXX 删除。1999 / 3 / 11
    **********XXX 添加,修改添加语句 1999 / 3 / 11**********
    INSERT INTO STUDENT VALUES“姓名”
**************************修改结束**************************
    THISFORM. REFRESH
    SELE&OLDTABLE                  &&环境恢复
```

第 8 章　调试

8.1　调试概述

当您的表单运行时出现象“找不到 PRARMETERS 子句”的错误时，怎么办 ——调试。

当您的程序运行时出现“数据类型不匹配”的错误怎么办 ——调试。

当您在操作中出现“更新冲突”怎么办 ——调试。

在您构造自己的应用程序时，面对出现的一个又一个错误，却又无从下手，怎么办 ——调试。

如果您的程序已没有任何错误，编译也顺利通过，该完成了吗 非也，还有一项重要的工作没有做——调试。

我想，当您学完上面的章节后，在使用中困难的不是如何编程，而是如何解决各种各样的错误，但在许多参考书中都 找不到合适的方法，所以，我们根据自己的使用经验，特地编写了这一章，给用户一个解决之道。

在您的程序完成后，交付用户使用前，调试也是很重要的一步，它由测试和修改组成，主要完成程序的查错、完善和修改等任务。注意，编译通过不等于没有错误，并且没有任何错误是不可能的。将一个未经调试的程序交给用户使用是很不负责任的行为，也会给自己日后带来不必要的麻烦 注：本书中所有程序已经过我们反复验证，保证无误，请放心使用 。当然，程序完成后的测试，最好由别人来进行，所谓“旁观者清”嘛，就像考试时，想必大家都经历过，自己看自己做的题，总也看不出错误来，“只缘身在此山中”。许多软件在发布以前，都会发放一些免费使用的、存在未知错误的测试版，供大家使用，所谓测试版即由此而来。有些公司用 α 版、β 版来表示经过不同程度测试的版本。除去商业目的 抢占市场等等 以外，就是希望通过广泛的测试发现错误，以完善产品。大家熟知的一些软件如 Windows 98、Netscape、Winamp 等等都发放过许多测试版。仅 Windows 98 就有好几个测试版，即使如此，Windows 98 中还是有许多 BUG，一次演示中，不是照样“在大名鼎鼎的比尔·盖茨面前崩溃了”吗 可见，调试工作对一个成品软件的重要性。

BUG 程序中的错误、漏洞的专用术语，有是也称为虫子。

调试，是一个对能力要求很高的工作，它需要您对所用的语言及其开发环境非常熟悉，包括它的 BUG，这样当出现一些问题时，您可以迅速找到问题的原因。它需要您对程序的流程有一个整体的认识，这样当错误发生时，您可以条理清晰的查找到错误可能发生的地方，总之，通过学习调试，将使您的程序更完善，也会使您的各种能力有很大的提高。由于调试有别于程序设计，没有什么固定的规则，而实际情况又多种多样，涵义所有可能的问题是没有必要也是不可能的，所以本着“授人以渔”的思想，着重讲述方法和技巧 其中穿插着大量例子 ，而在实际中，则需要您灵活运用，因为调试的特殊性，本章的例子与前几章没有很多联系，大部分是专门为本章设计的，可独立使用。

8.2　数据库系统中的常见错误及调试

几经考虑，我们还是把这节的标题加上了“数据库”三个字。因为说到底，我们的程序、表单、菜

单都是建立在数据库上的,数据库是最重要也是最容易出问题的地方。数据库可以说是重中之重,如果数据库没有建立得很合理、很完善、很强健、您的整个“作品”也不会是一个优秀的工程。希望您了解到这样一个事实当您的工程越接近完成，数据库中的细微修改对整个工程的影响越大，甚至是全部得重做! 在我们初做项目时,数据库设计的失误,使我们返工数次! 所以,作为一个数据库应用开发者,希望您首先建立好数据,完善好您的数据库,再去做其他事。在本节中,通过一个个从实际应用中精心挑选的实例,您将看到一些操作中经常会遇到的一些错误,也许您没有遇到,但希望您对这些有所了解,相信您定会有所收获。

下面以一个学生表的建立过程中的一些问题为例，来说明调试中应注意的问题假设已建立如图 8 - 1 所示的表结构。

考虑一个很常见的要求 :STUDENT 表的姓名字段不能为空。怎么办 您一定会说,加一条记录验证规则 NOT EMPTY 姓名 给姓名字段。见图 8 - 2。

这样就行了 请在命令窗口用 BROWS 打开这个表。然后 APPEND BLANK,系统会“嘀”地一声,提示您“违反了字段姓名的有效性规则”。怎么回事 原来当您用 APPEND BLANK 时,会给每个字段添一个空值,当然违反了记录验证规则。而此时验证规则又不能没有,怎么办

图 8 - 1　例子中使用的表结构

图 8 - 2　添加字段规则

有两种解决方法：一是添一个默认值，二是用 INSERT 语句，带上 VALUES 参数。在这里，我们用添加默认值的方法，因为这样可以尽可能使已有的程序不做改动，如图 8 - 3 所示。

图 8 - 3　修改默认值

当您修改了表的结构后，无论是多小的修改，您都一定要对所有涉及到该表的程序、表单进行调试，以保证无误。

为了在下面的例子中，您能得到和我们一样的结果，请您先去掉记录验证规则，稍后我们会再次加上并进行解释。

STUDENT 表的学号当然应该是惟一的，所以设置学号为主索引，如图 8 - 4。

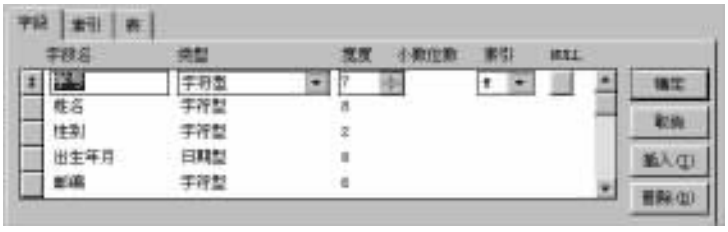


图 8 - 4　设置主索引

然后，添加 3 条记录，注意学着不要重复，如图 8 - 5 所示。

图 8 - 5　添加 记录后情况

然后，定位到学号为 1 的记录，使用 DELETE 命令删除该记录，然后再使用 APPEND BLANK 命令添加一条记录，修改其学号为 1，然后移动记录指针到其他记录，使该修改得到保存。但此时出错了，系统提示您 '主关键字不唯一"，很显然是和前面删除的记录中的学号发生了冲突，但是明明已经删除了该记录，怎么会不唯一呢

是不是没有删除除掉　用 BROWS 浏览一下，原来只在记录的前面加了一个删除标志　如看不到，请先打一个命令：SET DELETE OFF　，这会影响到主索引吗　当然会，怎么办　有两种解决方法：一是删除完一条记录后，立即用 PACK 彻底删除掉。二是给主索引添加一个筛选条件 NOT DELETE

。在本例中,我们使用添加筛选条件的办法。

图 8 - 6　设置索引的筛选条件

如图 8 - 6 所示,在索引标签中设置筛选条件。筛选条件使记录在排序时,只考虑符合筛选条件的记录。对不符合条件的记录不予以考虑,我们在这里设置为 NOT DELTET　表示在排序时排除已经删除的记录。

添加筛选条件后,让我们在来看看结。如图 8 - 7 所示。

	学号	姓名	性别	出生年月	邮政编码	家庭住址
▶	9401101	马天明	男	12/12/81	310000	浙江杭州
	9401102	江　山	男	03/14/82	116000	辽宁大连
	9401103	李　智	女	10/02/80	430000	黄沙洲和路123号
	9401105	田　园	男	04/11/81	230000	安徽合肥
	9401106	赵明山	男	09/05/82	710000	陕西西安解放路34号
	9401107	赵　晶	女	06/10/80	100000	北京西城区367号

图 8 - 7　加筛选条件的结果

不要看到添加没有问题就认为没 BUG 了,调试时,要对各种情况反复测试,反复进行添加删除操作,检查是否有 BUG。

用户提出,当添加学生时学号要能自动生成就好了,VFP 并没有这种功能,怎么办　我们想到了字段的默认值可以是一个自定义函数,为此,我们修改了学号的默认值为一个自定义函数 AUTOADD　,它使得添加记录时,学号自动加 1,如图 8 - 8 所示。

图 8 - 8　添加默认值函数

并在存储过程中定义函数 AUTOADD　,代码如下:

```
*******************************************
* 程序名:AUTOADD
* 目　的:用于使学生表学号字段自动增加
*
*                          作者:王杰
*                          1999 年 1 月
*******************************************
* 函数声明:
```

```
PROCEDURE AUTOADD
*设置排序索引为学号：
SET ORDER TO 学号
*到学号最大的记录
GO BOTTOM
*取出学号值给自定义变量 HAO
HAO = 学号
*将学号加一
HAO = HAO + 1
*返回 HAO 的值
RETURN HAO
*函数结束
```

请您查看上面的程序，好像没什么错误，让我们来测试一下：用命令打开表进行浏览，用 APPEND BLANK 试一试。不对，提示“数据类型不匹配”。哪里出错了 涉及到数据的只有三个地方，一个取学号，一个加一操作，一个返回学号，取学号不会有问题，加一呢，呀，学号是字符型的，原来如此。改吧，把学号改为数值型吧，当然也可以在程序中把字符型转换成数值型，但费点劲，这里我们假设您改了学号的类型。这下该没问题了吧！ 注：请打开浏览窗口 再测试一次，怎么回事，见图 8 - 9。

	9401105	田　园	男	04/11/81	230000
	9401106	赵明山	男	09/05/82	710000
	9401107	赵　晶	女	06/10/80	100000
	9501101	李丽霞	女	09/04/82	300000
	9501102	王　茜	女	02/06/81	154000

图 8 - 9　错误发生时的情况

最后一条记录学号为空而倒数第二条学号却加了一个一，试着定位到最后一条记录修改修改吧，不行！定位不到！怎么了 VFP 坏了 系统出毛病了 不，都不是。关闭浏览窗口，再打开，您会发现，记录并没有添上，而最后一条记录的学号加了一。

怎么回事呢 让我们分析一下，从现象上看，是记录指针发生了意外改变。查看程序，发现其中使用了 GO BOTTOM，只有它有可能修改了记录指针。让我们来仔细分析一下程序运行过程，当您添加一条记录时，默认值里的函数触发了，此时用 GO BOTTOM，记录指针会到哪呢 此时新记录在哪个位置呢 学号最大的记录在最后吗

分析可知，学号值最大的加了一，所以我们取出的学号是最大值，也就是说，GO BOTTOM 确实把记录指针指到了学号最大的那一条上，但返回前没有把记录指针还原到新记录上。噢，原来如此，那么，把指针还原到新记录上不就行了 但是，新记录在哪儿呢 我们很自然的猜想新记录应该在最后，但此时 GO BOTTOM 语句执行后，显然不是新记录。那么在已索引的最后一条记录之后会不会还有一条记录呢 为了验证我们的假设，我们在返回前加一条 SKIP 语句，有人会说，不对吧，已经 GO BOTTOM 了，再 SKIP 岂不是又犯了记录超出范围的错误 管它呢，试试再说，修改程序如下：

```
*********************************************
* 程序名 :AUTOSDD
* 目　的 :用于使学生学号字段自动增加
*
*                                  作者 :王杰
*                                  1999 年 1 月
*********************************************
* 函数声明 :
PROCEDURE AUTOADD
* 设置排序索引为学号 :
SET ORDER TO 学号
* 到学号最大的记录
GO BOTTOM
* 取出学号值给自定义变量 HAO
HAO = 学号
* 将学号加一
HAO = HAO + 1
* 将记录号往后跳一条
SKIP        &&修改后添加的语句
* 返回 HAO 的值
RETURN HAO
* 函数结束
```

让我们来测试一下成果,再用命令 APPEND BLANK 或打开浏览窗口用快捷键 Ctrl + Y,终于成功了! 见图 8 - 10。

	9401106	赵明山	男	09/05/82	T10000	陕西西安解放路34号
	9401107	赵　晶	女	06/10/80	100000	北京西城区357号
	9501101	李丽霞	女	09/04/82	300000	天津市河西区304号
	9501102	王　蕾	女	02/06/81	154000	黑龙江佳木斯市向东区
▶	9401111	姓名	女	/ /		

图 8 - 10　修改后的正确显示

真有趣,一条看似错误的语句居然使程序取得成功! 趁热打铁,下面我们来总结一下我们学到的东西。

我们看到,一条看似错误的语句成了程序成功的关键,您也许认为这只是瞎猫碰了个死耗子,其实,如同金庸的武侠小说中一些奇特的武功一样,这看似反常的一条语句却蕴涵了许多深刻的道理 :首先,它说明要想做好程序,必须大胆,敢于尝试,不被条条框框束缚住,敢想才行 ;其次,做程序也要求您对它的底层有一些了解,如果您不知道新添加的记录在物理顺序上一般在最后,您也就不会猜想到用 SKIP 去定位到新记录 ;再次,您应该对 VFP 中各个程序发生的时刻和先后有一个明确的认识,如果您认为默认值中的程序在新记录添加完成时发生,那您就一定会走入另一个误区不可能成功了!

我想问大家一个问题,上面的那个程序真的没有问题了吗　想一想,再回答。别被一点点胜利

冲昏了头脑，它其实还是有错误，只是现在没有发作而已！您看出来了吗 您是否注意到我们的测试其实并不完善 一个隐藏很深的、只在很特殊的条件下才会发作的 BUG 就藏在其中！让我们把它捉出来！

请大家考虑一下，有什么特殊情况我们在测试时没有想到呢 哈，当表中一条记录也没有时，程序还会正常运作吗 从程序上看，没有考虑这个问题但没有记录时，运行了 GO BOTTOM 之后会到一条空记录吗 会取回什么数据呢 如果是零的话，正好零加一是一，返回值便是一，便万事大吉了，测试一下吧！首先，用 USE STUDENT EXCLU 命令以独占方式打开 STUDENT 表，为确保表为空，用 ZAP 命令清空表，然后再用 APPEND BLANK 命令，系统“嘀”的一声，错了，错误如图 8 - 11 所示。

图 8 - 11 错误显示

显然，是 GO BOTTOM 和 SKIP 引起了错误，既然发现了问题，又知道了问题出现的原因，解决起来应该不费吹灰之力。修改程序如下：

```
*******************************************
*程序名 :AUTOADD
*目  的 :用于使学生表学号字段自动增加
*
*                                作者 :王杰
*                                1999 年 1 月
*******************************************
*函数声明：
PROCEDURE AUTOADD
*判断是否表为空
IF RECCOUNT    = 0        &&如果为空
RETURN!                   &&返回 1 程序结束。
ENDIF
*设置排序索引为学号：
SET ORDER TO 学号
*到学号最大的记录
GO BOTTOM
*取出学号值给自定义变量 HAO
HAO = 学号
*将学号加一
HAO = HAO + 1
*将记录号往后跳一条
```

SKIP　　&&修改后添加的语句

＊返回 HAO 的值

RETURN HAO

＊函数结束

为确保正确，让我们再多测试几次，添几条记录试试，清空了再试一次，怎么样　总算成功了！这个程序历经几次修改终于没有什么 BUG 了。

很显然，一个不完善的测试使我们遗漏了这个发生机率很小的错误。这更说明了测试必须是完整的、严密的，我们的程序中的 BUG 才会尽可能少。从中可以看出发现问题比解决问题更重要。这就要求您有敏锐的洞察力和严密的思维。所以，使您的测试尽量覆盖所有可能出现的情况这才有可能发现所有的 BUG!

学号字段的自动加一终于实现了，该把整个表再完善一下，还记得前面的例子中的姓名字段的记录检验规则吗　现在把它们加上吧。方法同前面的例子，这里就不再详叙了。好了，默认值早已测试过，而学号刚刚也测试了，应该没问题了吧。为了保险，我们还是来测试一下。

使用命令 APPEND BLANK 向表中添加一条记录，又出问题了，如图 8 - 12 所示。

图 8 - 12　错误提示

这个错误很怪，明明添加了默认值，前面的例子中也测试成功了，为什么还会出现违反记录验证规则呢

让我们来分析一下到底是什么原因。当只有学号的默认值时，测试很成功，当只有名字的默认值时也测试成功了，但是，当两个都有默认值时，却失败了，这是怎么回事呢　难道这两个默认值不能共存吗　还是有一个默认值没有起作用　如果是，那怎么才能知道默认值是否起了作用呢

为了测试的方便，我们把姓名的默认值也改为一个自定义函数，如下图 8 - 13。

图 8 - 13　修改默认值函数

程序写在存储过程中，如下：

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

＊测试用程序

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

PROCEDURE XINGMING_TEST

RETURN“姓名”

这显然和以前的默认值是等价的，只是用一个函数来实现。但是我们还是无法知道这个程序

是否能够执行了。有过编程经验的读者一定会说,设置一个断点。但很快您就会发现,在存储过程中是无法设置断点的!

断点:让程序发生中断的地方。在调试中,经常通过在有可能出现问题的地方设置断点来使程序在该处发生一个中断,使程序员可以逐条的调试程序。

这里,我向您介绍一个很实用的技巧:用 WAIT WINDOW 命令。当您无法中断程序或者调试中不需要调出调试器这样很强大但也很耗内存的东西。那就用 WAIT WINDOW 来解决问题吧。当您在程序中使用了 WAIT WINDOW 后 不带 NOWAIT 参数 ,程序运行到此处时会发生中断,等待用户按下一个键,WAIT WINDOW 的详细使用方法请参考帮助。

下面对两个默认值的程序修改如下:

```
*************************
* 测试用程序
*************************
PROCEDUREXINGMING_TEST
WAIT WINDOW"姓名默认值函数运行了!按任意键继续"
RETURN"姓名"
*****************************************
* 程序名:AUTOADD
* 目 的:用于使学生表学号字段自动增加
*
*                          作者:王杰
*                          1999 年 1 月
*****************************************
* 函数声明:
PROCEDURE AUTOADD
* 测试使用
WAIT WINDOW"学号默认值函数运行了!按任意键继续"
* 判断是否表为空
IF RECCOUNT    = 0       &&如果为空
RETURN1                  &&返回 1 程序结束。
ENDIF
* 设置排序索引为学号:
SET ORDER TO 学号
* 到学号最大的记录
GO BOTTOM
* 取出学号值给自定义变量 HAO
HAO = 学号
* 将学号加一
 HAO = HAO + 1
* 将记录号往后跳一条
SKIP                     &&修改后添加的语句
```

＊返回 HAO 的值

RETURN HAO

＊函数结束

我们来调试一下，再用命令 APPEND BLANK 您会发现在屏幕的右上方出现了一个小窗口当按下任意键后,又出现了另一个窗口。依次如下图 8 - 14 所示。

图 8 - 14　先后出现的 WAIT WINDOW 提示

看来两个默认值确实都起作用了,但还是不行! 难道真的是鱼与熊掌不可兼得得　是不是 VFP 本身的缺陷造成的呢　当时我在碰到这个错误时，也几乎认为是 VFP 的错，差一点与成功失之交臂。因为不想草率的下结论,我又单另做了一个表进行测试,也加上两个默认值函数,在函数中只简单的返回一个符合字段类型的值，这个测试很简单，我就不再详细说明了，请读者自己去试一下。结果发现两个函数并不互相影响,工作都很正常! 这说明 VFP 本身还是正常的,但问题出在那里呢

为了方便比较，我们把姓名字段的记录验证规则去掉，并把其他两个字段的默认值都添上一个自定义函数。分别为 TEST1 和 TEST2 并在存储过程中定义这两个函数,如下：

```
PROCEDURE TEST1
WAIT WINDOW"年龄的默认值函数运行了,按任意键继续"
RETURN 20
PROCDURE TEST2
WAIT WINDOW"地址的默认值函数运行了,按任意键继续"
RETURN"DD
```

现在，请您再使用 APPEND BLANK 命令来添加记录，您会发现四个函数依次发生，结果的确是后三个起了作用,唯独第一个没作用! 如图 8 - 15 所示。

9401105	田　国	男	04/11/81	230000	安徽合肥
9401106	赵伟山	男	08/05/82	710000	陕西西安解放路36号
9401107	赵　晶	女	06/10/80	100000	北京西城区367号
9501101	李福雅	女	09/04/82	300000	天津市河西区304号
9501102	王　雪	女	02/06/81	154000	黑龙江佳木斯市向阳区

图 8 - 15　出现错误时的显示

看来有戏了,让我们来比较一下它们之间的区别。直观的看,在学号字段前面的默认没值起作用,在它后面的默认值起了作用,学号字段的默认值函数与其他的有什么不同呢　它改变过记录指针! 难道问题是出在这里吗　难道改变指针会使以前发生的默认值函数失效吗　为了验证这一猜测,我们修改两个字段的先后顺序,如图 8 - 16 所示。

再来测试,用命令 APPEND BLANK,终于成功了! 好了,把测试用的东西全部去掉,把姓名字段的记录验证规则也添上,总算圆满完成了,别忘了再好好测试一下唷!

通过这个例子,我们又掌握了不少东西。更重要的,我认为是一种不轻易放弃,不轻易下结论的原则,这才是成功的关键所在。

图 8 - 16　调整学号和姓名的先后次序

经历了数次的的修修补补，经历了为断的调试、修改、再调试，一个看似很简单的学生表终于建立起来了。上面的这些例子碰到的问题都是我们在实际应用中遇到过的，但是在一般参考书上又找不到的东西，经过我们的精心设计，把他们集中到一个表的建立过程中来，希望在实际应用时能给您一些帮助，当然，上面的例子并没有涵盖所有的数据库应用中的错误，有些是因为很简单，一两句话便可说清，有些是因为太复杂，不易设计一个例子，有些是不常用，没必要浪费笔墨。稍后的汇总中我们会以很简单的几句话来简要的说明一下，都是些经验之 谈，如果您在使用中遇到什么问题，希望您一定去看看，定会有所收获。

在结束本节前，有一点想提醒读者，做程序要精益求精，要知道自己的不足，才能不断进步。比如在上面的例子中我们很费劲的建立了一个 SUTDENT 表，但是，它很完善吗　非也！其实，您很快就会发现，当用户使用一段时间，经过数次增删之后，这个程序只会使学号越来越大，不会找一个最合适的学号，即当您删除了一条学号不是最大的记录后，这个程序并不会把这个学号当成新添加记录的学号，而仍然是把最大的学号加一，这个程序很死板，没有一点智能，如何做一个更好的、更完善的程序呢　那是程序设计方面的问题，超出了本章的范围，请读者自己去尝试吧！

第 9 章　创建 HTML 帮助

帮助文件对用户来说是学习如何使用软件的一个很有价值的信息来源，对于一个便于用户使用的软件，联机帮助是必不可少的。在 Internet 上常常看到一些“高手”对微软的东西嗤之以鼻，但微软的软件却如此流行，这和它强大的帮助系统是分不开的。一个好的帮助可以使用户更快的学会使用软件，可以减少培训操作员的时间，还可以减轻维护费用。在 VisualFoxPro 中，可创建图形方式帮助或者 . dbf 样式的帮助。图形方式帮助可以是 Windows 标准“帮助”文件，或是 Web 样式的“HTML 帮助”文件。. dbf 样式帮助基于字符，并可灵活地转移到其它平台上。这种样式的“帮助”很容易创建，因为它基于 VisualFoxPro 表。这里我们只介绍 HTML 帮助的创建。

HTML 帮助是 Microsoft 所提供的用于创建适应 Internet 时代要求的帮助文件的解决方案。MicrosoftVisualStudio6. 0 包含有创建 HTML 帮助文件的 MicrosoftHTMLHelpWorkshop Hhw. exe 。本章将通过一个实例，演示如何应用该工具创建一个图形方式的帮助。

HTML 超文本识别语言 ，是标记和标识文档的一个系统，这样该文档可以在 WWW WorldWideWbd 上发布。按照 HTML 准备的文档包含引用图形和格式标记。可以使用 Web 浏览器例如，Microsoft Internet Explorer 查看这些文档。

9. 1　认识 Microsoft HTML Help Workshop 工具

HTML 帮助由 MicrosoftHTMLHelpWorkshop 创建，此软件包含在 VisualStudio 中并独立于 VisualFoxPro 之外。HTMLHelpWorkshop 提供一个完整的 HTML 帮助创作系统，并且包含有向下兼容的功能，此功能可以很方便地从已有的 Winhelp 项目中创建 HTML 帮助文件。

9. 1. 1　启动 MicrosoftHTMLHelpWorkshop

从开始菜单中选择 HTMLHelpWorkshop⇨HTMLHelpWorkshop，如图 9 - 1 所示。

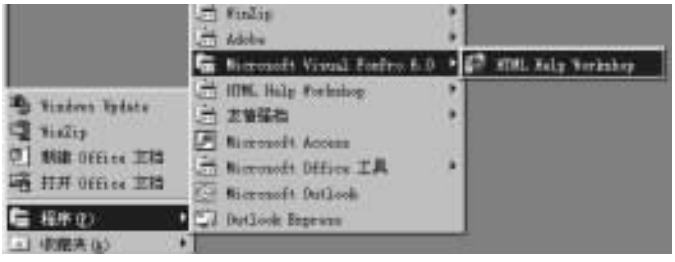


图 9 - 1　启动 HTML Help Workshop

9. 1. 2　HTMLHelpWorkshop 界面

图 9 - 2 所示为 HTMLHelpWorkshop 系统界面。我们不能详细介绍该系统的所有组成部分及其功能，只对下面要用到的软件进行说明，对于其他部分，读者可以参阅 HTMLHelpWorkshop 的联机帮助。

图 9-2　HTML Help Workshop 界面

1 Viewcompiledfile 查看已编译的文件 命令按钮：

单击该按钮可以选择查看一个编译过的帮助项目文件，其后缀为 .chm。

2 Contents 内容 选项卡：

点击该选项卡可以为帮助添加“内容”也就是目录列表 ，显示在导航栏 NavigationPane 内。

3 Add/Modify/window/definitions 定义或修改窗口类型 命令按钮；

单击该按钮可以定义帮助窗口的样式。

4 Save all proiect files and compilec 保存所有文件并编译命令按钮：

单击该按钮时可以保存所有文件并将项目编译为 .chm 文件，之后就可以用于我们的应用程序了。

9.1.3　HTMLHelpWorkshop 的文件类型

为 HTML LHelp Workshop 常用的文件类型，读者可以在 student h helps 目录下看到本章实例所用相应文件。

9.2　为帮助准备 HTML 文件

本节列出了将要用到的四个 HTML 文件，可用 HTMLHelpWorkshop 或记事本等工具编辑它们，保存时文件后缀须写为 .htm。关于 HTML 的详细情况请读者查阅有关资料。

这四个文件以及它所用到的图像文件都保存在本书应用程序实例如例的 Student h helps 目录下。

9.2.1　帮助主页文件 main.htm

```
〈HTML〉
〈HEAD〉
〈metaname =“GENERATOR content = Microsoft&reg HTMLHelpWorkshop4.1 〉
〈Title〉main〈/Title〉
```

```
〈/HEAD〉
〈BODYdackground =“bg. gif  bgcolor =“#FFFFFF  bgproperties =“fixed 〉
〈h1〉〈B〉学生管理系统帮助〈/B〉〈/h1〉
〈hr〉〈br〉〈br〉
〈imgsrc =“red - button - bg. gif“alt  1.  align =  middle 〉
〈U〉〈ahref =  gaishu. htm 〉概述〈/a〉〈/U〉
〈br〉
〈imgsrc =“yellow - button - bg. gif  alt =“2.  align =“middle 〉
〈U〉〈ahref =  anzhuang. htm 〉安装说明〈/a〉〈/U〉
〈br〉
〈imgsrc =  green - button - bg. gif  alt =  3.  align =  middle 〉
〈u〉〈ahref =  gongneng. htm 〉系统功能〈/a〉〈/u〉
〈/BODY〉
〈/HTML〉
```

9.2.2 “概述”文件 gaishu.htm

```
〈! DOCTYPEHTMLPUBLIC  -  / /IETF / /DTDHTML / /EN 〉
〈HTML〉
〈HEAD〉
〈metaname =“GENERATOR  content =  Microsoft&reg  HTMLHelpWorkshop4. 1 〉
〈Title〉gaishu〈/Title〉
〈/HEAD〉
〈BODYbackground =“bg. gif  bgcolor =“#FFFFFF  bgproperties =“fixed 〉
〈h2〉〈B〉概述〈/B〉〈/h2〉
〈ahref =“main. htm 〉〈imgsrc =“back. gif  alt =“目录  border =  1  WIDTH =  74  HELGHT =
33 〉〈/a〉
〈br〉〈br〉〈br〉〈br〉
这里是概述的内容
〈/BODY〉
〈/HTML〉
```

9.2.3 “安装说明”文件 anzhuang.htm

```
〈! DOCTYPEHTMLPUBLIC“ -  / /IETF / /DTDHTML / /EN 〉
〈HTML〉
〈HEAD〉
〈metaname =“GENERATOR  content =“Microsoft&reg  HTMLHelpWorkshop4. 1 〉
〈Title〉anzhuang〈/Title〉
〈/HEAD〉
〈BODYbackground =  bg. gif  bgcolor =  #FFFFFF  bgproperties =  fixed 〉
〈h2〉〈B〉系统安装〈/B〉〈/h2〉
```

〈aherf =“main. htm 〉〈imgsrc =“back. gif alt = 目录 border =“1 WIDTH =“74 HELGHT =“33 〉〈/a〉

〈br〉〈br〉〈br〉〈br〉

系统安装说明

〈/BODY〉

〈/HTML〉

9.2.4 “系统功能”文件 gongneng. htm

〈! DOCTYPEHTMLPUBLIC“ - / /IETF / /DTDHTML / /EN 〉

〈HTML〉

〈HEAD〉

〈metaname =“GENERATOR content =“Microsoft&reg HTMLHelpWorkshop4.1 〉

〈Title〉gongneng〈/Title〉

〈/HEAD〉

〈BODYbackground =“bg. gif bgcolor =“#FFFFFF bgproperties =“fixed 〉

〈h2〉〈B〉**系统功能**〈/B〉〈/h2〉

〈ahref = main. htm 〉〈imgsrc =“back. gif alt =“**目录** border =“1 WIDTH =“74 HEIGHT =“33 〉〈/a〉

〈br〉〈br〉〈br〉〈br〉

系统功能说明内容

〈/BODY〉

〈/HTML〉

9.3 创建 HTML 帮助的步骤

我们将先使用 HTMLHelpWorkshop 提供的向导建立帮助系统的框架，然后设置窗口类型，添加一个内容导航，最后编译、测试。

9.3.1 使用向导

操作步骤如下：

1 **在如图 9-2 所示的 HTMLHelpWorkshop 界面上选择菜单：File⇨New，出现一个对话框 图 9-3 。**

2 **选择 Project，单击“OK”按钮，出现另一对话框，系统询问是否转换已有的 WinHelp 帮助文件 我们不转换 。**

3 **单击“下一步”按钮，出现图 9-4 所示对话框。**

4 **单击“Browse”按钮，选择要建立项目的路径，键入项目名称“student. hhp”，单击“下一步”按钮，出现对话框 图 9-5 。**

图 9 - 3　新建项目

图 9 - 4　路径项目名称

图 9 - 5　选择已有文件类型

图 9 - 6　添加已有文件

5 选择 HTMLfiles . htm ,单击“下一步”按钮,出现对话框 图 9 - 6 。

6 单击“Add…”按钮,添加相应路径下的 HTML 文件 main. htm,gaishu. htm,anzhuang. htm,gongneng. htm ,单击“下一步”按钮,出现对话框 图 9 - 7 。

图 9 - 7 向导完成

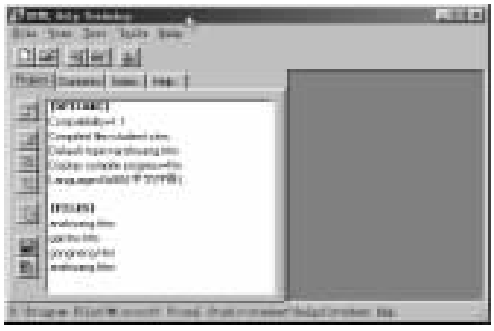


图 9 - 8 向导完成后系统界面

7 单击“完成”按钮,回到系统界面 图 9 - 8 。

9.3.2 设置窗口类型

操作步骤如下:

1 在图 9 - 8 所示的界面上,单击“Add / ModifyWindowDefinitions 添加 / 修改窗口定义 ”按钮,出现对话框,询问自定义窗口类型名字。

2 键入自己的窗口类型名字 mywindowtype,单击“OK”按钮,出现对话框 图 9 - 9 。

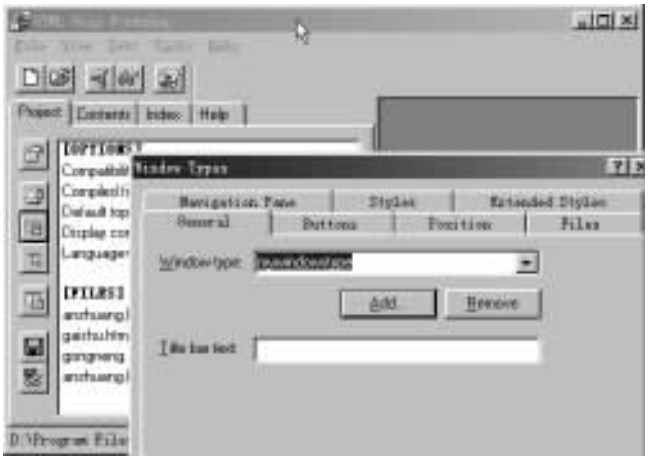


图 9 - 9 设置窗口类型

3 选择选项卡“General”,在“Windowtype :”文本框中输入“mywindowtype”,在“Titlebartext :”文本框中输入“学生管理系统帮助”,设置帮助窗口的标题。

4 选择选项卡“Buttons”,选择“Hide / Show”、“Back”、“Forward”、“Home”、“Print”,定义工具栏中出现的命令按钮。

5 选择选项卡“Files”,在“Toc :”文本框中输入“student. hhc”,在“Default :”文本框中输入“main. htm”,在“Home :”文本框中输入“main. htm”,指定内容文件表文件 student. hhc 文件将在稍

后建立　设置帮助主页 Home 及默认帮助页面 Default 。

6　选择选项卡“Position”，在“Left :”文本框中输入“10”，在“Top :”文本框中输入“27”，在“Width :”文本框中输入“551”，在“Heigth :”文本框中输入“386”，设定帮助窗口出现时的大小及位置，单位为像素　如非特别指出，本章所用长度单位均为像素　。

7　选择选项卡“NavigationPane”，选中“WindowWithNavigationPane”，在“NavigationPaneWidth :”文本框中输入“160”，设定帮助窗口包含导航栏 NavigationPane 并设置导航栏的宽度。

8　单击“确定”按钮，回到系统界面　见图 9 - 2 ，请比较，项目栏内容比图 9 - 8 增加了几项。

9. 3. 3　添加导航器

在 9. 3. 2 的步骤　5　中，我们输入了一个文件名“student. hhc”，此处给出该文件的建立过程。

1　在系统界面　见图 9 - 2　上，选择“Contents”选项卡，系统弹出对话框。

2　选择“Createanewcontentsfile　新建一个内容文件　”，单击“OK”按钮，出现对话框，询问路径及内容文件名。

3　输入文件名 student. hhc”，单击“保存”按钮，回到系统界面，如图 9 - 10 所示，只是没有图中的“帮助主题”、“概述”、“安装说明”和“系统功能”等条目，下面我们就来添加这些内容。

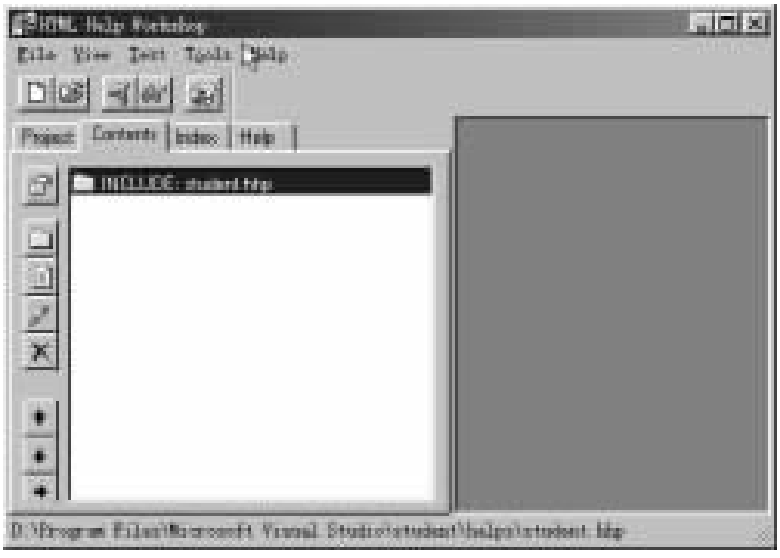


图 9 - 10　添加导航内容　一

4　单击“Insertaheading　插入一个标题头　”按钮，出现对话框，如图 9 - 11 所示，只是没有“帮助主题”和“main. htm”。

5　在“Entrytitle :”文本框中输入“帮助主题”，定义标题头的名字。

6　单击“Add…”按钮，选择 main. htm 文件，单击“OK”按钮，回到图 9 - 11 所示界面，在“Files / URLsandtheirinformationtypes :”列表框中有了一个文件　main. htm 。这样就建立了标题头“帮助主题”和 HTML 文件“main. htm”的关联，当单击导航栏中的“帮助主题条目时，帮助系统就会调用“main. htm”文件从而显示该帮助页面　以下方法相同，不在重复　。

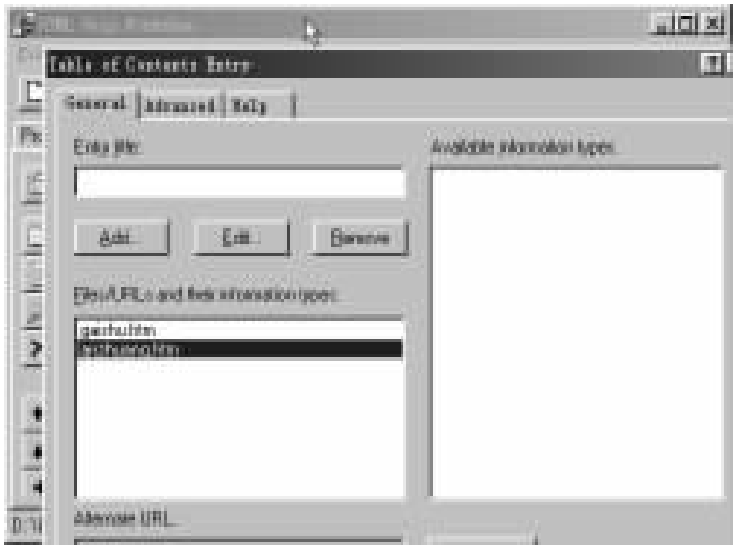


图 9 - 11　添加导航内容

7 单击“确定”按钮,回到图 9 - 10 所示界面,contents 选项卡中多了一个“帮助主题”条目。

8 单击“Insertapage 插入一个页 ”按钮,和“Insertaheading”的方法一样,在图 9 - 10 所示界面中添加标题 Entrytitle:为“概述”,相应文件为 gaishu. htm 的条目。

标题头 heading 和页 page 的关系是,一个标题头包含几个页。

9 重复 8 ,添加其他两个条目 “安装说明”和“系统功能” ,“安装说明”对应“anzhuang. htm”文件,“系统功能”对应“gongneng. htm”文件。

完成后就成为如图 9 - 10 所示的界面,“帮助主题”条目下包含了“概述”、“安装说明”、“系统功能”三个子条目。

9.3.4　编译测试

1 在系统界面 见图 9 - 2 上,单击“Saveallprojectfilesandcompile”按钮,系统对项目进行编译,生成 student. chm 文件。

2 在系统界面 见图 9 - 2 上,单击工具栏上“Viewcompiledfile”按钮,选择 student. chm 文件,单击“View 查看 ”按钮,系统将显示编译后的帮助文件,如不满意可回到系统界面对刚才的 student. hhp 进行进一步的修改。图 9 - 12 是我们所创建的帮助系统界面。

9.4　如何在 VisualFoxPro 中激活帮助

编译后的 HTML 帮助文件 student. chm 就可以联入我们的 VisualFoxPro 应用程序,方法是:

1 在主文件中加入代码 假设当前目录为应用程序安装目录 :

```
sethelptoshelp hstudent. chm
```

这样当系统执行“help”命令时就会激活我们建立的帮助系统。“help”命令可以放在一个按钮的单击事件中,也可以放在帮助菜单项中。

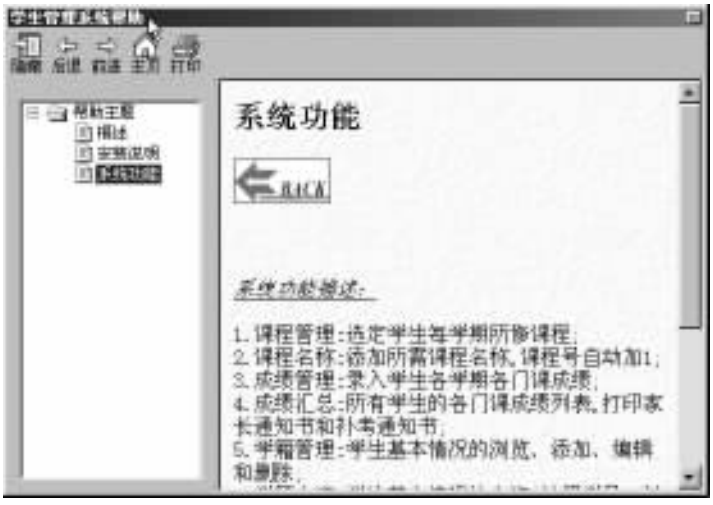


图 9 - 12　帮助预览

2 **设置快捷键 在主文件中加入代码：**

on key label F1 help

这样当我们击 F1 **键时可激活帮助系统。**

3 **如果想还原** VisualFoxPro **指定的默认帮助可执行命令：**

set help to

第 10 章　创建发布磁盘

在完成应用程序的开发和测试工作之后，可用“安装向导”为应用程序创建安装程序和发布磁盘。如果要以其他磁盘格式发布应用程序，“安装向导”会按指定的格式来创建安装程序和磁盘。

如果仅仅把文件复制到用户的机器上，应用程序有可能不能正常运行。而 Windows 的安装例程，比如由安装向导所创建的安装程序，会进行版本检查，并注册多个 DLL 和 ActiveX 文件。因此，为了确保正确安装，请使用“安装向导”。

本章通过一个实例介绍了应用安装向导创建发布磁盘的过程。若要运行安装向导，在 Visual-FoxPro 系统菜单中选择“工具⇨向导⇨安装。结果如图 10 - 1 所示。

图 10 - 1　选择要发布的目录

下面详细介绍安装向导的每个步骤：

1 定位文件。如图 10 - 1 所示，选择应用程序所在的目录，系统将包装该目录下的所有文解和子目录。例中应用程序所在目录为… hStudent。

2 指定组件。选择应用程序所必须的组件：

如图 10 - 2 所示，我们选择下列应用程序组件：

图 10 - 2　选择要安装的组件

“VisualFoxPro 运行时刻组件”：我们的应用程序需运行 VisualFoxPro 运行时刻文件 Vfp6r. dll 。这个 . DLL 文件自动包含在应用程序文件中，可以在用户的计算机上正确的安装。

“HTML 帮助引擎”：为应用程序中自定义的帮助文件　例如的 student. chm　提供 Microsoft-HTMLHelp 引擎。

3 磁盘映像。设置向导存放压缩后文件的路径，其中包含每种指定类型磁盘的映像。在运行向导之前可以建立磁盘映像子目录。如果想让向导来建立，在文本对话框键入新的名称。本例将压缩文件放在 hStudentsetup 目录下 图 10 - 3 。

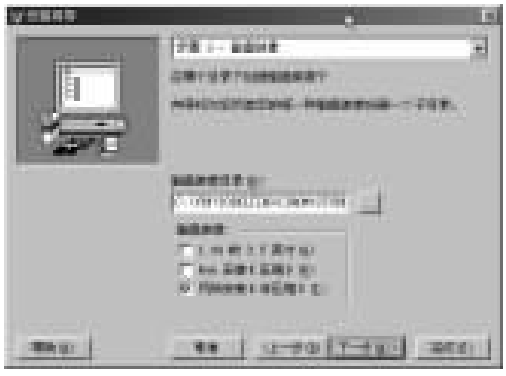


图 10 - 3　设置磁盘映像的路和包装盘格式

磁盘映像格式有三种选择，可多选：

①如果选择“1. 44MB3. 5 英寸”，向导将在发布目录中创建 3. 5 英寸软盘的映像。

②如果选择“网络安装　压缩　”，向导将创建密集压缩的安装映像，用于从 Web 站点快速下载文件。

③如果选择“网络安装　非压缩　”，向导将建立惟一的子目录来包含所有的文件。

我们的选择“1. 44MB3. 5 英寸”和“网络安装　非压缩　”，如图 10 - 3 所示。读者可从本书附带盘中找到相应路径安装例子程序。

4 安装选项。设置用户进行安装时，在安装屏幕上的一些信息提示，以及安装完后立即执行的程序命令　如 Readme 。设置如图 10 - 4 所示。

图 10 - 4 设置安装时对话提示

5 默认目标目录。设置进行安装时默认的路径名称与程序组名称 图 10 - 5 。

图 10 - 5　设置安装时默认的路径和程序组名称

安装程序将把应用程序放置在“默认目标目录”对话框指定的目录中。

如果在“程序组”框中指定了一个名称，当用户安装应用程序时，安装程序会为应用程序创建一个程序组，并且使这个应用程序出现在用户的“开始”菜单上。

选择在“用户可以修改”下面的选项，用户在安装过程中可以更改默认目录的名称和程序组。

6　改变文件设置。向导在表格中列出文件。可以通过单击要改变的项来改变对文件的设置 图 10 - 6 ，每列显示的设置有：

图 10 - 6　设置程序组项目

①文件 指定在用户机器上创建文件时使用的名称。

②目标目录 :文件能够安装在用户机器上的应用程序目录、Windows 目录或者 Windows / system 目录内。

③程序管理器项 :如果选择此选项，向导将显示“程序管理器项”对话框，从中可以指定程序项的属性 :说明、命令行和图标。选择 student. exe 后的复选框，出现对话框图 10 - 7。

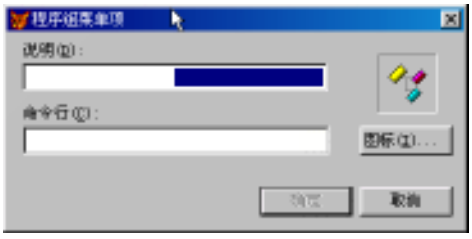


图 10 - 7　开始制作安装盘

在命令行可以使用嵌入的%s 序列代替应用程序目录,其中的“s”必须小写。要将程序安装到应用程序子目录中时，应使用%s，这可以确保在用户为应用程序子目录指定了不同于默认值的名称时,文件也可以安装到正确的目录中。本例中在命令行中输入：

%s hstudent. exe

student. exe 为编译后的可执行文件

如果指定了源树之外的图标,安装例程将在应用程序子目录中安装该图标。

④ ActiveX 项：如果用户在应用程序中包含了 ActiveX 控件则需选择该项，安装程序将在用户机器上注册该控件,保证程序正常运行。

7 完成。当选择“完成”时,向导开始创建应用程序磁盘映像 图 10 - 8 。

图 10 - 8　开始制作安装盘

安装盘制作完后,在机器的 hStudentsetup 目录下创建了 Disk144 和 Netsetup 两个子目录 图 10 - 9 。用户可以执行这两个目录下的 setup. exe 命令安装应用程序。

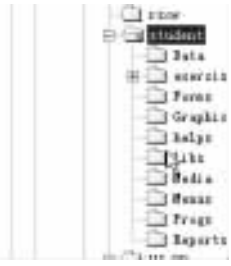


图 14 - 9　向导完后磁盘目录结构

第 11 章　远程数据连接

11.1　建立 Client / Server(客户 /服务器)

“网络就是计算机”。随着现代计算机技术的发展，SUN 公司的这句名言越来越被大家所接受。现在,网络的发展成为一个时代的潮流。局域网、Intranet、Intranet 的应用越来越广泛。相应的，单机数据库系统也变的不能满足用户的需要，取而代之的是用于网络的大型数据库的普遍应用。Oracle、SQL Serer 等大型数据库在中小型企业的应用开始普及。Client / Server 客户 /服务器 模式也不在只是一个新名词。可以说,现在的时代是网络的时代在这样的时代中,Visual FoxPro 又可以充当什么样的角色呢

在网络环境中,应该承认 Visual FoxPro 不是一个合适的服务器端数据库。与大型数据库相比，它缺乏很好的安全性和对网络的优化。但是在小型的网络中，当对安全性要求不是很高时，Visual FoxPro 完全可以胜任,Visual FoxPro 6.0 更是增强了这方面的功能。在众多的大型客户 /服务器系统中,大多选用 Oracle、SQL Server 作为后台数据库。而在客户端,则有许多工具可以使用,如 Visual Basic、DELPHI、Power Bulider 等,Visual FoxPro 也是其中之一。我认为 Visual FoxPro 是其中比较好用的一个前端开发工具。为什么选择 Visual FoxPro 呢 因为它本身就是一个数据库操作系统,大型数据库使用的 SQL 语言在 Visual FoxPro 中一样适用。这使得代码有很好的一致性、通用性。在单机上制作的软件,可以很方便的升级到网络上。Visual FoxPro 中众多的向导可以很迅速的帮助您完成这些操作。另外,Visual FoxPro 有很好的可视化界面,您可以通过鼠标的操作来完成复杂的远程访问,而无须编写代码。Visual FoxPro 还提供了远程视图、游离视图等工具使远程操作变得和本地操作一样容易。通过游离视图您甚至可以在网络不通的情况下浏览和修改服务器数据，当网络连通时再把数据回传到服务器,而这 看似复杂的操作在 Visual FoxPro 中只需通过简单的几条命令就可以实现。

Visual FoxPro 使用 ODBC Open DateBase Connect 来实现与远程数据库的连接。您可以在控制面板中设置它。ODBC 是由 Microsoft 公司提出的一个通用的数据库接口,已经成为业界的标准,被大多数数据库接受,每种数据库几乎都提供相应版本的 ODBC 驱动程序，当服务器数据库升级时，您可以简单的通过修改 ODBC 驱动，来使原来的程序适应新的数据库环境，而无须重做客户端程序。

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查询、报表和处理等优点与严密的多用户访问、海量数据存贮、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。

Client / Server 客户 /服务器 是一种应用程序的名称,它具有本地 客户 用户界面,但访问的是远程服务器上的数据。这种应用程序根据前端和后端产品的能力将工作分布到本地机和服务器上。远程视图和 SQL pass - through 二者都可以用来创建客户 /服务器应用程序,二者结合起来则可获得更强大的功能；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。

11.1.1　远程视图

远程视图使用远程 SQL 语法从远程 ODBC 数据源表中选择信息；本地视图使用 Visual FoxPro SQL 语法从视图或表中选择信息。您可以将一个或多个远程视图添加到本地视图中，以便能在同一个视图中同时访问 Visual FoxPro 数据和远程 ODBC 数据源中的数据。也可以在单个视图中访问远程数据和本地数据，即创建集成本地数据与远程数据的视图。方法如下：

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“升迁向导”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。另外，可以使用“ ”视图作为开发强大的客户/服务器应用程序的核心方法。

11.1.2　SQL pass - through

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，因而能够在很大程序上提高客户/服务器应用程序的性能。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执行。如果服务器支持外部联接的话，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联接。远程视图和 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，从而获得更强大的功能。

11.2　升迁 Visual FoxPro 数据库

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，最大限度地重现 Visual FoxPro 数据库中各表的功能。利用“升迁向导”可以实现以下功能：

1　将本地数据转移到远程服务器上。
2　将本地基表和本地视图转换为远程基表和远程视图。
3　将本地应用程序移植为客户/服务器应用程序。

11.2.1　把 Visual FoxPro 升迁到 SQL Server 上

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。

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，就可以启动升迁向导。

1　选择菜单：工具 ▷ 向导；
2　选择“升迁”，在从“向导选取”对话框中，选择“SQL Server 升迁向导”，启动 SQL Server 升迁向导；
3　在向导中，选择本地数据库；
4　选择表，并匹配这段数据类型；

5 选择目的数据库,设置数据库属性;
6 指定日志属性;
7 设置升迁选项;
8 选择“完成”按钮后,“升迁向导”开始将数据迁移到服务器上。

11.2.2 升迁到 Oracle 上

建立与 Oracle 服务器相联的命名连接或者建立好 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可以开始升迁。

先从“工具”菜单上,选择“向导”,然后再选择“升迁”;在“向导选取”对话框中,选择“Oracle 升迁向导”,开始升迁。

启动 Oracle 升迁向导:
1 选择本地数据库;
2 选择数据源;
3 选择表;
4 匹配字段数据类型;
5 选择表空间;
6 选择表空间文件;
7 指定簇信息;
8 指定簇表;
9 指定簇键字段;
10 设置升迁选项;
11 完成。

在选择“完成”之前,向导不会对服务器上进行任何操作,所以,在此以前的任意时候都可以选择“取消”来退出向导。在没有完成所有的向导步骤之前选择“完成”,“Oralce 升迁向导”会采用剩余步骤默认值。当提供了升迁所需要的基本信息之后,完成了“Oralce 升迁向导”,系统就开始将 Visual FoxPro 数据库导入到服务器上。

11.3 远程数据更新

有两种连接远程数据源的方法,一种是直接访问在机器上注册的 ODBC 数据源,另一种是用“连接设计器”设计自定义连接。使用远程视图,无需将所有记录下载到本地计算机上即可提取远程 ODBC 服务器上的数据子集。您可以在本地机上操作这些选定的记录,然后把更改或添加的值返回到远程数据源中。

11.3.1 远程视图

要了解远程视图首先要知道什么是远程数据。

远程数据:简而言之,就是当前数据库以外的数据。所以远程视图 remote view 也就是使用当前数据库之外的数据源的一种视图。

远程视图是一种强有力的技术,用它可以从一个远程服务器中选择所需要的数据并下载到一

个本地 Visual FoxPro 临时表中，同时可以查看并更新远程数据。视图具明长期性，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。

11.3.2　创建连接

使用“连接设计器”能够创建并修改命名连接。如果想为服务器创建定制的连接，可以使用“连接设计器”，创建的连接将作为数据库的一部分保存起来，并含有访问特定数据源的连接信息。因为连接是作为数据库的一部分存储的，所以仅在打开的数据库中才能使用“连接设计器”。

创建新的连接：

1 在“项目管理器”中选定一个数据库；

2 选定“连接”并选择“新建”；

3 在“连接设计器中，根据服务器的需要输入选项；

4 从“文件”菜单中，选择“保存”命令；

5 在“保存”对话框中，向“连接名称”框中输入连接的名称；

6 选择“确定”。

一旦建立了连接，“打开”对话框就会显示出来，从中可以选择远程服务器上的表。当表选定后，“视图设计器”就会打开。

11.3.3　创建远程视图

视图可分为本地视图和远程视图，本章着重介绍远程视图。远程视图使用远程 SQL 语法从远程 ODBC 数据源表中选择信息：本地视图使用 Visual FoxPro SQL 语法从本地视图或表中选择信息。您也可以将一个或多个远程视图添加到本地视图中，经便能在同一个视图中同时访问本地 Visual FoxPro 数据和远程 ODBC 数据源中的数据。

创建使用远程数据源 ODBC 的视图，必须存在一个数据库来保存视图，同时还需要存在数据源或命名连接。如果数据库淌有打开，系统将提示您打开数据库或创建数据库。在“远程视图向导”运行之。从“工具”菜单内“向导”子菜单中选择“查询”，在“向导选取”对话框中选定“远程视图向导”，可以在“选定”对话框的“远程数据”选项卡上设置远程视图和连接的默认选项，并在“连接设计器”中创建连接。

启动远程视图向导，如图 11 - 1 所示。

图 11 - 1　进入远程视图向导数据源选取

1 选取一个已有的数据源或到数据源的连接；
2 选择你希望包含到远程视图结果中的字段；
3 根据你所希望的排序方式选取用为排序依据的字段；
4 为表建立关系；
5 包含记录；
6 排序记录；
7 筛选出的记录同操作符框、值框一起来创建表达式，从而确定输出的字段；
8 完成。

11.3.4 用远程视图更新数据

远程视图对远程数据的修改可以通过对视图设计器中的“更新”条件选项的设置来控制，也可以控制表中指定字段更新的开关，并设置适合服务器的 SQL 更新方法。

1 使表可更新。

如果希望在表的本地版本上所作的修改能回送到源表中，需要设置“发送 SQL 更新”选项，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。

2 允许源表的更新。

在“更新条件”选项卡中，设置“发送 SQL 更新”选项。

3 更新所有字段。

如果需要，可以将表中的所有字段设置成可更新的。

4 使所有字段可更新。

在“更新条件”选项卡中，选择“全部更新”。

表 11-1 在“更新条件”选项卡设置 SQL WHERE 子句

希望达到的效果	应选
当清表中的关键字段被改变时，使更新失败	关键字段
当远程表中任何标记为可更新的字段被改变时，使更新失败	关键字和可更新字段
当在本地改变的任一字段在源表中已被改变时，使更新失败	关键字和已修改字段
当远程表上记录的时间戳在首次检索之后被改变时，使更新失败 仅当远程表有时间戳列时有效	关键字和时间戳

11.3.5 在视图中使用多个远程表

与远程数据源建立连接时，这个数据源可能包含许多相关表。您也可以从中选择需要的表。需要的话还可以调整这些选取表之间的关系，获得所需的信息。

创建多表远程视图的步骤如下：

1 从“文件”菜单中，选择“新建”命令，再选定“远程视图”，然后选择“新建文件”按钮；
2 在“选择连接或数据源”对话框中，选择一个已定义的连接或可用的数据源；
3 根据要求登录到服务器上；
4 在“打开”对话框中，选定您要用的第一个表；

5 选择“添加表”按钮以选定附加的表。

与本地视图一样,可以利用视图设计器中的“更新条件”选项卡,为远程视图设置更新方式。

如果需要的数据存储在远程数据源中，并且又与本地表中的信息相关，就可以在一个视图中合并本地数据与远程数据。

首先创建远程视图,从远程服务器上仅挑选出想操作的相关记录;然后利用本地视图,把该远程视图和相关的本地表联接起来。

在视图中合并远程和本地数据的步骤如下：

1 创建远程视图并加入一个或多个远程服务上的表；

2 创建新的本地视图并把刚创建的远程视图加入到其中；

3 将相关的本地表加到本地视图中,并根据公共字段把本地表和远程视图联接起来；

4 在本地视图中设置选定条件,然后运行该视图；

5 更新本地视图中的结果,使本地表和远程视图都得到更新；

6 关闭本地视图及远程视图,以更新远程服务器上的数据。

11.3.6　游离视图

将一些数据从主数据库中分离出来进行显示、收集或修改，这样即使断开与数据源的物理连接,也可以实现对自由数据的处理。Visual FoxPro 相应的提供了可用自由数据的功能。使用这种功能，可以用与主数据库相连的视图建立数据的子集，这样就可以脱离主数据库对自由数据进行操作。比如对一个大型数据库,如果只对某方面数据感兴趣,可以构造一个只包含相关信息的视图,随后将数据从主数据库中分离成自由数据,允许相关的各用户更新数据。最后,把变动过的数据提交给主数据库。在地理位置较运的情况下处理数据,条件不允许与主数据库相连,这时可以事先将有用的数据子集将主在便携机中,独立于主机数据库调整这些数据,等到可以与主数据库相连时,再用这些调整后的数据更新主数据库。也有些数据会随某一时刻的到来而产生变化。这些情况下,都适合使用自由数据。

操作游离视图的步骤：

1 在创建游离视图前,分析需求以决定需要用手游离数据库中的视图结构。首先确定脱机使用的数据子集，然后就可使用已有的视图或是创建新视图。如果已有视图返回的记录正是您脱机操作时想要的,可直接使用它,否则需要以编程方式创建视图。

游离视图中包含的数据存储在本地机数据库中的一个 . dbf 文件中。由于视图是游离的,所以只能以编程方式设置它的更新属性,而不能在“视图设计器”中修改游离视图。如果要在游离视图中修改数据,则要在孩子前确认视图可更新。

2 用 CREATE SQL VIEW 命令创建视图。如果使用已有视图。则可省去这一步。

3 用 CREATEOFLINE　使视图呈游离状态例如,如果要在客户机上更新成绩表,添加科目信息,需要当前成绩单以及考试科目信息。就可以建立一个称为 Gradeinfo 的视图,组合那些来自于成绩表和科目表的信息,可用下列代码创建游离视图：

CREATEOFFLINE "Gradeinfo" 。

4 为自由数据创建了视图后,就可在应用程序中像使用其他视图一样使用这些视图为添加、修改和删除记录。由于 Visual FoxPro 中的事件是单线程执行的，所以数据库中表只能以独占方式打开,也就是在同一时刻只能被一个用户使用。但如果把库中的有关数据分离成自由数据,创建游离数据库的视图结构,就可以让多个用户以共享模式同时访问同一数据库中的游离视图。另外,如

果不想保留对自由数据的任何修改,还可以恢复信息重新反映原始信息。

5 处理完自由数据后,就可以更新服务器中的数据了。

如果要在创建游离视图的计算机以外的机器上使用此视图,必须在这台机器 目标机 上建立主数据库文件 .dbc 的副本 确认目标面上的视图使用了 ODBC 数据源 并分析数据需求以决定所需的视图内容。

使用下列代码更新单条记录

```
USE Gradeview ONLINE EXCLUSIVE * 以独占方式打开该游离视图
GO TO 10 * 移到第 10 条记录
IF TABLEUPDATE 0,. F. ,wordarea * 如果出现更新冲突
* 处理冲突代码
ENDIF
```

游离视图是一种很实用的技术，使用游离视图可以在网络不通的情况下继续操作，在网络连通后将操作上传。但是在许多参考书中对游离视图讲得很少。下面我们以一个实际的例子向大家展示游离视图的使用方法。

游离视图在使用时，请大家注意必须使游离视图所在数据库保持打开并且为当前数据库，否则,所有操作将无法完成。

我们通过一个类来实现游离视图的数据传输和刷新。使用游离视图中的一个缺陷是在游离态无法刷新,所以您必须通过取消游离态来达到刷新的目的。但是,在取消游离态时如果远数据库断开。游离视图也就丢失了。所以在取消游离态时,您最好检测数据库是否边通。如何检测呢 我们上面曾提到要将远程视图和 SQL pass - through 结合使用,这里就用到 SQL pass - through 语句了。

下面类中刷新函数的原代码,它的输入参数为游离视图的名称：

```
**************************************
* 程序名 :REMOTEREFRESH
* 作　用 :刷新游离视图
* 说　明 :输入参数 :游离视图名称,字符型。使用 ODBC
*　　连接名为 DKYDATA
*　　作者 :× × ×
**************************************
LPARAMETERS VIEWNAME
WAIT WINDOW"正在与主服务器连接,请等待…"AT 15,45 NOWAIT
* * 使用 SQL pass - through 语句检测数据库是否连通
do while handle! = 0 &&循环等待连接结果
handle = sqlconnect "dkydata" &&与服务器连接
enddo
wait clear
IF handle <0　　&&连接句柄小于 0,数据库不通
 = messagebox "连线失败,请稍候重试! ! ",48,"信息窗口"
RETURN
ELSE
SELECT &VIEWNAME
USE
```

```
IF DROPOFFLINE  VIEWNAME     &&取消游离态
        USE & VIEWNAME
        RENUM = REQUERY          &&刷新
        IF RENUM = 1
            = MESSAGEBOX "刷新成功!",48"信息窗口"
        ELSE
            = MESSAGEBOX "刷新失败,主服务忙游离视图丢失!!",48,"信息窗口"
        ENDIF
        USE
        REcreate = CREATEOFFLINE  VIEWNAME  &&重新建立游离视图
        IF NOT REcreate
            = MESSAGEBOX "游离视图丢失,系统将退出,请重新入后使用恢复程序!!",48,"信
息窗口"
            QUIT
        ENDIF
        USE & VIEWNAME
        DO RESETINDEX. PRG WITH VIEWNAME &&重新设置索引
        USE
ELSE
        = MESSAGEBOX "连线失败,请检查线路是否畅通!",48,"信息窗口"
ENDIF
SET DEFAULT TO
USE &VIEWNAME order mytag SHAREN
SQLDISCONNECT  handle  &&清除连接
ENDIF
相对来说,传输数据的程序要简单的多:
*****************************************
*程序名:REMOTESEND
*作　用:刷新游离视图
*说　明:输入参数:游离视图名称,字符型。
*                                    作者:× × ×
*****************************************
LPARAMETERS VIEWNAME
USE & VIEWNAME ONLINE EXCLU        &&连线
IF tableupdate  2  . t.                  &&传输
  = MESSAGEBOX "OK! 传输成功,数据已输入主数据库 . ",48,"信息窗口"
ELSE
  = MESSAGEBOX "传输失败,数据未全部输入主数据库,请稍候重试 . ",48,"信息窗口"
ENDIF
use & VIEWNAME SHARED
```

第 12 章　与 INTERNET 连接

随着计算机的日益发展和普及，网络逐渐深入人心，Internet 已经成为网络的代名词。由于 MicrosoftWord 的广泛应用，本章将首先介绍使用邮件合并向导将数据库中的数据发送到 Word 文件的方法。Internet 为目前应用软件的广泛应用提供了必要条件，6. 0 版本的 VisualFoxPro 已具备了与 Internet 的接口，本章还将介绍如何使用 Internet 搜索向导建立 Wed 查寻页面的技术。使用 VisualFoxPro6. 0 可以很容易地创建与 Internet 一起使用的应用程序，也使得创建与其他基于 Windows 应用程序 如 MicrosoftExcel 和 MicrosoftVisualBasic 一起使用的应用程序变得很容易。

在本章中，我们通过两个向导的使用步骤来详细讲述 VisualFoxPro 在 Internet 方面的应用。在您的印象中对 Internet 也许只限于 WWW 浏览，但它只是 Internet 众多功能中的一个应用，其他像电子邮件、FTP 等也是 Internet 的重要功能。而其中的电子邮件是最早使用的也是应用最广泛的 Internet 功能。WWW 浏览是当前最流行的 Internet 功能。所以，我们选择这两个向导的使用作为本章的主要内容。

因为对 Internet 中的详细设计，如 HTML 语言，主页制作等超出了 VisualFoxPro 的范围，所以我们没有过多地涉及这方面的东西。

12. 1　使用邮件合并向导

邮件合并称为 VisualFoxPro 高级版本中的一个灵活的功能，可以实现与 Word 文档的数据交换。通过邮件合并，可以把数据库中的有关邮政信息转换成 Word 文档的形式进行通信、保存或排版打印。

通常，使用邮件合并向导，可按以下步骤进行：

1 选择菜单 :工具⇨向导。

2 在子菜单内选择“邮件合并”，如图 12 - 1 所示。

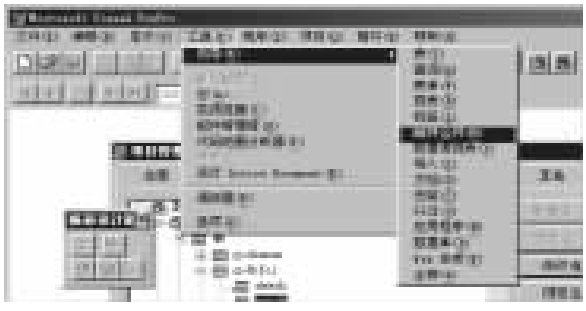


图 12 - 1　“邮件合并”对话框

或者从菜单的“工具”中选择“向导”的全部，从下拉菜单中选择“邮件合并”向导，如图 18 - 2 所示。

图 12 - 2　选择“邮件合并”向导

3 选择需要发送到 Word 的数据库表或视图，然后选择要使用的字段。确定后进入下一步。如图 12 - 3 所示。

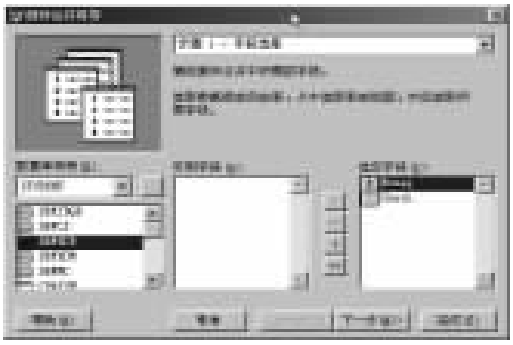


图 12 - 3　字段选取

4 选择好字处理软件后，就可以进入下一步了。确定建立新文档的方式，即新建一个文档或使用已有的 Word 文件，如图 12 - 4 所示。

图 12 - 4　文档类型选取

5 当确定了建立文件的方式后,就可以确定文档格式了。比如选择目录,如图 12 - 5 所示。

图 12 - 5　文档样式选取

到此,邮件合并工作准备就绪。

6 单击“下一步”,完成邮件合并向导,如图 12 - 6 所示。

图 12 - 6　完成邮件合并

7 向导工作完成后系统自动启动 Word 程序,并进入其中,继续运行。如图 12 - 7,Word 自动进行邮件合并功能,出现如图 12 - 7 所示的对话框。以后的操作完完全按照 Word 的功能进行。

图 12 - 7　Word 中的邮件合并

8 如果此时需要打印邮政标签或者创建或获取新的数据源,请更改文档类型,重新设置。如

图 12 - 8 所示，选择新的文档类型为标签。

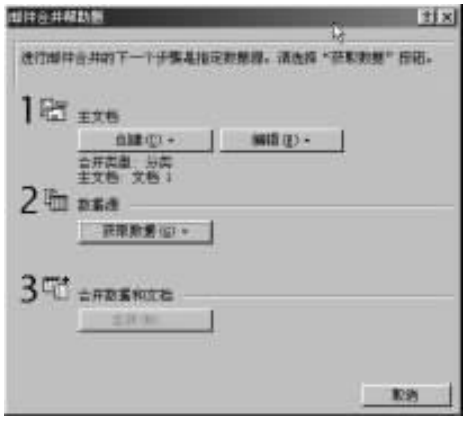


图 12 - 8　Word 中的邮件合并帮助器

图 12 - 9　设置标签样式

9 图 12 - 9 是设置用户自定义标签对话框。在图 12 - 10 中，单击“插入合并域”，可以设置标签格式，结果如图 12 - 11 所示。

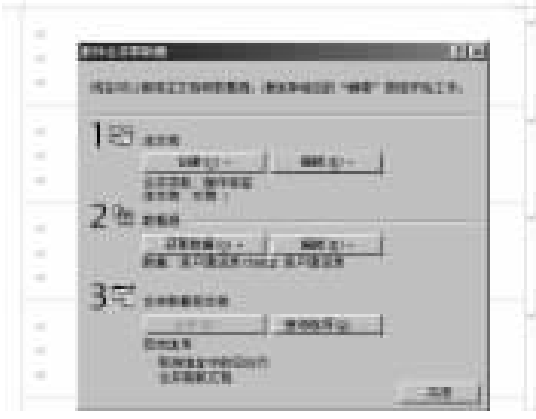


图 12 - 10　插入合并域

图 12 - 11　创建标签结果

10　最后，需要设置合并的一些条件，如记录数目等，以及对记录的一些限制和处理，如图 12 - 12 所示。

图 12 - 12　邮件合并

设置完毕，就可以完成操作了，并立即得到标签文档，结果如图 12 - 13 所示。打印出来的是信封标签，其中的 ID 号和地址等信息来自数据库中的表。

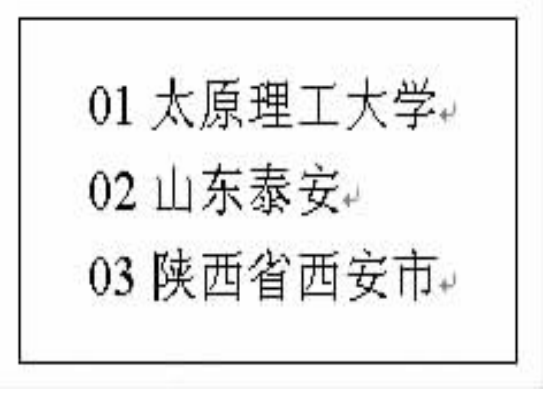


图 12 - 13　信封标签

12.2　用 Internet 搜索向导向 Internet 发布数据

VisualFoxPro6.0 与 Internet 有很强的结合能力。能够将数据发布在 Internet 上，极大地提高了信息的处理和查询能力。

要进入 Internet 搜索页向导，首先需要进入向导选择对话框，选 Internet 搜索页向导并确定。如图 12-14 所示。

图 12-14　WWW 向导选取

然后单击一步，开始建立 WEB 页。首先选择一个自由表。选定表后，就要选择待搜索字段。在浏览器中查询时，就是按照这个字段为顺序，所以该字段必须是索引字段。如果要有几个索引字段时，还要选定哪个是有效索引。如图 12-15 所示。

如同人需要有名字一样，搜索页也应该有个标题。有时甚至只有标题还不够，还须对标题另作说明。如图 12-16 所示。

图 12-15　选择搜索字段

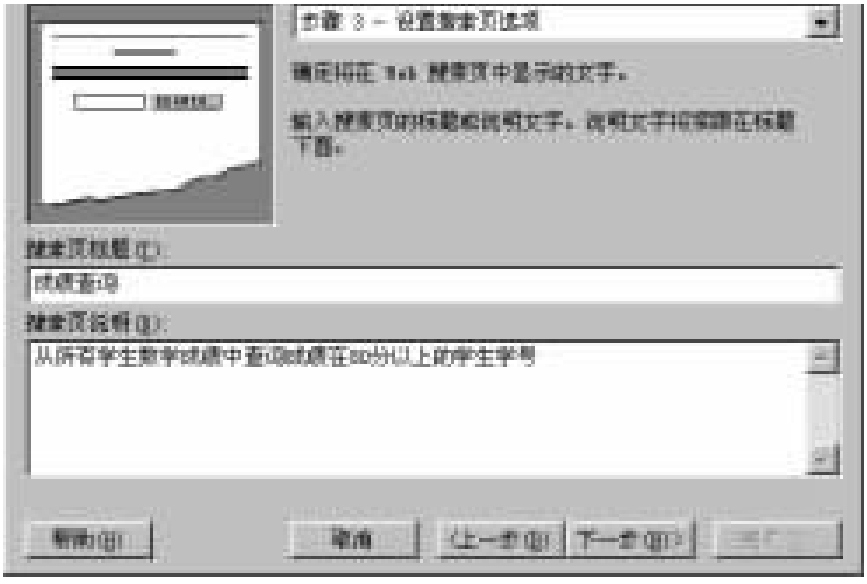


图 12 - 16　设置 WWW 搜索界面

设置页面图形时，可以选择一个图形文件作为背景或标题的图片，图片可以是 JPEG 或 GIF 格式。搜索结果的字段，最多为五个，这些字段是符合前面选择的搜索条件的记录。对搜索的结果，也可以设置是否加入图片、显示多少条记录，以及记录的来源等。记录的默认来源是 FoxProServer，如图 12 - 17 所示。

图 12 - 17　设置 WWW 结果界面

处理完成以后，就可以将结果保存 HTML 格式的文件，使用浏览器来查看。比如运行 IE，打开 HTML 格式的文件，就可以看到设计的结果。这时，只要输入条件，并单击“search”按钮进行查询，结果将被返回。有关 HTML 的内容，本章不作详细的讨论。

第 13 章　扩展 VFP 的功能

13.1　概述

在实际应用中，我们常常发现 VisualFoxPro 的功能不够用。比如我想通过程序操纵调制解调器、我想操纵 Windows 的注册表、我想直接操纵 I/O 地址　从 COM 口取数据等等。在 VisualFoxPro 中确实没有相应的函数能解决这样的问题，有些人因此认为 VisualFoxPro 功能太少，其实在 VisualFoxPro 中，这些问题还是能解决的。

作为一个从单机数据库系统发展起来的 VisualFoxPro，因为拥有着广泛的市场基础和众多的开发人员，最重要的是保持了与 FoxBASE、Dbase 的兼容和简单易学易用，使 Microsoft 公司没有放弃继续发展它的功能，最终在 Windows 的数据库开发系统中占有一席之地。

随着数据库市场的扩大，VisualFoxPro 的应用更加广泛。在 VisualFoxPro6.0 中已经又有了许多新的功能，如对客户/服务器的支持、面向对象的编程技术等等。虽然借助 Microsoft 公司强大技术力量，VisualFoxPro 的性能得到了不断的提高，但作为一个数据库系统，它的功能并不能和 VisualC++等相提并论。在实际应用中，常常会出现一些 VisualFoxPro 内置的特性无法解决的问题。这就需要我们来扩展 VisualFoxPro 的功能了。

既然 VisualFoxPro 不能满足用程序的要求，那我们只能借助于其他程序的强大功能来扩展 VisualFoxPro 的功能。幸运的是 VisualFoxPro 中提供了许多方法来扩展自己的功能。我们可以利用外部库—ActiveX 控件或动态链接库（DLL）来扩展 VisualFoxPro 的功能。使用外部库，可以向应用程序添加对象，从增强的文本框到日历以及其他功能齐全的应用程序。也可以通过应用程序接口（API）调用其他应用程序（包括 Windows 本身）提供的各种强大的功能。

13.2　如何扩展 VFP 的功能

从某种意义上讲，每种语言都有它的不足之处。VB 使用简单，功能也比较强大，但速度太慢。VC 功能强大，速度最快，但写起代码来相对要难很多。VisualFoxPro 也有许多功能不够完善。在编程技术发展迅速的今天，人们学会了取人之长补己之短。Windows 中许多公用的控件和库在 VisualFoxPro 中都可以拿来直接使用。在 VisualFoxPro 中您可以通过使用（ActiveX 控件（.ocx 文件），ActiveX 对象和动态链接库（.dll 文件）来扩展 VisualFoxPro 功能。外部库不仅允许您访问其他程序，而且可以访问 Windows 本身。例如，可以使用一个 ActiveX 控件直接对 Windows 注册表进行读取和更新，也可以通过链接一个 Windows 的 DLL 来调用系统级别的函数。通过这些调用，我想大多数的应用功能都可以满足了吧。

关于 ActiveX 的详细使用方法，请参见本书的第十七章。在那里您将会看到对 ActiveX 的详细的讨论。

如果上面的方法仍不能满足您在应用程序中的需要，怎么办呢（VisualFoxPro 还是有办法解决的。您可以自己通过其他语言进行扩展，使用一个 32 位编译器，例如 MicrosoftVisualC++4.0 或更

高版本，创建一个 ActiveX 控件 .ocx 文件 或用 VisualFoxPro 的库文件 .fll 文件 来完成您的需要。从这里能看出，VisualFoxPro 完全可以实现像 VisualC++等语言一样的强大功能。至于如何使用 VisualC++创建这些文件，已经超出了本书的范围，您可以从一些详细讲述 VisualC++的书中得到帮助。

过多的讲述 Windows 中每一个 ActiveX 控件和 DLL 是没有必要的，对它的使用的函数和方法的详细列举对读者来说没有很大的用处。在下一节，我们将通过一个很简单的例子介绍一下对 DLL 的引用。

13.3 DLL 的使用实例

DLL 全名是 DynamicLinkLibraries，中文名是动态链接库。整个 DLL 统称 DLLs。动态链接库是 Windows 操作系统的核心和精华部分，因为 Windows 系统依靠它来实现 Win32API 的应用程序接口的功能。

动态链接库是一个可执行文件，它包含一个可以被多个进程同时使用的函数库。当代码中的函数调用在运行中与位于 DLL 内的实际代码进行动态链接时，发生动态链接。DLL 是可以自升级的，可以通过升级 DLL 来提高应用程序的性能。Windows 定制控件已存储在 DLLs 中，甚至 Windows 自己提供的公用控件也包含在 DLLs 中。由于它们有良好定义的接口和被许多语言调用的能力，在代码重用和软件构件领域内，DLLs 向前迈出了重要的一步。

Windows 本身主要由 DLL 集合构成，如 USER32.DLL，GDI32.DLL 和 KERNEL32.DLL。这三个 DLLs 构成 Win32 应用程序设计接口 API 的核心。可见 DLL 在 Windows 中的重要性，可以说通过 DLLs 您可以解决所有的 Windows 中的问题，DLLs 就是 Windows 的底层。这也是我们选择它作为向读者演示扩展 VisualFoxPro 应用例子的原因。

如果您需要调用的函数在某个 DLL 中，必须先链接该 DLL 库，再调用 DLL 中的相应函数。当然在调用一个 DLL 函数之前，必须了解该函数的调用协议，包括函数的名称，参数的数目和类型以及返回值类型等必要的信息。

在 VisualFoxPro 中，只能使用为三十二位环境编写的 DLLs。但是，如果需要访问一个十六位的 DLL，可以使用 Foxtools.fll 中合适的函数来实现。有关磁详细内容，请参阅联机帮助 MSDN 中的 Foxtools Foxtools.hlp 主题。

如果您要调用一个 DLL 函数，请先使用 DECLARE - DLL 命令注册 DLL 函数，注意函数的名称区分大小写，这和在 VisualFoxPro 中使用变量是不同的。下面是一个注册的例子：

```
DECLAREINTEGERGetActiveWindowINwin32api
```

它说明您要使用 win32api 中的 GetActiveWindow 函数，该函数没有输入参数，返回值是整型的。该函数将显示 VisualFoxPro 主窗口的句柄。

如果指定 WIN32API 为库名称，VisualFoxPro 将在 Kernel32.dll、Gdi32.dll、User32.dll、Mpr.dll 和 Advapi32.dll 中查找被调用的 32 位 WindowsDLL 函数。

注册后，您就可以像调用其他 VisualFoxPro 函数一样调用注册了的 DLL 函数。下面的程序将注册 WindowsUSER 系统中 DLL 库里的 GetActiveWindow 函数，该函数将显示 VisualFoxPro 主窗口的句柄。GetActiveWindow 无参数，但返回一个一位整数：

```
DECLAREINTEGERGetActiveWindowINwin32api
```

GetActiveWindow

包含所要注册函数的 DLL 文件必须存放在 VisualFoxPro 的默认目录中，或系统目录中，如 Windows 或 System 目录，或者在 DOS 目录中。否则，将提示找不到该 DLL。

如果要调用的函数和 VisualFoxPro 中已存在的函数 本地函数或者前面声明的 DLL 函数 重名，您可以为重复的函数名字取一个别名，然后用别名来调用它。下面的例子为 GetActiveWindow 取了个别名 GetWinHndl，然后通过别名调用它，效果和上一个例子是一样的。

DECLAREINTEGERGetActiveWindowINwin32apiASGetWinHndl

GetWinHndl

注意，在退出 VisualFoxPro 之前，所链接的 DLL 函数一直保持有效，因此，在每个工作期中只须声明一次。不过它要占用一定的内存资源，如果已经不需要调用 DLL 中的函数，可以执行 CLEARDLLS 命令将其从内存中清除以释放资源。

执行 CLEARDLLS 命令时，将从内存中清除所有已声明的 DLL 函数。

如果您需要获得更多的关于使用 DLL 的信息，请运行 Visual Studio… hSamples h Vfp98 hSolution hSolution hWinapi 中的示例表单 Systime. scx。有关如何向 DLL 函数传递参数的其他实例，请参阅 VisualStudio… hSamples hVfp98 hClasses 中的程序 Registry. prg。

如果在 VisualFoxPro 中使用的数据是数组，则在传递给 DLL 函数之前，必须遍历该数组，把它联结到一个用 C 语言样式的数组的单个字符串。如果 Windows 函数需要 16 位或 32 位的值，则在链接为字符串之前，必须将该值转换为等价的十六进制的形式。在传递包含数组数据的字符串时，VisualFoxPro 将该字符串的地址传送给 DLL，该 DLL 将其作为数组处理。实例程序请参阅 VisualStudio… hSamples hVfp98 hSolution hWinapi 中的示例表单 Syscolor. scx。

第 14 章　FoxPro 中级教程

14.1　更多对象

编辑框对象 editbox 

使用编辑框控件可以编辑诸如字符型变量、数组元素、一般字段以及备注型字段的内容，而用得最多的地方就是编辑备注型字段。

VFP 所有的标准编辑功能都能在此使用，比如剪切、复制、粘贴等。编辑框中的文本可以垂直滚动，并且是自动换行的。

复选框对象 checkbox 

复选框用于在两种状态之间切换，比如“真”、“假”或者“是”、“否”等等。

当被选中时，一般条件为真，这时在复选框中有一个钩，其 value 值为 1，否则为 0，也可以是“. t. ”或“. f. ”，如设置了 controlsource，其中的变量也一样。

在 caption 属性中输入文字，可显示在复选框的边上。

在线条对象中供下载的秒表程序里面有对复选框的应用。

微调控件对象 spinner 

微调控件可以让用户在一个范围内对数值进行微调，所谓微调就是将数值一点点地增加或减少，而增加减少的方法就是按向上或向下的箭头，每按一下加 1 或减 1，您还可以直接将数值通过键盘输入进去。

微调控件有个 keyboardhighvalue 属性，它限制了键盘输入的最大值，既然有最大值就肯定有最小值喽，最小值的限制属性就是 keyboardlowvalue。

对鼠标输入自然也有上、下限制，属性分别是 spinnerhighvalue 和 spinnerlowvalue。

在线条对象中供下载的秒表程序里面有对微调器的应用。

页框对象 page 

页框是一种容器型控件，其中可以包含若干页，每一页又是一个容器控件，用法就好象表单，也就是说可以在其中放入其它各种控件，如果您高兴还可以在其中放入一个页框，单击各页左上角的标签来选择页。但是整个页框仍是一个控件，它也必须放入表单这个更大的容器才能使用。

页框的主要属性就是 pagecount，该属性的值决定了页框中页数的多少，隐含是 2。

页框最主要的一个作用是当要显示的内容在一屏显示不完时，可将要显示的内容分成若干页放到页框中，比如在一个表单上显示或修改一个有很多字段的记录，就可这样做。

注意：象上面的做法有个很大问题，就是我们调用表单或页框的 refresh 方法时只有当前活动的页才被刷新，也就是说，如果您将一条记录的多个字段放在了不同的页，当您从一个记录跳到另一个记录，比如有一个向我们的人事管理软件编辑人员表单中的“下一个”按钮被按下，接着调用页框的 refresh 方法，比如执行如下语句：

thisform. pageframe1. refresh

那么就只有您所看到的页面被刷新，即其中的字段显示的是下一条记录的内容，而其它看不见的页面中的字段还是显示的原来的内容，如这时您选择其它页面就可以看到这种情况，无疑这将带来混乱。解决的办法是要刷新页框中的每一个页面，比如：

thisform. pageframe1. page1. refresh

thisform. pageframe1. page2. refresh

. . .

每个页面的标签文字可用页面 page 的 caption 属性设定。

在线条对象中供下载的秒表程序里面有对页框的应用。

形状对象 shape 

用于创建矩形、圆角矩形、圆和椭圆。

这里最重要的属性是 curvature，用于设定曲率，为 0 时没有曲率，形状为矩形；99 为最大曲率，为圆；0 - 99 之间的数值可创建圆角矩形，将圆角矩形压扁到一定程度，使其两边的直线没有，而是圆角对圆角就成了椭圆。

另外设置 backcolor，可改变它的填充颜色。

在线条对象中供下载的秒表程序里面有对形状的应用。

命令按钮组对象 commandgroup 

创建一组命令按钮使您可以同时对多个按钮进行设置，比如移动它们的位置，如果有 5 个按钮是分开的，要分别设置它们的位置，假如 5 个按是和某一个主题有关的，就可以把它们设为一个命令按钮组，这样只要设好这个组的位置，各按钮也就各就各位了。当然也可以分别设置每个按钮的属性、事件程序等。

使用 buttoncount 属性可以设置组中按钮的个数。

列表框对象 listbox 

列表框用于显示一个选项列表，在其中您可以选择所需要的项目。

可用于选择的数据类型有值、别名、SQL 语句、查询、数组、字段，具体使用哪种由 rowsourcetype 属性确定。

较常用的是值和字段，如果选择的是值，也就具体指定几个选择项，如“男”、“女”，那么就将这些候选值放入 rowsource 属性，各项目之间用逗号隔开，字符不需引号。

如果是字段，就将字段名放入其中，最好带上所在表的别名，那么当程序运行时该字段的所有记录（除了用 set filter 等语句屏蔽掉的记录）就将成为侯选项。

当一个项目被选定后，其 value 以及 controlsource 都将是这个值。

可以在程序设计时给 value 设个初始值，这样在运行时光标就会停在这个值上。如果在编程序时不可预知候选项将会是些什么值，如使用字段常常是这样，那么就在程序中编入相应语句，在列表打开之前先给它的 controlsource 变量赋值，要注意的是一般不要赋候选项中没有的值。

选中某个选项后，controlsource 变量的值就是该选项的字符。

最后再提醒一下，rowsource 和 rowsourcetype 属性都要做相应设置。

在线条对象中供下载的秒表程序里面有对列表框的应用。

图像对象 image 

图像对象用于显示图片，5.0 版只能显示 bmp 格式的图形文件，而 6.0 可显示除 bmp 外包括 jpg、gif 等当前流行的图形文件。

但图像对象只能显示图片，而不能修改。然而它同样有许多的属性、事件和方法，可以响应各种事件并在程序运行时更换图像文件。

这里最主要的属性就是 picture，将其设置为所需要的图形文件名，当程序运行时，该图像就会在表单显示出来。

在线条对象中供下载的秒表程序里面有对图像的应用。

容器对象 container 

容器对象就好象只有一个页面的页框，在它里面可以放入各种控件。

您可能会问，用它有什么好处呢，直接将控件放在表单上不就行了吗？

好处在于您可以把一些相关控件放在一起，比如要移动位置时只需移动容器就行了，里面的所有控件都会一起移动，另外我们可能在运行过程由中程序根据某种情况把一些控件取消或添加到表单，如果把它们放在一个容器中，就可以很方便地一起取消或添加。

选项按钮组对象 optiongroup 

选项按钮组是一个容器，其中包含若干选项按钮，它可以让您在一组按钮中选择一个，它是单选的，即您选了一个按钮，原来所选的按钮就释放，始终只能有一个按钮被选中。

被选中的按钮是怎样的不需要我再→铝税伞＊使用 buttoncount 属性可以设置组中按钮的个数。

在程序运行时选了第几个按钮，它的 value 属性就是几，如果您设置了 controlsource 属性为一个变量，那么该变量的值也是这个数，您可以通过这个变量或其 value 值得知是那个按钮被选了。

在线条对象中供下载的秒表程序里面有对选项按钮组的应用。

组合框对象 combobox 

当选择了组合框以后，组合框将打开，显示一个选项列表，您可以从其中选择一个。

组合框组合了文本框和列表框的功能，您可以在文本框部分中输入信息，也可以在列表框部分中选择。它的优点是不象列表框要占很大地方，缺点是要点击一下拉下列表部分才能选择。

与列表框一样也有 rowsource 和 rowsourcetype 属性，使用方法与列表框一样，都要设置。

计时器对象 timer 

计时器的主要作用就是当到一定时间激活一事件程序。

程序运行时该控件是看不见的，只在后台运行。

其主要属性是 interval，将该属性设为所需要的时间值，注意该值是以毫秒为单位的，即 1 秒应设为 1000。

主要事件是 timer，当计时时间一到，该事件即被激活，其中的程序将被执行。

比如我们可以在程序启动时，显示一欢迎窗口，在其中放入一计时器，interval 设为 3000，timer 事件中写入 thisform.release，这样显示 3 秒钟后窗口自动消失。

激活事件后，计时器重新回零，如果计时器仍然有效，则又开始另一次计时。因此我们可以用该控件控制一段程序每隔一定的时间重复执行。

线条对象 line 

线条是一图形控件，作用就是在表单上画线，主要用来美化我们的表单。

该控件有几个主要属性：height、width、lineslant。

这里的 height 不是指线条的长度，width 也不是指粗细，这两个属性构成一个矩形（当然这个矩形是看不见的），线条则为这个矩形的对角线，由此可确定线条的长度和倾斜度。而倾斜的方向由 lineslant 确定，将其设为“ ”则从左上向右下倾斜，设为“ / ”则从右上向右下倾斜，隐含为“ ”。

线条的粗细由 borderwidth 确定。

应用一些技巧也可以让该控件有实际作用，比如可以让它做为一个秒针，在计时器控制下改变它的 height 和 width 属性，使它走动。

14.2　用新的控件改进人事管理（一）

这一节我们用上一节讲的一些控件来对前面的人事管理软件进行改进，使其更易用，更不容易出错，最主要当然是为了进一步说明这些控件的用法。

先讲第一个改进，用选项按钮  来选择性别

在新增和修改人事档案的表单中，本来性别是要输入的，虽然只能输入“. t. ”或“. f. ”，倒是不会出现输入其它字符的错误，但是很不直观，所以我们用选项按钮来改进。

（1）、将表单上输入性别的文本框换为选项按钮组，选项按钮组做好后隐含两个按钮，这里正好我们需要两个，因为性别就两种（大概不会有第三种）。

但是按钮隐含的排列是上下排列，这样很占地方，我们将其调整为水平排列，方法是：先把选项按钮组拉的比较大，使两个按钮都能看见，暂时将别的控件盖住也不要紧。然后在属性窗口中选择 optiongroup1 下面的 option2，然后将其放到与 option1 水平排列的地方，如不能精确调整它的位置，看一下初级教程第七课中的精确调整对象位置。

接着设置标题，将 option1 的 caption 设为“男”，option2 该设为 . . . . . ，设好如图 1。

图 1

将 optiongroup1 的 controlsource 设为 xb，为什么要这样，直接设为“性别”字段不行吗？不行！因为性别字段的值是逻辑型，而 optiongroup1 的值为数值型，因此要用一个数值型的变量来接收 optiongroup1 的选择值，然后用适当方法将其变换为逻辑型存入数据表。具体方法下面会讲。

⑵、设置 xb 的初始值，进入“编辑人员”(bjry)表单的修改，在其 load 事件中输入如下代码：

```
public xb
if 性别
xb = 1
else
xb = 2
endif
```

在这里先将 xb 设为公共变量，因为 load 事件程序是一个子程序，当这个程序运行完毕，其中所有的私有变量（即在本程序中创建的变量）都将释放，之后就再也找不到这个变量了，但这个变量在其它地方又要用，故将其设为公共变量

注意：这段代码不能放在 init 事件中，这是因为表单的 init 事件是在所有控件的 init 事件发生之后，而当控件的 init 事件发生时，即初始化时，就需要这个变量，而这时还没有，所以会发生找不到变量的错误，而 load 事件则是发生在所有事件之前，即表单启动的时候。

然后根据当前记录的性别字段设置 xb 的值，如果是“真”就设为 1，即“男”，否则为 2，即“女”，这样进入表单后选项按钮的黑点就会根据 xb 的值来显示。

注意：在需要逻辑表达式作判断时，如果是逻辑变量，因为其本身就是一个逻辑表达式，为“真”的话就直接写这个变量名，为假的话就写为“. not. 变量名”，而不要写成“变量名 = . t. ”或“变量名 = . f. ”

⑶、将“新增”按钮的 click 事件程序改为如下：

```
＊根据选项按钮所做的选择，将相应的值存入性别字段
if xb = 1
replace 性别 with . t.
else
replace 性别 with . f.
endif
append blank &&增加一条空记录
xb = 1 &&将 xb 设为 1
＊这是一个新记录，还不知道是男或女，因此一律设为 1，即“男”，作为初始值
thisform. text1. refresh &&将 text1 的内容刷新，下同
thisform. optiongroup1. refresh
thisform. text2. refresh
thisform. text4. refresh
thisform. text5. refresh
thisform. text6. refresh
thisform. text7. refresh
thisform. text8. refresh
thisform. edit1. refresh
```

```
thisform. text1. setfocus &&将焦点设到 text1
```

仔细看看与以前有什么不同

⑷、将“上一条”按钮的 click 事件程序改为如下：

```
skip  - 1 &&记录指针向上跳一行
* 以下一段程序检测记录指针是否到了开头,如果是给出提示,并把指针定位到第一个记录
if bof    &&假如指针已到开头
wait  已到开头  window nowait &&在右上角显示的提示,鼠标或键盘一动提示消失
go top &&将指针定位到第一个记录
endif &&假设结束
* 根据当前记录的性别字段设置 xb 的值,如果是“真”就设为 1,否则为 2
* 在需要逻辑表达式作判断时,如果是逻辑变量,因为其本身就是一个逻辑表达式,
* 为“真”的话就直接写这个变量名,为假的话就写为“. not.  变量名”,
* 而不要写成“变量名 = . t. ”或“变量名 = . f. ”
if 性别
xb = 1
else
xb = 2
endif
thisform. text1. refresh &&将 text1 的内容刷新,下同
thisform. optiongroup1. refresh
thisform. text2. refresh
thisform. text4. refresh
thisform. text5. refresh
thisform. text6. refresh
thisform. text7. refresh
thisform. text8. refresh
thisform. edit1. refresh
thisform. text1. setfocus &&将焦点设到 text1
```

“下一条”按钮也做相应改动,看看自己会不会做。

⑸、将“退出”按钮的 click 事件程序改为如下：

```
* 根据选项按钮组所做的选择,将相应的值存入性别字段
if xb = 1
replace 性别 with . t.
else
replace 性别 with . f.
endif
thisform. release &&本表单释放
```

14.3 用新的控件改进人事管理(二)

这一节讲用组合框选择输入部门和职务。之所以要用选择输入,是因为对于同一值不同人在不同时候输入是不一样的,比如"人事部门",有人可能会输入成"人事部",再加上无意识的输入错误,使得一个部门会有好几种说法,这将给查询、统计等带来很大问题,因此我们对于一个其内容只有有限个选择的字段的输入,最好采用选择输入。

从我们前面所讲过的控件中可以看出,复选框、选项按钮组、列表框、组合框都可以用作选择输入,那么时候该用哪一个控件呢?一般原则是:

1、对于具有"是"、"否"两种选择的,用复选框;

2、选项是固定的,并且选项不太多,可以用选项按钮组;

3、对于选项是可变的,并且选项不是非常多,可用列表框;

4、对于选项是可变的,并且选项非常多,可用组合框,因为选项非常多的情况下,用列表框找起来是很麻烦的,而组合框可以直接输入,只是在必要的时候才查一下。另外如果不希望该输入控件占太多地方,也可用组合框,因为它平时只占一行,拉下时才显示框,而列表框始终要占一块位置,当然您高兴也可以让列表框只占一行,但那样操作起来一定别扭。

要实现选择输入,就要将可选择的项目事先准备好,对于选项来说有两种情况,一种是固定的、一种是可变的。比如性别就是固定的,而部门就可能是不固定的,因为一个单位随时可以增减部门。

固定的选项可以在编程时编好,而不固定的一般就要有个数据表来存放选项,而且这个表可由使用的人任意增删和修改,下面我们就来讲讲怎样实现这种功能。

在人事档案数据库中建立一个数据表,就一个字段,字段名为"部门";

在菜单中加一项"维护",下面有一个子菜单项"部门字典"使用的命令是:

do form bmzd name bmzd

建立一个表单,界面如图 1;

图 1

在“新增”按钮的 click 事件中写入如下程序：

```
if this. caption = 新增  &&假如本按钮的标题为“新增”,表示第一次新增
this. caption = 继续新增  &&将本按钮标题设为“继续新增”
endif &&结束假设
select bmzd &&选择部门表
append blank &&加一空记录
thisform. grid1. readonly = . f.  &&将表格设为非只读,即可以修改
thisform. grid1. refresh &&表格刷新
thisform. grid1. setfocus &&将焦点放到表格上以利于输入
```

在“修改”按钮的 click 事件中写入如下程序：

```
if this. caption = 修改  &&如果本控件的标题是“修改”
this. caption = 结束修改  &&将本控件的标题设为“结束修改”
thisform. grid1. readonly = . f.  &&将表格设为非只读,即可以修改
else &&否则
this. caption = 修改  &&将本控件的标题设为“修改”
thisform. grid1. readonly = . t.  &&将表格设为非只读,即可以修改
endif
thisform. grid1. refresh &&表格刷新
thisform. grid1. setfocus &&将焦点放在表格上
```

在“删除”按钮的 click 事件中写入如下程序：

```
select bmzd &&选择部门字典
*虽说目前肯定是 bmzd 表,不用选也可以,但这是为了防止以后程序做了某种改动,
*造成当前表不是 bmzd 而使程序出错,这样就万无一失了。
delete &&删除当前记录
thisform. grid1. refresh &&表格刷新
thisform. grid1. setfocus &&将焦点放在表格上
```

在“恢复”按钮的 click 事件中写入如下程序：

```
select bmzd &&选择部门字典
*虽说目前肯定是 bmzd 表,不用选也可以,但这是为了防止以后程序做了某种改动,
*造成当前表不是 bmzd 而使程序出错,这样就万无一失了。
recall &&恢复当前记录
thisform. grid1. refresh &&表格刷新
thisform. grid1. setfocus &&将焦点放在表格上
```

在“退出”按钮的 click 事件中写入如下程序：

```
select bmzd &&选择部门表
locate for delete    &&查找是否有做了删除标记的记录
if found    &&假如找到
pack &&将做删除标记的记录彻底删除
endif &&结束假设
thisform. release &&关闭本表单
```

在表格下的 column1 下的 text1 控件的 lostfocus 事件中写入如下程序：

＊当处于新增状态时，即 command1 的标题为“继续新增”，

＊一旦光标离开本格，表示新增结束，就将表格设为只读，

＊同时将标题设回“新增”

```
if thisform.command1.caption = 继续新增
thisform.grid1.readonly = .t.
thisform.command1.caption = 新增
endif
```

存盘退出

这样用于输入和维护部门选项的子功能就做好了。

下面在编辑人员的表单中设置一组合框用于输入部门：

将原来用于输入部门的文本框删除；

换上一组合框，如图 2；

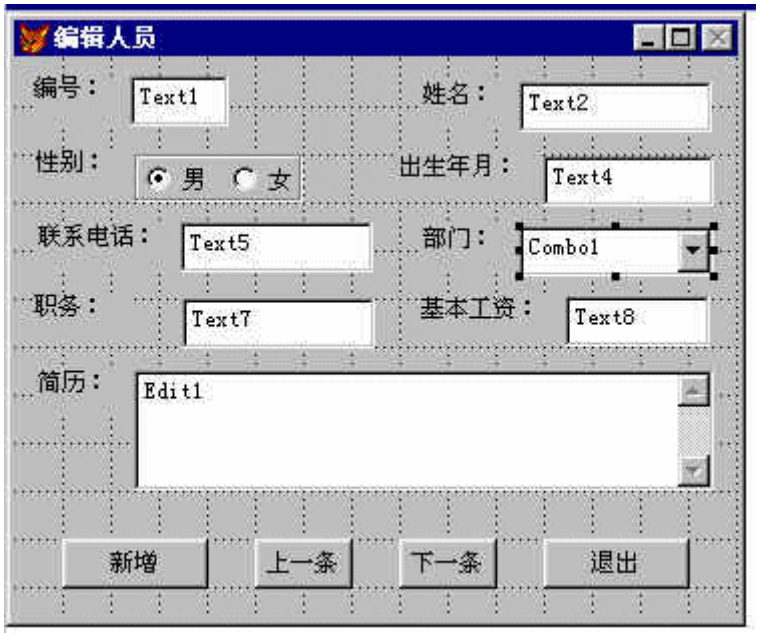


图 2

将组合框的 controlsource 设为“rsda.部门”；

将 rowsourcetype 设为“6 - 字段”；

将 rowsource 设为“bmzd.部门”；

重新设置所有控件的 tabindex，设置方法在第七课的第九条“设置控件的 tabindex 位置”中有讲解。

将菜单的初始化代码改为如下：

```
set talk off &&关闭命令响应
set safety off &&当覆盖磁盘上的文件时不提示，当程序编好后，不会错误覆盖文件
set date ansi &&设置日期为“年.月.日”方式
set century on &&设置年为 4 位数表示
```

use rsda &&打开人事档案数据表
select 0 &&选一空工作区（新增语句）
use bmzd &&打开部门表（新增语句）
dkda = . f.　&&设置一变量用于检测档案是否打开，真为打开，假为关闭

14.4　用新的控件改进人事管理（三）

在我们的人事管理软件中，您可能会感觉输入简历的地方小了，这一课我们就来讲利用页框对这一问题进行改进。

进入 bjry 表单的修改状态；

将表单拉大，把所有控件移到一边；

按工具栏上的页框按钮  ，在表单上做上页框，如图 1；

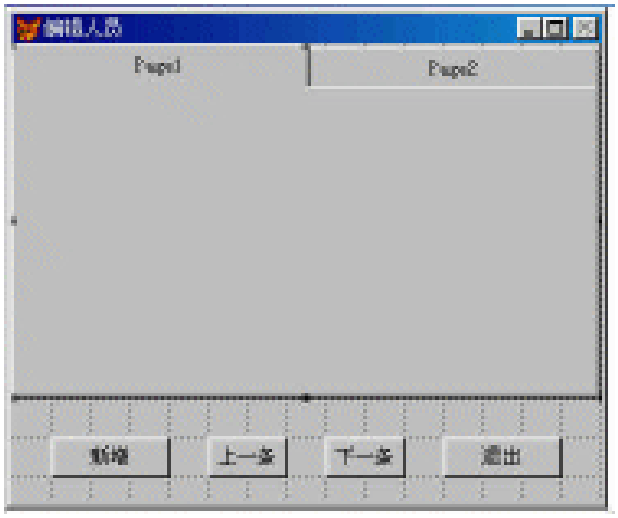


图 1

在属性窗口中选择 pageframe1 下的 page1，将其 caption 属性设为“基本情况”，再将 page2 的 caption 设为“简历”；

回到表单把除简历和按钮以外所有控件剪切，再选择 page1，将控件粘贴到 page1 中，如图 2，接着按同样方法把简历控件粘贴到 page2 中，如图 3，由于这个页面中只有一个控件，页标题已说明是“简历”，故“简历”标签可以删掉。

表单恢复原状；

将“新增”按钮的 click 事件程序改为如下：select rsda &&选择人事档案表
＊根据选项按钮所做的选择，将相应的值存入性别字段
if xb = 1
replace 性别 with . t.
else
replace 性别 with . f.

图 2

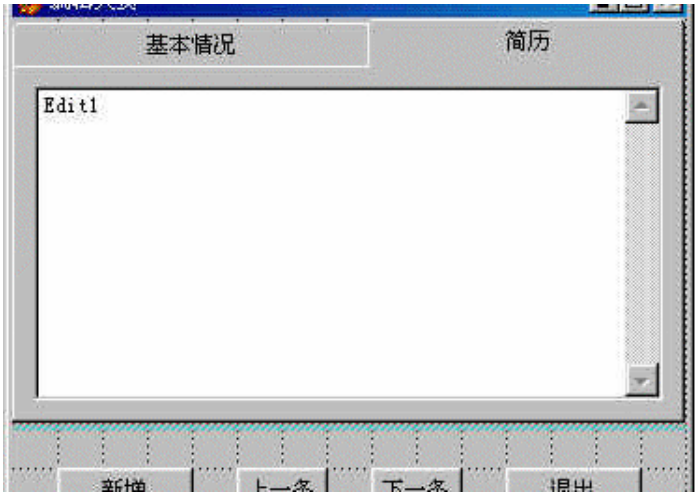


图 3

```
endif
append blank &&增加一条空记录
xb = 1 &&将 xb 设为 1
*这是一新记录,还不知道是男或女,因此一律设为 1,即"男",作为初始值
*将部门设为部门字典中的第一个记录,这样可防止部门没有选择而为空
select bmzd
go top
select rsda
replace 部门 with bmzd. 部门
thisform. pageframe1. page1. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page2. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page1. text1. setfocus &&将焦点设到 text1
```

将“上一条”按钮的 click 事件程序改为如下：skip - 1 &&记录指针向上跳一行
＊以下一段程序检测记录指针是否到了开头，如果是给出提示，
＊并把指针定位到第一个记录
if bof　&&假如指针已到开头
wait　已到开头　window nowait &&显示提示，鼠标或键盘一动提示消失
go top &&将指针定位到第一个记录
endif &&假设结束
＊根据当前记录的性别字段设置 xb 的值，如果是“真”就设为 1，否则为 2
＊在需要逻辑表达式作判断时，如果是逻辑变量，因为其本身就是一个逻辑表
＊达式，为“真”的话就直接写这个变量名，为假的话就写为“. not. 变量
＊名”，而不要写成“变量名 = . t. ”或“变量名 = . f. ”
if 性别
xb = 1
else
xb = 2
endif
thisform. pageframe1. page1. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page2. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page1. text1. setfocus &&将焦点设到 text1
将“下一条”按钮的 click 事件程序改为如下：skip &&记录指针向下跳一行
＊以下一段程序检测记录指针是否到了结尾，如果是给出提示，
＊并把指针定位到最后一个记录
if eof　&&假如指针已到结尾
wait　已到结尾　window nowait &&显示提示，鼠标或键盘一动提示消失
go bottom &&将指针定位到最后一个记录
endif &&假设结束
＊根据当前记录的性别字段设置 xb 的值，如果是“真”就设为 1，否则为 2
＊在需要逻辑表达式作判断时，如果是逻辑变量，因为其本身就是一个逻辑表
＊达式，为“真”的话就直接写这个变量名，为假的话就写为“. not. 变量
＊名”，而不要写成“变量名 = . t. ”或“变量名 = . f. ”
if 性别
xb = 1
else
xb = 2
endif
thisform. pageframe1. page1. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page2. refresh &&将页面及其控件的内容刷新
thisform. pageframe1. page1. text1. setfocus &&将焦点设到 text1

14.5 更多事件

激活事件 Activate

当激活表单、表单集或页对象时发生。

本事件与 init 事件不同，init 是在表单创建时发生，对于一个表单它只发生一次，而本事件是在表单成为当前使用的表单时发生，一般活动的表单其标题栏是蓝色的，而不活动的是灰色的，当用鼠标单击一个不活动的表单时就会激活一个表单，对于一个表单此事件可多次发生。

释放事件 Destroy

当释放一个对象时发生。比如用 thisform.release 关闭一个表单时。

应用于几乎所有对象。

按键事件 KeyPress

当用户按下并释放某个键时发生。

该事件会返回两个参数，nkeycode 用以标识被按下的键值，其键与 inkey 函数的返回值相同。nshiftaltctrl 标识所按下的控制键 shift、ctrl、alt ，其返回值分别为：

shift 1

ctrl 2

alt 4

根据这两个值便可知道所按下的是什么键，并可根据所按下的键执行相应的程序。

如果该控件是属于一控件数组中的一个控件，还会返回一个 nindex，标识其为数据中的第几个控件。

滚动事件 scrolled

在表格中移动滚动条时发生。

该事件将返回两个参数：

nindex：惟一标识控制数组中的某个控件。

ndirection：指定用户如何滚动表格控制中的内容。可能的参数有：

0 上箭头

1 下箭头

2 单击垂直滚动条滚动块上面的区域

3 单击垂直滚动条滚动块下面的区域

4 左箭头

5 右箭头

6 单击水平滚动条滚动块左边的区域

7 单击水平滚动条滚动块右边的区域

如果在程序代码中调用 doscroll 方法，也发生此事件。

应该避免在 scrolled 事件中制造等待状态（例如，wait window），因为当滚动表格时会出现屏幕

刷新,如果有等待可能会出问题。

到达事件 when

在控件接收焦点之前此事件发生。

该事件会返回一个参数 nindex ,惟一地标识控件数组中的某个控件。

此事件有点像 gotfocus,但该事件中的代码必须返回一个逻辑值,如果是"真",该控件可以接收焦点,否则不能收到焦点。

可以利用该事件使得在某些条件下一个控件不能接收焦点,比如这种情况,前一个控件输入了"不知道",那么下一个控件就不需要输入了。

对于列表框控件,每当用户单击列表中的项或用箭头键移动使焦点在项之间移动时,when 事件发生。

应用于所有控件。

行列变换后事件 AfterRowColChange

当用户将光标移到表格的另一行或列时发生。

这个事件会返回一个参数 ncollndex ,该参数为最新选择的行号或列号。

我们可以利用该事件显示一个记录的详细内容,比如有一个数据表有很多记录,用表格不能将所有记录在一屏显示出来,要移来移去才能看全,可以在表格中显示其主要的几个字段,然后用另外一些文本框显示其详细内容,每当表格的 aferrowcolchange 事件发生时就将其它的文本框刷新,这样当记录指针移到一个新记录时,文本框中的数据也会相应变化。

另外与此类似的还有一个 beforerowcolchange 事件,即在用户将光标移到表格的另一行或列之前发生。

获得焦点事件 GotFocus

当对象接收到焦点时发生。

该事件会返回一个参数 nindex ,该参数用以惟一标识控件数组中的某个控件。

应用于所有控件。

另有一事件为 lostfocus,即当对象失去焦点时发生。

鼠标按键事件 MouseDown

当用户按下一个鼠标键时发生。

该事件返回 5 个参数:

⑴ nindex 标识控件数组中的一个控件,仅当控件是控件数组的一部分时,才传送此参数;

⑵ nbutton 标识按的哪个键:1 (左),2 (右),4 (中);

⑶ nshift 标识控制键 (shift ctrl alt) 的状态

shift 1

ctrl 2

alt 4

⑷ nxcoord 存放鼠标指针的横坐标

⑸ nycoord 存放鼠标指针的纵坐标

与 click 事件不同,可以用此事件区别左、中、右键,还可以识是否有控制键与鼠标键组合。

应用于所有对象。

另外有以下一些相关事件：

mousemove：鼠标移动时。

mouseup：释放鼠标键时。

mousewheel：滚动鼠标轮时，有一种鼠标上面有一个可滚动的轮子。

时间到事件 timer

当计时器的计时时间到时此事件发生。也就是经过了 interval 属性中指定的毫秒数时。

该事件会返回一个参数 nindex，惟一标识控件数组中的一个控件。

删除事件 Deleted

当用户在表格的记录上做删除标记、清除删除标记，或者执行 delete 命令时，此事件发生。

这个事件会返回一个参数 nrecno ，为被删除的行记录号。

改变值事件 InteractiveChange

当用键盘或鼠标改变控件的值时发生。

在每次交互地更改对象时，都要发生该事件。例如，当用户在文本框中键入字符时，每一次击键都会触发该事件。

该事件会返回一个参数 nindex ，对于控件数组中的控件，指定唯一标识号。

应用于：

复选框，组合框，命令组，编辑框，列表框，选项组，微调，文本框。

改变尺寸事件 Resize

当调整对象大小时发生。

该事件会返回一个参数用于唯一标识控制数组中的控制。

我们可以利用此事件调整表单中对象的位置，使得表单大小调整时都可保持其中控件排列美观。

应用于：

列，容器，表单，表格，页框。

合法校验事件 valid

在控件失去焦点之前发生。

有点象 lostfocus，但不同的是该事件要求其中的程序最后要返回一个数据，可以是逻辑型的，如果返回的是“真”，则控件失去焦点，否则控件不会失去焦点。

这可以用来校验输入的值是否合法，如不合法，则光标不离开，直到输入合法值为止。例如如下程序：

```
if this. value  <18
wait window  输入值必须大于等于 18   nowait
return . f.
else
return . t.
endif
```

该程序的作用是，当控件（文本框、微调等）中输入的数值小于 18 时，光标不会离开，并提示输入不正确，如大于等于 18 则可以离开。

也可返回数值,对应于以下情况:

若返回 0,则控件不失去焦点,类似于上面的"假";

若返回正值,则该值指定焦点向前移动的控件数。例如返回 2,则焦点下移两个控件,而并非按正常跳到下一个控件;

若返回负值,则该值指定焦点向后移动的控件数。例如返回 - 1,则焦点上移一个控件,也就是回到进入当前控件之前的哪个控件。

应用这一点可以实现这样的功能,当输入某一值时,下一个控件不用输入了,而直接跳到后面的某个指定的控件上,至于跳到哪个就用返回的数值指定喽,或者输入了某个特殊值时回到上面某个控件重新输入。

14.6　更多方法

激活单元格方法 ActivateCell

将光标移到表格的某个单元格。

语法:

表格 . ActivateCell 行号 列号

参数:

行号、列号 :用于指定所要激活单元格所在的行和列。

备注:

有时为了做相对移动,即将光标移到当前所在行的下一行,或当前所在列的下一列,那么就要知道当前行或列的位置,用表格的 activecolumn 和 activerow 可获得。

移动方法 Move

移动一个对象。

参数:

左边界 :指定对象新位置的左边界位置。

上边界 :指定对象新位置的上边界位置。

宽度 :指定对象新的宽度。

高度 :指定对象新的高度。

重置时间方法 Reset

将计时器的计时时间重置为 0,也就是说让计时器重新开始计时。

语法:

计时器 . reset

滚动方法 DoScroll

使表格做滚动操作。

语法:

表格 . DoScroll 滚动方向

参数:

滚动就是用于确定滚动的方向(废话),可做的设置如下:

参数 方向

0 上一行

1 下一行

2 上一页

3 下一页

4 左移一列

5 右移一列

6 左移一页

7 右移一页

隐藏方法 Hide

将表单、表单集或工具栏隐藏起来,也就是将它们的 visible 属性设为 .f.。

语法:

对象.Hide

置点方法 PSet

在表单上画一个点。

语法:

表单.PSet 横坐标 纵坐标

参数:

横坐标 相对于表单左边的距离。

纵坐标 相对于表单上边的距离。

全部设置方法 SetAll

将一个容器对象(如表单)中的所有控件设置同一个属性。

语法:

容器.SetAll 属性名

参数:

容器 指定哪个容器中的控件要做设置。

属性名 指定控件的哪个属性要做设置。

值 指定属性要设为何值。

类 指定一个类名,以便对所有基于此类的控件做设置,不是基于此类的就不设置。

14.7 优化菜单

一、跳过

我们在使用 WINxx 软件时,经常可以看到它们的某些菜单项在某种情况下被屏蔽掉(字成为灰色),这时这些菜单项就不可用,这也叫跳过。

这是一个非常有用的功能,比如当一个数据表没有打开时,就不能执行查询的功能,这时如能

将“查询”菜单项屏蔽岂不很酷，那么 VFP 能否做到这一点呢？能！而且很容易，方法是：

进入菜单设计器；

找到需要屏蔽的菜单项；

按该项后的选项按钮参见图 1；

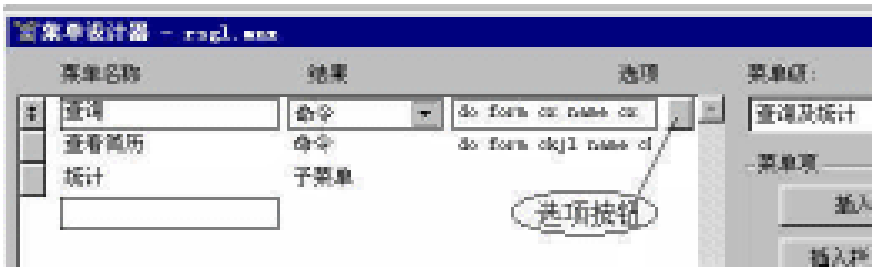


图 1

按后出现图 2，在“跳过”文本框中输入一个表达式，其值以后会是逻辑值，当此值为“真”时该菜单项就被屏蔽，比如图 3；

图 2

图 3

输入完后，确定，会看到选项按钮中有一个钩，表示选项中有内容，如图 4；

将所有菜单项的跳过逻辑表达式设好，重新生成新的菜单程序，关闭菜单设计器就行了。

当程序运行时，在启动菜单之前必须先为所有的跳过变量赋值，否则菜单启动后会发生变量找不到的错误，因为菜单程序会根据这些变量值的“真”、“假”来确定这些菜单项是否可用。

图 4

如果您希望菜单一启动某个菜单项就不可用，那么在菜单启动前就给相应的变量赋值“. T. ”，否则为“. F. ”。

在程序中什么时候要想让这个菜单项不可用，只需要将变量设为“真”即可，又想让它可用，再设为“假”。

比如我们可以给前面的人事档案程序的查询等菜单项设一跳过表达式：

. not. dkda

这里为什么要加个“. not. ”呢？因为在菜单的初始化代码中，该变量是设为“假”的，因为这时档案还没有打开，菜单一启动，查询功能应该不可以用，我们前面说过，当跳过表达式为“真”时不可用，这时 dkda 变量为“假”，我们又想“查询”不可用，于是就取其反面，加上一个“. not. ”，整个表达式的值就为真了，反之，当 dkda 为“真”时，即档案已打开，这时表达式就为假，因此“查询”就可以用了。

二、给菜单加说明

在许多的软件中我们都会看，当鼠标指到某个菜单项时，在下面的状态栏上可看到该菜单项的详细说明，这个功能是怎么实现的呢，方法如下：

进入菜单设计器；

找到要加说明的菜单项；

按选项按钮；

在“信息”一栏输入相应的说明，注意，字符要加引号，如图 5；

图 5

确定；

重新生成。

三、给菜单加分隔线

为了使菜单易于查看和调用，常常需要在菜单中加入分隔线，如图 6，下面讲做分隔线的方法：

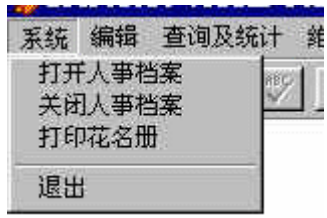


图 6

进入菜单设计器；

找到要做分隔线的地方，在菜单名称中输入“\-”，其它都不用变，如图 7。如果需要做分隔线的地方已经有菜单项了，那么就按“插入”插入一空栏再输入；

图 7

重新生成即可。

四、增加热键

如果要给某个菜单项增加热键，方法如下：

进入菜单设计器；

找到要加说明的菜单项；

按选项按钮；

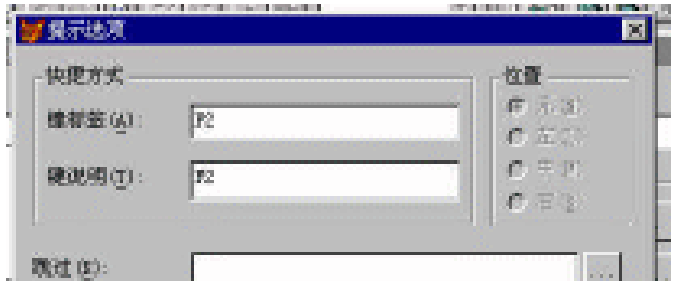


图 8

用鼠标点一下“快捷方式”的“键标签”；

按下您所需要的键，比如“F2”或者“Ctrl + A”，按子之后出现如图 8；

这时在键说明中也会出现与键标签中一样的内容，键说明是用于显示在菜单上以提示菜单项的热键是什么，如图 9。当然您也可以不要隐含的这个说明，而输入自己的说明，比如“ A”。

图 9

确定及重新生成，再去看看菜单有什么变化，是不是更酷了。

五、插入栏

很多朋友一定想在自己的软件中做出象 VFP 中的菜单中的一些功能，比如剪切、复制、粘贴等，这些功能如果自己编可就太麻烦，如能直接调用 VFP 的相应功能那就太好了，完全可以做到，方法是：

进入菜单设计器；

进入一个子菜单，注意这些功能是不能加在主菜单上的；

将光标放在适当位置，按“插入栏”按钮，如图 10；

图 10

这时会弹出一个插入栏选择框，找到您所要插入的功能，如图 11，然后按“插入”按钮；

插入后的效果如图 12；

剩下的事情不用我再说了吧。

这样您的菜单也就有了 VFP 的相应功能了，而且还可以使用它的快捷键，如 ctrl + x、ctrl + c 等等。

注 意：不能在您的菜单中禁止系统菜单，也就是不能用这个命令：

set sysmenu off

图 11

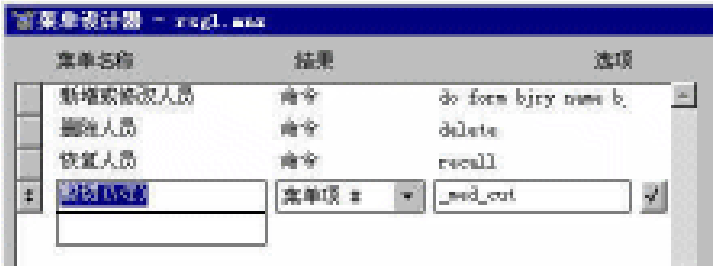


图 12

否则系统菜单中的功能就不能用了。如要不出现系统菜单应使用：

set sysmenu to

菜单设计器会自动在生成的程序中加入这一句，一般您不去动生成的程序（mpr）就不会有问题。

六、增加快捷键

这里所说的快捷键与上面所说的热键有什么不同呢？说起来要一大堆话，相信您看了图 13 就明白了，制作的方法为：

图 13

进入菜单设计器(老生常谈);

找到所要的菜单项,在菜单名称后加一个括号,括号中加一个您想要的字母或数字,在该字符前面加上"《"如图 14,如果是英文菜单,直接在菜单名称中找一个字符就行了。

图 14

好了。

按同样办法可以给一些控件加快捷键,比如要给一个按钮加上快捷键,用同样方法设置它的 caption 属性就行了,设好后如图 15,运行时按"Alt + Q"即可调用"退出"按钮。

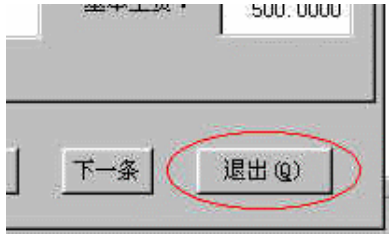


图 15

七、制作快捷菜单(弹出式菜单)

我们都知道在 WIN95/98 中可以按右键弹出一个快捷菜单,那么在 VFP 中怎样制作快捷菜单呢?方法如下:

在项目管理器的"其它"中选择"菜单",按"新建";

出现如图 16,按"快捷菜单";

图 16

出现与制作顶部下拉菜单类似的菜单设计器,这时可按制作下拉菜单同样的方法输入所需要的菜单项等内容;

制作完后同样需要"保存"、输入文件名、"生成";

调用的方法也与下拉菜单一样,即在需要的地方输入命令:do 快捷菜单名.mpr。比如您可以

在一表单或某个控件的“rightclick”事件输入这个命令，那么当您在表单或这个控件上按鼠标右键时就会弹出这个菜单如图 17。

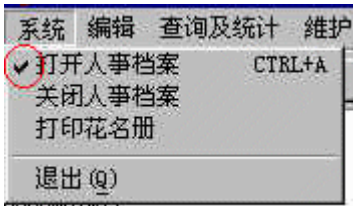


图 17

注 意：控件的 rightclick 事件优先于表单的 rightclick 事件，也就是说您把命令放在了表单的 rightclick 事件中，而没有放在控件中时，那么在这个控件上按右键是不会有作用的。

14.8　报表及标签

一、分类及总计报表的设计

将要分类的字段索引，比如您的报表要按“部门”分类，那么相应表中“部门”这个字段必须索引；

在项目管理器中选中“报表”；

按“新建”；

按“报表向导”；

选择“分组 / 总计报表向导”；

选择所需要的字段，比如选择“编号”、“姓名”、“部门”、“职务”、“基本工资”，下一步；

选择分组依据，即按哪个字段分组，最多可以选择三个分组字段，必须选了第一个才能先第二个，这里我们以“部门”作为第一分组依据，以“职务”作为第二个分组依据，即在同一部门中又按职务分小组，另外可以选择是否需要“总计”或“小计”，总计是指第一分组的合计，小计是指第二及第三分组的合计，选择好后，如图 1，下一步；

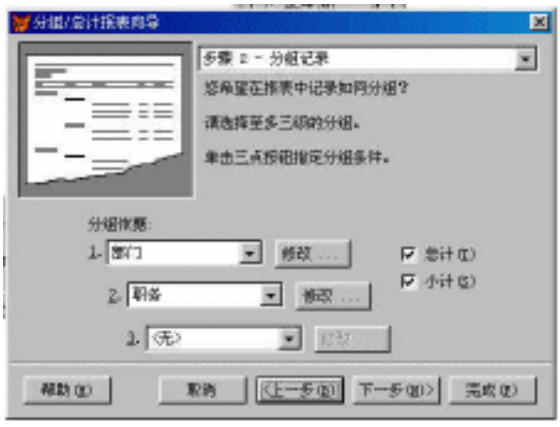


图 1

窍门：这里还可以对分组依据的字段进行一些修改，按依据字段后面的“修改”按钮，会弹出一个对话窗：

如果是分组字段是字符型的,可选择“整个字段”或“字段的前几个”字符做分组依据,一般情况下当然都是整个字段,那也就不用修改了,在某些情况下可能需要按前几个字符来分组,比如商品分类,可能会有 A-1、A-2、B-1...,如果我们希望按分类分组,而又不希望分得太细,即 A-1、A-2 等都算 A 类,分为一组,那么就可选起始第 1 个字符作为分组依据,否则 A-1 和 A-2 就会分在不同的两个组。

如果是数值型的字段,可按一定数量级间隔分组,比如按工资分组,如果不修改这个分组依据就会 110 是一组、120 也是一组...,如此可能不知会有多少组,只要数值不一样就是一组,这比不分组可能还糟,为此我们可以选“100s”作为分组依据,即每隔 100 为一组,200 和 299 都属一组。

如果是日期型,可选择按“星期”、“月”、“年”来分组。

选择在组中排序的字段,比如选择“编号”,最多可选三个字段,下一步;

选择报表样式,可选择“帐务式”,下一步;

输入报表标题,比如“部门工资统计表”,按“完成”;

输入报表文件名,比如“bmgztj”,当然还要选好适当的目录,然后“确定”。

在项目管理器中预览报表可看到类似图 2 情况。

部门工资统计

08/17/00

部门	职务	编号	姓名	基本工资
财务				
	会计			
		2	张丽	200.00
		3	莫琳	300.00
		17	余梅平	400.00
	分类汇总会计:	22		900.00
	经理			
		7	苏清	500.00
		12	庄稼	500.00
	分类汇总经理:	19		1,000.00
	分类汇总财务:	41		1,900.00
电脑				

图 2

注意:在 VFP6.0 中分组/总计报表和一般报表已合并了,您选择分组依据就是分组/总计报表,不选择就是一般报表。另外,不光可以求合计,还可以求平均值、最大值、最小值等等。

二、报表的调整

有时用报表向导做出的报表不能完全满足要求,需要手工对做好的报表进行调整。

首先需要在项目管理器中找到所要的报表,选择它,然后再按“修改”,进入报表设计器,在这里我们一般可以看到四个带区:

标题:如图 3 中的“人员花名册”,该带区的内容显示在报表第一页的开头。

页标头:该带区的内容显示在每页的开头。

细节:为报表的主要内容,紧接在页标头下打印,虽然我们在这里看到只有一行,但打印时却不一定只有一行,而是有多少条记录符合打印的条件就会有多少行,这里的一般都是变量名或字段名,也叫“域控件”(很难理解的名称),它们都有个方框框住,打印时会换为相应的内容。

页注脚:打印在每页的底部,一般用于打印页号等内容,在这里一般会看见一个域控件:

“页”+alltrim str _pageno

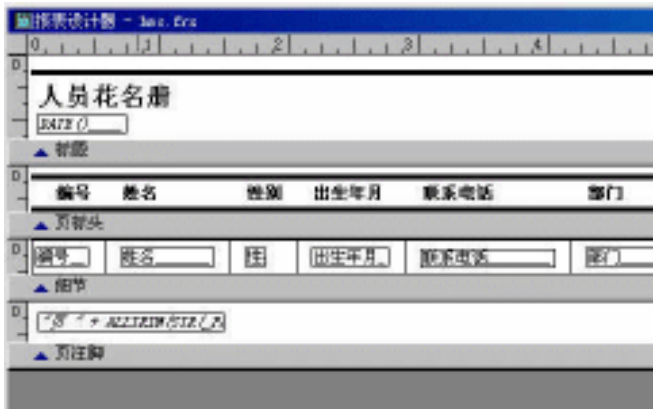


图 3

这就是由报表向导自动产生的页号。

在这里我们可以做如下的调整：

调整带区宽度。将鼠标放在带区上，鼠标光标会变为一个上下的箭头，这时按住鼠标左键可上下调整该带区的宽度。比如，一般向导所做的报表，其标题、页标头和页注脚都很窄，这样打出来的报表就会上、下将纸顶得很满，因此应把这些带区调高些，如图 4。

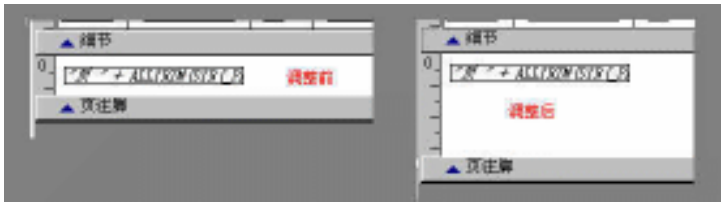


图 4

移动控件。报表上的所有东西（标签、域控件、线条、矩形等）都是可以移动，如果您觉得位置不满意，可按您自己的喜好把它们移来移去，就像设计表单一样。比如上面所说的为了不让报表把纸顶的太满，可将带区调高些，对页注脚调高就行了，但对标题和页标头，带区调高后还应将里面的内容往下移，才不会使上面顶的太满。另外页号在左边，这不符合我们的习惯，可将其移到右边。

窍门：除了将控件在带区中移动，还可以从一个带区移到另一个带区。西方的习惯也许是只在第一页打印报表的总标题，也就是那个标题带区，但我们中国人一般喜欢在每页上都有相同的标题（可能老外比较小气，那么一点纸也要节约），也就是说我们不需要标题，而只要页标头就行了，为此我们可以将页标头区加宽，将标题中的内容移到页标头中，不需要的线条可以删掉，标题区的内容全部去掉后，将其高度调到最窄。

修改控件。首先调出报表工具栏，在菜单的“显示 / 报表控件工具栏”，如要修改标签，比如标题，按下工具栏上的标签工具，再到要修改的标签上按一下，就可以对其进行修改了。

如要修改域控件，双击所要修改的域，会出现一个窗口，如图 5，在表达式中作需要的修改，例如，向导生成的页号也不符合我们习惯，什么“页 1”，这简直的就是老外说中国话，那么您就可以在表达式框中将其改为：

alltrim str _pageno　+ 页

或 第 + alltrim str _pageno　+ 页

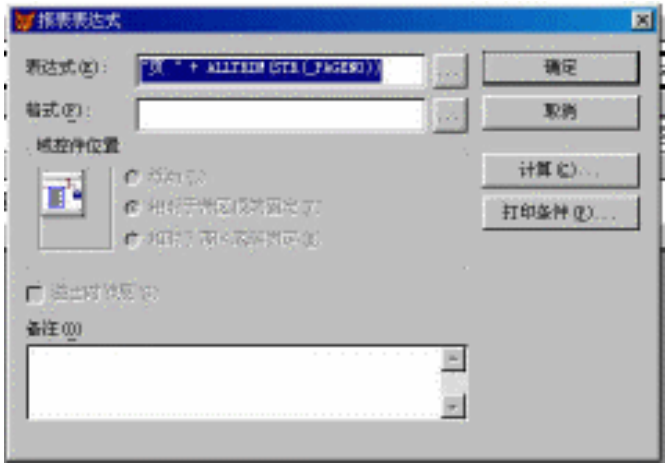


图 5

再要不,什么都不要,就是一个

alltrim str _pageno

14.9 查询与视图

一、查询

这里所说的查询与用 locate **命令查询记录不太一样,这里的查询是一个名词,它可以说是一个表的子表(或叫子集),即将一些符合某些条件的记录筛选出来形成一个子表,而且此子表可象一个表一样保存起来以供随时调用。**

一定会有人要问了:这样把子表保存起来,要是主表的内容改变了,那么子表不是与主表不符了,这时再看子表不是就是错的?

其实它并不是保存的子表,而保存的是形成子表的命令,换句话说,一个查询就是一个程序,打开查询就是运行这个程序,该程序将主表中的数据按一定条件筛选出来。那么您说上面的问题还存不存在?

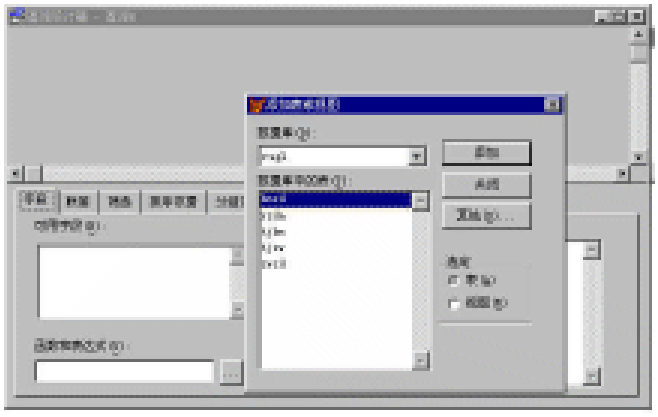


图 1

好了，下面我们就来看看怎样做一个查询，我们仍以人事管理软件为例，做一个专门显示电脑部门人员的查询：

首先打开查询所需要用到数据库，注意不是表，表可以先不打开；

在项目管理器中选择“数据 / 查询”，按“新建”，并选择“新建查询”（等您学会手工方式建立查询后，您应该会使用查询向导了）；

出现图 1，在其中选择所需的数据库及表，选好后按添加，即将需要的表添加到查询的数据环境中，可添加任意多个表，如果是自由表，按“其它”选择，添加后可在查询的数据环境中看见该表如图 2，添加完后按“关闭”，得到如图 3；

图 2

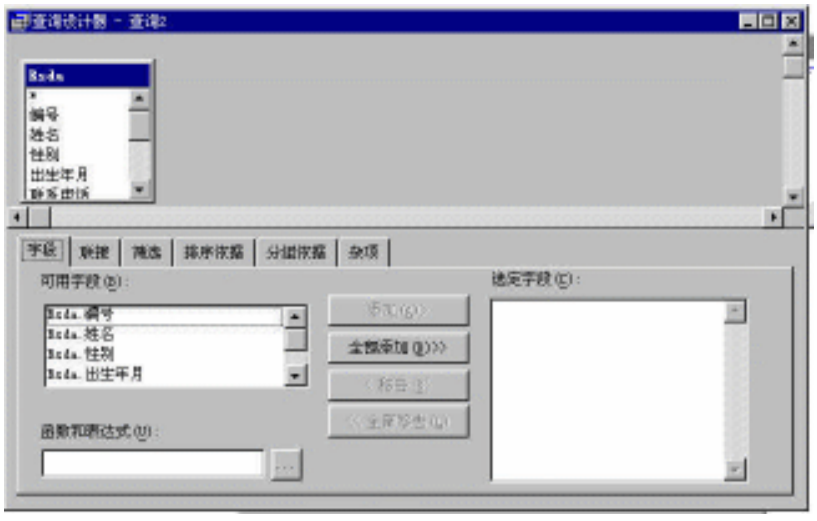


图 3

在“字段”页中选择查询所需要的字段，按“添加”，选择好后如图 4；

在“筛选”页中选择筛选的条件，选择好后如图 5；

这样一个简单的查询就做好了，在查询上按右键，选择“运行查询”，便可看到查询的结果；

图 4

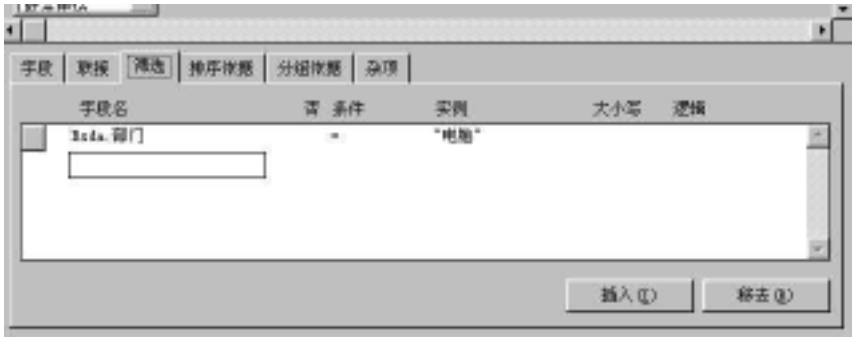


图 5

如没有问题,“保存”、“取文件名”,然后“关闭”。

在项目管理器中的“查询”下面可看到刚做好的查询名称,按“运行”即可看到查询的结果。我们前面讲过,查询实际上是一个程序,它被存在“xxx. qpr”文件中,实际上与“xxx. prg”文件是一样的格式,可在程序中用“do xxx. qpr”来调用它。

排序:您可以选择查询的结果按那个或那些字段排序,只需在“排序依据”中选择相应字段即可。

无重复记录 如果希望在查询结果中没有重复的记录,可在“杂项”中选择“无重复记录”。

显示排在前面的若干记录:一般情况下我们会要显示符合筛选条件的所有记录,但有时我们可能会只需要看排在前面的 10 个记录,比如找出分数最高的前 10 名,就可以按分数先排序,然后在“杂项”中将“全部”的钩去掉,在“记录个数”输入所要 10。还可以显示查询结果中 50% 的记录,知道怎么做吗?记住 如要选择前若干记录,必须有排序。

分组:如果要把一个字段中相同的内容做为了一个组,即显示一条记录,可以在分组中选择所需要的字段,但如果只是简单的分组,就和无重复记录差不多了,一般要结合统计函数使用,即在字段中输入“函数和表达式”,比如您可以选择“部门”字段,另外再输入一个表达式“sum rsda. 部门 ”,然后在分组中按“部门”分组,猜猜会出现什么结果。其它可用的函数还有 average min max count 等,不过这些函数只能在这儿用,不能在别的地方用,为什么?以后讲 SQL 语言时再讲。另外还可以按条件显示分组的信息。

查询结果的去向

一般情况查询结果是以一个临时表的方式来存放,并用 browse 显示出来。不过您还可以选择其它方式来存放结果,单击右键,选择“输出设置”,看到了吧。

二、多表查询

如果希望从多个表中获取数据来形成查询结果，可建立多表查询，比如人事管理的职务库(zwzd)中有个职务工资字段，而 rsda 中没有，我们就可以做个多表查询来看每个人的职务工资：

将所需要的表添加到数据环境。当您添加新表时，系统会让您建立与新表与原有表的联接，添加后如图 6，可以看到两个表的职务字段有一根线连在一起；

图 6

另外可在需要的时候将一个表的字段拖到另一个表的字段上形成联接。也可以双击那根线或在“联接”页中修改联接。

选择各表中所需要的字段；

其它的就和前面讲的差不多。

进一步

连接的方式有以下几种，各自的含义如下：

内部连接：只返回完全满足连接条件的记录；

左外部连接：返回前一个表中的所有记录以及后一个表中匹配的记录；

右外部连接：返回后一个表中的所有记录以及前一个表中匹配的记录；

全外部连接：返回两个表中的匹配和不匹配的所有记录。

三、视图

“视图”，又是一个难以理解的名词，这都是老外创造出来的名词 view，我们的翻译也就照着字面直译。其实视图和查询差不多的，不同在哪呢？我们知道查询的数据来源于一个或几个表，这些表称之为基表，“查询”就是对基表进行“查看”，您是不能通过它来修改基表的，而视图就是这一点不同，您可以在视图上修改数据，并将修改后的数据传回给基表。

既然视图和查询是基本一样的，我们就不多讲怎样来创建视图了，创建的方法也是差不多的，主要就来讲讲它们不同的地方。

进入视图设计器，我们可以看到与查询设计器不同的就是多了一个“更新条件”页面（那是当然的，因为视图可更新，而查询不可更新嘛），如图 7，在这里我们要做以下一些设置：

设置“发送 SQL 更新”，这样才使基表可更新，至于什么是 SQL，以后再讲；

设置每个表的“关键字段”，关键字段是用来使视图中的修改与基表中原始记录相匹配，必须设置了关键字段，该表才可修改，设置的方法为：在作为关键字段的字段名前面的钥匙处打上钩。每个表只能有一个关键字段；

设置“可修改字段”,在需要修改的字段名前面的铅笔处打上钩,一般说来关键字段应是不可修改的,但您一定要修改也可以;

“SQL WHERE 子句包括”又是什么意思呢?在多用户环境中,当要修改基表时,可能会有冲突,也就是您要改的时候,人家也要改,那么是不能两个人同时改一个数据的,当别人正在改一个数据时,您就不能改,怎样知道别人是否改了某个字段呢?这里就要为系统指定检测的方法,可以只检测“关键字段”,一般设置为“关键字段和已修改字段”。有人会问了,“关键字段和可更新字段”什么时候用呢?虽然它是可更新字段,但在实际中不是已修改字段,既然没修改就不用检测嘛。有时候会有这种情况,当一条记录中的某个字段改了,就不允许再改其它字段,这时候就有用了。

“使用更新”也就是更新方式,一般用“SQL UPDATE”,用“SQL DELETE 然后 INSERT”会使修改的记录跑到最后。

好了,这样一个视图就做好了。使用视图就和使用表一样,可以用 USE 将其打开,也可以使用其它命令象操作表一样来操作视图。

重要一点

当您在使用中打开视图并作了修改时,所修改的内容在您离开当前记录时自动地进入基表(这叫记录缓冲,关于缓冲我们以后再讲),如果您想在没有离开该记录时就更新基表,可使用如下命令(函数):

tableupdate

如果您想放弃对当前记录的修改,使视图回复原状,在没离开当前记录前,可使用如下命令(函数):

tablerevert

如果已离开了当前记录,那就不行了。

14.10 OLE 控件

OLE 控件又是个什么洋玩艺,先来解释一下 OLE,它是英文“Object Link and Enbed”的缩写,意思是对象的链接和嵌入。

这里所说的对象,与我们以前课程中所讲的对象又有所不同,不是指表单、文本框之类的东西,而是指 VFP 以外的东西,比如一幅图片、一段声音、一个 EXCEL 图表、一个 WORD 文件等等,我知道您这时一定很兴奋:难道在 VFP 的程序中也能放入这些东西吗?我的软件不就有了多媒体的功能,那岂不是帅呆了。

是的,的确是可以,下面我们就告诉您怎样做。

一、通用字段

VFP 的数据表有一个通用型字段,可以放各种各样其它格式的文件,比如图片、声音等,在这里就是以 OLE 的方式放入的。

将其它文件放入通用字段的命令是:

APPEND GENERAL 通用字段名

比如我们可以给人事档案数据表(rsda.dbf)加一个字段“照片”,将一个人的照片用扫描仪扫好,以文件名“庄稼.bmp”存放在当前目录中,打开 rsda 数据表,将记录指针走到“庄稼”这条记录上,然后执行如下命令就可以将庄稼的照片放在相应的记录中了:

append general 照片 from 庄稼 . bmp

具体到程序中可以在增添和修改人员的表单中增加一个文本框,在其中输入照片文件名,在该文本框的 lostfocus 事件中加入如下代码:

```
zpwjm = alltrim  this. value    &&将文本框的内容放入一个变量
if file  zpwjm    &&如果该文件存在
append general 照片 from &zpwjm  &&将文件放入照片字段
else  &&否则
messagebox  文件名不存在!  64  注意      &&给出提示
endif
```

如用删除通用字段中的内容,可用如下语句:

```
append general 通用字段名
```

多学一招

另外还可以交互式地将文件放入通用字段:

在浏览状态,将光标放在通用字段上,按"ctrl + pagedown"键,或者双击该字段,出现一表单,如图 1;

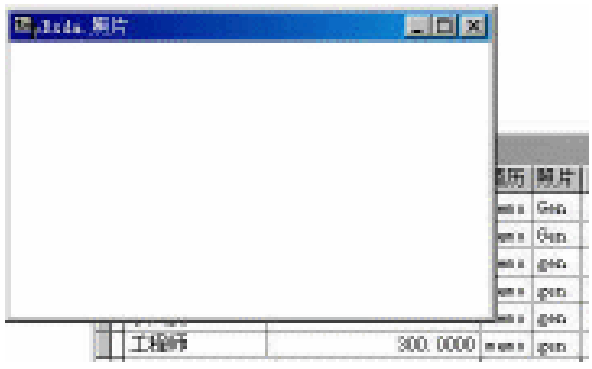


图 1

调用菜单上的"编辑 /插入对象",如图 2;

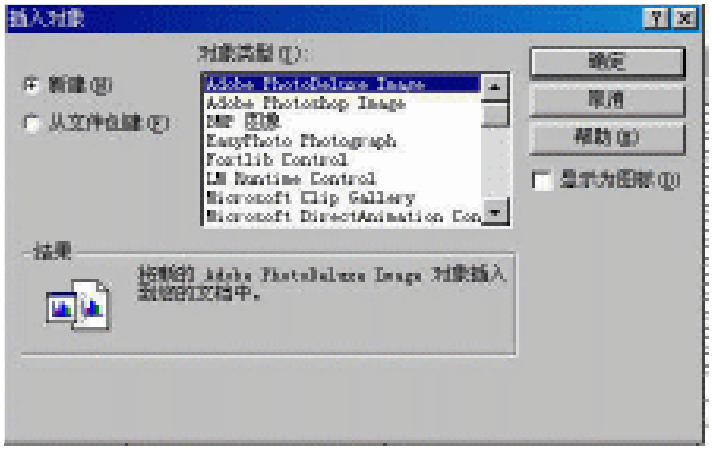


图 2

在出现的对话框中选择“从文件创建”；

输入文件名并确定，即可在显示通用字段的表单中看到插入的文件；

关闭该表单。

如要修改通用字段中的内容，可在第 2 步双击所看到的对象。

如要删除通用字段中的内容，可在第 2 步调用“编辑/清除”。

在第 3 步，您也可以选择“新建”，然后选择对象类型，比如 WORD，然后您就可以打入一篇文章，之后存放在这个通用字段中。

通用字段中如果有了内容，在浏览时可看到字段中显示“Gen”，字母以大写开头，否则是以小写开头。

二、链接和嵌入

链接和嵌入是两个概念，也就是说我们将 VFP 之外的对象放入 VFP 时，既可采用链接的方式，也可采用嵌入的方式，它们有什么不同呢？

嵌入：是指将对象真正放入了 VFP，或者说是复制了一份放进来。好处是如果它的数据源丢失了，它仍然还在；缺点是数据源发生了变化，它不会随之而变化，比如照片换了，数据表中的照片不会自动更换，除非调用命令重新加入才行。另一个缺点是会使表变得很庞大。

链接：没有将对象真正放入 VFP，而只是放了个地址进来，每次要看时，就到这个地址去取。显然，其优缺点与嵌入正好相反。

在第一点中我们采用的都是嵌入，如要链接，append general 跟“link”子句；交互式时，如果是从文件创建则在对话框中选择“链接”，如果是新建，因为直接是在通用字段中建立这个文件，不存在什么数据源，所以肯定是嵌入。

三、OLE 绑定型控件

所谓“绑定”，就是说该控件是与通用字段绑在一起的（好象看不出通用字段的意思嘛），也就是说它的数据源是一个通用字段的内容，我们可以将一个 OLE 绑定型控件放在表单以显示通用字段的内容，方法为：

进入表单设计器；

调 OLE 绑定型控件工具；

在表单上适当地方画出控件，就象设置一个 image 控件一样，如图 3；

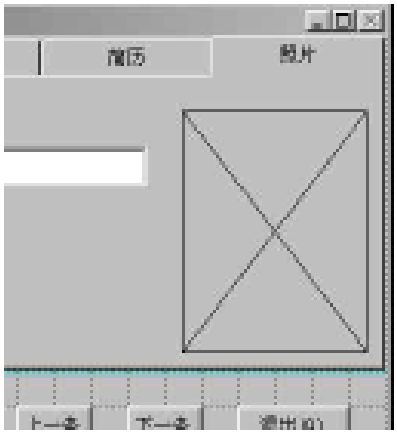


图 3

设置其 controlsource 属性为一通用字段的字段名，比如“rsda. 照片”，如图 4。

图 4

当运行表单后，我们就能看到所要的效果了，并且当表中记录指针移动时，表单中的照片也会随之变化。

四、OLE 容器型控件

类似于在通用字段中放入对象，当调用工具栏上的 OLE 容器型控件工具，在表单上拉出该控件后，会出现一个与在通用字段中插入对象类似的对话框，插入的方法也与在通用字段中一样，也就是直接将对象插入到表单中，而不象绑定型控件到通用字段中去取。

与在通用字段中插入对象时的对话框有一点不同，就是多了一个“插入控件”，如图 5，这是用于插入 ActiveX 控件的，关于 ActiveX 控件要到高级教程再讲了，这一课就此 Bye - Bye。

图 5

五、重要方法程序

OLE 控件有一个重要的方法程序，就是 doverb，该方法程序可以调用插入对象的宿主程序（如一般在电脑中 BMP 格式文件的宿主程序是画笔），对对象进行编辑、运行等。

语法：

Object. DoVerb

动作参数

指示 OLE 对象作怎样的动作。如果省略参数，将执行宿主程序并调用该对象，比如一个 BMP 图像，就调用画笔程序并将该图像打开，类似在表单上双击该 OLE 对象。

以下为标准的动作参数：

参数值 动作

0 执行隐含动作。

- 1 打开对象供编辑。如果程序支持在现场激活，则对象可在表单中打开，如 WODR、EXCEL 等都可以。

- 2 在另一个窗口中打开对象。

-3 如果是嵌入的对象,启动对象,并将宿主程序隐藏。这非常有用,比如表单上嵌入的是一个声音文件,可以用此参数启动声音播放器(如录音机),将声音播放出来,但录音机并不显示出来。

-4 假如对象支持现场激活,那么对象将在现场激活,并显示用户界面。如果对象不支持现场激活,将产生错误。

-5 假如将焦点放在了 OLE 控件上,将打开一个窗口使对象可编辑。如果对象不支持单击激活,则将产生错误。

-6 如果对象已进入编辑状态,可用此参数放弃所做的修改。

下面举例说明怎么使用这个方法:

您可以在表单上放入一个绑定型 OLE 控件,数据源为表中的一个通用字段,字段的内容为声音,比如可以是人事档案中人员自我介绍的讲话录音,加在表单上放入一按钮,按钮中写入如下语句:

```
thisoform. oleboundcontrol1. doverb  -3
```

这样按下此按钮就可以将当前记录中的声音放出来,并且不会显示出播放声音的软件。

14.11 网络版软件初步

一、原则

网络软件的最重要原则就是不能两个人同时修改一个数据。否则就会造成冲突,不知以谁的修改为准。

二、普通的解决办法

最普通的办法是在要修改数据时将数据表以独占方式打开,所谓"独占"就是使得数据只能自己使用,而别人不能使用,既不能修改,也不能查看。独占打开表的命令是在 use 命令中加 exclusive 子句,比如:

```
use rsda exclusive
```

这样就不会出现两个人同时修改一个数据的情况了。

三、共享数据

如果您只是查看数据,而不进行修改,那么您应该以共享方式打开表,否则别人也就没办法查看数据了,如果大家都是查看,不会有任何的冲突发生,即使查看的是同一个数据。

共享方式打开表的命令是 use ... share ,比如:

```
use rsda share
```

对于数据库也是一样的处理:

```
open database ... share
```

独占打开则是:

```
open database ... exclusive
```

四、友好的提示

如果您以独占方式打开了表,别人无论用独占或共享方式打开表都会出错,错误是"不能存取文件",错误代码是 1705 ;另外如果有人以共享方式打开了表,其它人试图以独占方式再打开,也会出现同样的错误。

如果在程序中出现这样的错误,可能使操作者莫名其妙,如果能给出比较友好的提示就好了,方法是用 on error 语句捕捉错误,一旦捕捉到错误即给出一个友好的提示,比如在需要共享打开数

据时：

on error　err = error

＊上面这条就是捕捉错误的陷阱，如在下面任何地方出现错误就将错误代码存入 err 变量，并不显示错误。

use rsda share　&&如果这时出现错误，错误代码就会存入 err，并且表不会打开，也不会出现错误提示。

if err = 1705　&&如果错误代码是 1705

＊给出自己的提示

messagebox　数据正在被其它人修改，您暂时不能使用，请稍候再试。　16　注意

endif

on error　&&取消捕捉错误

或者在需要独占打开数据时：

on error　err = error

use rsda exclusive　&&以独占方式打开表，如果有人已打开表则会出错，不论是独占还是共享。

if err = 1705

messagebox　数据正在被其它人使用，您暂时不能进行修改，请稍候再试。　16　注意

＊因为操作人员可能并不懂什么叫独占，而此时以独占方式打开表，肯定是要修改，故告诉他不能修改。

endif

on error

五、提高网络效率

学了以上的知识，您就可以编一个简单的网络软件了，但是这种软件的网络效率很低，为什么这么说呢？因为一旦有人独占数据，其它人就都不能看了，假如有几十个人要看这个数据，而该数据又被某人独占，可想而知其工作效率的低下。另外一个缺点就是可能始终都有人在以共享方式打开表，您根本没有机会对数据进行修改。

为了解决效率问题，因此引进“锁”的概念，锁的意思就是将数据锁上后，别人不能修改但可以查看，而且在别人查看时也可以对数据进行加锁修改，这样就解决了上面所说的问题。

给整个表加锁的命令为一个函数：

FLOCK

即给指定工作区或别名的表加锁，如果省略工作区或别名，则给当前工作区的表加锁。

该函数的返回值为逻辑值，如果返回“真”，代表加锁成功，如果“假”，则不成功。假如加锁不成功，表明有其它人已经锁住了该文件，目前您不能对其进行加锁，也就是不能修改其中的内容，但您可以查看。

讲到加锁就不能不讲一下这个命令：

SET REPROCESS TO 次数 COND TO AUTOMATIC

这个命令的意思是设置尝试加锁的次数或时间。如果 set reporcess to 1，那么当执行 flock 时，如果不成功立即就返回 . f. ，如果设置为 5，那么系统会连续尝试锁 5 次，如果 5 次都不成功加锁函数才返回 . f. ，如果跟上 SECOND 子句，则会连续加锁 5 秒钟。

隐含设置为 set reporcess to 0，系统将会无限次加锁，至到成功为止，并返回 . t. ，期间可按 ESC

键中止尝试,则返回 .f. ,set reporcess to automatic 也是同样的意思。

如果设置 set reporcess to -1 ,那么不但是无限次加锁尝试,而且不能按 ESC 中止,这一般都是不会用到的。

在程序中我们一般只需要试 1 次就行了,故一般在程序开头设置 set reporcess to 1。

一个典型的加锁程序如下:

```
select rsda
jscg = flock
if jscg
*进入数据的修改处理
else
messagebox 其它人正在修改数据,您不能修改,请稍候再试。 16 注意
endif
```

数据修改完后应尽快将锁解除,以便他人能够进行数据修改,解锁的命令为:

```
UNLOCK
```

注意:那么是不是说有了锁,独占打开表的方式就没有用了?不是的,有些操作必须要独占整个表,比如这些命令:INSERT INSERT BLANK MODIFY STRUCTURE PACK REINDEX and ZAP。因为它们会使表中的所有记录发生改变或者使表的结构发生改变,那么此时其它人是不能查看数据的。当您的表不是以独占方式打开,系统将会拒绝执行这些命令。

14.12 RUSHMORE 技术

一、概念

进入 FOXPRO 经常会碰到 RUSHMORE 这个东东,那么什么是 RUSHMORE 呢?我们都知道利用索引来查询数据会快得多,比如在建立了索引的字段中查询,seek 就比 locate for 要快得多,这就是因为 seek 利用了索引查询,而 locate for 没有。

那么能不能让 locate for 也能利用索引加快查询的速度呢?RUSHMORE 技术就是起这个作用的。不光是 locate for,其实凡是带 for 的命令都要做查询,比如:count for 就是统计有多少符合条件的记录,那么自然就要将符合条件的记录查询出来。而 RUSHMORE 技术可以让所有带 for 的命令加快执行速度,所以也叫优化查询。

小知识:为什么利用索引查询会比不利用查询快呢?一般查询是按顺序查询,比如一个表中有 1000 条记录,符合条件的记录在第 750 条,一般的查询就从第 1 条记录开始,一条条比较是否符合条件,直到找到符合条件的为止,很显示它必须进行 750 次比较。如果记录是按从小到大按顺序排列的,那么就可以不按顺序来查,可以先查第 500 条,如果需要找的记录比第 500 条大,那么符合条件的一定在 500 - 1000 之间,按同样方法再从 500 - 1000 的中间去查,就可找到 750 条,而 750 条正是要找的,因此只需两次便能找到需要的记录。

二、使用

别以为 RUSHMORE 这个词看起来深奥难懂,其实使用起来并不是很难,一般说来您只需要将有关的字段建立了索引就行,当执行有 for 的命令时,FOXPRO 系统会自动根据索引去查询符合条件的记录,您不需做任何事,连将所需索引设为主索引都不需要,当然如果不是结构化复合索引,您

必须在查询前将索引文件打开。

三、不能使用 RUSHMORE 的情况

如果您在命令中用了范围子句,可能 RUSHMORE 会不起作用。我们知道范围子句有四种:all rest next record n (n 用于指定一记录号),其中 all 和 rest 不会影响 RUSHMORE,但如果使用了 next 和 record n,则 RUSHMORE 将不起作用。

如果建立索引时使用了 for 子句,则这个索引不能被 RUSHMORE 所采用,比如:

index on 姓名 for 基本工资> 200 tag gz

这个索引就对 RUSHMORE 没有任何帮助。

索引表达式中如果使用了 . not. ,那么这个索引也不能被 RUSHMORE 使用,比如:

index on . not. delete tag notdel

遇上这种情况您可以采用 set delete on ,然后再使用不带 . not. 的索引表达式。

当命令中使用了 while 子句。

如果您的电脑内存比较少,RUSHMORE 可能也不能起作用。不过我想现在的电脑很少会内存较少的。

对于一个已经按某个索引排序了的表,不能使用 RUSHMORE ,所以想要使用 RUSHMORE 发出命令前最好用 set order to 或 set order to 0 命令将索引设置在不对表进行控制的状态下,您不必一定要把索引关闭,对于复合索引也不可能关闭。如果一定要按某种顺序排列记录,可使用 sort 命令对记录进行实际的排序,当然这要权衡利弊了,因为 sort 命令也是很花时间的,如果为了查询时节省一点点时间,而花了大量的时间去排序,那就划不来了。假如排序一次后会大量地进行查询,那么还差不多。

当然,不能使用 RUSHMORE 并不会影响命令的正常执行,只是速度会慢一些,如果是一个很大的表,比如几百万条记录,可能会慢很多。

四、不能使用 RUSHMORE 的表达式

并不是所有的 FOR 后面跟的表达式都可以使用 RUSHMORE 来加快查询的速度,以下的表达式就不能使用 RUSHMORE 来优化:

带有 isblank 和 empty 函数的表达式是不能优化的,比如:

locate for empty 姓名

带有包含运算符 $ 的表达式。

locate for 稼 $姓名

表达式必须与索引表达式精确匹配,比如:

index on upper cu_name tag name &&索引表达式是 upper cu_name

locate for cu_name = ACME &&不能优化,因为 cu_name 与 upper cu_name 不是完全一样的

locate for upper cu_name = ACME &&则可以优化

五、组合表达式的优化

如果是两个表达式用逻辑运算符 . or. 或 . and. 组合起来,情况就比较复杂了,下表对这种情况做了说明:

序号 表达式 1 操作符 表达式 2 查询结果

1 可优化 . and. 可优化 全部可优化

2 可优化 . or. 可优化 全部可优化

3 可优化 . and. 不可优化 部分可优化

4 可优化 . or. 不可优化 不可优化

5 不可优化 . and. 不可优化 不可优化

6 不可优化 . or. 不可优化 不可优化

7 - . not. 可优化 全部可优化

8 - . not. 不可优化 不可优化

比如：

locate for 姓名 = 庄稼 . and. 日期 <{12 /30 /96} &&是全部可优化的,符合第 1 条

locate for 姓名 = 庄稼 . or. 稼 $姓名 &&不可优化,符合第 4 条

locate for . not. 姓名 = 庄稼 &&全部可优化,符合第 7 条

locate for . not. 稼 $姓名 &&不可优化,符合第 8 条

六、关闭 RUSHMORE

在有些极特殊的情况下 RUSHMORE 可能导致命令执行结果不正确,比如使用 RUSHMORE 的命令在执行过程中修改了 for 子句中的索引关键字，当然这种情况是很少发生的，如果有这种情况,就应关闭 RUSHMORE 功能,使整个 VFP 系统关闭 RUSHMORE ,可用如下命令：

```
set optimize off
```

如要打开则是：

```
set optimize on
```

如果只是关闭某一条命令的 RUSHMORE 优化功能,可在该命令后加 nooptimize 子句,例如：

```
locate for 姓名 = 庄稼 nooptimize
```

七、看看 RUSHMORE 的威力

运行这个程序：rushmore. prg,您就会知道 RUSHMORE 的利害了,源程序如下：

```
set talk off
set safety off
close table all
clear
＊设置隐含路径
DQML = SYS 5 + SYS 2003
CXLJ = SYS 16
FOR JSQ = 1 TO LEN CXLJ
CXZF = LEFT RIGHT CXLJ JSQ 1
IF CXZF = ’
CXLJ = STUFF CXLJ LEN CXLJ - JSQ + 1 JSQ
EXIT
ENDIF
ENDFOR
SET DEFAULT TO &CXLJ
if file sjb. dbf &&假如 sjb. dbf 存在
use sjb index sjb &&打开 sjb. dbf 及索引
set order to 0 &&将表设为不按索引排序,但保持索引打开
else &&如果 sjb. dbf 不存在
```

```
create table sjb  name c 10   &&创建 sjb. dbf
index on name to sjb &&建立索引
set order to 0 &&将表设为不按索引排序,但保持索引打开
endif
if reccount   <300000 &&假如表中没有三十万条记录
@2 5 say  正在生成数据表,请稍候 . . .
zap &&清空表
*在表中添加三十万条相同的记录
ks = second   &&记录开始时间
for jsq = 1 to 300000
append blank
replace name with  庄稼
dq = second   &&记录当前时间
sysj =  dq - ks  / jsq * 300000 -  dq - ks  &&计算剩余时间
if mod  jsq  5000  = 0 &&如 jsq 正好能被 1000 整除,即每隔 1000 显示进度及剩余时间
@2 30 say str  jsq / 300000 * 100  3  +  %  &&显示增加的进度
@2 40 say  剩余时间: + str  sysj  4  +  秒  &&显示剩余时间
endif
  endfor
*将第一百、二十九万条记录改为另一值
go 100
replace name with  庄愚
go 290000
replace name with  庄愚
endif
@4 5 say  正在执行统计操作,请稍候 . . .
set optimize off &&关闭 RUSHMORE 优化
ks = second   &&记录操作开始时间
count for name =  庄愚  to jg &&进行统计操作
js = second   &&记录操作结束时间
@4 5 say  不使用 RUSHMORE : + str  js - ks  5  3  +  秒,统计结果为: + str  jg  2  &&显示
操作所用时间
set optimize on &&打开 RUSHMORE 优化
ks = second
count for name =  庄愚  to jg
js = second
@6 5 say  使用 RUSHMORE : + str  js - ks  5  3  +  秒,统计结果为: + str  jg  2
```

14.13 数组在应用

数组实际上就是一组变量，它们有同样的名称，但有不同的下标，也可以叫编号，比如：a 1 、a 2 、a 3 ……，这些变量叫做数组的元素。

这些变量可以同一般变量一样使用，比如：

```
a 1 = 庄稼
replace 姓名 with a 1
```

那么用数组有什么好处呢？当我们要对一组变量进行处理时，就会体现出它的好处了，比如我们想将某个数据表中连续的若干记录中某个字段的内容放入变量，如果不用数组程序就会是这样

```
a1 = 姓名
skip
a2 = 姓名
skip
a3 = 姓名
skip
. . .
```

显然这是很麻烦的，尤其需要的变量很多时，而最主要的是，如果变量的个数是不定的，那就几乎难以实现。

而用数组就可以解决问题，假设需要 100 个变量，程序就可以是这样：

```
dimension a 100    &&定义数组的个数
for counter = 1 to 100
a counter  = 姓名
skip
endfor
```

＊counter 根据循环从 1 到 100，即完成 a 1 = 姓名、a 2 = 姓名、……、a 100 = 姓名。

＊并且每循环一次记录向下跳一个，也就将连续 100 个记录的“姓名”字段的内容赋给了 100 个变量。

而代表数组个数的 100 也可以用变量，即：

```
dimension a ac    &&定义数组的个数
for counter = 1 to ac
a counter  = 姓名
skip
endfor
```

这样只要在每次定义数组个数前给 ac 不同的值就可以实现数组个数是可变的了。

注意：从上面的程序中可以看出，数组在使用之前必须定义，且要说明数组中元素的个数。定义的命令是：

```
DIMENSION 数组名 1 元素个数  数组名 2 元素个数  . . .
```

二维数组

我们上面讲到的是一维数组，即一个元素由一个下标来确定，那么二维数组就是由两个下标来确定一个元素，比如：a 1 1 、a 1 2 、a 1 2 、a 2 1 、a 2 2 ……。

二维数组有什么好处呢？比如还是上面的例子，但我们需要把“姓名”和“电话”都放到变量中，我们就可以这样来做：

```
dimension a 100 2    &&定义数组的个数
for counter = 1 to 100
a counter 1  = 姓名
a counter 2  = 电话
skip
endfor
```

如果不用二维数组而用一维数组，那做起来会很麻烦，不信您自己试试看，另外后面使用起来也很不方便。

实际上上面这个二维数组就相当于一个具有 2 列、100 行的表，a 1 1 是第一个人的姓名、a 1 2 是第一个人的电话、a 2 1 是第二个人的姓名、……，使用起来非常直观。

14.14　子程序和函数

当我们编写一个软件的时候，通常都会遇到这样的情况：同一段程序在不同的地方要执行很多次。虽然 WINDOWS 为我们提供了”复制”，”粘贴”的功能，但是如果您不想被人家笑是个外行的话，我建议您还是把它们写成子程序的形式，也就是我们通常用到的扩展名为 PRG 的程序文件，而不要去”复制”+ ”粘贴”。更重要的是一味的复制大量的重复代码会使你的程序冗长、繁索，大大降低程序的性能和可读性。而使用子程序则会提高程序的利用率，缩短开发周期。

编写一个子程序文件非常简单，只要我们：

打开项目管理器

翻到“代码”页

选中“程序”

按下“新建”按钮

就一切 OK 了。看看下面的图你就会知道这是多么的容易。

看到“程序 1”哪个子窗口了吗？那就是我们要输入代码的地方，只不过是我为了减小图形，所以把它给缩小了。

好了，让我们动手来写我们的第一个子程序。这个子程序非常的简单，只是显示一个小窗口，上面来上那么一句“你好 VFP”。让我们在“程序 1”的子窗口写下以下的代码：

```
MESSAGEBOX“HELLO VFP”   &&显示提示窗口
RETURN  &&返回主程序
```

搞定！让我们保存一下，以便可以在以后的程序中调用它。单击右上角的小 X，如果 VFP 发现你对子程序进行了修改，它就会问你是否保存子程序的修改，选”是”并给它起个名字，就象用 WORD 保存一篇文档一样。VFP 会自动为它加上 . PRG 为扩展名的，这一点不用我们费心。假设我们存为 HELLO. PRG。

这里的的 MESSAGEBOX　函数用来显示 HELLO VFP 窗口，RETURN 命令用来返回调用的子

程序的主程序，当子程序执行到 RETURN 语句时返回到子程序调用语句的下一条语句继续执行。

小心：要是你把代码写到 RETURN 的后面的话，是不会被执行到的。每个子程序至少有一个 RETURN 语句。

保存好子序程后，让我们在命令窗口中用 DO 命令来调用它。直接在命令窗口中写入以下的命令：

DO HELLO

这时你就会看到一个小窗户跳出来，上面写着“HELLO VFP”，当然你也可以在你的程序中的任何地方用

DO HELLO

来调用这个程序，比如在一个命令按钮的 click 事件中，或在一个表单的 init 事件中。

到此为止，我们这个简单的 HELLO VFP 子程序就全部做完了。怎么样，是不是很容易？

带参数的子程序

在实际的编写软件的过程中我们经常会遇到这样的事情，子程序的某些地方要随着不同的情况要灵活的改变，就象我们刚才编写的”HELLO VFP”程序中，我们可能会要求它不要老是显示出’HELLO VFP’这句只同 VFP 打招呼的话，而是根据我们的需要显示非富多彩的提示信息，比如显示个’吃了吗？”之类的问候。可我们又不想建那么多的 PRG 文件。这时我们就要引入子程序中的’参数’的概念。

“参数”简单的说就是我们要向子程序传递的信息。它即可以是一个字符串，也可以是一个整数。下面我们就来编写一个带有参数的子程序，看过之后我不说，您也能知道参数是干什么用的了。

按照上面讲过的建一个新的子程序文件，取个名字为 NEWHELLO.PRG。在它里面写入下面的代码：

```
PARAMETERS C_HELLO &&用来接收参数。
MESSAGEBOX  C_HELLO
RETURN
```

参数用 PARAMENTERS 引导，若多于一个参数，则参数之间用逗号分开，这些东东称为形式参数，因为它们只是一个参与运算的符号，只是一个形式的量，所以叫做形式参数。

下面我们来调用这个新的子程序。在命令窗口敲入下面的代码

DO NEWHELLO WITH 吃了吗？

这时你就会看到屏上又跳出一个小窗口，不同的是这次的问候语变成“吃了吗？”这句了。我们再在命令窗口写入下面的代码：

DO NETHELLO WITH 还没呢，您想请我搓一顿儿呀！

又跳出一个小窗口，问候语又变成“还没呢，您想请我搓一顿儿呀！”这句了，有意思吧？

WITH 关键字指出将要传递给子程序的参数，PARAMETERS 定义的形式参数接收的就是它的值。

小心：要强调的是形式参数和实际的个数必须一致。就好比有两个小伙子，而只分配给他们一个姑娘，不打架才怪；其次是二者的顺序必须一致，最后二者对应参数的类型必须一致。否则的话。嘿，我不说你也知道。

子程序还有一项很好的功能，它可以通过参数从子程中返回值，然后在程序中的其它地方便可以引用这个参数（或叫变量）进行其它的运算。为了更好的说明我们现举一个例子，这回我们不用 HELLO 哪个例子了，再用它的话，说不定就该让我们点票子，下馆子了。

我们举一个查询的例子来说明返回值的用处。

按照上面讲的过程建一个新的子程序文件，取个名字为 MYLOCATE. PRG。在它里面写入下面的代码：

```
PARAMENTERS TJ    &&定义形式参数 TJ
USE 人事管理   &&打开人事管理表
GO TOP   &&使指针指向首记录
LOCATE FOR NAME = TJ   &&开始查找
IF EOF
RETURN . F.   &&如没找到就返回 . F.
ELSE
RETURN . T.   &&如找到就返回 . T.
ENDIF
```

用 RETURN 语句，后面跟上一个表达式，这样我们就可以把一个值返回。如果不指定表达式，则返回真值 . t.

下面我们来调用它，新建一个程序：

```
C_NAME = 庄稼    &&设定要查询的名字
DO MYLOCATE WITH C_NAME TO C_RETURN &&调用子程序
 C_RETURN  &&显示查询结果
```

用 TO 关键字来接收子程序的返回值，如果找到名字为“庄稼”的员工就会在屏上显示 . T. ，否则的话就显示 . F.

过程文件

也许有的细心的读者会问：“如果是一个大型的软件要用很多的子程序，岂不是要建很多的 PRG 文件？”。问的好。对于较大型的程序，太多的 PRG 子程序即不便于管理，还会由于每个子程序文件做为单独的文件来处理，这样会带来加载文件时的开销。这就要引入 ’过程 ’概念。

其实过程和子程序是一回事儿，只不过是把多个子程序保存在一个 PRG 文件中。在 PRG 文件里每个子程序单独定义罢了。而保存多个子程序的 PRG 文件我们称为 ’过程文件 ”。

在过程文件写子程序不能象我们单独建立子程序时一样直接就开始写代码。那样的话分不清哪段代码属于哪段子程序了。要先对子程序进行定义，也就是定义一个过程。

过程的定义形式是：

```
PROCEDURE 过程名
RAMETERS PARS1ARS2….
过程体
RETURN TO
```

PROCEDURE 是标识过程定义的关键字，“过程名 ”是标识此过程的名字，它就象人的名字一样，只不过人的名字是用来区分人的而“过程名 ”用来标识标识过程的。但有一点“过程名 ”与人名不太一样，人的名字可以重复，虽然也会出麻烦但问题不是很大，“过程名 ’就不一样了，它要求必须是唯一的，最好还能表达过程的主要功能。电脑可没有那么聪明会区分两个同名的过程。

PARAMETERS 和 RETURN 的用法和我们讲过的子程序的用法一样,就不多唠叨了。

调用过程的方法也和调用子程序的方法一样。但是要注意一点,在调用过程时要指出包含在哪个过程文件中。方法有两个:

一、指出定义过程的过程文件。

二、必须在调用前加上一个语句:SET PROCEDURE TO 过程文件名(带扩展名)

它打开指定的过程文件,该文件中的过程就可以直接用 DO 过程名 调用了。

讲完了过程以后我们就可以把前面的 NEWHELLO 子程序和 MYLOCATE 子程序和并到一个过程文件里了。我们用 MYPRG.PRG 来保存。

```
*****MYPRG.PRG*****
PROCEDURE NEWHELLO  &&定义第一个过程
PARAMETERS C_HELLO &&用来接收参数。
MESSAGEBOX  C_HELLO
RETURN
PROCDEURE MYQUERY && 定义第二个过程
PARAMENTERS TJ &&形式参数
USE 人事管理
GO TOP
LOCATE FOR NAME  =  TJ
IF EOF
RETURN . F.
ELSE
RETURN . T.
ENDIF
```

我们调用时用下面的形式调用就行了.

```
SET PROCEDURE TO MYPRG.PRG  &&将过程文件打开
DO NEWHELLO WITH "吃了吗 "
DO MYLOCATE WITH "庄稼" TO TC
? TC
```

函数

说完了子程序和过程 咱们再来说说函数。其实函数和过程是十分相似的,只有一点小小的差别。(注意:我们这里讲的是自定义函数,而不是系统的函数。因为这些函数都是我们根据自己的需要,由自己动手定义和编写的,所以叫做自定义函数。)

首先在定义时所用的关建字不是相同的。函数的定义形式如下:

函数定义形式是:

```
FUNCTION 函数名
RAMETERS 参数表
函数体
RETURN
```

FUNCTION 是函数的标识符,后面是函数名,同过程名一样,它是该函数的唯一标识,它的名

字不允许同 FOXPRO 中的其它关键字及函数、过程同名。PARAMETERS 和 RETURN 关键字的用法是和过程及子程序是相同的。每一个函数有一个函数体，它也是由一系列的代码组成，来完成特定的任务。

其次在调用上也和过程有一些差别。

函数的调用：

函数名

函数可以出现在任何表达式中，并与 FOXPRO 的标准函数一样调用。当 FOXPRO 执行到表达式中的自定义函数时，就钻进函数里执行函数体的代码，在遇到 RETURN 语句时，便用表达式的值返回并替代函数名参与调用函数时的表达式的计算。

最后在用法上也要灵活运用。如果要在表达式中使用且只返回一个值，并且该值只在一处调用，则应定义为函数；若定义为过程，则值作参数返回，返回的参数才能应用到表达式中，但可将一个值用在多个地方。在其他情况下，不论是用在表达式中，也不论是否返回值或多少个值，则应定义为过程。

下面我们定义一个自定义函数 MYQUERY()。他的功能与 MYLOCATE 子程序一样。我们也把他加入到 MYPRG. PRG 过程文件中。新的 MYPRG. PRG 过程文件的内容如下：

```
*****MYPRG. PRG*****
PROCEDURE NEWHELLO  &&定义第一个过程
PARAMETERS C_HELLO &&用来接收参数。
MESSAGEBOX  C_HELLO
RETURN
PROCDEURE MYQUERY && 定义第二个过程
PARAMENTERS TJ &&形式参数
USE 人事管理
GO TOP
LOCATE FOR NAME  =  TJ
IF EOF
RETURN . F.
ELSE
RETURN . T.
ENDIF
FUNCTION MYQUERY && 定义函数
PARAMENTERS TJ &&形式参数
USE 人事管理
GO TOP
LOCATE FOR NAME  =  TJ
IF EOF
RETURN . F.
ELSE
RETURN . T.
ENDIF
```

在调用时的形式如下

SET PROCEDURE TO MYPRG.PRG

MYQUERY 庄稼

到此为止,关于子程序、过程和函数的使用方法就介绍完了。

差点忘了,还有一个重要的问题没有讨论,就是变量的作用域的问题。这个问题可是十分重要的哟!0

在 FOXPRO 中,在子程序、过程或函数内定义(下面将这三者一并简称为子程序)的变量是局限于过程和函数内部的,当返回主程序时这些变量就不存在了,除非用 PUBLIC 指明它是公共变量,比如:

public tj

那么返回主程序后 tj 这个变量就还可以用。也就是说,在一个过程或函数内定义的内存变量,在这个过程或函数体内便一直起作用,一旦从过程或函数体中退出,它便不复存在了,因为这时 FOXPRO 会把它所占用的内存空间收回作别的用。如果要使它们在过程或函数以外起作用,应用 PUBLIC 关键字来指明它是公共变量。

但主程序中的变量在子程序中可以用,如果在子程序中不想用主程序中变量,可在子程序中用如下命令将其屏蔽:

private 变量名 1 变量名 2 ...

也可用如下语句屏蔽所有主程序的变量:

private all

也可在主程序中指明一个程序只能在本程序中使用,而不能在其下的子程序(或其上的主程序,怎么主程序还有主程序?等会讲)中使用,这种局部于过程或函数的内存变量也称局部变量或私有变量可以用 LOCAL 关键字来声明:

local 变量名 1 变量名 2 ...

在整个程序中起作用的变量也称全局变,用 PUBLIC 关键字来声明。

提示:主程序、子程序是相对的,比如 A 程序调用 B 程序,那 A 是主程序,B 是子程序;而 B 又可以调用 C 程序,那么这时 B 就是主,C 就是子了,也就是说爷爷是爸爸的爸爸。

请看下面的例子:

```
***A.PRG***
use sjb
do b
***B.PRG***
locate for 姓名 = 庄稼
if found
do c
endif
return
***C.PRG***
messagebox 找到啦!
return
```

下面是一些常用命令的语法说明。

1) PROCEDURE 命令

作用 :定义一个过程

语法：

PROCEDURE 过程名

RAMETERS PARS1ARS2….

过程体

RETURN TO

PROCEDURE 是标识过程定义的关键字，“过程名”是标识此过程的名字，它就象人的名字一样，只不过人的名字是用来区分人的而’过程名’用来标识标识过程的。但有一点“过程名”与人名不太一样，人的名字可以重复，虽然也会出麻烦但问题不是很大，“过程名”就不一样了，它要求必须是唯一的，最好还能表达过程的主要功能。电脑可没有哪么聪明会区分两个同名的过程。

如果过程有参数，就用关键字 PARAMETERS 后面跟随用逗号隔开的参数表，它起到向过程传递参数的作用。“过程体”是一系列 FOXPRO 的语名及其它成分，正是由它们来完成过程的主要功能。当过程执行到 RETURN 语句时返回到过程调用语句的下一条语句继续执行。要是你把代码写到 RETURN 的后面的话，嘿。。。。。不被执行可别来找我。还有一点，每个过程至少有一个 RETURN 语句。没有了 RETURN 是万万不行的。就象人没了空气一样。

2) FUNCTION 命令

作用 :定义一个函数：

语法：

FUNCTION 函数名

RAMETERS 参数表

函数体

RETURN

FUNCTION 是函数的标识符，后面是函数名，同过程名一样，它是该函数的唯一标识，它的名字不允许同 FOXPRO 中的其它关键字及函数、过程同名。由于这个函数不是 VFP 的亲生儿子，而是我们自己定义的，所以我们称它为自定义函数。函数可以有参数，用时把它们列在关键字 PARAMETERS 的后面，彼此以逗号分离开。在退出函数时，用 RETURN 语句，后面跟上一个表达式，这样我们就可以把一个值返回。如果不指定表达式，则返回真值 . t. 。每一个函数有一个函数体，它也是由一系列的代码组成，来完成特定的任务。

3) DO 命令

作用 :调用过程或子程序

语法：

DO 过程名 TH 参数表 | 过程文件

WITH 关键字指出将要传弟给过程的参数，IN 关键字指出定义过程的过程文件。

4) SET PROCEDURE TO 文件名

作用 :指出包含将要调用的过程或函的的过程文件。

14.15 宏替换的应用

我们来学习 FOXPRO 的一个十分强大而又非常容易掌握的功能：宏替换。就从名字上看来这不是个容易对付的家伙，在开始讲宏替换之前，让我们先来看一看这样的一个例子：

假设我们有两个表，1997 级学生成绩表（1997.DBF）和 1999 级学生成绩表（1999.DBF）。这时我们要求用户在浏览表之前先输入年份，以便我们来打开用户指定年度的学生成绩表。也许你会说："这太容易了，俺会呀"然后给出下面的方案：

先用一个文本框来接收用户的输入的年份，然后调用下面的代码来打开表：

```
DO CASE
CASE THISFORM.TEXT1.VALUE = 1997
   USE '1997.DBF
CASE THISFORM.TEXT2.VALUE = 1999
USE 1999.DBF
ENDCASE
BROW
```

不错，您的这段代码表面上是达到了要求。但是它存在着几个问题：

问题 1：当学生成绩表很少时（如：上例只有 1997.dbf 和 1999.dbf 两个表），用上面的代码可以轻松摆平。但是如果表很多时，假设有十个年度的学生成绩表（1980.DBF - - - 1990.DBF）。这下惨了，代码就会变成下面的样子：

```
DO CASE
CASE THISFORM.TEXT1.VALUE = 1980
USE 1980.DBF
CASE THISFORM.TEXT2.VALUE = 1981
USE 1981.DBF
CASE THISFORM.TEXT2.VALUE = 1982
USE 1982.DBF
CASE THISFORM.TEXT2.VALUE = 1982
USE 1982.DBF
……
```

＊为了节省你的网费，我的纸，我就不继续打了，我想您应该明白哪些麻子……代表什么了吧……

```
ENDCASE
```

这样一来代码非常的冗长。但这还不是最糟的，更惨的在下面（天啊～～）。

问题 2：当随着学生的升学，用来保存学生每年学习成绩的表也会动态的增加，这时上面的代码就完全失去了做用。用上面的代码无法预先知道到底会有多少个学生成绩表，也就不可能事先设 N 个 CASE 来判断年份了。

宏替换："不要担心，少要害怕，俺宏替换来也"。宏替换是 FOXPRO 为我们提供的一项强大的功能。它的运算符是"&"，作用就是提取字符串的现值，返回值是字符串常数。

看到这里您的第一反应可能会是："WHAT？什么乱七八糟的，你说的是什么意思？"。

别忙，接着往下看您就明白了。下面就让我们用宏替换功能来完成上面例子的功能，是又好又省事儿，包您满意。这次假设我们有 20 个表（1980. DBF 一 2000. DBF）分别存放每一年的学生成绩，放心我们绝对不会用 20 个 CASE 的：

```
C_YEAR = THISFORM. TEXT1. VALUE  &&用变量 C_YEAR 来接收用户输入的年份。
USE &C_YEAR
＊上面这一句利用宏替换来根据 C_YEAR 的值来找打开对应的表，等价于
＊Use '1999. dbf'，只不过引号的表名是随 C_YEAR 的值而变罢了
BROW
```

搞定，收工。嘿。怎么样，是不是又省时又省力呀？

再看下面的例子：

```
a = b
b = 庄稼
 a  &&显示的结果是"b"
 &a  &&显示的结果是"庄稼"
```

也就是说宏替换将当前变量的值作为真正需要的变量，并返回这个变量的值。

VFP 的很多命令的某些地方不能使用变量，比如上面的 use 命令，你不能写成：

```
a = 1997. dbf
use a
```

这样就成了打开 a 这个表，这时就必须用宏替换，

```
use &a
```

类似的还有当 locate for 后面跟的条件不是固定的，你的程序可以允许操作者选择不同的字段来查询，可能是：

姓名 = . . .

也可能是：

电话 = . . .

你不能直接把 for 后面的内容整个作为一个变量，这时宏替换也可以起作用，你可以把条件整个放入一个变量，比如字段名放在一个变量中 zdm，要查的内容放在 nr：

```
tj = zdm +  + nr +
locate for &tj
```

如果这时 zdm = 姓名 ，nr = 庄稼 ，这就相当于：

```
locate for 姓名 = 庄稼
```

那么只要 tj 的内容是个正确的逻辑表达示，不论是什么内容都可以运行，这将给编程带来很大方便。

多学一招

你甚至可以把整条语句放入一个变量：

```
tj = locate for  + zdm +  + nr +
&tj
```

酷吧！

注意:值得注意的是,宏替换&不能递归的引用自已。比如下面的用法就是错误的:

STORE "&ZL" TO ZL

&ZL

赋值语句与宏替换巧妙地组合,能实行灵活的查询和程序控制,提高程序通用性,使程序更加简练,运行效率更高,龙其在处理一些不确定对象时,宏替换将扮演非常重要角色。

最后补充一点,虽然宏替换功能强大,但是也不要太频繁使用,正是由于宏替换灵活性,使得它执行时要进行一些转换,如果太多的使用宏替换 可能会影响程序的性能。

所以在一些不是必须的场合下,尽量避免使用宏替换 ,就象好吃的东西吃得多了也会起腻一样,宏替换用的太多也会使你的程序不舒服的。

14.16 程序的测试与调试

当写完程序的所有代码后您一定会想:"终于大功告成了,哈……"。

慢着!如果您真的以为万事大吉就错了,还有一件非常重要的事情等着我们去做,那就是对程序的测试和调试。

据说很多年前当一个小虫子在一台计算机中使一些晶体管不能工作时,才第一次用到了调试 DEBUG 一词。所以,术语 DEBUG 也就是找出程序中的"小虫子 BUG "。不管它的由来到底是什么,它的目的就是查找使程序失败或产生不正确结果的原因。

在开发应用程序中应将测试作为一个独立的并且是有计划的任务。测试软件的方法有很多种。而有些开发者只在完全编写完应用程序后才进行测试。VFP 所具有的交互功能使与开发过程中并存的测试变得更容易也更有效果。问题是管理跟踪与开发过程并存在测试所花费的时间比较困难。这一课将考察各种测试技巧并分析它们的优缺点。

数据驱动和逻辑驱动的测试

测试应用程序的两个要素是:有效性和范围。有效性测试是检查应用程序是否对特定的输入产生预期的结果。范围是检查所有的语句是否都已被测试执行,任何没有被执行的代码都有可能隐藏故障。

有效性测试有两种基本的方法。第一种方法是数据驱动的,它不需要知道关于程序的工作方式方面的知识,而主要集中在对现实世界或虚构的数据进行采样的基础上 挑选出一系列测试数据集合,然后使用这些数据运行程序,看它是否产生了预期的结果。

另一种方法是逻辑驱动的方法,这种方法需要程序编码的广泛知识,它试图测试程序可以执行的每个路径,还通过使用接近和超过已知的实际上存在限制的数据,对程序如何处理这些数据限制进行检测。

这两种方法各有其优点和不足:

数据驱动应用程序的优点包括它有意识或无意识地对程序进行假设。但程序员经常"假设"一个程序绝不会执行某种动作,从而也就不能彻底地测试它。而有时程序员假定是正确的部分往往正是出现问题的部分。数据驱动方式的主要不足是不能保证测试数据集合覆盖了所有的程序路径和循环。

逻辑驱动的方式测试弥补了数据驱动测试的弱点。如果设计得好,它可以测试整个系统中的每一行代码。它明显的缺点是对于大型应用程序来说,全部测试每一行代码需要多重数据测试。更进

一步说，它需要花费很多时间开发出必要的数据集合以保证对每一行代码的测试。

测试技术

一旦开始进行编码，讨论的重点就从纸上思维方式的检查转移到正规的编码和实际物理测试上，下面描述了一些技术：

正式的代码审查是对重要代码的仔细的检查，它提供了对代码标准、注释的使用、变量的命名、局部和全局变量的使用等的反馈。

模型化或原型化是使用可以获得的工具快速地创建应用程序的外壳。它的目的是显示系统的整体功能。在 VFP 中，它牵涉到创建与一个简单菜单相连系的基本表单和报表。虽然表单是有功能的，但它们也许没有包括最终的事件捕获。与之类似，报表也许没有包括选择和排序的选项。目前这种技术被称为快速应用程序开发。

语法检查测试代码的基本正确性。它检查命令和文本串的拼写、表达式的有效性以及命令的基本结构。Foxpro 在编译中进行的大部分是这种工作，当然有些语法问题只有到运行程序时才会表现出来。

单元测试用来检查一些独立的语句组。例如，在设计一个新的类定义时，使用单元测试检查与它的方法有联系的代码。为了实现单元测试，必须编写一种称为驱动程序的特殊程序。驱动程序不成为最终代码的组成部分，而是用来设置测试代码段所必须的程序环境，如初始化变量。

虽然可以分别测试各个过程和函数。但是，它们的正确运行并不能完全保证程序作为一个整体也可以正确的工作。单元测试可以减少整个程序系统错误产生的可能性。测试主要是对模块之间接口的连接进行检查，它的目的是何证数据和逻辑正确地从一个模块传递到另一个模块，其中包括了使用正确的参数类型和大小。它还可以寻找由被调用过程或函数重定义而对公共或私有变量进行的意外的修改。

功能测试检查程序工作的主要特性是否象预期的一样。当从菜单中选定了一个报表选项时，是否能得到所选定的那个报表或者是另一个报表，甚至是一个表单呢？如果一个表单中显示出的消息说，按下 F2 来显示可能值的列表，它就会检查在你按下 F2 时是否真的出现了这个列表，它不需要对报表结果或按 F2 时显示的那个列表是否是包括了正确的数据进行验证。

强度测试检查程序的边界条件。它对程序如何响应极限值进行测试。例如，如果程序员负责记录每周的工资单，它也许会检查，如果输入 170 作为雇员的一周工时数，程序会做些什么？一周的工时最大值 7 乘以 24，即 168，对网络的强度测试主要是当多个用户运行程序时，程序如何动作，程序是否能处理数据的并发更新？记录和文件锁是否工作正常，而同时又能最小化其他用户不能访问数据的时间

程序的运行速度归于性能测试一类。VFP 提供了用多种方式编写出大多数特性的灵活性的方法，但这些方法的性能并不完全相同，有些方法比其他的性能好得多。性能测度通过识别程序的那些部分花费了最多时间来寻找能提高速度的地方。然后就可以试着用其他的编程方法来提高速度。

创建测试环境

创建一个测试环境要考虑两个重要问题：硬件和人的问题。在测应用程序时，测试它的硬件当与计划使用它的配置相同。如果要在网络上运行应用程序，则在一个单独的系统上测试它会漏掉与记录和文件共享有关的潜在问题。

在测试快结束时使用真实的数据是很重要的。因此，推荐的开发模式是首先创建和实际收集数据所必须的哪些模块。然后用户可以使用它们输入真实的数据，这些数据又可作为以后开发的测试

数据。

当然,所有的开发工作都应与真实数据是物理隔离的。在开发过程中很有可能会发生破坏真实数据的错误。例如:错误的输入真实数据的路径而不是测试数据的路径会破坏真实的数据。这也意味着应该定期备份测试数据测试程序。

不是每个人都擅长测试的。实际上,有些人比其他人更乐于进行测试。应用程序的开发者通常是最差劲的测试者,他们太了解程序而且倾向于总是输入正确的数据。他们下意识不想使自己的程序失败,虽然他们也很清楚地认识到进行测试的必要性。

另一方面,其他人员不会和程序有感情,也就能在测试中更加不留情面。虽然"无情的"测试者会发现更多的问题,但他们必须与开发者进行讨论。程序毕意是别人创建的产品,这点无法否认。

VFP 提供了很多不同的方法来完成同一个任务,所以不要因为与你的方法不同而批评别人的方法。如果存在另一种方法能更好、更有效地完成任务,可以向其他程序员演示它为什么更好。帮助别人学习,而不是强求他们学习新的方法。

确定测试何时结束

有人也许会说测试永远也不会结束,直到应用程序变得过时,废弃不用为止。实际应用中总会出现数据的新组合,大多数程序会不断发生变化,在这里加个新特性或在那里加个字段。每个新特性都会引入产生错误的可能性。程序路径也可能发生变化,新的变量可能覆盖其他例程中的同名变量,而且其他的副作用也会影响现在的代码。

不管使用哪种方法都几乎不可能确定什么时候找出所有的故障。更多的时候是根据找出一个故障与放任它存在的代价做出的猜测决定。相反,仅仅为了如期完成测试计划就宣布测试结束是危险的,缺少远见的。测试应当一直持续到每个参加工作的人都对应用程序的性能感到有信心才结束。每个程序员进行的代码测试的方法都略有不同,因此,即使仅有两名程序员在一起进行测试,都会显著提高整体的检测率。

跟踪错误的方法

尽管前面讨论了避常见错误和测试的方法的开发,但是错误仍然会发生。最首要的任务是在发生错误时找到它并修改它。

在程序执行过程中发生错误时,VFP 显示出一个出错提示框,并给出简单的出错信息。在信息下方有三个按钮;<终止> <挂起> <忽略>。大多数情况下不想忽略错误。如果是为其他用户编写程序,你决不会希望忽略错误。但事情总有例外。如果程序因为找不到颜色集而失败,就可能希望忽略这个错误。大部分错误是不能忽略的。如果 VFP 不能定位一个表格,则忽略错误没有任何意义。因为某些时候可能会用到这个表格,所以可将问题存档并退出程序。

作为开发者,读者会发现《挂起》是测试中的一个选项。它停止程序的执行而不是从内存中删除当前的变量。任何当前已打开的表格仍然保持打开,它们的记录指针也在各自的位置上。这时我们可以打开调试窗口(debug)和跟踪窗口(trace)来查看正在被执行的源代码及各种变量和表格的状态。因为已经在前面的章节里讲过了调试工具的使用方法,在这里就不重复了。

实际上,可能不希望让客户看到这种 VFP 的出错信息。而是你自己的错误处理程序来捕获所有的错误,错误处理程序将发生错误时的系统存档,并安全的退出应用程序。

下面就讲一下如何编写自己的错误处理程序。

ON ERROR 正是我们想要的命令,它可把程序定向到一个错误处理程序中,它的主要任务是提供关于错误的信息。以下这条命令的执行将使后续程序的错误定向到 MYERROR 错误处理程序

中：

```
ON ERROR DO MYERROR WITH error　message　message 1　sys 16　lineno
```

ON ERROR 的语法如下：

```
ON ERROR
```

其中，这项指出当程序出现错误时将要执行的命令，它即可以是一条简单的命令也可以调用其他的过程。如果省略它的话将使前次定义的 ON ERROR 失去作用。

在上面的例子中我们定义的是当程序错误时执行

```
DO MYERROR WITH error　message　message 1　sys 16　lineno
```

命令来调用名为 MYERROR 的错误处理过程。其中 error　message　message 1　sys 16　lineno　为传递给过程的参数，它们分别传递“错误号”，“错误信息”，“返回导致这个错误的程序源代码”“当前执程行的程序名”，“正在执行的那一行程序的行号”给错误处理过程。在 MYERROR 过程中可以通过检查错误号来确定错误的类型。

从用户的角度来说主要存在三种主要类型的错误。第一种是轻微错误，其中包括的错误有找不到颜色集，或者打印机或软盘驱动器没有就绪。有些情况下只要登记错误，然后跳过引起错误的命令并继续执行，如找不到颜色集。

其他情况下，在发给用户的消息后是一个处理错误的 RETRY 或 RETURN 命令。使用这种方法的例子是程序试图从软盘驱动器上读文件时，驱动器中没有软盘或者程序试图写盘时软盘是写保护的。在这种情况下，将显示给用户消息，告诉他们如何用 RETRY 纠正错误。

另一个例子假设使用 SKIP 在记录中移动时超过了文件的尾。它的错误号为 4. 在遇到这种类型的错误时不必取消程序。为了纠正问题，程序只要把记录指针重置到表格的最后一条记录上即可，程序清单如下：

```
prodedure myerror
lparameters lnErrorNo lcMessage lcErrorLine lcmodule lnErrorLineNo
if lnErrorNo = 4
messagebox“到了记录尾了”
gotobottom
else
cancel
endif
return
```

另外我们还可以用一个文本文件来记录程序的出错信息，以便程序员查阅。可以用记录本建一个名为 ERROR. TXT 的文件，COPY 到程序所在的目录下。然后在错误处理程序中把各种错误信息写到文本文件里。下面是一个简单的小例子。

```
prodedure myerror
lparameters lnErrorNo lcMessage lcErrorLine lcmodule InErrorLineNo
local myerror
c = ttoc datetime　+ 错误号　+ str lnErrorNo + 提示信息　+ lcMessage +
 代码　+ lcErrorLine + 文件名　+ lcmodule + 行号：+ str InErrorLineNo
a = fopen　error　12
```

```
b = fread a 1024
do while ! feof a
b = fread a 1024
enddo
fput a c
fclose a
d = messagebox c 2 + 16 + 0    错误
do case
  case d = 3
cancel
case d = 4
retry
case d = 5
return
endcase
```

第 15 章　FoxPro 高级教程

15.1　配置 VFP 环境

15.1.1　概述

工作环境规定了 VFP 工作时的某些行为。配置工作环境意味着您可以通过某些方法使 VFP 的行为满足自己的需要 - 必须的或者是某些个人偏好的。

与任何完善的开发工具一样，VFP 也是一个复杂的系统，复杂的一个方面就表现在它具有极大的灵活性 - 同一种工作,由于 环境 的不同,可能会产生不同的效果。

工作环境是个整体的概念,我们使用 环境参数 一词来描述构成工作环境的个体。那么就可以说,工作环境就是由一组环境参数来决定的,配置工作环境就是设置这组环境参数,以满足不同的需求。这显然有两方面的任务,第一、您要知道有那些环境参数,第二、明确怎样配置这些参数。

工作环境涉及到很多方面,例如输入输出、数据处理、开发、调试等等。可以通过分类对所有环境参数进行描述。所有的环境参数都有一个原始的默认值,一般的环境配置工作,事实上只需要修改少量的环境参数即可。

有一个问题需要明确，VFP 的工作环境 是指 Visual FoxPro 工具的开发环境。另外一种情况,使用 VFP 开发的应用程序,在编译成执行文件后,工作在 VFP 运行时的工作环境下。这两者是有区别的,显而易见的区别是,运行时环境涉及的问题比开发环境少得多,例如开发、调试方面的问题。一般来讲,运行时的环境是开发环境的一个子集,但严格说来这是不对的,二者仍存在一些差别,不仅在环境参数的类别与数量方面,也存在于配置方法上。本文的叙述主要针对开发环境而进行,对二者存在的差别也将被提及。

15.1.2　默认的工作环境

VFP 安装完成后,其 内部 保存着对所有环境参数的设置,这些参数可以称做是原始设置。您永远也不能够去修改这些原始设置。但 VFP 提供一种机制允许您每次启动时用自己的设置去覆盖这组原始设置。方法是是使用 Windows 注册表及配置文件。

启动时,VFP 首先会检查 Windows 注册表中的有关设置,然后据此修改环境参数。然后还会查找一个配置文件,该文件默认的名字为 config. fpw,其中也存放着一些环境配置信息。VFP 将读取这些信息,并再次修改环境参数。如果 VFP 在 Windows 注册表中没有找到配置信息或者没有找到配置文件,这没有关系,VFP 将以原始配置进行工作。您甚至可以主动让 VFP 在启动时放弃读取注册表及配置文件。

VFP 不会主动把环境参数写进 Windows 注册表中,但它提供了一种方法使您方面地将环境参数写进注册表,写入不仅针对修改过的参数,而是所有参数,您没有去修改的参数值将以原始值写入。配置文件则不同,它只保存您写入的参数,某种意义上,它相当是一份注册表中的环境参数表的勘误。

保存在 Windows 注册表中的环境参数以及配置文件中的设置称做 默认工作环境

15.1.3 配置文件

先读取注册表再查找配置文件的方法似乎有点复杂,但有些时候需要这样做,例如工作在运行时环境的应用程序,启动时不去读取注册表,这使得配置文件成为必须。另外,配置文件创建于FoxPro 的早期版本,兼容性使它必须被保留。VFP6 以前的版本在安装时将自动建立一个这样的文件(在安装目录下),而到了 VFP6,系统不再自动建立该文件,需要时由用户自行建立。该文件是一个文本文件,可以使用任何文本编辑器来建立它。

VFP 启动时将按照下列路径查找该文件:当前工作目录 安装 Visual FoxPro 的目录 DOS 路径中列出的目录 配置文件有默认的名字(config.fpw),默认 意味着你可以去改变它。这有利于您同时拥有多个配置文件,以便满足不同的需要。指定另外不同于 config.fpw 名字的配置文件的方法是在启动 VFP 时增加命令行开关项。

配置文件中不仅仅存放环境配置信息,例如可以在配置文件中设置 VFP 启动后立即去执行一个程序文件。这使得您可以在配置环境方面使用一种技巧,即将环境设置命令保存在一个程序文件中,然后让 VFP 启动时去执行它。

启动过程也可以通过使用命令行参数来定制,通过定制启动过程,可以得到不同的环境默认设置,以下是对环境配置有影响的一些情况:

开关 描述

- A 忽略默认配置文件以及 Windows 注册表设置。
- C 放弃使用配置文件。
- c <文件> 指定默认配置文件 Config.fpw 之外的配置文件(必要时包含路径)。

15.1.4 更改默认设置

VFP 启动完成后,工作环境被设置成默认状态。您可以即时地更改环境参数,这有几种方法,各种方法可能适合于不同的情况。

交互式修改

在命令窗口输入环境设置命令,执行后该修改会立即生效。绝大部分的环境参数是通过 SET 命令设置的,另外还有一些通过系统变量存储以及极少数的函数设置。在早期的 XBASE 工具中,例如 dBASEII 中,仅有 20 几条 SET 命令,而今天,VFP 中已经有 120 多个 SET 命令了。交互式的修改方法是枯燥的,因此只用于临时性的修改,例如调试等。

程序化执行修改

可以将一组环境设置命令组织成一个程序文件,运行该程序,就相当于在命令窗口一一输入执行了这组命令,有些情况下是需要这样做的,例如在开发环境下同时调试多于一个项目的时候。可以将该环境设置程序存为一个过程或直接写在项目的主文件中。事实上,对于应用程序,因为最终要脱离开发环境运行,所以大部分都是这样做的。

用命令方式读取 Windows 注册表中的设置

让 Visual FoxPro 再次读取自己的注册表设置,并且使用当前的注册表设置更新自己。 语法为:SYS(3056) 说明:包含 1 选项可以只根据注册表设置进行更新,不包括 SET 命令和文件位置。Visual FoxPro 在 Windows 注册表的选项对话框中保存设置。当 Visual FoxPro 启动时,读取这些设置,并利用这些设置进行配置。SYS 3056 可更新包含有文件名和路径的系统变量,而系统变量

由 SET 命令和注册表设置。当您选择设置为默认值时,选项对话框指定的设置写入注册表。其中一些影响 Visual FoxPro 环境的设置含有 SET 命令,例如 SET BELL,SET LOCK 等。注意 SYS 3056 不更新来自 Config. fpw (即 Visual FoxPro 配置文件) 的设置。

有关环境设置的命令

有很多环境设置的命令,在本文的附表中,以 选项 对话框中的分类列出了相应的环境设置命令,还有一些命令没有包含在 选项 对话框中,但它们也是关于环境的设置命令,本文仅将它们列出来,具体的意义及语法可参见随机文档。

SET ALTERNATE SET ASSERTS SET AUTOSAVE

SET COLOR OF SCHEME ET COLORS SET SET CONSOLE

SET COVERAGE SET CPCOMPILE SET CURSOR

SET DEBUG SET DEBUGOUT SET DEVICE

SET DISPLAY SET EVENTLIST SET EVENTTRACKING

SET FIXED SET FULLPATH SET FUNCTION

SET MACKEY SET MEMOWIDTH SET NULL

SET NULLDISPLAY SET OLEOBJECT SET PRINTER

SET READBORDER SET SPACE SET STATUS

SET SYSMENU SET TEXTMERGE SET TEXTMERGE DELIMITERS

SET TYPEAHEAD

无论是交互式还是程序化执行命令,其修改都是临时的,只对当前工作有效,VFP 系统关闭后,这种修改不能被保存,下一次运行,如果需要同样的设置,您不得不再一次输入该命令或执行该程序。要想使修改成为永久,必须将修改保存在 Windows 注册表中或构建配置文件。

修改注册表

有两种方法:直接修改 Windows 注册表或使用 VFP 开发环境提供的 选项卡 对话框。对于直接修改 Windwos 注册表的方法,没有人会建议您这样做,除非您首先对 Windwos 注册表的结构非常熟悉,其次对 VFP 在 Windows 注册表中的各种参数的意义比较了解,第三有某些特殊需要以至于不能使用常规的方法完成。常规的方法,是使用 VFP 的 选项卡 对话框来完成环境配置。

使用 选项 对话框

使用 选项卡 对话框来进行系统的配置比直接修改 Windows 注册表的方法至少有两个好处,第一,很多配置参数在 选项卡 上是以一种 易于理解的方式 表达的,您只需在某些 问题 前打勾或者在很多答案中选择一个,这就象您填一份表格一样来得轻松。第二,在 选项卡 上进行的配置,一旦完成后,立即生效,而直接修改 Windows 注册表的方法却不是这样,只能是在下一次运行 VFP 时生效。

在开发环境下,执行 工具 菜单中的 选项 菜单项,会出现 选项 对话框。对话框上有 12 个卡页,分 12 个方面对环境进行配置。本文附表中详细描述了其中的每一项设置。事实上,真正需要修改的并不多,很多项设置,原始值应该能满足要求。现在的问题是,当你修改完成后,可有两种选择,按 确定 按钮退出对话框,您的修改只会保存在当前环境中,随着系统的关闭而失效 按 设置为默认值 按钮退出对话框,当前的设置将会被写进 Windows 注册表从而成为新的默认值。

15.1.5　构建配置文件

配置文件可以使用任何文本编辑器进行编辑,可在其中写入四种内容,在在书写格式上也有一

些特殊的约定：

SET 命令：不写 SET 关键字，例如：DEFAULT = HOME + FP 系统变量：与命令格式相同，例如：_SPELLCHK = SPLLCHK.EXE 特殊术语：

COMMAND

指定 Visual FoxPro 启动时所运行的 Visual FoxPro 命令。使用如下语法，COMMAND = cVisualFoxProCommand 。注：cVisualFoxProCommand 表示要执行的命令。

EDITWORK path

指定文本编辑器放置工作文件的位置。由于工作文件会变得很大，所以需要指定一个具有大量空间的位置。默认的位置是启动目录。

INDEX extension

指定 Visual FoxPro 索引文件的扩展名。默认扩展名是 .idx。

LABEL extension

指定 Visual FoxPro 标签定义文件的扩展名。默认扩展名是 .lbx。

MVCOUNT

设置 Visual FoxPro 可以含有变量的最大数目。这个值的范围是从 128 到 65 000；默认值是 1024。

OUTSHOW ON | OFF

废止通过按下 SHIFT + CTRL + ALT 键在当前输出前面隐藏所有窗口的能力。默认设置为 ON。

PROGWORK path

指定 Visual FoxPro 保存程序高速缓存文件的位置。为了得到更快的性能，特别是在多用户环境中，可指定一个快速的磁盘（如果可能，可指定本地磁盘或 RAM 磁盘）。允许为高速缓存指定最小 256K 的空间（但是，文件可能会变得很大）。默认位置是启动目录。

REPORT extension 指定 Visual FoxPro 报表定义文件的扩展名。默认扩展名是 .frx。

RESOURCE pat 指定 FOXUSER 源文件的位置。file 参数是可选的 如果不包含这个参数，Visual FoxPro 就查找 Foxuser.dbf 文件。如果指定的文件不存在，则创建该文件。默认值是启动目录（path）和 Foxuser.dbf（file）。

SORTWORK path

指定 SORT 和 INDEX 这样的命令将工作文件存放的位置。由于工作文件会比保存的表大两倍，所以需要指定一个具有大量空间的位置。为了得到更快的性能，特别是在多用户环境中，可指定一个快速的磁盘（例如本地磁盘）。默认的位置是启动目录。

TEDIT editor 指定当您使用 MODIFY COMMAND 或 MODIFY FILE 命令编辑程序文件时，所使用的文本编辑器的名称。包含可选子句 /N 和 TEDIT，可以指定 Windows 文本编辑器（例如，Microsoft Word for Windows）。默认编辑器是 Visual FoxPro 编辑器。

TITLE title

指定在 Visual FoxPro 主窗口的标题栏出现的标题。默认标题是 Microsoft Visual FoxPro 。

TMPFILES drive

如果在其他选项中没有指定，则本项指定保存临时的 EDITWORK、SORTWORK、和 PROGWORK 工作文件的位置。由于工作文件会很大，所以需要指定一个具有大量空间的位置。为了得到更快的性能，特别是在多用户环境中，可指定一个快速的磁盘（例如本地磁盘）。默认的位置是启动目录。

15.1.6　ON 命令

ON ERROR 命令

设计得再好的程序，在运行时也不可避免地会发生错误，这些错误可能是程序自身的错误，也可能是系统环境引起的或是用户错误地操作 如错误地移动/删除文件等 引起的等。

因此，程序员有责任编写出可以捕捉错误的程序并尽可能地处理这些错误。要捕捉程序中发生的错误使用 ON ERROR 命令。你可以在 ON ERROR 命令后跟随一个错误处理程序的名字：ON ERROR DO ERRORHANDLER，或者在 ON ERROR 命令后跟随一条赋值语句：ON ERROR glError = .T.。

注意，在程序中全程使用类似于 ON ERROR glError = .T. 的命令是极不负责任的和令人憎恶的，这有可能会使用程序陷入死循环而使用用户不得不强行退出系统 强行关断电源等 ，这样做极有可能破坏用户的数据文件。

ON ESCAPE 命令

指定在程序或命令运行过程中，按下 ESC 键时所执行的命令。

语法为：ON ESCAPE mmand

ON SHUTDOWN 命令

指定当试图退出 Visual FoxPro 时所要执行的命令。

语法为：ON SHUTDOWN mmand

15.1.7　一些重要的设置

以下是几乎每一个应用程序都要进行的设置：

_VFP 系统变量

指向当前运行的 Visual FoxPro 应用程序对象。

语法

_VFP. PropertyNameValue

- 或 -

_VFP. Method

参数描述

PropertyName 指定应用程序对象的属性。

eValue 指定属性的值。Method 指定应用程序对象的方法。

说明

通过 _VFP，您可以访问对象集合 Objects collection 。

可设置的属性有：

ActiveForm Application AutoYield

Caption DefaultFilePath Forms

Full NameHeight Left

Name OLERequestPendingTimeout OLEServerBusyRaiseError

OLEServerBusyTimeout Parent StartMode

StatusBar Top Version

Visible Width

SET SAFETY

决定改写已有文件之前是否显示对话框，或者决定当用表设计器或用 ALTER TABLE 命令对表结构进行修改后，是否重新计算表或字段规则、默认值以及错误信息。

语法为：SET SAFETY ON | OFF

SET PROCEDURE TO

设置运行时的命令文件

SET CLASSLIB TO

设置运行时的类库文件

SET MEMOWIDTH TO

指定备注字段和字符表达式的显示宽度。

语法为：SET MEMOWIDTH TO nColumns

SET MULTILOCKS ON

决定能否使用 LOCK 或 RLOCK 锁定多个记录。

语法为：SET MULTILOCKS ON | OFF

若要在程序中使用表缓存必须设置 SET MULTILOCKS 为 ON。

SET HELP TO

激活或废止 Visual FoxPro 联机帮助或指定的帮助文件。

语法为

SET HELP ON | OFF

- 或者 -

SET HELP TO Name

SET DELETED

指定 Visual FoxPro 是否处理标有删除标记的记录，以及其他命令是否可以操作它们。

语法为：SET DELETED ON | OFF

SET EXCLUSIVE

指定 Visual FoxPro 在网络上以独占方式还是共享方式打开表文件。

语法为：SET EXCLUSIVE ON | OFF

SET NOTIFY

确定是否显示某种系统信息。

语法为：SET NOTIFY ON | OFF

SET BELL

关掉或打开计算机铃声，并设置铃声属性。

语法为：

SET BELL ON | OFF

- 或者 -

SET BELL TO requency nDuration | cWAVFileName nDuration

SET NEAR

FIND 或 SEEK 查找记录不成功时，确定记录指针停留的位置。

语法为：SET NEAR ON | OFF

SET EXACT

指定比较不同长度两个字符串时，Visual FoxPro 使用的规则。

语法为：SET EXACT ON | OFF

SET CONFIRM

指定是否可以用在文本框中键入最后一个字符的方法退出文本框。

SET ESCAPE

决定是否可以通过按 ESC 键中断程序和命令的运行。

语法为：SET ESCAPE ON | OFF

SET DATE

设置日期格式

语法为：SET DATE AMERICAN | ANSI | BRITISH | FRENCH | GERMAN | ITALIAN | JAPAN | USA | MDY | DMY | YMD

SET CENTURY

设置是否在日期的年中显示世纪值

语法为：SET CENTURY ON | OFF

ON SHUTDOWN DO

指定当试图退出 Visual FoxPro 时所要执行的命令。

语法为：ON SHUTDOWN mmand

15.1.8　进一步订制 VFP 的向导和生成器

作为最好的数据库管理系统，VFP 给我们提供了高度的可自定义的交互式的开发环境 interactive development environment IDE 。你可以修改菜单，安装新的生成器和向导，实现开发工具条，修改项目管理器的行为，以及很多其它事来使你的 IDE 更高效。这些定制甚至比我们所知道的和喜欢的 VFP 语言更好，连相对不熟炼的 VFP 开发者都可以按他们自己的方法定制开发环境。

本节将配合实际程序示例，向你展示如何建立简单的工具增加你和你的开发组的效率。我们将着眼于修改 VFP 的向导和生成器来提供额外的或自定义功能，并使用 VFP 6 中的 BuilderD 技术创建你自己的生成器。

修改 VFP 的向导和生成器

如果你象我一样，你可能不常用 VFP 的生成器和向导，因为它们不完全能满足你的需要。可能它们不具备足够的灵活性或可能你只是不喜欢他们的工作方式。直到 VFP 6 以前，没有办法改变生成器和向导的行为，因为 Microsoft 没有提供源代码。但是，现在我们不但获得了生成器和向导的源代码，也获得了类浏览器，组件管理器的源代码。

生成器和向导的源代码可以在 VFP 主目录下的 TOOL SOURCE 目录中的 XSOURCE. ZIP 文件中找到。当你解压该文件时，它建立一个 VFPSOURCE 目录，其中包含了所有的源代码。向导的源代码 可以在 WIZARDS 目录下找到，生成器的在 BUILDERS 目录下　虽然一些 WIZARDS 目录中的公共文件也被生成器使用 。

因此，现在我们有了源代码，我们可个按我们的需要来修改生成器和向导，对吗　好了，一个较好的办法是建立新的生成器和向导时，使用源代码中的大多数类和程序，但以派生子类并复制和修改 PRGs 或建立封装 PRGs 的方法来忽略他们原有的行为。本文将详细说明如何对各生成器和向导这样做。

一但你建立了你自己的生成器和向导，怎样告诉 VFP 使用你的而不是原来的吗 生成器是注册在 BUILDER. DBF 且向导是在 WIZARD. DBF 中，两个表都在 VFP 主目录下的 WIZARDS 子目录中。这些表具有相同的结构，如下表所示。

字段描述

NAME 生成器或向导描述名。

DESCRIPT 生成器或向导的说明。

BITMAP 未使用。

TYPE 生成器或向导是对哪一对象类型的。在 BUILDER. DBF 情况下，它一般是一个对象的基类 虽然已有以 MULTISELECT，AUTOFORMAT 和 RI 字段 。对于 WIZARDS. DBF，它可能是 FORM，REPORT，和 QUERY。

PROGRAM 包含生成器和向导的 APP 文件名。

CLASSLIB APP 文件中要实例的类。

CLASSNAME APP 文件中的主要类的类库。

PARMS 传递到生成器或向导的参数。

当你调用一个生成器时，VFP 调用 _BUILDER 系统变量中指定的程序 默认是 BUILDER. APP 。BUILDER. APP 查看它所在的环境 例如，生成器是被那一个对象调用 ，在 BUILDER. DBF 表中查找与环境匹配的记录 例如，在 TYPE 字段中查找对象的基类 ，并调用注册后的生成器。除系统变量 _WIZARD 外，向导也是以相同的方法处理，_WIZARD 是用于指向可在 WIZARD. DBF 找到的 WIZARD. APP。

要告诉 VFP 使用一个不同的生成器或向导，插入一条说明如何运行你的生成器或向导的新记录到 BUILDER. DBF 或 WIZARD. DBF 表中。如果你想在运行新的向导或生成器的同时也可以选择运行原有的生成器或向导，在 WIZARD. DBF 表中保留原有的生成器或向导记录。如果你想替换原有的，简单的修改它的 TYPE 值为另外的值 我使用一个 X 前缀，如 XGRID ，而不是删除该记录

使用该方法，你可以简单的以恢复 TYPE 的值来恢复使用原来的生成器或向导。 我们将考查创建一个 Grid 生成器的替代物，参照完整性生成器，Upsizing 向导，和 WWW 搜索页向导。

创建更有用的 Grid 生成器

VFP Grid 生成器提供了一种快速方法来整合 grid 的列并建立你所希望的视觉外观。但是，关于该生成器，有我所不喜欢的一些东西：

它不会自动调整列的宽度。必须自己调整列的宽度以适应数据宽度的需要。

在生成器的外观页面中的控件类型组合框只列出了已存在于列中的 VFP 基类和类 没有办法添加你自己的类到该列表中。

它创建的列和列头是 VFP 的基类。你也许想用你自己的类替代它 必须是以编程的方式定义的 ，例如，单击一个列头而按该列排序。

要建立一个 VFP Grid 生成器的替代物，首先建立了 SFGRIDBLDR 项目 它在你解压该文档所附的示例程序时建立的 GRID 子目录中 并添加以下文件：BUILDER. VCX 在 VFP 向导源目录中的 BUILDER UILDERS 子目录中 ，GRIDBLDR. VCX 在 BUILDER RIDBLDR 目录中 ，THERM. VCX，WIZCTRL. VCX，WIZMOVER. VCX 所有都在 WIZARD ZCOMMON 目录中 ，DUMMY. PRG，和 WBGRID. PRG 在 BUILDERS 目录中 。我如何知道要添加那些文件到项目中呢 这很简单 :只需要查看 GRIDBLDR 项目的内容。

其次，派生 GridBuilder 类到 SFGridBuilder　在 SFGRIDBLDR. VCX 类库中　并复盖 Reset-Columns 方法，该方法用于调整选定列的宽度到适当的大小　我只实现了该想法，未实现上述列表中的其它两个　。该方法的代码在下面列出。这里要注意两个有趣的东西。首先，我希望写很多复杂的代码，根据字段宽度、字体和字号来计算出列的宽度，等等。有趣的是，它放弃了生成器中已存在的任务的方法　SetColWidth　，而只是传递字段宽度到该方法中，生成器传给它一个不同的值。其它事情是分配到 loColumn。Width 的计算宽度被注释了。理由是我还没有调试好，改变列宽后，当生成器关闭时，会造成问题，所以我把列宽设置为，列宽是保存在 wbaCols 中。　如果你需要，wbaCols 是一个公共数组。我没有创建生成器，因此不要责怪我采用这种方式

因此，效果是当一个字段添加网格中时，它没有立即获得适当的列宽，但是一旦你执行了一些其它操作　添加另一个字段，转到生成器的另一个页面中，关闭生成器等等。它就会取得适当的列宽。

```
local lnRows
lnI
lcField
loColumn
lcHeader
loHeader
dodefault
lnRows = alen wbaCols  1
for lnI = 1 to lnRows
lcField = wbaColsI  2‖
if not empty lcField
loColumn = evaluate  wbaControl.  +
wbaColsI  7‖
wbaColsI  1‖ = This. SetColWidth fsize lcField
loColumn
* loColumn. Width = wbaColsI  1‖
endif not empty lcField
next lnI
```

最后我为我的生成器建立了 GRIDMAIN. PRG　使用了 BUILDERS RIDBLDR 中的 GRID-MAIN. PRG 的办法　并使它成为项目的主文件。该程序用 SET CLASSLIB 命令添加 SFGRIDBL-DR. VCX 到打开的类库中，这样它可以找到我们的 SFGridBuilder 类。GRIDMAIN 也自动注册它自己到 VFP BUILDER 表中　如果是直接运行该 APP 文件　。

要查看该生成器的行为，建立并运行 SFGRIDBLDR. APP 来作为 grids 生成器注册它。如果你在运行该 APP 后，观察观查 BUILDER. DBF，你会发现它是　未注册的　，原始的 grid 生成器的 TYPE 字段由　GRID　改成了　XGRID　并在它后面为它自己添加了一条 TYPE 字段为　GRID　的新记录。下一步，新建一个表单，拖动 grid 到表单中，并调用生成器。新的生成器看起来与原来的没有任何区别，但是当你添加字段到表格时，你会发现字段宽度自动调整到了适当大小。

创建一个更好的参照完整性生成器

我有一小点关于 VFP 自身的参照完整性　RI　生成器的问题：

当你单击确认按钮来保存修改后的 RI 时,它两次 而不是一次 让你确认。不是我太自信,但是我相信取消按钮的用途是允许我们返回 我单击确认按钮因为它是确认,因此我不需要一条 你真的,真的确认你要这样做吗 的确认对话框。

在你压缩数据库前,你不能运行 RI 生成器。

我确实砂需要它备份我的当前的储存过程为一个叫做 RISP. OLD 的我总是要自己删除它的文件。

生成的代码中至少有一处 BUG ,应该用正确的功能来防止它。关于该 BUG 的细节,参见 VFP 中的错误处理一文 。

它生成的代码既多又缺少注释。要试着理解这些代码是做什么的是一个艰苦的过程,而且如果是在一个复杂的数据库中,RI 代码可能超过 VFP 编译后的代码的 64K 限制。如果你看过 Jim Booth 和 Steve Sawyer 写的 Visual FoxPro 6.0 应用程序开发的有效技术 一书,你会有一个较好的快速,简洁的子程序来维护 RI。Steve 的 NEWRI 子程序是数据驱动的,因此它是在运行时而不是在生成时检查哪些规则需要强制执行。结果是清楚的,紧密而不是 VFP RI 生成器生成的大实用的代码。

它所支持的规则仅是忽略、级联和限制。你所需要的其它选项又怎样呢,例如无效 设置子表的外部关键字为 . NULL. 或指定一个新值 设置外部关键字为的默认值

修正这些项相对容易些,因为我们有 RI 生成器的源代码。我建立了 SFRIBUILDR. APP,一个 VFP RIBUILDR. APP 的替换物。我没有实现附加的规则 上面提及的最后一项 但是我处理了其它的所有项。我首先建立了 SFRIBUILDR 项目 它在解压后的程序的主目录下的 RI 目录中 并添加了 VFP 的 RIBUILDR. VCX 在 VFP 向导源程序所在目录下的 BUILDERS IBUILDR 目录中 。

接着,我派生 VFP RI 生成器类到 SFRIBUILDR. VCX 类库中的 SFRIBuildr。我忽视 overrode 了 Load 方法以便在数据库中有被删除的记录时不会发生错误 你可以参见我注释的已存在的代码 。我忽视了确认按钮的 Click 方法,以便不出现确认对话框和不复制当前的储存过程到 RISP. OLD 中,并修复了生成的代码中的错误。同时,如果它检查到 NEWRI. PRG 存在于你的系统中 与 SFRIBUILR. APP 在同一目录 ,它会把该代码放入数据库的储存代码中而不会生成其它代码。这个修改需要一个新的方法 RIMakeNewTr 用一个不同的名字建立触发器 _RI_Handler 而不是原来的 RI 生成器的生成的 _RI_ <action> _ <table> ,如 _RI_Delete_Customer 。因为这些方法中有大量的代码,在这里就不写出来了 但是,如果你查看提供的源代码,你会看见我确实只注释了很少的代码行,并添加了很少的代码。

最后,我从 BUILDERS IBUILDR 目录中复制 RIMAIN. PRG 到我的生成器所在的目录,添加它到项目中,并使它面为主程序。我修改了该程序所指向的类库,使它能找到我的 VFP RI 生成器类的子类所在的类库 SFRIBUILDR. VCX 并自动注册该生成器到 VFP BUILDER 表中 如果是直接运行该 APP 。

生成并运行 SFRIBUILDR. APP 导致该文件作为 RI 生成器注册取代了通常的 RIBUILDR. APP。要看这一点,按以下步骤进行:

移动本文附带的示例文件到 WIZARDS I 目录。

如果你有 Steve 的 NEWRI. PRG,将其复制到该目录。

运行 COPYDEMO. PRG。该程序将复制 VFP TESTDATA 数据库到 DATA 目录中,这样我们不会改动原始的数据库。

运行 DELETETEST. PRG。这会演示一个 VFP RI 生成器生成代码的流程。第一次浏览显示 ALFKI 客户的几个订货,然后试着删除该客户。因为 CUSTOMER 到 ORDERS 表间存在着级联删

除,且 ORDERS 到 ORDITEMS 表间的规则为限制,该客户将不能被删除。但是,请注意错误对话框出现了六次　而不是你所希望的一次　,然后另一个浏览显示该客户已被删除　CUST_ID 是 . NULL. 。

因为我们已经把数据弄乱了,再一次运行 COPYDEMO. PRG 重新复制 TESTDATA 数据库到 DATA 目录中。

以独占方式打开 DATA ESTDATA 数据库。

生成并运行 SFRIBUILDR. APP 来注册它为 RI 生成器。

打开数据库设计器,选择修改参照完整性,然后在 RI 生成器对话框中,单击确认按钮。注意没有出现确认对话框。选择修改储存过程并注意代码的不同　如果你有 NEWRI. PRG,该代码会放入储存过程中　否则,生成的 RIDelete 和 RIUpdate 方法代码中的 bug 已修正 。如果你有 Steve 的 NEWRI. PRG,修改 CUSTOMER 表,并注意各触发器方法的名字。同时,你不会看到 RISP. OLD 文件了。

再次运行 DELETETEST. PRG。这一次,你只会看到一次错误信息且 ALFKI 客户没有被删除。

创建一个更好的升迁　Upsizing　向导

Jim Falino 在使用升迁向导时失败了,因此他创建了一个具有他所希望的行为的版本。

向导在每一个具有至少一个数值,浮点,通用 或备注字段的表中,建立一个 timestamp 字段。Jim 不希望向导自动建立任何 timestamp 字段,因此他移去了该功能。

向导为具有主关键字段的表自动建立一组索引。这在有时是令人满意的,Jim 决定在默认情况下不建立组索引,因此在建立主关键字时,他加入了 NONCLUSTERED 子句。

由于数据库对象名字在 SQL Server 中必须唯一,对于具有非唯一名的东西不能升迁。例如,如果你为多个表的主关键字段建立了相同索引名的索引则你不能升迁关键字索引　如使用　ID　为每一个表的主关键字段 。Jim 的方案是重命名主关键字索引为 UQ_ <table name> ,因为名字在在 SQL Server 中是不重要的。

可用的表的列表框滑有排序　表出现的顺序是它们在 DBC 中的顺序。在一个拥有上百个表的 DBC 中,要找到一个特定的表是件痛苦的事。Jim 设置了 SuperMover 对象的 SortLeft 属性 用于指明左边的列表框是否需要排序　为 . T.

向导可以为 DBC 中的每一个表,用相同和定义建立一个远程视图。但是,但是它命名视图与原表同名,并重命名表为一个唯一名。由于有些人喜欢对视图使用命名约定 例如,在原表名前加上一个 V　,Jim 修改了向导中的这一点。

Jim 修复了一个问题 :向导有时会重新排列字段复合索引的顺序。

VFP 数值型字段类型升迁为浮点型,即使 SQL Server 中有数值型数据类型也一样。这致使 VFP 中多于两位小数的数值型字段被截短,并且 VFP 数值字段没有小数点的会升迁为两位小数。

15. 1. 9　创建你自己的向导

Rod Paddock 和 John Petersen 也在他们的书中也提供了一些使用 VFP 向导类来建立你自己的向导的细节。这样做的优点是,有人已经建立了所有的 引擎 来管理向导处理,且你的向导将与其它的 VFP 向导具有相同的感观。

用 BuilderD 创建生成器

在查看 VFP6 中的 FoxPro 基本类时　FFC ,我注意到一件事是它们都具有叫做 Builder 和 BuilderX 的自定义属性,并且各个类的 BuilderX 是设置为 = HOME　+　Wizards uilderD

BuilderDForm 。我知道这些属性的作用 他们告诉 VFP,这个类的生成器名 ,但为什么每一个类都要指定相同的生成器呢 更有趣的是,各个类以不同的选项调用类似的生成器表单当使用相同的类时将会发生什么情况呢 如图 1 所示。

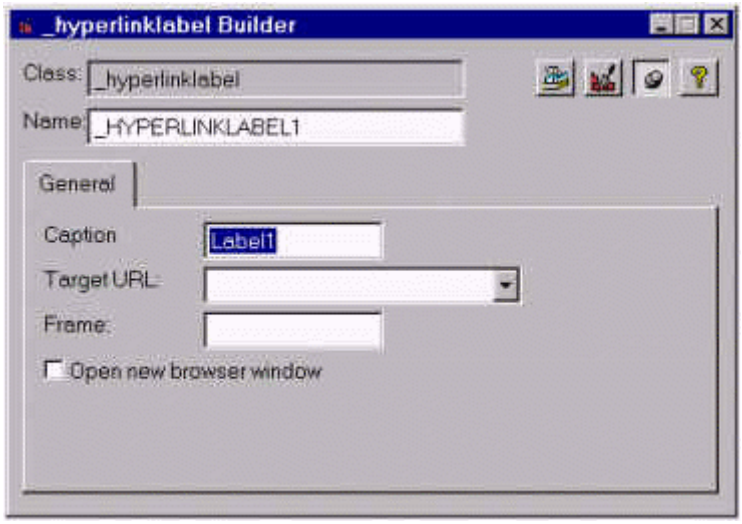


图 1

首先介绍一点背境。可能正象你可能意识到的,VFP 生成器可能以多种方式调用,但最常见的是可能是在对象上右击鼠标并从出现的菜单上选择生成器。这会使 BUILDER. APP 被执行。BUILDER. APP 检查是否选定的对象 我们称为 目标对象 具有 BuilderX 属性,如果有,运行该属性指定的程序或实例化该该属性指定的类 如果生成器是一个类,将指定一个类库,逗号及类名 。如果它没有 BuilderX 属性但有一个 Builder 属性, 生成器运行该属性指定的程序或实例化该该属性指定的类 我们将看到为什么会同时用两个属性指定生成器 。如果两个属性都不存在,将使用该对象基类的默认生成器。 你可以在你的类中建立自定义的 Builder 和 BuilderX 属性 甚至在你的基类中 然后为每一个类填写适当的生成器名字。要使用两个属性的理由是 BuilderX 为该类指定一个自定义生成器 而 Builder 为一组公共的类 如 comboboxe 或 grid 指定一个所需的生成器。正如我们稍后会看到的一样, 我们可以在 BuilderX 属性中指定的生成器中, 单击一个按钮来调出在 Builder 属性中指定的生成器。 因此, 我们可以很容易地在 Builder 和 BuilderX 属性中指定生成器。而不是花很多时间来建立你自己的生成器,特别对于不常使用的类。

BuilderD

你可能意识到了 BuilderB 技术 用来建立生成器是容易的和快速的。BuilderB 是一组你可以派生来建立你自己的生成器的类。你可以为你想用生成器维护的目标对象的各属性添加一个控件到生成器子类。虽然 BuilderB 使得建立生成器更容易,但你必须为每一个你想用生成器管理的类建立一个新的生成器子类。对于我们这样的懒人来说, 幸运的是, Ken 以数据驱动的方式增强了 BuilderB。这种新技术叫做 BuilderD , D 的意思是 动态 Dynamic , 可以在 Ken 的网站 www. classx. com 找到, 也包含在 VFP 6 中 在 VFP 主目录下的 WIZARDS 目录中的 BUILDERD. VCX 。 BuilderD 由一系列的类组成, 但主要的一个是 BuilderDFor 这是一个数据驱动生成器表单 注意这是 VFP6. 0 的基本类中指定的 BuilderX 类 。正如你在图中看到,该表单中的文本框用于输入目标类的名字和类名, 按钮提供的功能可以调出类浏览器和显示帮助, 一个页框、

两个页面,但没有用于管理属性值的控件,如图 2 所示。

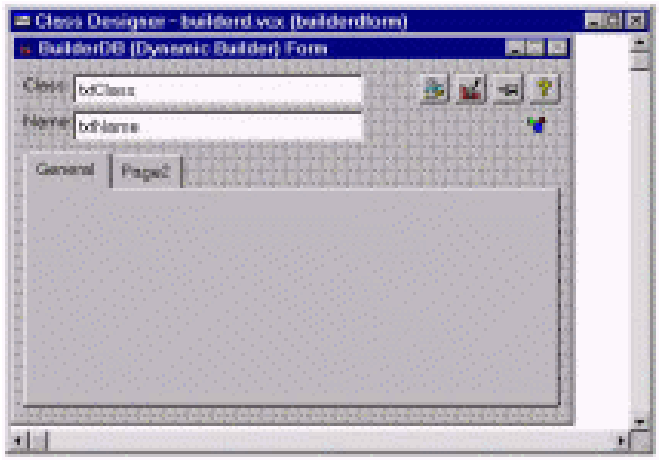


图 2

以下是该表单在实例化时,如何组合适当的控件:

Init 方法调用 SetObject 方法,它调用 AddObjects 方法 该代码事实上是在 BuilderDForm 的父类 BuilderBaseForm 中 。

AddObjects 方法调用来自 BuilderDB 类的表单上的 oBuilderDB 对象的 AddObjects 方法。

BuilderDB. AddObjects 是一个复杂的方法,但其基本操作是打开生成器定义表 默认情况下是在 VFP 主目录下的 WIZARDS 目录中的 BUILDERD. DBF,但可以修改 cBuilderTable 属性来指定另一个表 ,查找目标对象类的记录,查找所有相似的记录,并使用该记录中的信息在页框中的一个或多个页上建立控件。这些控件是基于 BUILDERD. VCX 中的类的, 诸如 BuilderCheckBox 和 BuilderTextBox,它们知道如何绑定属性到目标对象。

这些步骤的结果是,一个生成器可以管理目标对象的一个或更多的属性。一个指定的生成器可以事实上做得比这些还要多,例如放置代码到目标对象的方法或对象的容器中,介是目前我们只使用最简单的方法。

BuilderD 表

让我们进一步了解生成器定义表 BUILDERD. DBF,因为理解它的结构是建立你自己的生成器的关键。显示了 BUILDERD. DBF 表的结构 在该表中, property control 意思是生成器中的、维护一个目标对象的指定属性的控件。

字段 用途

TYPE 定义记录类型。它包含了两个内容 如果这是一个类记录它是“CLASS”,如果这是一个属性记录则它是“PROPERTY”。

ID 记录识别符,通常是类的名字。

LINKS 连接到该记录的其它记录的 ID 值列表 以回车符分隔 。该字段的更多细节在下面描述。

TEXT 如果这是一个“CLASS”记录,则为生成器表单的标题提示。

如果这是一个“PROPERTY”记录,则为属性控件的标题提示。

DESC 属性控件的状态条文本。

CLASSNAME 对于“CLASS”记录,是类的名字。BuilderDB 搜索该字段中指定的、一个包含目标

对象类的记录。

对于“PROPERTY”记录，是要为属性控件实例的类的名字。若该字段为空，将使用 BUILDERD.VCX 中的默认的类 逻辑属性是 BuilderCheckBox 其它属性是 BuilderTextBox 。你指定的任何其它类都必须是 BuilderD 类的子类，因为这些类具有被 BuilderDForm 使用的特殊的属性和方法。

CLASSLIB 包含有 CLASSNAME 中指定的类的类库。该属性可以包含一个表达式 例如 HOME + “WIZARDS UILDERD.VCX” 或一个常量 在这种情况下，在表达式事放上方括号。对于“PROPERTY”记录，如果该字段为空且指定了 CLASSNAME，则假定为 BUILDERD.VCX。对于“CLASS”记录，BuilderDB 搜索一个包含有与该字段内容值相同的记录 如果必要的话，在求值后作为目标对象的 ClassLibrary 属性，保留该值为空可建立一个类的生成器而不必担心它是在哪一类库中。

MEMBER 对于“CLASS”记录，其值为空。对于“PROPERTY”记录。如果为空，ID 必须包含属性名。

HELPFILE 包括有该类的帮助内容的 CHM 文件名。如果为空，就使用当前帮助文件。

HELPID 帮助主题 ID。

TOP 属性控件的 Top 设置。如果为 0，BuilderDForm 将该控件放在先前一个控件的下面 页面上的第一个控件放置在 BuilderDB.nTop 属性中指定的位置 。

LEFT 属性控件的 Left 设置。如果为 0，BuilderDForm 将该控件放在先前一个控件的左边 在 BuilderDB.nLeft 中指定 。

HEIGHT 属性控件的 Height 设置。如果为 0，就使用该控件的默认 Height 值。

WIDTH 属性控件的 Width 设置。如果为 0，就使用该控件的默认 Width 值。

ROWSRCTYPE 如果类使用的属性控件 在 CLASSNAME 中指定 是一个 combobox，该 combobox 的 RowSourceType 设置。

ROWSOURCE 如果属性控件是一个 combobox，该 combobox 的 RowSource 设置。例如，如果 ROWSRCTYPE 是 1 值 ，ROWSOURCE 将包含一个用于 combobox 的、以逗号分隔的值的列表。

STYLE 如果属性控件是一个 combobox，该 combobox 的 Style 设置。

VALIDEXPR 一个用于验证属性值的表达式。

READONLY 如果属性控件中只读的则为 .T.。

UPDONCHNG 如果属性控件的值写到目标对象的属性后被修改，则为 .T. 就是说，交互式方式 。

UPDATED 决定记录最后修改 未被 BuilderD 使用，仅是一个信息 。

COMMENT 关于记录的注释 未被 BuilderD 使用 。

USER 用户信息记录 未被 BuilderD 使用 。

下面显示了两个生成器的记录，VFP6.0 的基本类 _HyperLinkBase 和 _HyperLinkLabel 类。我没有显示 BUILDERD.DBF 表中的所有字段，出于空间的考虑 仅列出与讨论有关的字段。

```
TYPE ID LINKS CLASSNAME CLASSLIB
CLASS _HyperLinkBase Ctarget
cFrame
lNewWindow _HyperLinkBase HOME + FFC—Hyperlink.vcx
CLASS _HyperLinkLabel Caption
```

_HyperLinkBase _HyperLinkLabel　HOME　+　FFC－Hyperlink. vcx

. _HyperLinkBase 和 _HyperLinkLabel 类的生成器记录

TYPE ID TEXT CLASSNAME ROWSRCTYPE ROWSOURCE

PROPERTY cTarget Target URL：BuilderComboBox 1 www. microsoft. com / vfoxpro

PROPERTY cFrame Frame：

PROPERTY lNewWindow Open new browser window

PROPERTY Caption Caption

_HyperLinkBase 和 _HyperLinkLabel 生成器管理的属性定义记录

在上面的 _HyperLinkBase 记录中，我们看见 CLASSNAME 和 CLASSLIB 指定了该生成器使用的类　注意 CLASSLIB 包含了一个在运行时求值的表达式而不是一个硬编码值　，在 LINKS 中列了出该生成器可管理的类的属性。显示在表 3 中的 cTarget PROPERTY 记录指明该属性将被 BuilderComboBox 控件管理，且该 BuilderComboBox 的 RowSource 包含一个 Microsoft VFP 网站的 URL　目前是 msdn. microsoft. com / vfoxpro 。cFrame 将被 BuilderTextBox 对象管理 因为它是一个字符串属性且该类没有定义　，lNewWindow 将有一个 BuilderCheckBox 对象　因为它是一个逻辑属性　。

看起来是如此简单，对吧　好了，LINKS 字段事实上可以更加复杂。首先，假如在 LINKS 字段中指定的类记录的 ID 没有匹配记录，该 ID 就被假设为属性名。

这样，你可以简单的在 CLASS 记录的 LINKS 字段中指定它管理的属性来实际建立一个生成器。当然，你必须保持属性的 captions 和属性的名字相同，没有状态条文本、默认的类和属性大小，但这些对于一个快速的和 dirty 生成器来说都不算太坏。

第二个复杂之处是，CLASS 记录的 LINKS 字段中指定的 ID 可以指向另一个 CLASS 记录而不是一个 PROPERTY 记录。在这种情况下，该类　继承　所有指定类的连接。你可以在其中的 _HyperLinkLabel 记录中看到这一点　它的一个连接是 _HyperLinkBase，因此它不仅用该类的生成器来管理 Caption 属性　specifically 在它的 LINKS 字段中列出　，同时也管理 cTarget、cFrame 和 lNewWindow 属性，因为这些在 _HyperLinkBase 的 LINKS 字段中指定。

第三，PROPERTY 记录可以连接到另一个 PROPERTY 记录。在这种情况下，记录 继承 连接记录的所有非空字段。最后，如果你用@ <caption> 格式指定它，LINKS 可以包含生成器中的页框的标题　例如，@Properties 使用　Properties　作为页的标题　。

创建一个生成器

让我们在 TEST. VCX 类库中创建一个名为 TestCheck 的 VFP CheckBox 的子类作为开始。在该类中添加一个叫做 BuilderX 的自定义属性并设置它的值为 = HOME　+　Wizard uilderD BuilderDForm 。然后在该类上右击鼠标并从快捷菜单中选择生成器。我们得到一条 没有注册该类型的生成器 的错误。这也说得通，因为我们还没有定义它　尽管这令人不愉快，在稍后它将自动为我们建立一个　。按以下步骤做：

在命令窗口中打入 USE HOME　+　WIZARDS UILDERD　，然后打入 BROWSE。

从表菜单中选择　添加新记录　。

在 TYPE 字段中输入　CLASS　，ID 字段中输入　TestCheck　，LINKS 字段中输入　Enabled　，AutoSize 和　Caption　在输入各字段后按回车键　，TEXT 字段中输入　My Test Builder　，CLASSNAME 字段中输入　TestCheck　，CLASSLIB 字段中输入　TEST. VCX　。

关闭浏览窗口并在命令窗口中打入 USE 来关闭该表。

在 TestCheck 上右击鼠标并选择生成器。

酷不酷，嗯 完全是你自己的生成器，仅在一分钟左右就建立了！去掉 Enabled 的选择并输入一个不同的 Caption，并可看到这些属性已被修改。

注意不要被为属性建立记录所迷惑 Enabled 和 Caption 记录已存在于 BUILDERD.DBF 表中，因此我们只需重用它们，而且由于不存在 AutoSize 记录，BuilderD 只假设我们想管理那个属性。

让我们再建立一个并看看以多小的输入可以得到一个可工作的生成器。在 TEST.VCX 类库中创建一个名为 TestText 的 VFP TextBox 的子类，向它添加一个属性 BuilderX 并设置它的值为 = HOME + Wizards uilderD BuilderDForm 。在 BUILDERD.DBF 中添加一条记录并只指定 TYPE CLASS ，ID TestText ，LINKS ReadOnly 和 CLASSNAME TestText 我们可以跳过 CLASSLIB 但 CLASSNAME 是必须的 。关闭该表，然后调出该类的生成器。

这也是一个可以工作的生成器。单击生成器中的生成器按钮 它调出另一个生成器 如果 Builder 属性存在并非空，在 Builder 属性中指定的生成器，否则就是该类的基类的默认生成器，在此例中是 VFP TextBox 生成器 。这样，即使我们可以为一个类指定生成器，只要愿意，我们仍然可以访问更多的普通生成器。

预置 - 内建的生成器

我最近用 BuilderD 建立了两个生成器，一个为我的 SFGrid 类另一个为 SFPageFrame。SFGrid 生成器维护 DeleteMark 和 RecordMark 属性 你可以很容易地添加其它你可能经常修改的属性 但由于它们容易在属性窗口中修改，所以这不是我要建立该生成器的真正的理由。真正的理由是因为我喜欢使用 grid 的 columns 中的 SFGridTextBox 对象 SFTextBox 的一个子类，具有一小部分不同的属性设置，以提供它们在 grid 中出现的外观 ，并尽量用 SFGridTextBox 对象替换一般的 TextBox 对象。

因此我建立了一个按钮 SFBUILDERS.VCX 类库中的 SFGridTextBoxButton 类 来自动进行该工作。以下是该按钮的 Click 方法代码：

```
local loColumn
loControl
lcName
for each loColumn in Thisform.oObject.Columns
for each loControl in loColumn.Controls
if upper loControl.Class = TEXTBOX
lcName = loControl.Name
        loColumn.RemoveObject lcName
loColumn.NewObject lcName SFGridTextBox
 SFCtrls.vcx
endif upper loControl.Class = TEXTBOX
next loControl
next loColumn
wait window SFGridTextBox added to each column
timeout 2
```

BuilderDForm 有一个引用到目标对象保存在它的 oObject 属性，因此生成器表单上的许多控件可以引用或修改目标对象的东西。在这些代码中，Thisform.oObject.Columns 引用 grid 的 Columns 集

合被生成器影响。

这里有一个副作用：我怎样才能在我的生成器表单中得到该按钮　我可以为我的生成器在 BUILDERD 表中建立一个记录，但那会添加按钮到页框中的一个页上，而我确希望该按钮象其它的生成器按钮一样出现。

因此我建立了一个　button loader　类　SFBUILDERS. VCX 类库中的 SFBuilderButtonLoader 类，该类添加一个按钮到生成器表单然后返回 . F. 这样，它实际上没有实例化。SFBuilderButtonLoader 在 BUILDERD 表的当前记录的备注字段 USER 中查找　使用类实例化的记录　要添加到表单的按钮的类名和类库　注意下面代码中的 BUILDERD 是以　BUILDER　别名打开的　。对于 SFGridTextBoxButton，例如，我在 BUILDERD 表中建立一条记录，以　SFGridTextBoxButton　作为它的 ID 但　SFBuilderButtonLoader　是一个类，并在 USER 字段中输入　SFBuilders SFGridTextBoxButton　。这告诉 SFBuilderButtonLoader 添加一个 SFGridTextBoxButton 到表单中。以下是 SFBuilderButtonLoader 的 Init 方法代码：

```
local lcClass
lnPos
lcLibrary
lnTop
lnLeft
loControl
* 如果 BUILDERD 表已经打开且定位在要载入的按钮记录上
* 我们将从 USER 字段中读取类和类库名并添加该按钮
* 到表单。
if used  BUILDER  and
lower  BUILDER. CLASSNAME  = lower  This. Class
with Thisform
lcClass  =  BUILDER. USER
lnPos  = at  lcClass
if lnPos >  0
lcLibrary  =  left  lcClass  lnPos  -  1
lcClass  = substr  lcClass  lnPos  +  1
    else
lcLibrary  =
endif
* 添加按钮。如果我们成功了，在表单的按钮区找第一个打开 裂缝
. NewObject  lcClass  lcClass  lcLibrary
if type  .  + lcClass +  . Name  =  C
lnTop  = . cmdClassBrowser. Top  +
. cmdClassBrowser. Height  + 5
lnLeft  = . cmdClassBrowser. Left
for each loControl in . Controls
if loControl. Left  =  lnLeft and
```

```
loControl. Top  =  lnTop
lnLeft  =  lnLeft  +
. cmdClassBrowser. Width  +  6
if lnLeft  +  . cmdClassBrowser. Width >  . Width
lnLeft  =  . cmdClassBrowser. Left
lnTop  =  lnTop  +  . cmdClassBrowser. Height  +  5
endif lnLeft  +  . cmdClassBrowser. Width >  . Width
endif loControl. Left  =  lnLeft . . .
next loControl
* 设置按钮位置到 裂缝 位置,并使其可访问。
with . &lcClass
. Top  =  lnTop
. Left  =  lnLeft
. Visible  =  . T.
endwith
endif type  .  +  lcClass  +  . Name  =  C
endwith
endif used  BUILDER  . . .
return . F.
```

在 BUILDERD **表中** SFGrid **记录的** LINKS **字段中有** DeleteMark、RecordMark **和** SFGridTextBox，**因此该生成器获得这些控件。**

现在我可以为我的 SFGrid **对象调出我的** BuilderD **生成器，使用** VFP Grid **生成器　单击** BuilderD **表单上的生成器按钮　来快速建立** grid **中的列，然后单击我的添加** SFGridTextBox **按钮来让这些列使用** SFGridTextBox **对象。**

对于 SFPageFrame，**我也希望相似的功能：一个添加代码到页框中各页的** RightClick **方法的按钮　这允许在页面上右击鼠标明时为页框或整个表单提供快捷菜单。象** SFGrid **生成器一样，我在** BuilderD **表中建立了一个记录来实例化一个** SFBuilderButtonLoader **对象，然后添加一个** SFCode-PageButton **对象到生成器表单。以下是添加代码到各页的** SFCodePageButton **的** Click **方法：**

```
local lcCode
loPage
lcCurrentCode
lcCode  =  This. Parent. ShowMenu  +  chr  13
for each loPage in Thisform. oObject. Pages
lcCurrentCode  =  loPage. ReadMethod  RightClick
if not lcCode $ lcCurrentCode
loPage. WriteMethod  RightClick  lcCurrentCode  +
iif  empty  lcCurrentCode  chr  13  +  lcCode
endif not lcCode $ lcCurrentCode
next loPage
wait window  Code added to each page  timeout 2
```

要添加这些新的生成器到你的系统中，复制 SFBUILDERS. VCX 和 VCT　在本文档的源代码文件中可以找到　到一个目录，然后打开示例文件提供的 NEWBUILDERS 表，并修改 CLASSLIB 字段到包含路径的 SFBUILDERS. VCX 文件名。打开 BUILDERD 表，然后 APPEND FROM NEWBUILDERS 表，添加 Builder 和 BuilderX 属性到你的类中，并在 BuilderX 中输入 = HOME + Wizards uilderD BuilderDForm 。

其它生成器类

我早些时候注意到，SFBUILDERS. VCX 中的 BuilderCheckBox 和 BuilderTextBox 类都是派生自同一个 BuilderD 类。在附加的右击鼠标行为中，这两个类具有 nTop 和 nLeft 属性，当这些属性改变时，用 assign 方法设置它们的自定义的 lMoved 属性为 . T. 。这允许我们查觉到什么时候一个属性控件被移动了 在属性控件生成器中的 Left 和 Top 是绑定至 nLeft 和 nTop 而不是直接绑定到 Left 和 Top 。只有当一个属性控件移动时，我们保存它的 its Left 和 Top 值到 BUILDERD 表。 SFBUILDERS. VCX 中的加一个类是 SFPropertyCaption。该控件用于管理属性控件的 caption。

为什么用一个特别的类来做这件事　为什么不用一个普通的 BuilderTextBox 对象

理由是如果属性控件是一个 BuilderCheckBox，它有一个 Caption 属性，因此控件管理该属性。但是，如果属性控件是一个 BuilderTextBox，它没有 Caption 属性但确有一个组合的 label 对象，因此控件必须管理对象的对象的 Caption 属性。由于 BuilderD 类只管理单个对象的属性，SFPropertyCaption 不得不超越 override 少量方法来允许它处理这种情况。

SFBuilderPropertyComboBox 类是 BuilderComboBox 的一个子类。它提供一个目标对象的所有可写属性的一个列表 用 AMEMBERS　得到所有属性的列表然后检查各自的 PEMSTATUS　来排除只读的 。虽然它的主要目的是修改属性控件的 cProperty 属性 用于检查要管理的属性 ，它有两个有趣的行为。

首先，如果属性控件是一个 BuilderTextBox 但你只修改了它管理的逻辑属性　如 Enabled ，它移去 BuilderTextBox 及组合的 label 对象，并放一个 BuilderCheckBox 在这个地方。如果你改变了一个逻辑属性为另一个类型它会做相反的事。

第二，如果你输入了一个不存在的属性名，你会被提示建立该属性。如果你同意，生成器使用 AddProperty 来添加新属性到目标对象。我的确不推荐在一个拖放到表单中的对象上这样做，因为这等于实例化编程　如果你真的需要一个新属性，你可以考虑用子类来代替。

总之，这是一个经一个步骤在类中建立新属性并用生成器管理它的快速方法。 SFBuilderAddButton 是一个为了属性控件使用 SFBuilderButtonLoader 类而添加到生成器表单的按钮类　派生自 BuilderCommandButton 。SFBuilderAddButton 的 Click 方简单地调用属性控件所在的生成器表单的 AddPropertyControl 方法来建立一个新的属性控件，然后转换当前生成器表单来管理新属性控件。

这意味着你可以单击添加属性按钮添加一个新属性到生成器，并在生成器中管理它。这是一种添加多个属性控件的快速方法。

试一试 ：让我们看一看 SFBuilderBuilderForm　它实际上比对它的描述更易于使用。在 TEST. VCX 中建立一个叫做 MyTestText 的 VFP TextBox 类的子类。添加一个 BuilderX 属性并设置它的值为 SFBuilders SFBuilderBuilderForm ，然后调出该类的生成器。

注意尽管我们没有在 BUILDERD 表中建立任何记录，我们仍然得到了一个生成器表单 当然，表单上没有控件，但我们马上会改变这一切 。

这是因为 SFBuilderBuilderForm 在不能找到该类的记录时，自动地为该类在 BUILDERD 表中建立了一个 CLASS 记录。

单击添加属性控件按钮。你会注意到一个文本框和标签出现在生成器表单上，但随后另一个生成器表单出现在该生成器表单的面上。这个新的表单是我们刚添加的属性控件的生成器。在属性 combobox 中，选择 Tooltiptext 并修改 Caption 为 Tool Tip Text 和 Width 到 250。移动该生成器表单在一旁并注意原表单上的属性控件也同样修改了。

单击第二个生成器表单上的添加另一属性按钮并注意另一个属性控件已添加到原生成器表单中，并且该生成器表单现在可以管理它了。为属性选择 Statusbartext 并修改 Caption 为 Status Bar Text 和 Width 到 250。添加另一个属性控件，并为该属性选择 Readonly 并修改 Caption 到 Read - Only 这时，注意原生成器属性控制改变了 checkbox。关闭新的生成器表单。图中显示了我们建立的生成器，图中显示了属性控件生成器，如图 3、图 4 所示。

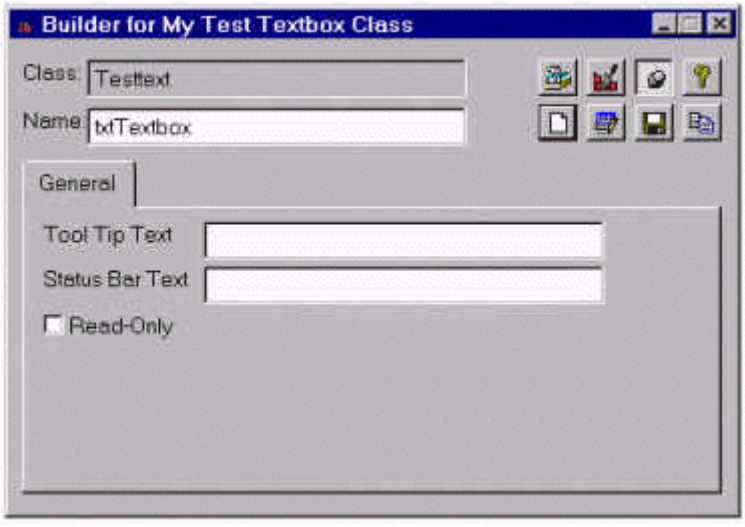


图 3

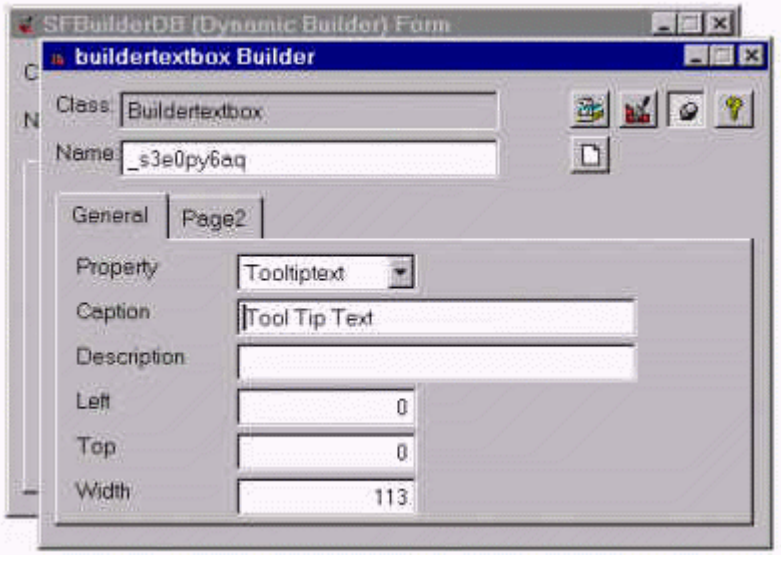


图 4

要修改控件的属性，在控件上右击鼠标并从快捷菜单中选择修改属性修改控件 你以前看到过的相同的属性控件生成器会出现。要移去属性控件，从菜单中选择移去属性控件。要重置目标对

象的该属性的值为默认值,选择重置为默认值。

让我们修改生成器表单的标题为更为合适的东西。单击修改生成器标题按钮并在随后出现的 SFBuilderBuilder 生成器表单中输入　我的测试文本框类生成器　。关闭第二个生成器。

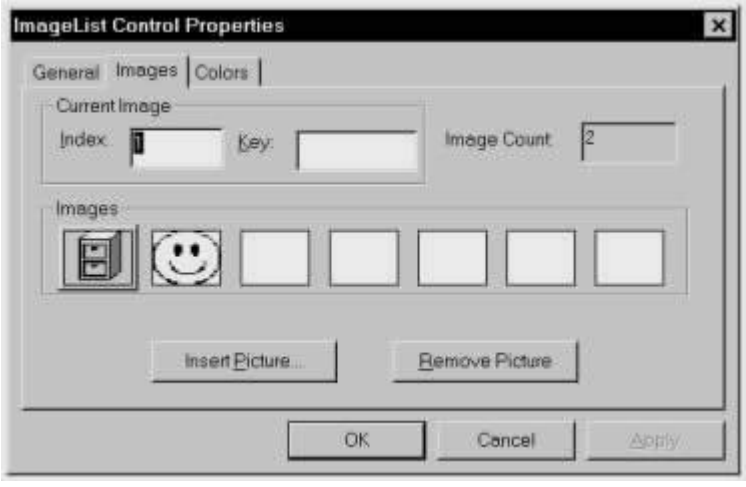
如果你在关闭生成器时没有保存，下一次你再调用 MyTestText 对象的生成器时，你的设置不会起作用。若要保存生成器的定义到 BUILDERD 表中,单击保存生成器按钮。如果你想导出生成器定义到另一个表中，单击导出生成器按钮并在出现的对话框中输入文件名。然后你可以发送该表给一些人,他们可以将其导入他们的 BUILDERD 表并访问你建立的生成器。

15.2　使用 Visual FoxPro 的 ActiveX 控件

15.2.1　ImageList 控件

ImageList 控件非常简单，但因为我论述的其它控件要用到它　所以我们将首先讨论它。ImageList 控件预载入一些图象 因此其它控件 如 TreeView 和 ListView 控件　可以拥有图象资源。此外,它自己并不做任何用有的事情。

虽然该控件有少量属性,事件和方法,你可能不会用到它们。通常,你会使用 ImageList 控件的属性表　可在其上右击鼠标调出　来为另一个控件载入你需要的图象并设置图象的大小和颜色。



要载入图象到 ImageList 控件，简单地从 ActiveX 控件工具条上拖动一个到表单上，修改它的名字,调出 ImageList 控件的属性表,选择 Images 页,并插入你需要的图形。各图形的索引值　从 1 开始　用于其它控件选择一个图象。例如我们将很快看到的 TreeView 控件,肯的一个 Add 方法来添加一个新节点,并使用了相关 ImageList 控件的索引号作为节点使用的图象。

15.2.2　TreeView 控件

VFP 5.0 的新的 ActiveX 控件 TreeView 是一个有力的、可视的、吸引人的工具　你可以将它使用于许多应用程序。 但是　其使用的复杂和技术文档的简单使很多人望而却步。本文探索了一些使用 TreeView 的有用的技术。

VISUAL FoxPro 5.0 包含了很多新的 ActiveX　原来的 OLE　控件　这些控件可以为你的应用程

序增加很多新功能。这些控件包括 TreeView、ListView、StatusBar、和 CommonDialog 控件 允许你为你的应用程序建立 Windows 95 风格的界面。ActiveX 控件很容易找到 在表单控件工具栏中的查看类菜单中选择 ActiveX 控件 会出现 30 个新的控件。简单的从工具条拖放一个控件到一个表单中 就象使用 VFP 自身的控件一样 给它一个名字 设置一些属性 等等… 在附加到 VFP 的属性表单 每一个控件都有一个定制的属性表。要访问这个属性表 在控件上右击鼠标并从出现的快捷菜单中选择适当的项。

15.2.3 Common Dialogs 控件

如果你用过 FoxPro GETFILE 和 PUTFILE 函数。你可能对它们多少有些遗憾：

没有办法改变对话框的标题。

PUTFILE 函数总是询问用户是否想复盖已存在的文件。

使用两个函数时。指定的路径必须存在 否则将显示一条错误信息。

为了得到更大的灵活性。你需要使用随 VFP 5 同时发布的 Common Dialogs 控件 在 INDOWS YSTEM 目录下的 COMDLG32.OCX 中 。该控件称为 Common dialogs 是因为它可以显示文件。颜色。字体和打印对话框。所有这些对话框提供了比 VFP 的相同功能更大的灵活性。例如。当在 VFP 中使用 GETFONT 和 GETCOLOR 函数时。你控制不了所有的东西 如非 TrueType 字体是否可用或用户是否可自定义颜色。Common Dialog 中的字体和颜色对话框就有这种功能。

由于时间限制。在本章中我们只集中讨论文件对话框。如果你想知道关于颜色 字体和打印对话框的更多信息。请参阅 ActiveX 控件的帮助文件。注意 Common Dialogs 控件没有出现在帮助文件的目录页中。但可以在选定控件后按下 F1 键或在帮助索引中搜索 Common Dialog 来找到它们。

15.2.4 Calendar 控件

Calendar 控件位于 MSACAL70.OCX 中。帮助文件名为 MSACAL70.HLP。该控件提供了在你的应用程序中包含日历的能力。该控件的一个显而易见的用处是当用户在日期型字段上右击鼠标时显示一个日历让用户选择一个日期值。

方法和事件

Calendar 控件中的方法主要用于编程地控制日期，包括 NextDay，NextWeek，NextMonth，PreviousDay，PreviousWeek 和 PreviousMonth。当然也可以提供按钮或其它方法来调用这些方法，但由于用户可以在日历中单击各种控件来修改日期，所以我没有在这方面作过多的探索。

除 Click，DblClick 和 KeyPress 这样的常用事件，Calendar 控件还有 AfterUpdate，BeforeUpdate，NewMonth 和 NewYear 事件，这些事件允许你在用户改变了某些东西后，执行一些可能需要的特殊的处理。我想你会用到的最常用的事件是 DblClick，该事件可用于当用户选择了一个日期后，释放或隐藏 Calendar 控件。

属性

Calendar 控件的属性比它的方法和事件更有趣。你可能想设置一些属性，如颜色 BackColor，DayFontColor，GridFontColor，GridLinesColor，TitleFontColor ，字体 DayFont，GridFont，and TitleFont ，和控件的其它初始显示属性 DayLength，FirstDay，GridCellEffect，MonthLength，ShowDateSelectors，ShowDays，ShowHorizontalGrid，ShowTitle，and ShowVerticalGrid ，这些属性可以通过右击菜单从 Calendar 控件属性表中进行访问。Value 属性包含在日历控件中选定的日期，Day，Month 和 Year 属性包含日期的相关部分的值。

通常，你会设置控件的 Value 来指定一个日期 例如，在控件的 Init 事件中 ，该日期是被高亮显示的默认日期，在用户选定一个日期后，可以从 Value 中读取用户选择的日期值。

示例

我创建了一个包含日历控件和一些按钮的容器类 源代码中的 ACTIVEX. VCX 类库中的 SFCalendaron 类 和一个 SFDateSpinner 类 源代码中的 CONTROLS. VCX 类库中 ，适用于在用户从右击菜单选择 日历 时实例化 SFCalendar。

SFDateSpinner 类是一个容器类，拥有一个文本框和数码器 只有上下键头是可访问的 。数码器用于增加和减少文本框中的日期值，文本框的 KeyPress 方法中的代码模仿快速填充日期按键。当用户在文本框上右击时，调用 ShortcutMenu 方法来显示一个快捷菜单 采用硬编码并封闭于控件的该方法中 。如果用户从菜单中选择了 日历 ，会调用 ShowCalendar 方法。该方法从 cCalendarClass 属性 该属性的默认值是 SFCalendar 是指定的类中，实例化一个对象。NEWOBJ. PRG 用于确保存该类所在的类库是打开的。

SFCalendar 控件接收一个对象参数，因此它可以在对象释放前修改对象的值为用户选择值。当用户双击一个日期或选择 保存 或 退出 按钮时，该对象被释放。

15. 2. 5　ProgressBar 控件

ProgressBar 控件位于 COMCTL32. OCX 中，其帮助文件是 CTRLREF. HLP。该控件给我们一个 Windows 95 风格的进度条，就象你从一个驱动器中复制一个大的文件到另一个驱动器时 Windows 95 所显示的进度条一样。该控件可用于较费时的操作而你又想向用户显示处理进程时。示例包括执行长的计算 如工资表计算 ，在打印前执行的复杂查询，保存记录等。进程条可以在处理了每条记录或某一批记录时更新，或在任务的各步中进行更新。

方法和事件

ProgressBar 控件响应一些与其它 VFP 控件相关的方法和事件：Click，Drag，DragDrop，DragOver，MouseDown，MouseMove，MouseUp，Move，ShowWhatsThis 和 ZOrder。

属性

与 Calendar 控件相比，ProgressBar 控件的属性比其方法或事件更为有趣，主要是因为它是视觉控件。许多影响控件外观的属性更易于在设计时从 VFP 的属性表中或 ProgressBar 控件的属性表中进行设置。这些属性包括 Align 决定控件位置是否可移动或它是否自动靠向表单的上，下，左或右边沿 ，Appearance 平面或立体 和 BorderStyle。

我们更感兴趣的是运行时的属性 Min,Max 和 Value。Min 和 Max 提供了控件值的范围,默认值是 0 到 100。控件的条的长度是由 Value 属性控制。

示例

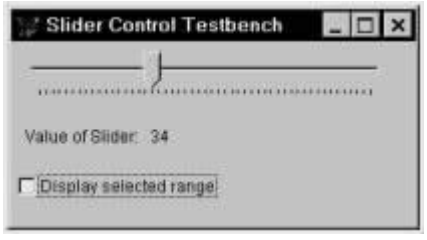
源代码中的 ACTIVEX.VCX 包含一个 SFThermometer 类。该类是一个带有进程条,一些标签和一个取消按钮的容器类。它的自定义方法 SetTitle 和 SetMaximum 用于设置 Main 标签的标题和 ProgressBar 控件的最大值。要更新温度计,用 Update 方法。它接收两个参数:温度计的当前值和表单中的 Current task 标签的标题。传递到 Update 的值的转换取决于表单的 lPassPercent 属性的设置。如果 lPassPercent 是 .T.,Update 期望一个百分比值 如果 lPassPercent 是 .F.,如果 lPassPercent 是 .T. 希望一个值并用该值和最大值的比来计算百分比。

SFThermometer 使用了一种非常有趣的技术来允许从一个 Hard 循环中中断。Update 方法检查是否鼠标移动到了取消按钮上及该按钮是否被按下。如果是,它用新的 doevents 命令来允许对事件进行处理 也就是说,取消按钮按下 ,然后设置一个用户取消处理的标志。循环调用 Update 方法检查该标志以决定是否需要继续处理。我原先在 Update 方法中使用 doevents 来允许按下取消按钮 而不是检查鼠标位置和按下状态 ,但这样要花大量的时间!

PROGRESS.PRG 是一个示例程序用于展示 SFThermometer 的使用。PROGRESS 表单只是有一个 SFThermometer 容器,PROGRESS.PRG 运行该表单来显示温度计。

15.2.6 Slider 控件

Slider 控件在 COMCTL32.OCX 中,其帮助文档是 CTRLREF.HLP。Slider 控件与音响中的音量控制滑动块相似;它用一个条提供控制的范围值和一个可以沿着条拖动的指针来指示选定值。该控件常用于输入数值型的值,但更多的是用于 定位 或 性质 对话框类型而不是数据输入,TextBox 或更适于数据输入。



方法和事件

Slider 控件响应一些与 VFP 的控件的方法和事件相同的方法和事件:Click,Drag,DragDrop,DragOver,GotFocus,KeyDown,KeyPress,KeyUp,LostFocus,MouseDown,MouseMove,MouseUp,Move,Refresh,SetFocus,ShowWhatsThis 和 ZOrder。

Change 事件与其它控件的 InteractiveChange 事件相似;它在 Value 属性改变时激发。在沿着条拖动滑杆时,Scroll 事件连续不断地激发。

ClearSel 方法清除控件的选定区域 见下述 。GetNumTicks 返回控件中的 tick 数。

属性

Slider 控件的许多属性影响控件的外观,它们在设计时很容易从右击菜单中调出 VFP 属性表或 Slider 控件属性表进行设置。它们包括 BorderStyle,LargeChange 当按下 PgUp 或 PgDn 或在 slider 的左边或右边单击鼠标时,slider 改变的 tick 数 ,SmallChange 当按下左右箭头时,slider 改

变的 tick 数，Orientation 横向或纵向放置，TickStyle ticks 沿着顶/左，底右边沿交叉的出现或不出现 和 TickFrequency。

我们在运行时更感兴趣的属性是 Min，Max 和 Value。Min 和 Max 提供控件值的范围，默认值是 0 和 100。slider 的沿着控件的位置由 Value 属性控制。

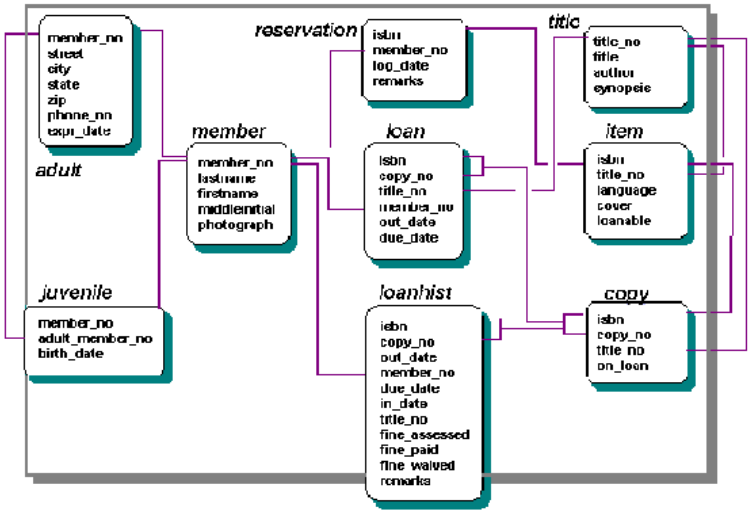
SelectRange 属性控制 slider 是否显示选定范围。如果 SelectRange 设置为 .T.，SelStart 和 SelLength 属性决定选定范围的起始位置和长度。

15.3　远程视图与 SQL pass - through 的区别

15.3.1　示例数据

本文中使用的示例数据来自 SQL Server 6.5 基本图书馆应用程序。该图书馆应用程序用于跟踪它的成员，它的图书和借出。图书馆的数据库方案如下图所示。

图书馆数据库方案：



表结构

图书馆应用程序的主要的表之一是 Member 表，图书馆中的每一个成员具有一条记录。一个有趣的情况是少年成员 其信息保存于 Juvenile 表中 必须有成人 其信息保存于 Adult 表中 担保。对于少年和成年人，分别设计两个单独的表 Adult 和 Juvenile。这种设计方法节约了磁盘空间，因为一但你知道担保人，所有的少年的地址信息就是多余的。另外，少年的期满日期与成年人的相同。将来，你不必介意成年人的出生日期而只需注意少年的出生日期，因为在他们的第 18 个生日后他们就成为了成年人。

以下 SQL Server 语句用于创建 Member，Adult 和 Juvenile 表：

```
CREATE TABLE member
   member_no     member_no     NOT NULL IDENTITY  1  1
lastname          shortstring   NOT NULL
firstname         shortstring   NOT NULL
```

```
middleinitial     letter          NULL
photograph        image           NULL
CREATE TABLE adult
  member_no       member_no       NOT NULL
street            shortstring     NOT NULL
city              shortstring     NOT NULL
state             statecode       NOT NULL
zip               zipcode         NOT NULL
phone_no          phonenumber     NULL
expr_date         datetime        NOT NULL
CREATE TABLE juvenile
  member_no       member_no       NOT NULL
adult_member_no   member_no       NOT NULL
birth_date        datetime        NOT NULL
```

Member 中的 member_no 字段在添加新记录时会由 SQL Server 自动生成。该字段是一个 Identity 列。起始值为 1 ,增量值也是 1。这样在表中输入的第一条记录的 member_no 值就是 1。对于后来插入到表中的记录 member_no 的值自动增加 1。当添加一条记录时如果客户没有指定 member_no 的值。SQL Server 自动维护它并询问客户使用什么值。

在 Adult 和 Juvenile 表中的 member_no 不是 Identity 列。这些记录中的值必须与 Member 表中相应的 member_no 值相匹配。当新记录添加到图书馆库时,一个记录首先会添加到 Member 表中。SQL Server 的全局变量 @@Identity 包含了自动生成的 member_no。然后添加到 Adult 或 Juvenile 表中的记录的 member_no 值将使用 @@Identity 中的值。

申明参照完整性

在早期版本的 SQL Server 参照完整性是通过使用触发器强制执行,这与 Visual FoxPro 强制参照完整性相同。SQL Server 6.0 添加了可申明的参照完整性,这允许你定义你自己的作为数据结构一部分的参照完整性规则。第一步是在各表中创建基本关键字约束,如以下代码所示:

```
ALTER TABLE member
ADD CONSTRAINT member_ident PRIMARY KEY CLUSTERED
 member_no
ALTER TABLE adult
ADD CONSTRAINT adult_ident PRIMARY KEY CLUSTERED
 member_no
ALTER TABLE juvenile
ADD CONSTRAINT juvenile_ident PRIMARY KEY CLUSTERED
 member_no
```

基本关键字约束创建一个唯一索引,用于强制 member_no 的唯一性。在示例中创建一组索引用于对数据进行物理排序。

定义可申明的参照完整性的第二步是在相关表之间创建外部关键字约束,如以下代码所示:

```
ALTER TABLE adult
ADD CONSTRAINT adult_member_link FOREIGN KEY  member_no
```

```
REFERENCES member  member_no
ALTER TABLE juvenile
ADD CONSTRAINT juvenile_member_link FOREIGN KEY
 member_no  REFERENCES member  member_no
ALTER TABLE juvenile
ADD CONSTRAINT juvenile_adult_link FOREIGN KEY
 adult_member_no  REFERENCES adult  member_no
```

第一个 Alter Table 定义了一个 Member 和 Adult 表之间的关系。这是一个一对一关系，虽然这里没有代码指明或强制是这种类型的关系。第二个 Alter Table 在 Member 和 Juvenile 表部定义了一个关系。最后一个 Alter Table 在 Adult 和 Juvenile 表之间定义一个关系。这是一个一对多关系。

15.3.2　使用视图尝试一

第一种创建图书馆应用程序的方法是使用远程视图。视图易于设置，可以参数化，因此可以一次只返回一个或少量的记录，支持行缓存和表缓存以及事务处理。拥有这些内置的能力，只有在疏忽的的情况下你才不会考虑使用视图来创建客户服务器应用程序。

视图

在图书馆数据库中 Visual FoxPro 版 你可以找到远程视图 vAdultMember 和 vJuvenileMember。下面的 SQL 语句用于定义这两个视图：

```
SELECT Member. member_no  Member. lastname  Member. firstname
Member. middleinitial  Adult. street  Adult. city
Adult. state  Adult. zip  Adult. phone_no  Adult. expr_date
FROM dbo. adult Adult  dbo. member Member
WHERE Adult. member_no  =  Member. member_no
AND Member. member_no  =  nMemberID
SELECT Member. member_no  Member. lastname  Member. firstname
Member. middleinitial  Juvenile. adult_member_no
Juvenile. birth_date  Adult. street  Adult. city  Adult. state
Adult. zip  Adult. phone_no  Adult. expr_date
FROM dbo. adult Adult  dbo. juvenile Juvenile
dbo. member Member
WHERE Adult. member_no  =  Juvenile. adult_member_no
AND Juvenile. member_no  =  Member. member_no
AND Member. member_no  =  nMemberID
```

两个视图是非常直截了当的。成员的名字在 Member 表中地址在 Adult 表中。少年的出生日期和担保人可以在 Juvenile 表中找到。两个视图基于相同的连接 并确定各表的主关键字、标记其它字段是可更新的以使它们是可修改的。这些出现在视图设计器中的更新条件标签中。

载入表单

表单 MEMBVIEW. SCX 使用了这两个视图。以下是表单的 Load 方法代码。因为两个视图是用 NoData 选项打开 所以表单最初没有载入数据。然后设置视图游标为开放式行缓存。

```
Open Database library
```

```
Use vAdultMember In 0 NoData
 = CursorSetProp  Buffering  DB_BUFOPTRECORD
  vAdultMember
Use vJuvenileMember In 0 NoData
 = CursorSetProp  Buffering  DB_BUFOPTRECORD
  vJuvenileMember
```

定位一个成员

用户可以输入一个成员的 ID **并按下定位按钮。这样做就向视图所需的参数** nMemberID **提供一个值。以下是定位按钮的** Click **事件代码：**

```
nMemberID = Val ThisForm. txtMemberID. Value
Select vAdultMember
 = ReQuery
If RecCount  vAdultMember  = 0
Select vJuvenileMember
 = ReQuery
If RecCount  vJuvenileMember  = 0
lcMessage =  设有该 ID 成员。
 = MessageBox lcMessage  MB_ICONINFORMATION
  <略去的代码>
```

代码首先检查用户是否修改了数据。然后保存输入的 ID **到变量** nMemberID **中。**Adult **视图首先被请求。如果没有找到该** ID **则请求** Juvenile **视图。如果没有找到该** ID **的记录则该** ID **是无效的。如果找到了则该成员的信息显示在表单中。**

添加一个成员

当用户单击 Add **按钮时他们会得到一外空的表单。在用户按下** Save **按钮前记录并未被保存。**Add **按钮的** Click **事件中的代码首先检查用户是否修改了当前成员的信息。如果没有记录改变则发布一个** TableRevert **。然后添加一个空记录到视图中并刷新表单。用户可以在空的记录上输入成员信息。当用户按政** Save **时，**Visual FoxPro **将发送新的记录到** SQL Server**。**

保存修改

视图的一个好的功能是后端为你处理这些。TableUpdate **函数用于保存对视图的修改到源表中。**Visual FoxPro **自动处理这一点。以下是** Save **按钮的** Click **代码：**

```
If TableUpdate
 = MessageBox lcMessage  MB_ICONINFORMATION
Else
ThisForm. ShowError
Endif
```

取决于当前成员是成年人或少年，vAdultMember **或** vJuvenileMember **视图被选定。然后发布一个** TableUpdate **。如果成功，数据被保存。否则向用户显示发生了什么问题。 注意因为使用了** TableUpdate **，它可以处理新添加的成员或对成员信息的修改。**

删除一个成员

要删除后端上的一条记录只需在视图中删除它。以下是 Delete **按钮的** Click **事件代码：**

```
Select  ThisForm. cViewInUse
Delete
If TableUpdate
 = MessageBox  This member has been deleted.
MB_ICONINFORMATION
Append Blank
……
Else
ThisForm. ShowError
 = TableRevert
Endif
```

在询问用户确认删除选定视图的记录后，记录从视图中删除并发布 TableUpdate 。如果成功，记录从后端删除。否则，向用户显示错误原因。

问题

Visual FoxPro 不是奇妙的吗 客户服务器是多么的容易，你可能会想 该表单是非常简单和易于使用。但这里有三个问题。

不能添加新成员

要测试这些问题，单击 Add 按钮，添加一个新成员并单击 Save。很快你会看到如图所示的信息。

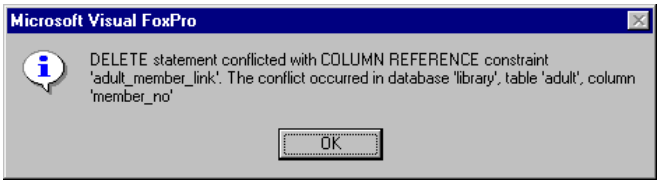
在按下 Add 按钮时，一个空记录添加到 vAdultMember 视图游标中。该视图由两个表组成，Adult 和 Member。在按下 Save 按钮时，Visual FoxPro 发送 name 信息到 Member 表。SQL Server 自动提供 member ID。因此很好。Visual FoxPro 也发送地址信息到 Adult 表。但关于成员的 ID，它没有发送任何东西。该字段保持为空，这是非法的并造成 TableUpdate 失败。

要处理这一问题，新记录需要首先添加到 Member 表，并且结果 ID 需要根据地址信息放到 Adult 表。没有理由认为 Visual FoxPro 的视图知道这些。为什么会这样 Visual FoxPro 不知道是什么 ID，且不知道在 Adult 表中发送它。

不能删除成员

在图书馆数据库中 SQL Server 版 在 Adult 和 Juvenile 表及 Member 和 Loan 间有着参照完整性。如果一个成员负责有少年成员或她/他还有未还清的借书，你就不能删除他/她。你可以期望 SQL Server 拒绝该删除并发送加一个错误提示。

但是，如果一个成员没在相关少年和借书，你可以删除一个他/她。试着这样做你会看到一条如下图所示的错误信息。



该错误,如果你理解 SQL Server 语言,它告诉你 Adult 和 Member 表间参照完整性冲突。这将会在试着在删除 Adult 记录前删除 Member 记录时发生。这明显的是发生在视图中。对于删除处理记录,必须首先删除 Adult 然后删除 Member。但是,Visual FoxPro 如何知道这些?

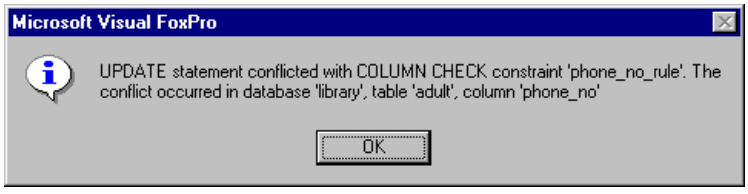
不可理解的错误信息

如果你想删除一个负责有少年的成员,你会被终止。如果你想删除一个借有图书的成员,你会被终止。但是,SQL Server 将发送回 Visual FoxPro 一个类似于上图 的错误信息。这些信息对于用户来说是完全不能理解的。

考虑另一个示例。Adult 表具有以下约束检查定义:

```
ALTER TABLE adult
  WITH NOCHECK
ADD CONSTRAINT phone_no_rule CHECK  phone_no LIKE
```

这些约束强制电话号码以 206 区号开始。如果插入或更新的记录的电话号码违反了这个规则,数据将被拒绝。但是,用户看到的怪异的提示信息如下图所示。



15.3.3 使用视图尝试二

那么,是不是在这种情况下视图是没有用处呢 你是不是要向它们投降了 而不准备进行更多尝试了。上面的问题是视图是由两个表组成,而且你没有控制 Visual FoxPro 如何插入或更新记录。在图书馆数据库方案中,有一些你必须遵循的特殊规则。

作为一个可选的方法,表单 MEMBVEW2.SCX 使用三个视图而不是两个。视图 vMemberTable,vAdultTable 和 vJuvenileTable 都是基于单个的 SQL Server 表。

载入表单

表单仍然是以无数据方式打开。三个视图都是以 NoData 选项打开并设置为开放式行缓存。

三个视图都基于想同的共享连接。视图的 ConnectHandle 属性保存着视图使用的连接的句柄该句柄将会被立即使用。

* 视图使用的连接句柄是什么

```
ThisForm.nHandle = CursorGetProp  ConnectHandle
  vMemberTable
```

定位一个成员

当用户想要查看一个新的成员的信息时，他们仍然需要提供一个 member ID 并按下 Locate。该表单中的代码与早期版本稍稍不同的是，Member 视图首先被 requery。如果没有找到记录你立即知道 ID 是无效的。Adult 视图其次被 requery。如果没有找到记录则成员肯定是一个少年。

保存修改

在早期版本的表单中，当用户按下 Save 时，简单地发布一条 TableUpdate　。正象你所看到的一样，对于特定结构的表那是不够的。该版本的表单采用了一种更复杂但更成功的方法。如果用户当前添加了一条记录，保存按钮的 Click 事件代码的前半段将运行。

```
If ThisForm. lAddMode  = . T.
Wait Window  保存新成员信息 . . .  NoWait
```

代码首先添加新成员到 Member 表。然后添加一条记录到 Adult 表。两个表都可能添加一条新记录，也可能都不添加。因此必须启动一个事务处理。连接的 Transactions 属性用于控制事务处理过程。在表单的 Load 方法中，保存了视图的连接句柄到表单的 nHandle 属性中。SQLSetProp　使用它来启动一个事务处理。

```
＊ 开始一个事务处理
 = SQLSetProp  ThisForm. nHandle   Transactions   2
```

vMemberTable 视图被选定并发布一条 TableUpdate　。若无问题则新成员的名字信息已添加到 Member 表中。并且成员会有一个成员 ID。如果 TableUpdate　失败用户会看到发生了什么事并用 SQLRollback　函数回滚事务处理。

```
＊ 添加新成员到 member 表
Select vMemberTable
  If Not TableUpdate
ThisForm. ShowError
＊ 回滚事务处理
 = SQLRollback  ThisForm. nHandle
Return
Endif
```

如果 TableUpdate　成功则添加一条新记录到 Member 表中。接着是找出分配了什么成员 ID。SQLExec　用于发送一条 Select 语句到 SQL Server。该 Select 取回 @@Identity 的值　它保存了最后插入到表中的 Identity 值。在我们的情况下是成员 ID。SQLExec　使用视图使用的相同的连接句柄。这是有效的，因为它不需要另一个与 SQL Server 的连接。

```
＊ 找出新成员的 member_no
If SQLExec  ThisForm. nHandle   Select @@identity   < 0
 <略去的代码>
```

在 vAdultTable 视图中的 member_no 被 @@Identity 的值替换。这强制保持 Member 与 Adult 表间的关系。注意这包括了修改 Adult 表的主关键字段。没有其它选择。该成员的终止日期设置为从本日起一年的时间，并用 TableUpdate　来保存该信息到 Adult 表。如果发生错误事务处理回滚。这会删除 Member 表中的记录。

```
＊ 添加新成员到 adult 表
Select vAdultTable
＊ member_no 是主关键字但因足够的理由在视图中更新。
```

＊ 终止日期是从今日起一年时间。

```
Replace member_no With sqlresult.exp
expr_date With DToT GoMonth Date 12
If Not TableUpdate
```

<略去的代码>

如果在此时此刻一切正常则用 SQLCommit 函数提交事务处理。如果提交失败则回滚以前的所有操作。

＊ 一切正常因此提交事务处理。

```
If SQLCommit ThisForm.nHandle < 0
ThisForm.ShowError
```

＊ 回滚事务处理

```
= SQLRollback ThisForm.nHandle
Else
```

<略去的代码>

如果用户没有添加新成员,,是代码稍稍要简单一些。仍然要启动事务处理。首先保存 Member 表中的信息然后保存 Adult 和 Juvenile 表中的信息。如果一切正常则提交事务处理。否则回滚。

```
Else
Wait Window 保存成员信息... NoWait
```

＊ 开始事务处理

```
= SQLSetProp ThisForm.nHandle Transactions 2
```

＊ 保存信息到 member 表

```
Select vMemberTable
If Not TableUpdate
```

<略去的代码>

＊ 保存信息到 adult 表

```
Select vAdultTable
If Not TableUpdate
```

<略去的代码>

＊ 一切正常因此提交事务处理。

```
If SQLCommit ThisForm.nHandle < 0
ThisForm.ShowError
```

＊ 回滚事务处理

```
= SQLRollback ThisForm.nHandle
Else
= MessageBox 该成员信息已经 +
保存。 MB_ICONINFORMATION
Endif
Endif
```

删除一个成员

每一个表使用一个视图不仅允许解决添加新成员的问题,也解决了不能删除成员的问题。当用

户按下 Delete 按钮时启动一个事务处理。Adult 或 Juvenile 表中的成员记录首先被删除。Delete 应用于视图并且 TableUpdate　发送它到 SQL Server。如果 TableUpdate　未成功则回滚事务处理。

```
* 在视图连接上开始一个事务处理
= SQLSetProp ThisForm.nHandle　Transactions　2
If RecCount　vJuvenileTable　= 0
Select vAdultTable
Delete
Else
Select vJuvenileTable
Delete
Endif
If Not TableUpdate
ThisForm.ShowError
* 回滚事务处理
= SQLRollBack ThisForm.nHandle
<略去的代码>
```

如果相应的 Adult 或 Juvenile 记录已经删除，就可以删除 Member 表的记录。Delete 也有可能失败，例如该成员还有未结清的借书。在这种情况下回滚事务处理且 Adult 或 Juvenile 记录放回到相关的表中。

```
Select vMemberTable
Delete
If Not TableUpdate
ThisForm.ShowError
* 回滚事务处理
= SQLRollBack ThisForm.nHandle
<略去的代码>
```

如果两个 TableUpdate　都成功了则用 SQLCommit　函数提交事务处理。如果未成功则回滚事务处理。

问题

第一个版本的表单中的三个问题现已解决了两个。可以添加新成员和删除一个成员了。如果违犯了参照完整性删除将会失败。例如，假设你想删除一个没有相关 juveniles 但有未结清的借书的成员时。

因此，语无伦次的错误信息的问题任然存在。你可以编写代码来分解错误信息并寻找特定的短语。然后将 SQL Server 信息转换为用户可以理解的内容。这是很大的工作量并要求非常熟悉 SQL Server 的错误信息。它也使转换你的应用程序到另一个后端 如 Oracle 变得非常困难。

15.3.4　使用 SQL pass - through

与使用视图相比的另一个选择是单独地使用 SQL pass - through。这就意味着你发送 SQL 语句到后端并明白地告诉它要做什么。如果你想添加一条记录可发送一个 Insert。要保存记录可发送一个 Update。显然这比使用视图要做更多的工作。但是它允许你完全地控制发生什么和什么时候将

会发生。

载入表单

表单 MEMBEXEC.SCX 是一个与上面的表单相似的表单，它只是使用 SQL pass - through 而不是视图。以下是表单的 Load 方法代码。

```
Open Database library
ThisForm.nHandle = SQLConnect cnLibrary
If ThisForm.nHandle < 0
ThisForm.ShowError
ThisForm.lConnected = .F.
Else
lcSQL = Select member.member_no lastname firstname +
  middleinitial street city state zip +
  phone_no expr_date birth_date = null +
  adult_member_no = null +
 From member adult +
 Where member.member_no = -99
If SQLExec ThisForm.nHandle lcSQL c_member < 0
ThisForm.ShowError
ThisForm.lConnected = .F.
Endif
   = CursorSetProp Buffering DB_BUFOPTRECORD c_member
Endif
```

在表单载入时，SQLConnect 用于使用保存在 Visual FoxPro Library 数据库中的 cnLibrary 连接来建立到 SQL Server 的连接。如果 SQLConnect 失败，除了退出和回家外，你没有别的事可做。

如果连接正常，以发送到 SQL Server 一个搜索成员号 -99 的 Select 语句来创建了一个空的游标。即使没有返回记录，Visual FoxPro 也会创建该游标。然后设置该游标的开放式缓存。这样做的原因是可以在该表单上使用缓存了的游标。这反映了一些视图的易用性。

定位一个成员

当在表单上使用视图时，定位一个成员只需简单地为视图参数赋值并重新获取值 requery 。当使用 SQL pass - through 时要稍稍复杂一些。要获取成员信息，要创建一条 Select 语句并发送到服务器。在下面的代码中你可以看到这是一个 Union Select。如果成员存在于 Member 表和 Adult 或 Juvenile 表中。

```
lcSQL = Select member.member_no lastname firstname +
  middleinitial street city state zip +
  phone_no expr_date birth_date = null +
  adult_member_no = null +
 From member adult +
 Where member.member_no = adult.member_no +
  And member.member_no = +
   AllTrim ThisForm.txtMemberID.Value + +
```

```
Union    +
Select member. member_no   lastname   firstname      +
  middleinitial   street   city   state   zip       +
  phone_no   expr_date   birth_date       +
  adult_member_no     +
From member   adult   juvenile     +
Where member. member_no  =  juvenile. member_no     +
  And adult. member_no  =      +
    juvenile. adult_member_no     +
  And member. member_no  =      +
    AllTrim  ThisForm. txtMemberID. Value
If SQLExec  ThisForm. nHandle   lcSQL     c_member       < 0
```

<略去的代码>

如果 c_member 游标为空则没有输入的 ID 成员存在。否则所有成员信息都在游标中。游标设置了行缓存,且表单控件用游标中的成员信息填充。

Union 允许你发送一个 Select 到服务器并获取该成员的所有信息。在早期的示例中,需要对两个或三个视图进行重查询 requery 而不是象这里一样只需要一个 SQLExec 。注意,如果你使用视图设计器,你不能创建一个带 Union 的远程视图。但你可以用 Create SQL View 命令来创建这种视图。

添加一个成人

当用户按下 Add 按钮时，一条空白的记录添加到 c_member 游标中。这与早期的视图示例不同。

```
Select c_member
 =  TableRevert
Append Blank
```

仅出于可读性的原因,添加新成员的代码位于表单的 AddMember 方法中。因为添加一个成员包括向两个表添加记录,因此开始一个事务处理。只在使用视图的表单中,使用 SQLSetProp 函数来开始事务处理。

```
 =  SQLSetProp  ThisForm. nHandle     Transactions     2
```

当用视图来访问远程数据时,你可以依靠 Visual FoxPro 来为你处理大多数的后台工作。例如,在早期的表单中,你看到了要发送一个 Insert 或 Update 到服务器,你只需要发布一条 TableUpdate 。Insert 或 Update 的语法都由 Visual FoxPro 为你做了。

这里的表单合作 SQLExec 函数来发送 SQL 语句到服务器。这就意味着你必须自己构造 SQL 语句。在开始事务处理后,构造一个 Insert 语句来添加新记录到 Member 表中。

```
*  添加新成员到 member 表
lcSQL  =   Insert member   lastname   firstname       +
 middleinitial   photograph       +
 Values        +
AllTrim  ThisForm. txtFirstName. Value    +
          +
```

```
AllTrim ThisForm.txtLastName.Value +
        +
AllTrim ThisForm.txtMiddleInitial.Value +
        + NULL
If SQLExec ThisForm.nHandle lcSQL < 0
ThisForm.ShowError
* Rollback the transaction
= SQLRollback ThisForm.nHandle
Return
Endif
```

你现在需要知道 SQL Server **为新成员指定的** member_no**。代码与早期表单相似。**

```
* 找出新成员的 member_no
If SQLExec ThisForm.nHandle Select @@identity < 0
 <略去的代码>
nNewMemberID = sqlresult.exp
```

然后构造一个 Insert **来添加新记录到** Adult **表。从服务器获取的** @@Identity **值用于该** Insert **中来正确地连接** Adult **和** Member **中的记录。**

```
* 添加新成员到 adult 表
lcSQL = Insert adult member_no street city state +
                zip phone_no expr_date +
 Values + AllTrim Str nNewMemberID + +
AllTrim ThisForm.txtStreet.Value +
        +
AllTrim ThisForm.txtCity.Value +
        +
AllTrim ThisForm.txtState.Value +
        +
AllTrim ThisForm.txtZip.Value +
        +
AllTrim ThisForm.txtPhoneNumber.Value +
         +
TToC DToT GoMonth Date 12 +
If SQLExec ThisForm.nHandle lcSQL < 0
 <略去的代码>
```

如前所述,如果一切正常,则提交事务处理否则回滚。

保存修改

保存已存在的成员的信息的代码在表单的 UpdateMember **方法中。要保存信息,发送一条** Update **语句到服务器。表单中的更新语句如下**

```
Update <table> Set <column1> = <value1>
 <column2> = <value2>
```

等等...

这相当地直截了当，虽然构造一个要发送到服务器的 Update 语句看起来有些庞大。你知道表的字段名和表单上的控件中的值。你可以一个字段接一个字段地生成 Update 的 Set 部分。但是，象上面这样书写可以使程序更清晰。你也可能不想浪费 SQL Server 的时间来更新没有修改的字段。这里的代码使用了 OldVal　函数，使用游标缓存使这样做变为可能，要将游标中各字段的值与原始的值进行比较。只有当它被修改了时才让它成员 Update 语句的一部分。另外，远程视图自动地这样做了。

```
lcSQL =
* 更新该成员到 member 表
If c_member.firstname <> OldVal c_member.firstname
lcSQL = lcSQL + firstname = +
AllTrim ThisForm.txtFirstName.Value +
Endif
If c_member.lastname <> OldVal c_member.lastname
lcSQL = lcSQL + lastname = +
AllTrim ThisForm.txtLastName.Value +
Endif
<略去的代码>
```

如果 Member 表中没有字段被修改，变量 lcSQL 中将没有内容且不会保存信息到表中。如果有要保存的数据，构造剩下的 Update 语句并发送到服务器。

```
If Len lcSQL > 0
* 添加 Update，去掉最后一个逗号，
* 添加一个 Where 子句
lcSQL = Update member Set +
Left lcSQL Len lcSQL - 2 +
Where member_no = +
AllTrim ThisForm.txtMemberID.Value
If SQLExec ThisForm.nHandle lcSQL < 0
<略去的代码>
```

然后对 Adult 进行与上面相同的处理。接着要做的是我们非常熟悉的事。如果一切正常提交事务处理否则回滚。

删除一个成员

使用 SQL pass - through 相对于远程视图的一个优点是，你增加了对发生什么和何时发生的控制。当用户单击 Delete 按钮时运行的代码就是一个好的例子。

有很多理由可能使你不能删除一个成员。如果成员有相关的 juveniles 或成员有未结清的借书时任何 Delete 都会失败。可以很容易地通过发送一个 Select 到服务器来检查这一点。这里的代码使用 SQLExec　来检查这两个条件。如果两者都为真，会显示一个友好的信息且不会进一步发生什么情况。

```
* 首先检查是否有一个活动的 juveniles
lcSQL = Select member_no From juvenile +
```

```
 Where adult_member_no =     +
ThisForm. txtMemberID. Value
If SQLExec  ThisForm. nHandle   lcSQL    < 0
ThisForm. ShowError
Return
Else
If RecCount   sqlresult     <>  0
lcMessage =   该成员不能删除 .     +
 他是一个活动的少年的成年。
 = MessageBox lcMessage   MB_ICONINFORMATION
Return
Endif
Endif
* 现在检查成员还有活动的借书
lcSQL =   Select member_no From loan     +
 Where member_no =     +
ThisForm. txtMemberID. Value
If SQLExec  ThisForm. nHandle   lcSQL    < 0
ThisForm. ShowError
Return
Else
If RecCount   sqlresult     <>  0
lcMessage =   该成员不能删除。     +
 他还有活动的借书。
 = MessageBox lcMessage   MB_ICONINFORMATION
Return
Endif
Endif
```

如果还需要执行额外的检查,代码可以放在以上代码的后面。你可以完全控制要检查的内容和顺序。如果所有的检查都成功且成员可以删除,则开始一个事务处理。

在 Member 表和 Loanhist 及 Reservation 表单定义了关系。每一次借书和还书都在 Loanhist 表中有一条记录。成员预约的每一本书在 Reservation 表中有一条记录。如果成员被删除,需要删除这两个表中的相关信息。需要首先删除它们,否则会违犯参照完整性。

```
* 删除该成员的借书情况 loan history 记录
lcSQL =   Delete loanhist Where member_no =     +
AllTrim  ThisForm. txtMemberID. Value
If SQLExec  ThisForm. nHandle   lcSQL    < 0
 <略去的代码>
* 删除该成员的借书预约 loan reservation 记录
```

```
lcSQL = Delete reservation Where member_no =  +
AllTrim ThisForm. txtMemberID. Value
If SQLExec ThisForm. nHandle lcSQL  < 0
 <略去的代码>
```

要删除一个成人成员必须首先删除 Adult 表中的记录然后才能删除 Member 表中的记录。这也是事务处理的一部分,因此如何有任何错误发生则回滚所有处理。

```
* 删除成员
lcSQL = Delete adult Where member_no =  +
AllTrim ThisForm. txtMemberID. Value
If SQLExec ThisForm. nHandle lcSQL  < 0
 <略去的代码>
lcSQL = Delete member Where member_no =  +
AllTrim ThisForm. txtMemberID. Value
If SQLExec ThisForm. nHandle lcSQL  < 0
 <略去的代码>
```

如果所有删除都没有问题则提交整个事务处理且成员被删除。然后用户会看到一个空的屏幕,因此添加一条空记录到 c_member 并刷新表单显示。

问题

如何折衷地使用远程视图和 SQL pass - through

这是更多的工作

显然,使用 SQL pass - through 比使用远程视图需要做更多的工作。视图为你做了大量的工作,维护与服务器的通信和传递 Inserts,Updates 和 Deletes。视图是易于设置和使用的。

你拥有更多的控制

你看到了两种观点的示例,使用远程视图的一个问题是在 Visual FoxPro 和后端间的通信上你只有较少的控制。对于多数据库方案,这不是一个问题。但是,在这里使用的方案中出现了问题。问题用每一个表使用一个视图得到了缓解但任然存在。当视图不能给你以所需的数据输入能力时,问题会更为突出。

当你使用 SQL pass - through 时,你可以完全控制 Visual FoxPro 与后端的通信。你构造 SQL 语句并用 SQLExec 来发送它们到服务器。如果需要执行数据验证和检查商业规则,你可以决定在什么时候和如何做。

错误信息可以更友好

因为你在控制什么发生,并且你手动的进行数据验证,因此你可以控制错误信息。在可以预见一些事发生时,你可以截取 SQL Server 错误信息并显示给用户一个可以理解的信息。由于不可预料的事件你也会遇到 SQLExec 失败的问题,如网络故障。对于这些,你可以决定是否分解信息或以原始方式显示它们。

一方面它提供了较少的互用性

该方法的一个问题是从某种程度上牺牲了互用性。通过 SQLExec 发送到后端的 SQL 语句是用后端语言编写。在本示例中是 SQL Server。当转换到 Oracle 时需要重写多少代码

虽然不同后端的基本的 Select,Insert,Update 或 Delete 格式没有太大的不同。因此这里的示例

可以很容易 转移到其它后端。但是,重要的一点是取决于你使用的 SQL 语句的复杂性,这可能限制了你交换后端的能力。当然,如果应用程序是为一种后端且只用一种后端,这将不是大的问题。

15.4 select SQL 命令

15.4.1 select - SQL 的工作流程

再复杂的 SQL 命令,也是由一些基本的结构组成的。所以在看、去做一条很复杂的 SQL 命令时,要会把它一级一级的折分,最后折成最简单的,这样才容易理解。而这个折分过程,如果不熟悉 SQL 命令的工作流程,那就比较难折分了。

大体来说,它是先根据联接条件 即联接条件 on 中的表达式 ,把几个的表合成一个临时表,然后根据 where 中的条件进行过滤,过滤出来的结果根据分组条件再把这个临时表分成一组一组,然后对分别对些组进行字段计算, 最后又得出一个临时表, 然后又根据 having 中的条件对这个临时表进行再次过滤,最后输入到指定的地方,如数组、表等。它中间生成的临时表对用户来说,是完全透明的,用户是不可能使用、也不能创建,它是由系统自己创建、自己使用、自己撤除,完全不受用户控制的。我举个例子 :有二个表 :提货单 thd、提货单明细 thdmx:

thd

提货单号 提货日期

thdbh thrq

01 2000/01/02

02 2000/01/15

03 2000/02/01

thdmx

提货单号 产品编号 提货数量

thdbh cpbh Thsl

01 001 5

01 003 15

01 005 12

02 001 13

02 002 14

02 005 20

03 002 14

现在有个要求:要统计 day1 = 2000/01/01 至 day2 = 2000/01/20 这段时间内, 提货数量大于 10 的产品有那些,它们各自的总提货量是多少。命令如下:

```
sele cpbh sum thsl
from thd join thdmx
on thd.thdbh = thdmx.thdbh. and. thsl> 10. and. betw thrq day1 day2
grou by cpbh
into curs temp1
```

为什么把 thsl> 0 和 betw thrq day1 day2 这二个条件表达式放在 on 那里,参见 on、where 与

having 的区别。它的工作流程：

首先根据 on 中的过滤条件，对所涉及的表进行预处理。过程如下：

两个表根据 thd. thdbh = thdmx. thdbh 进行合并，变成一个这样的临时表：

thdbh1	thrq	thdbh2	Cpbh	thsl
01	2000 / 01 / 02	01	001	5
01	2000 / 01 / 02	01	003	15
01	2000 / 01 / 02	01	005	12
01	2000 / 01 / 02	02	001	13
01	2000 / 01 / 02	02	002	14
01	2000 / 01 / 02	02	005	20
01	2000 / 01 / 02	03	002	14
02	2000 / 01 / 15	01	001	5
02	2000 / 01 / 15	01	003	15
02	2000 / 01 / 15	01	005	12
02	2000 / 01 / 15	02	001	13
02	2000 / 01 / 15	02	002	14
02	2000 / 01 / 15	02	005	20
02	2000 / 01 / 15	03	002	14
03	2000 / 02 / 01	01	001	5
03	2000 / 02 / 01	01	003	15
03	2000 / 02 / 01	01	005	12
03	2000 / 02 / 01	02	001	13
03	2000 / 02 / 01	02	002	14
03	2000 / 02 / 01	02	005	20
03	2000 / 02 / 01	03	002	14

合并是按照笛卡尔积进行计算的，即二个表都有 10 个记录，那积就会有 10 * 10 = 100 个记录，这是很历害的。但因为 on 条件中有过滤条件，所以 VFP 并不会这么笨，它会把符合这个条件的记录才放到临时表中的。这样，结果就少了很多记录了。结果出来后，再根据 on 后来的过滤条件 thsl> 10. and. betw thrq day1 day2 进行过滤，这样 thsl 大于 10 而且提货日期是在 day1 至 day2 的记录最后才出现在这一步的临时表中。

01	2000 / 01 / 02	01	003	15
01	2000 / 01 / 02	01	005	12
02	2000 / 01 / 15	02	001	13
02	2000 / 01 / 15	02	002	14
02	2000 / 01 / 15	02	005	20

现在轮到分组了。根据产品编号进行分组，它具体的分组方法我不知道是怎样，我想可能是这样的：

象在投票选举时点票那样，在上面那个临时表从头到尾扫一次，每经过一记录时，它就看一下，当前的产品编号是不是一个新的组，如果是就新增一个分组记号，相当于新增加一个被选举人，然后在它下面加上 thsl 的值，全部记录数完了，就看看有多少个分组标记，各个分组又有多少 thsl。结

果就是以下的样子：

001 13

002 14

003 15

005 32

这就是这条命令的结果了。然后把它生成一个 cursor 表，命令就完成了。如果再深入一点，把要求改成某段时间内全部产品的提货情况，如果没有进货记录，那就是 0，一样要出现在结果表里。

这时，就涉及到三个表了：产品表 cpb、提货表 thd、提货明细表 thdmx。我们先用内联接来把这三个表联接起来。

```
sele cpb. cpbh  cpb. cpmc  sum  iif  isnull  thdmx. thsl   0  thdmx. thsl    as thsl
from cpbh join thdmx
join thd
on cpbh. cpbh = thdmx. cpbh
on thdmx. thdbh = thd. thdbh. and. betw  thd. thrq  day1  day2
grou by cpbh
into curs temp1
```

根据内联接的定义，即某个产品编号在产品表和提货明细表中都存在的记录，才会出现在结果表中，如果某种产品没有提货，那在提货明细表就没有这个记录，这样，也就不会出现在结果表中。那样就符合要求中的 全部产品 这个条件了。所以我们要把 产品表 左联接 提货明细表，这样不管这种产品有没有提货，它都会出现在结果表中，只是以 null 的值出现。但这个现象可以消除的。

命令过程也和上一个要求那样，先进行联接，只是这条命令的中间临时表比上面更大，因为是三个表的记录数相乘。但经过联接条件过滤后，就会剩下这些内容：

001 02 2000/01/15 02 001 13

002 02 2000/01/15 02 002 14

003 01 2000/01/02 01 003 15

004 null null null null null

005 01 2000/01/02 01 005 12

005 02 2000/01/15 02 005 20

然后也一样进行分组，分组时的判断过程也一样，只是在累加的时候有点不同：

sum iif isnull thdmx. thsl 0 thdmx. thsl ，是先用 isnull thdmx. thsl 检查 thsl 是不是 null，如果是，则 iif 就返回 0，如果不是，则返回 thdmx. thsl。然后外层的 sum 就根据 iif 返回的数值进行累加，最后做为这个分组的累加值。

还有一种使用方法，说出来也可以加深命令当中的 sum 函数的处理过程。

人事表 rsb：

姓名 xm 年龄 age

张三 25

李四 32

王五 28

现在想统计一下各个年龄段 20 - 30 31 - 40 的人数是多少。

```
sele sum iif betw age 20 30  1 0  as _20 - 30 sum iif betw age 31 40  1 0  as _31 - 40
from rsb
grou by zc
into curs temp1
```

因为这条命令没有过滤条件、联接，所以不需事先预处理，一来就进行分组。和投票中点票一样，在 黑板 上写划出三列：

zc　　_20 - 30　　_31 - 40

这样，在 rsb 中从头扫到尾时，每经过一记录时，都用 iif betw age 20 30 1 0 检查这个人的年龄是不是处于 20 - 30，如果是就在 _20 - 30 这一列下面加一横，否则就不加。第二个 iif 也这样处理。如果当前的年龄是 53，那二个 iif 都是返回 0，即二列都不加，相当于废票。全部记录都点完了，然后就用 sum 进行合计了，结果就出来了。

因为二个 sum 都是扫描完后再合计的，它不象 sum 命令。sum 命令会移动记录指针，而 sum 函数不会，所以不必怕使用这个函数会造成不良后果。而且二个 iif 都有自己的判断条件，两者不互相重合，所以一个记录不会重复计算 除非你的命令设计错了。

如果使用 union 参数把两条运算结果的格式一模一样的命令合在一起，那也是一样的。它是把其中的每一节 select 命令单独运行 它单独运行时的运行流程跟上面说的一样 ，最后才把每一节的结果首尾相接后，再根据最后那节的 orde by 进行排序。所以一条带 union 的 SQL 命令，只能有一个 orde by。而每一节 select 命令，却可以有自己的 grou by，它自己的 grou by 只对这一节有影响，是不会影响到其它节的运算的。更不会对最后结果有影响。

这里举个例子 :我想统计一段时间内的提货、进货情况。这里要涉及到 5 个表 :产品表、提货表、提货明细表、进货表、进货明细表：

```
sele cpbh  sum  thdmx. thsl  as thsl  100000 - 100000 as jhsl
from thd join thdmx
on thd. thdbh = thdmx. thdbh
grou by cpbh
union
sele cpbh  0  sum  jhdmx. jhsl
from jhd join jhdmx
on jhd. jhdbh = jhdmx. jhdbh
grou by cpbh
orde by cpbh
into curs temp1
```

如果看了上面的解释，应该可以理解每一节 SQL 命令的意思。这里要说的是：

1、union 要求前后两节 SQL 命令产生的表，在结构上要完全一样，包括字段的顺序也要一样。所以为了达到这个要求，在第一节，就要人为建立一个字段 jhsl，而在第二节命令，也要建立一个字段 thsl 以对应。

2、用 SQL 产生的表，它不象 crea table 那样可以直接指定字段的类型、长度，而是在根据生成的临时表中第一个记录的长度来确定的。所在在第一节，如果不使用 100000 - 100000 而是直接使用 0，这样产生的 jhsl 这个字段，它的长度就只有 2 个字节了。所以只有使用这种方法，才能使得这个字段的长度有 7 字节。在字符串也有这样情况，如果第一个记录的长度是 12 个字节，那以后的记录

中，超过 12 个字节的内容就会给它去掉，这就是为什么有时在结果表中会出现字符串不完整的情况。解决方法也差不多，在字段列表那里人空加几个空格去。

15.4.2 on、where、having 的不同之处

这里有个例子来比较一下过滤条件放在 on、where、having 会有什么的不同之处：

表 recdbf 内容如下：　还有一个 tempyf 的辅助表，记录 12 个月

日期	性质	yf
2000 年 7 月 3 日	特大	1
2000 年 7 月 9 日	特大	2
2000 年 9 月 3 日	特大	3
1999 年 3 月 2 日	一般	4
1999 年 3 月 4 日	一般	5
2000 年 1 月 3 日	一般	6
2000 年 2 月 1 日	一般	7
2000 年 2 月 3 日	一般	8
2000 年 3 月 4 日	一般	9
2000 年 8 月 7 日	一般	10
2000 年 11 月 2 日	一般	11
1999 年 2 月 3 日	重大	12
2000 年 2 月 3 日	重大	
2000 年 5 月 2 日	重大	
2000 年 8 月 9 日	重大	

现在的要求是要统计 yy 年中十二个月的事故记录中，一般、重大、特大各有多少。如果没有事故的，则以 0 表示。

我们首先要把今年的记录过滤出来，过滤条件就是 YEAR 日期 = yy，然后按月份分组统计。

这样一来，如果某个月没有事故记录，那分组后的结果就没有该月的记录，这样不符合要求。所以做个临时表 yf，该表有十二个记录，分别代表 1 至 12 月，用它来左联接 recdbf，这样，即使某个月没有事故记录，也会出现在最后的结果当中，只是以 null 的形式出现罢了。但我们可以使用 isnull 函数来判断它是不是 null 值，如果是，则 iif 会把它变为 0，然后交与 sum 进行统计。

总体设想搞好后，现在就开始写命令了。开始之前先说明：tempyf.yf = MONTH recdbf.日期是 yf 表与 recdbf 表的联接条件，是一定要在 on 的，这个不在讨论范围。我们要讨论的是 YEAR 日期 = yy 这个条件放在什么地方会有什么样的结果。

首先把过滤条件放在 on 这里：

```
SELECT tempyf.*
    SUM IIF ISNULL recdbf.日期 .OR. AT 一般 recdbf.性质 =0 0 1 AS 一般
    SUM IIF ISNULL recdbf.日期 .OR. AT 重大 recdbf.性质 =0 0 1 AS 重大
    SUM IIF ISNULL recdbf.日期 .OR. AT 特大 recdbf.性质 =0 0 1 AS 特大
  FROM tempyf LEFT OUTER JOIN recdbf
    ON tempyf.yf = MONTH recdbf.日期 .AND. YEAR 日期 = yy
```

```
    GROUP BY tempyf. yf
```

其中 yy = 2000,表示统计 2000 年的数据。

用 where 的命令如下：

```
SELECT tempyf. *
      SUM IIF ISNULL recdbf. 日期 . OR. AT 一般 recdbf. 性质 = 0 0 1 AS 一般
      SUM IIF ISNULL recdbf. 日期 . OR. AT 重大 recdbf. 性质 = 0 0 1 AS 重大
      SUM IIF ISNULL recdbf. 日期 . OR. AT 特大 recdbf. 性质 = 0 0 1 AS 特大
   FROM tempyf LEFT OUTER JOIN recdbf
      ON tempyf. yf = MONTH recdbf. 日期
   GROUP BY tempyf. yf
where YEAR 日期 = yy          &&注意,条件从 on 移到这里来了
```

用 having 的命令如下：

```
SELECT tempyf. *
      SUM IIF ISNULL recdbf. 日期 . OR. AT 一般 recdbf. 性质 = 0 0 1 AS 一般
      SUM IIF ISNULL recdbf. 日期 . OR. AT 重大 recdbf. 性质 = 0 0 1 AS 重大
      SUM IIF ISNULL recdbf. 日期 . OR. AT 特大 recdbf. 性质 = 0 0 1 AS 特大
   FROM tempyf LEFT OUTER JOIN recdbf
ON tempyf. yf = MONTH recdbf. 日期
GROUP BY tempyf. yf
having YEAR 日期 = yy          &&注意,条件从 on 移到这里来了
```

on 的结果如下,这是正确的：

YF	一般	重大	特大
1	1	0	0
2	2	1	0
3	1	0	0
4	0	0	0
5	0	1	0
6	0	0	0
7	0	0	2
8	1	1	0
9	0	0	1
10	0	0	0
11	1	0	0
12	0	0	0

用 where 的结果如下：

YF	一般	重大	特大
1	1	0	0
2	2	1	0
3	1	0	0
5	0	1	0

7 0 0 2

8 1 1 0

用 having 的结果如下：

YF 一般 重大 特大

1 1 0 0

2 2 2 0

5 0 1 0

7 0 0 2

8 1 1 0

9 0 0 1

11 1 0 0

各位看到有什么不同吗？

on 是把先把 recdbf 中不是 2000 年的记录过滤掉，剩下的就是 2000 年的了，再用 tempyf 去和它们进行外联接，其结果可用：

```
sele tempyf.* recdbf.日期
from tempyf left join recdbf
ON tempyf.yf = MONTH recdbf.日期 .AND.YEAR 日期 = yy
orde by yf
```

来查看，这个中间结果出来后，再用 isnull 把空值的记录变成 0 或 1，然后由 sum 去统计，结果就出来了。

而 where 呢：

1、它是先把 tempyf 外联接 recdbf 相当于 sele tempyf.* recdbf.* from tempyf left join recdbf on tempyf.yf = mont recdbf.日期 ；

2、然后把不是 2000 的记录过滤掉，这里要注意的是，如果某个月没有记录的话，那在第一个步骤后日期那里是 null 值，这当然不是 2000 的记录，所以就给这个条件给过滤出去了，所以下一步的 sum 之后就只剩下那有记录的那个月了，象 4、6 月等几个月就没有了；

3、然后进行 sum …… 。

再看 having：

1、第一步和 where 一样；

2、第二步不同，它是先 sum ，这里的 sum 可不管你是 1999 年还是 2000 的，先累加起来再说，这时，1999 和 2000 年的 2 月份都有 重大 这个记录，sum 的结果是 2，这里用第三个步骤去分辨这个 2 之中那个是 1999 年的，那个是 2000 的，这当然分不清啦，所以也错了；

3、根据步骤 2 来把 2000 的过滤出来。

所以 on、where、having 这三个都可以加条件的子句中，on 是最先执行，where 次之，having 最后。有时候如果这先后顺序不影响中间结果的话，那最终结果是相同的。但因为 on 是先把不符合条件的记录过滤后才进行统计，它就可以减少中间运算要处理的数据，按理说应该速度是最快的。

根据上面的分析，可以知道 where 也应该比 having 快点的，因为它过滤数据后才进行 sum，所以 having 是最慢的。但也不是说 having 没用，因为有时在步骤 3 还没出来都不知道那个记录才符合要求时，就要用 having 了。

在两个表联接时才用 on 的，所以在一个表的时候，就剩下 where 跟 having 比较了。在这单表查

询统计的情况下，如果要过滤的条件没有涉及到要计算字段，那它们的结果是一样的，只是 where 可以使用 rushmore 技术，而 having 就不能，在速度上后者要慢。

如果要涉及到计算的字段，就表示在没计算之前，这个字段的值是不确定的，根据上篇写的工作流程，where 的作用时间是在计算之前就完成的，而 having 就是在计算后才起作用的，所以在这种情况下，两者的结果会不同。

在多表联接查询时，on 比 where 更早起作用。系统首先根据各个表之间的联接条件，把多个表合成一个临时表后，再由 where 进行过滤，然后再计算，计算完后再由 having 进行过滤。由此可见，要想过滤条件起到正确的作用，首先要明白这个条件应该在什么时候起作用，然后再决定放在那里。

15.4.3　四种链接的区别及用法

链接：

作为动词，它表示将两个或多个表的内容结合在一起并产生一个结果集，该结果集对每个表的列和行进行合并。表的联接一般都使用它们共有的数据。例如，您可以对有一个共同 pub_id 列的 titles 表和 publishers 表联接，产生一个包含书名信息和出版商信息的结果集。

作为名词，表示对表进行联接的过程或结果，如在术语 内部联接 中表示对表联接的一种特殊的方法。

联接条件 (join condition)

一个比较子句，它指定了表是如何通过它们的联接字段相联系的。最普通的联接条件是相等(一个等联接)，在等联接中联接字段的值必须相同。例如，您可以通过在 titles 表和 publishers 表的 pub_id 列中查找相匹配的值联接这两个表。然而，任何比较运算符都可以是比较条件的一部分。

内部联接 (inner join)

一个联接，在该联接中只有当联接字段的值满足某些特定的准则时才将两个表的记录进行结合并添加到一个查询结果中。例如，在查询设计器视图中，表之间的缺省联接是一个内部联接，它只有当联接字段的值相等时才从两个表中选择记录。

外部联接 (outer join)

一个联接，该联接还包括那些和联接表中记录不相关的记录。您可以创建一个外部联接的三种变形来指定所包括的不匹配行：左外部联接、右外部联接和完全外部联接。

左外部联接 (left outer join)

一种外部联接类型，在该联接中包括第一个命名表 (左边的表，它出现在 JOIN 子句的最左边) 的所有行。右边表中没有匹配的行不出现。例如，您可以在 titles 表和 publishers 表之间创建一个左外部联接，以包括所有的书名，不论书名有无出版商的信息。

右外部联接 (right outer join)

一种外部联接，在该联接中包括第二个命名表 (右边的表，出现在 JOIN 子句中的最右边) 的所有行。不包括左边表中没有匹配的行。例如，titles 表和 publishers 表之间的一个右外部联接将包括所有的出版商，甚至包括那些在 titles 表中没有书名的出版商。

以上是 MSDN 中对链接的定义。现在我们就从这四种链接所使用的不同方法来看他们的结果有什么不同。

titles 表

sh 书号	ph 出版商编号
232342	001

0432 003

82478123 005

publishers 表

ph 出版商编号 mc 出版商名称

001 红虎

002 rmh

003 hazl

现要把这两个表的内容合成如下的表结构：

sh 书号 ph 出版商编号 mc 出版商名称

现在看看采用四种链接方法的结果会有什么不同。先说说他们的命令：

内联接：

```
sele titles.sh publishers.ph publishers.mc
from titles inner join publishers &&内联接中的 inner 是可以省略的
on titles.ph = publishers.ph
```

外联接：

```
sele titles.sh publishers.ph publishers.mc
from titles outer join publishers
on titles.ph = publishers.ph
```

左联接：

```
sele titles.sh publishers.ph publishers.mc
from titles left join publishers
on titles.ph = publishers.ph
```

右联接：

```
sele titles.sh publishers.ph publishers.mc
from titles right join publishers
on titles.ph = publishers.ph
```

大家可能看到，除了在 join 之前的那个关键字不同之外，其他地方是一模一样的，链接条件 即 on 那一部分 也是一样的。结果：

内链接：

232342 001 红虎

0432 003 hazl

全链接：

232342 001 001 红虎

Null Null 002 rmh

0432 003 003 hazl

82478123 005 Null Null

左链接：

232342 001 001 红虎

0432 003 003 hazl

82478123 005 Null Null

右链接：

232342　　001　001　红虎

Null　　Null　002　rmh

0432　　003　003　hazl

所以我们很容易记住：

1、左链接：就是以 join 的左边那个表为 主 ，以 titles. ph = publishers. ph 为判断标准，不管右边的表有没有对应的记录，都要把左边表的记录放在结果中去，但右边表没有相应的记录那应该放个什么数值进去？答案是就放个 Null，表示没有。在左链接中，某记录在右边表，却不在左边表，那是不放进去结果去的，原因是左边表才是 主 ，要不要放由它决定：它有的，就一定放进去，它没有的，就不要了。

2、右链接：和左链接一样，只不过为 主 的一方调过来了，换成是由右边做 主 。

3、内链接：和左、右链接不同，它一定要左、右两边都有的记录才会放进结果，如果有某个记录不存在于任何一边，那这个记录是不会出现在结果中去的。

4、外链接：跟内联接相，反，相当于左、右链接的合并：不管什么情况，只要某个记录出现在这两个表，就一定会出现在结果中去，然后象左、右链接的处理方法一样，用 Null 来填充没有对应值的字段。

注：以上说的 有 、没有 ，意思是以 titles. ph = publishers. ph 为判断标准来下决定的。比如当前 titles 表的 ph 是 002 ，而在 publishers 中，没有一个记录的 ph 的值是 002 的，所以就说 002 这个值在 titles 有，在 publisher 中没有，这样 titles. ph 为 002 的记录就会被选中，最后放在结果中去。

大家如果想一下，这个 on 的作用跟 where、having 似乎有点类似，都是起到过滤的作用：根据条件选取所取的记录，而根据命令的工作流程，这个 on 是比 where、having 都要早执行的，而它里面的条件表达式又不一定是 titles. ph = publishers. ph 的形式，还可以继续扩充，变成一个很复杂的条件表达式，从而完成一个很有效的、where 和 having 都不能实现的过滤功能。具体的比较请看 on、where、having 的区别 一节。

刚才举的例子，表中的 ph 都是没有重复的。现在以内联接为例，举个判断字段中内容有重复的例子：

Temp1　　temp2

Aa　　aa

1　　1

1　　2

2　　2

3　　2

```
sele temp1. aa  temp2. aa
from temp1 join temp2
on temp1. aa = temp2. aa
```

运行结果是：

1　1

1　1

2　2

2　2

2　2

很明显,有些记录重复了几遍。temp1.aa 中的虽然只有 1 个 2,但 temp2.aa 有 3 个 2,所以结果就会有 1*3=3 个 2 了。如果 temp1.aa 而 2 个 2 的话,那结果就会有 2*3=6 个 2 了。

知道了这一点,在做多表链接查询的时候很有用。你要考虑第一、二个链接后的结果跟第三个表链接时,会不会出现这种情况?如果有,那是不是你想要的?如果有,那怎么处理?有些朋友说做这个命令的结果中有些记录会比正确的结果大几倍,就要看看是不是出现了这种重复算的情况。

学会了链接,在开始做之前,先要说一个很重要的问题:在视图设计器来看多个表的联接关系,它们之间的链接是用一条线连接起来的,看起来就象一串糖葫芦。如果一个表同时和三个表联接,那看起来就象一支分叉的树枝了,那这种情况结果就不对了。大家可能不明白我在说什么,我举个例子大家就会明白了。

有一个产品表、一个进货明细表、一个出货明细表,现在的要求是要求产品表中所有的产品的进、出情况,也就是把三个表象 join 命令那样合成一个表,如果没有相应的进、出记录,也照样列出来但不计较 null 值。刚开始学的朋友很可就会这样做:

1、在设计器里添加这三个表;

2、然后用产品表中的产品编号分别与其它二个表左链接,这样产品表中就有二个链接　也就是二条线了　;

3、然后把三个表的字段都做为输出字段。

但结果呢?不对。只有一个表的记录出现在结果中,即使把四种链接类型都试一下,结果都是不对的。

为什么呢?我估计是以下原因:如果产品表只与进货表链接的话,系统根据产品表和进货表的联接关系,以产品表为左表,和进货这个右表组成一个临时结果,然后又以临时表为左表,再去找进货表的右边表。而进货表的右边没有表,这时系统就停止链接,交给 where 去过滤了。但现在产品表同时跟二个表左联接,系统会自动选其中一个先进行链接,链接结果出来后,这个临时结果的右边就没有表了,系统就停止链接动作了。剩下的出货表、退货表都还没链接,所以那个表等于没用。

解决的方法是:进货表用进货表的产品编号全链接产品表,然后产品表又用产品表的产品编号全链接出货表,进、货表的顺序可以调过来,但产品表一定要在中间,且两个链接类型都是全链接,否则结果都不对。这样的链接情况,在设计器里按链接中的各个表的左右顺序排起来,很直观的:就是一串!没有分叉。这个方法的实现过程就是:

进货表全链接产品表,即使某种产品没有进货,但得出来的结果也一样有这个记录,只是它的进货内容是 null 值。然后这个临时结果又跟出货表全链接,这次的结果就前一步差不多,有出货内容的记录就有出货数量,否则就是 null 值。因为没有分叉,所以全部表都链接进去了,结果也就对了　当然如果链接类型错了,结果也是不对的　。

看了刚才那个问题之后,还有一个问题也要说一下。在刚才那个例子中,如果产品表中某个产品编号出现了重复,有 N 个记录的编号相同,而在进货表里这个编号的记录也出现 M 个,这样一来,结果就有点不同了。首先在进货表跟产品表的全链接结果里,这个编号就会出现 N*M 次,就不是一次了。然后这个临时表再去跟出货表全链接时,即使这个编号在出货表里出现一次,但在最后的链接结果中,这个编号还是会出现 N*M 次,那它的出货记录也重复了 N*M 次了。如果现在要 sum　出货记录的话,那出货数量就会放大了 N*M 倍了,进货记录也不准了。所以如果产品表中的编号有重复的话,那结果就很可能会不对了。

但产品表的编号没有重复,那结果就一定会正确呢?也未必。大家试一下,假设进货表和产品表的编号 001 都是只出现一次,但出货表中就出现了二次。那最后的结果中 001 还是出现了二次,二次的产品名称、进货数量都是相同的,只是出货数量不同而已。如果这时 sum ,结果还是不对。

15.4.4　union 的使用

union 可以将几条 SQL 命令合成一条,要求是这几条命令生成的表,在字段个数、字段类型、字段长度、字段顺序上都完全一样。以下这种情况,一般都要使用它的:

把几个结构完全一样的表的记录都加在一起,最后生成的表,在结构上跟那几个表也完全一样,但记录数就是那几个表的记录数的总和。

举个例子:我想统计一段时间内的提货、进货情况,最后生成的表是这样的:

产品编号 cpbh　产品名称 cpmc　提货数量 thsl　进货数量 jhsl

先用二条命令分别计算提货数量和进货数量,生成二个临时表,最后用 union 合成一个表。

```
sele thdmx. cpbh  sum  iif  isnull  thsl   0  thsl    as thsl
from cpk left join thdmx
on cpk. cpbh = thdmx. cpbh
grou by cpbh
into curs temp1
sele jhdmx. cpbh  sum  iif  isnull  jhsl   0  jhsl    as jhsl
from cpk left join jhdmx
on cpk. cpbh = jhdmx. cpbh
grou by cpbh
into curs temp2
```

现在 temp1 和 temp2 的格式跟最后的结果有点不同,都是少了一个提货数量 进货数量 ,不能直接使用 union 联合。所以我们要人为给每个临时表加个对应的字段,命令如下:

```
sele cpbh  thsl  10000 - 10000 as jhsl
from temp1
union
sele cpbh  0     jhsl
from temp2
into curs temp3
```

在每节 SQL 命令,都加了一个字段,它的值都是零 没有嘛,当然是零啦 。这样一样,每节 SQL 命令生成的表在结构上就完全一样了,就可以使用 union 了。大家试一下,如果都不加个字段的话,那虽然不会出错,但结果的结构就跟要求不一样了。

在上面那条命令,每一节都可以使用 as 来给字段重新起名。如果在第一节使用了 as ,则以后的则可能不用了。否则的话,就是最后使用 as 的那节才起作用,前面的都无效了。

temp3 出来了,就可以使用分级合并了。

```
sele cpbh  sum  thsl   as thsl  sum  jhsl   as jhsl
from temp3
grou by cpbh
into curs temp4
```

现在这个 temp4 就是最后正确的结果了。

做了这么多步，大家应该明白这类联合统计的命令是怎样做的吧。但精于求精的我们是不会满足的，还可以对上面的那么多个步骤进行简化：

```
sele thdmx.cpbh sum iif isnull thsl 0 thsl as thsl 10000 - 10000 as jhsl
  from cpk left join thdmx
on cpk.cpbh = thdmx.cpbh
grou by cpbh
union
sele jhdmx.cpbh 0 sum iif isnull jhsl 0 jhsl
from cpk left join jhdmx
on cpk.cpbh = jhdmx.cpbh
grou by cpbh
into curs temp1
sele cpbh sum thsl as thsl sum jhsl as jhsl
from temp1
grou by cpbh
into curs temp2
```

大家看一下，使用了 union 是不是更简洁了？这里只统计进、出情况，如果再加上退、报废等情况，采用第一种方法就要使用五条命令，产生 6 个临时表。而采用第二种方法，无论再加多少种情况，都只需 2 条命令和 2 个临时表就可以。

只要记住 union 中每节 SQL 命令产生的结果中，字段个数、字段类型、字段长度、字段顺序上都完全一样的，才能进行联合。

而在分析一段很长的、又使用了 union 的 SQL 命令，可以按 union 分成一条一条 SQL 命令，然后再去分析每一条 SQL 命令，看每条命令是什么表采用什么链接类型进行链接，过滤条件是什么，按什么进行分组，进行什么样的字段汇总函数。只要懂得了 SQL 命令的工作流程顺序，再复杂的 SQL 命令都可以很快就看明白。

15.4.5 group by 分组的应用

首先先说说分组是怎样工作的。举个例子：

表：temp1 有以下记录：

```
bh   sl
aaa  1
aaa  4
ccc  2
bbb  5
aaa  9
bbb  7
```

现在要统计一下 temp1 有中多少种编号，各种编号的总数量又是多少。很明显，这是使用分组。

```
sele bh sum sl as total_sl
```

```
from temp1
grou by bh
orde by bh
into curs temp2
```

命令的运行过程就象投票选举中的点票过程一样。它中间到底是采用什么样的技术来储存中间结果我不知道，但不这重要，我现在假设它使用临时表 实际上我想不大可能 ，这样对结果没影响，但又容易理解。

1、是逐个扫描记录。每遇到一个新的编号，就为这个编号建立一个临时表 再次重申，使用临时表只是我的假设布局， 真正的处理方法我不知道 ， 然后把这个记录的内容放进对应的临时表中去。如果不是新编号，就直接把记录内容放进对应的临时表中去。

2、全部记录扫描完了，各种编号对应的临时表也产生了，现在就是对每一个临时表采用 sum 进行统计了。比如编号为 aaa 的临时表是 cursor1，那它的内容就是：

```
Aaa   1
Aaa   4
Aaa   9
```

现在进行 sum 统计了，它的效果就相当于 sele bh sum sl as total_sl from cursor1。每一个编号的临时表都要进行这样的 sum 统计，然后把每个编号的统计结果联合起来，就想使用 union 一样，最后得出的结果就是：

```
aaa   14
ccc   2
bbb   12
```

3、然后再给 bh 排序，结果就是：

```
aaa   14
bbb   12
ccc   2
```

看了上面那个例子，再看下面这条命令，想必也可以理解了吧：

```
sele cpk. cpbh cpk. cpmc sum iif isnull thdmx. thsl 0 thdmx. thsl as total_thsl
from cpk left join thdmx
on cpk. cpbh = thdmx. cpbh
grou by cpk. cpbh
orde by cpk. cpbh
into curs temp1
```

产品表左联接提货明细表，这样没有提过的产品，在联接后的临时表中，对应的提货数量就是个 null 值。到了分组，第一步完成后，生成的临时表中也仍然有 null 值，所以在第二步进行 sum 命令时，就要使用 iif isnull 来过滤了。过滤的结果就是把 null 值改为 0，其它的不变，然后再由 sum 进行汇总。最后第三步再排序，结果就出来了。

以上是分组的最基本用法。现在说说分组的一些古怪用法。

一、使用 recn

例子一：

有种情况，一大段本来是应该连续的号码，但中间地漏掉了一些号码，现在要把这些漏了的号

码找出来。恐怕很多人,一来就是使用 scan endscan,再加上一些中间变量以作为判断依据,最后写成的代码有一大段,运行起来慢得要死。但如果采用以下这种方法,我想即使它不是最好,但也是很好的了。数据如下:

1、6、3、5、2、7、9、10、15、16

在这些数据当中,要找出漏掉了的 4、8、11、12、13、14。

1、首先排序

sele bh from temp1 orde by bh into curs temp1

结果是:

1、2、3、5、6、7、9、10、15、16

如果源表中的物理记录顺序和号码的顺序一样就不用做这一步了。

2、取 temp1 中各个记录的 recn ,即记录号,然后把编号减去记录号

sele bh val bh - recn as aa from temp1 into curs temp2

结果就是:

Bh	aa
1	0
2	0
3	0
5	1
6	1
7	1
9	2
10	2
15	6
16	6

3、对 aa 使用分组,取每组中最小的 bh 和最大的 bh

sele min bh as minbh max bh as maxbh from temp2 grou by aa into curs temp3

结果如下:

minbh	maxbh
1	3
5	7
9	10
15	16

4、现在结果很明显了,某个记录的 minbh 跟它上面那个记录的 maxbh 中间相差的,就是漏掉的号码。现在才使用 scan endscan 就容易了。

以上方法适用于数据量大、第三步的结果比源表少很多记录的情况下才会发挥效果。因为 scan 的速度不能和 Select - SQL 相比,何况它还要作条件判断。虽然这种方法使用了二条全遍历的 SQL 命令,但这是没过滤条件的 SQL 命令,速度是很快的,只是分组要多点时间而已。同时只适用于编号没有重复的情况。如果编号有重复,那在第一步的时候,就要使用:

sele bh from temp1 goru by bh orde by bh into curs temp1

把多余的编号去掉。如果想了解那些编号是重复的,就可以使用:

sele bh coun bh as aa from temp1 grou by bh orde by bh havi aa> 1 into curs temp1

结果就是重复的编号,对应的 aa 字段就是重复的次数。

在第二步,好象可以不用二条命令就可以得出结果了,只是暂时还没想到,不知各位对些有什么看法。

上面这种方法有个缺点:就是使用了 sum 函数,而它是只能统计数值型的数据,其它字符型、日期型等是不能统计的。所以如果想把一个表中所有记录的产品名称都串起来,变成一个字符串,那是不能的。但对于日期型,变通一下,有时还是可以使用的。

二、日期相减的结果是数值,从而可以使用 sum 函数。

例子二:

编号 名称 入库日期

1 网卡 2000. 09. 02

1 网卡 2000. 09. 03

1 网卡 2000. 09. 05

2 vfp 2000. 09. 05

2 vfp 2000. 09. 06

说明:把编号和名称相同,入库日期相间为 1 的记录合并为一条,并加入另一个表里,如结果:

编号 名称 入库日期

1 网卡 2000. 09. 03

1 网卡 2000. 09. 05

2 vfp 2000. 09. 06

要完成上面的要求,用两个步骤:

1、sele * from 表 1 orde by 名称 入库日期 into curs temp1

2、sele 编号 名称 max 入库日期 入库日期 - recn as dd

from temp1

grou by 名称 dd

into curs temp2

问题的关键是那个 入库日期 - recn as dd ,因为已经按日期按好了顺序,所以 9 月 2 号减 1 和 9 月 3 号减 2 的结果都是相同的,但 9 月 5 号减 3 就是另外一个数了,所以这样就把这些日期分开了,就可以用分组了。假设表内容如下:(xm 是字符类型,其中的 1 代表上午上班,2 上午下班,3、4 如此类推)

这个例子,到在处理第二步的时候,其实就跟第一个例子一样了,都是把一个字段减去记录号,然后根据结果进行分组。

例子三:某员工某月的打卡记录 temp1 如下:

打卡时间 SJ 项目 XM 状态 ZT

2000/09/1308 01 00AM 1 On time

2000/09/1312 00 00AM 2 On time

2000/09/1302 00 00PM 3 On time

2000/09/1305 59 00PM 4 Early

2000/09/1208 00 00AM 1 On time

2000/09/1212 00 00AM 2 On time
2000/09/1202 00 00PM 3 On time
2000/09/1206 00 00PM 4 On time
2000/09/1408 00 00AM 1 On time
2000/09/1412 00 00AM 2 On time
2000/09/1402 01 00PM 3 Later
2000/09/1406 00 00PM 4 On time
2000/09/1508 00 00AM 1 On time
2000/09/1512 00 00AM 2 On time
2000/09/1502 00 00PM 3 On time
2000/09/1506 00 00PM 4 On time

xm 中的 1、2、3、4 分别代表早上上班、早上下班、下午上班、下午下班。zt 中 on time 表示准时，early 表示早退，later 表示迟到。在以上数据，把每天的上下班状态用以下的格式列出来：

日期 上午上班时间 上午下班时间 下午上班时间 下午下班时间 上午上班状态 下午状态

很明显，这是根据打时间进行分组。如果使用：

```
ele day sj as dd
iif xm = 1 sj {} as 上午上班时间
iif xm = 2 sj {} as 上午下班时间
  from temp1
grou by dd
```

这个方法不行，原因我详细说一下，可能会有点罗嗦。先说说 13 号这天的数据。在第一个记录，在第一个 iif ，xm = 1，上午上班时间就是早上八点。到了第二个记录，xm = 2，所以第一个 iif 的就是{}，即空白日期。到了第三、四个记录，上午上班时间都是空白日期，所以到最后的结果就是 13 号这天，上午上班时间是空白日期而不是早上八点！其他日期、除了下午下班时间之外其他时间都是如此。按前面说的分组过程，是先对每天的记录进行分组，然后按顺序一个个记录的计算 iif 的结果，每计算一次 iif ，都更新上午上班时间的值，这样在每天四个上下班时间中，前面三个记录都不起作用，只有第四个才起作用了。

那是不是就不能用分组了呢？那又不是，我们有 sum 函数，在 sum 函数里，它可以累计前面那三个记录的值，但要是数值型的字段才行。而现在却是日期型，所以我们要转换一下。命令如下：

先定义一个全程变量：initsj = {1900 - 01 - 01 00 00 00}，即 1900 年 1 月 1 日零时，用它来做基准时间。

```
SELECT DAY sj AS dd
initsj + SUM IIF xm = 1 sj - initsj 0 AS sj1
initsj + SUM IIF xm = 2 sj - initsj 0 AS sj2
initsj + SUM IIF xm = 3 sj - initsj 0 AS sj3
initsj + SUM IIF xm = 4 sj - initsj 0 AS sj4
IIF SUM IIF xm = 1 .AND. zt = on time 1 0 = 1 on time NO time
IIF SUM IIF xm = 3 .AND. zt = on time 1 0 = 1 on time NO time
FROM 数据 1! temp5
```

GROUP BY 1

结果如下：为了看得更清楚，我省略了两个字段 sj2　sj3，DD 是表示日期

DD	SJ1	SB3	Exp_6	Exp_7
12	2000/09/12 08:00:00 AM	2000/09/12 02:00:00 PM	On time	On time
13	2000/09/13 08:01:00 AM	2000/09/13 02:00:00 PM	NO time	On time
14	2000/09/14 08:00:00 AM	2000/09/14 02:01:00 PM	On time	NO time
15	2000/09/15 08:00:00 AM	2000/09/15 02:00:00 PM	On time	On time

这条命令的奥妙就在于日期可以相减，结果是一个数值，然后用 sum　进行累加；而日期加一数值，结果还是日期。在求早上状态时 xm = 1　，如果不是早上上班的时间，就累加零，否则就累加上班时间与基准时间的差。最后把结果再加上基准时间又得回原来的上班时间。

而字符那里，如果直接使用第一条命令那种做法，也是不行的。原因也一样，前面三个记录的结果都让第四个记录的值给覆盖了。因本例特殊点，具有唯一性，所以还可以 sum　+ iif　的方法。但 sum　不可以处理字符串，所以要用 iif　转换为数值型。

工作原理跟刚才计算时间那样，用内部的 iif　把 xm = 1　. AND. zt = on time 即上午准时上班的记录设为 1，而其它的时间或上午不是准时上班的就是 0，然后用 sum　累加，最后又用外面的 iif　对 sum　的结果进行判断，如果结果是 1，就表示上午是准时上班。因为只有一个上午上班记录 xm = 1　，而只有 zt = on time 才表示准时上班。

在决定是否用分组前，应先确定以哪些定段作为分组依据。分组依据确定后，就要检查一下，你认为某些记录是应该在同一个分组内的，但如果直接就去分组的话，这些记录又不是同一个分组内　看看第一和第二个例子　，就要想办法找出这些记录有什么共同点，然后根据这些共同点转换一下，得出一个相同的中间值　第一个例子就是减去各自的记录号从而找出共同点　。然后才根据这个共同点进行分组。

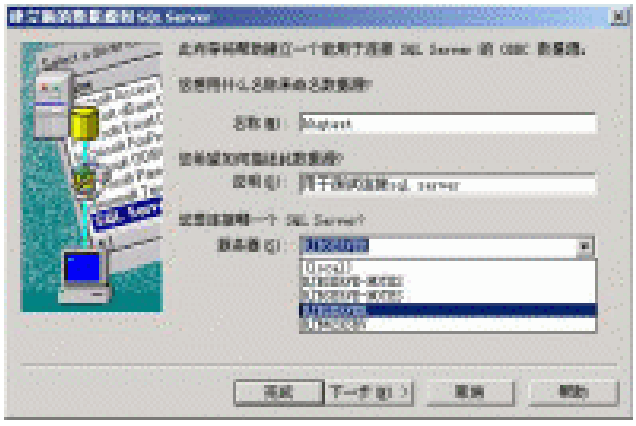
15.5　关于客户/服务器方式中的数据连接

15.5.1　ODBC 数据源的建立

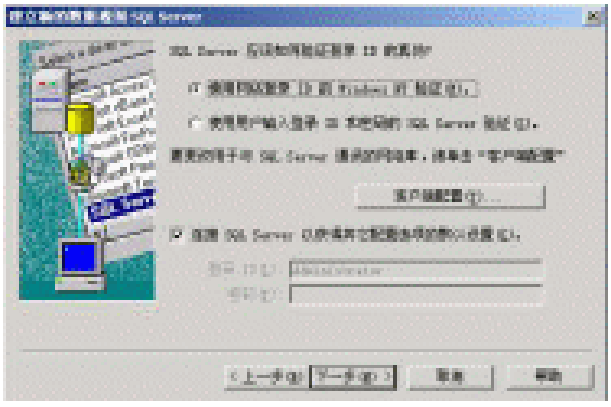
1. 打开 ODBC 数据源管理器，选择“用户 DSN”或“系统 DSN”后单击‘添加’

在‘创建数据源’窗口选中‘SQL Server’驱动程序后单击‘完成’。

2. 在‘名称’处输入一个你想好的 DSN data source name 名字 例中为 bhqtest ,‘说明’可以不写 ‘服务器’一栏可试选一下,如没有 除 local 外 ,可键盘输入要连接到的服务器的机器名 例中为 rjbserver——见‘实验环境’一节 . 单击‘下一步’.



3. 在‘如何验证 ID’窗口中,单击‘客户端配置’可配置网络协议等 .



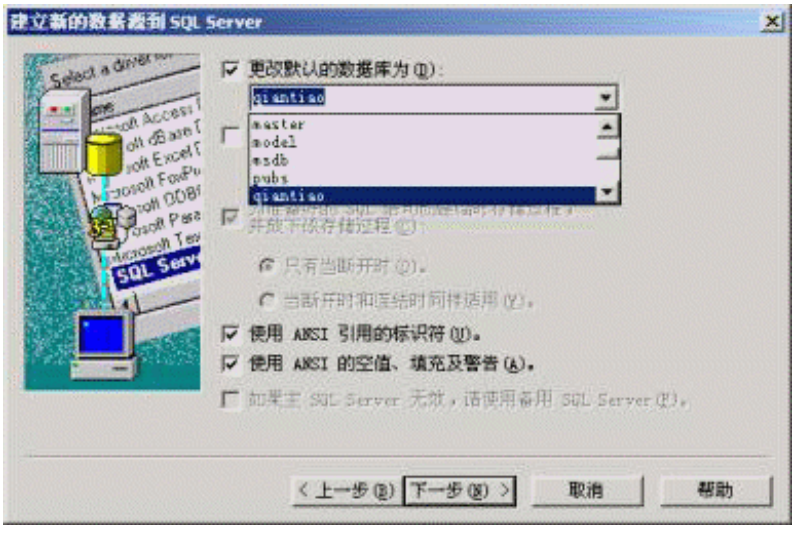
在此可指定使用服务器用户,也可指定使用后台数据库用户
两者选择的不同之处如下图

输好后单击‘下一步’—— 这之后可能要稍等一会儿 去验证你的 ID 了 .

至于对这两者用哪个好，我认为是使用后台数据库用户 . 因为数据库系统对用户的权限设置更细致、精确 从数据库——表——记录，都可设操作权限 .

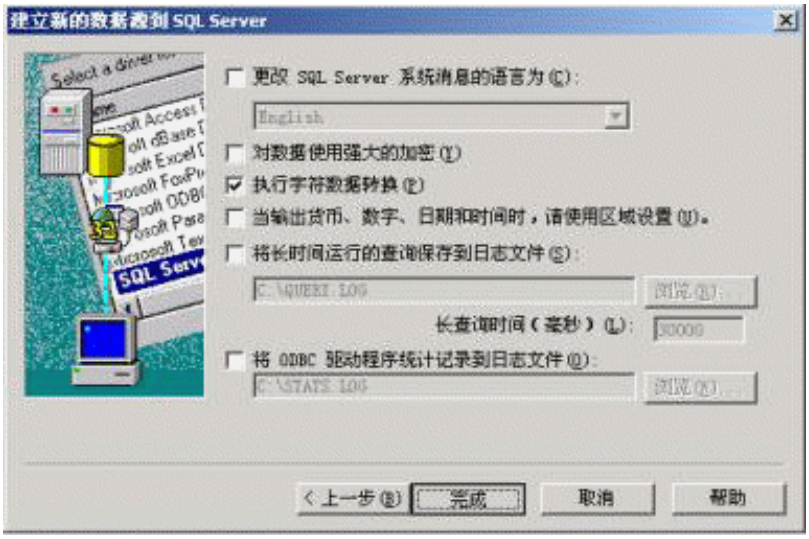
另外，一般不提倡使用管理员 administrator 或 sa 登陆

4. 复选‘更改默认的数据库为’复选框，在其下的下拉列表中选择你要连接的数据库名 例中为 qiantiao——见‘实验环境’一节 . 单击‘下一步’.



在此处，如果没有数据库名列表可选，那就是你在此步之前的几步填写有问题，单击‘上一步’重来吧！

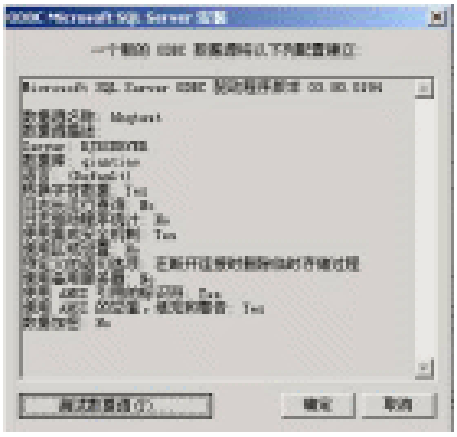
5. 该窗口内的参数一般不用改写 . 特殊需要时可试改之 . 单击‘完成’

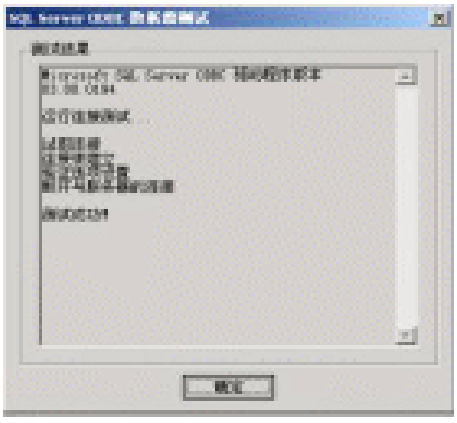


6. 接下是配置列表

单击‘测试数据源’就进行‘连接测试’了 . 结果列在下图

如在该处测试失败，请重来或请网络工程师测试网络 .





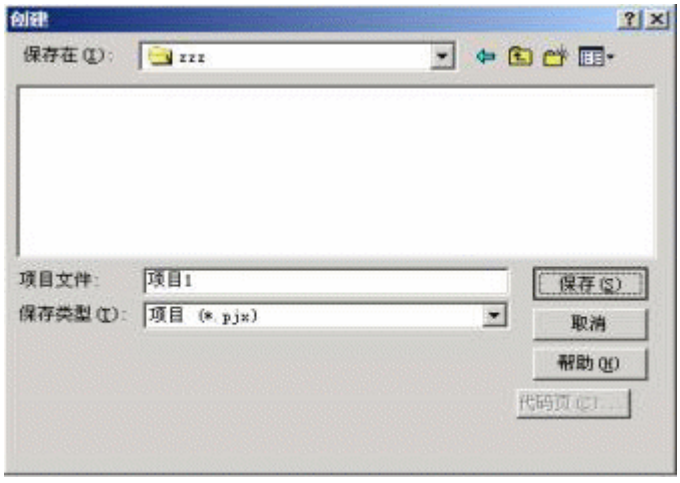
7. 测试通过后,在‘ODBC 数据源管理器’中可看到已建好的数据源名 例中为 bhqtest

至此,ODBC 数据源的建立、测试即告完成——别忘了单击‘确定’!

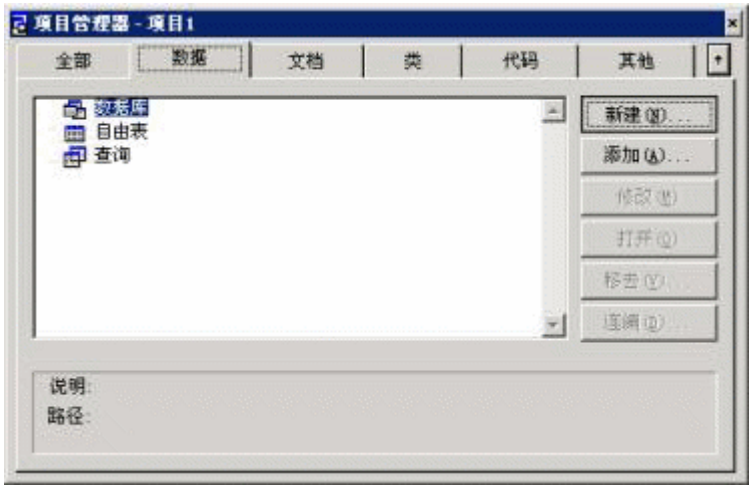
15.5.2 连接的建立

1. 在 VFP 中新建一个项目

选好路径,填写名称

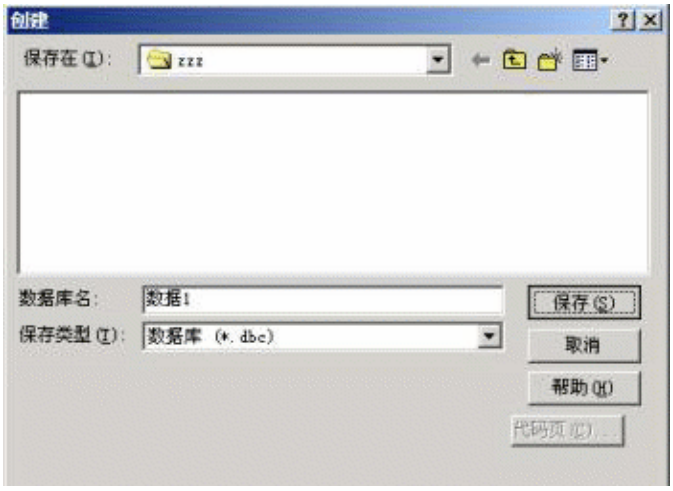


单击‘保存’例中为‘项目 1’

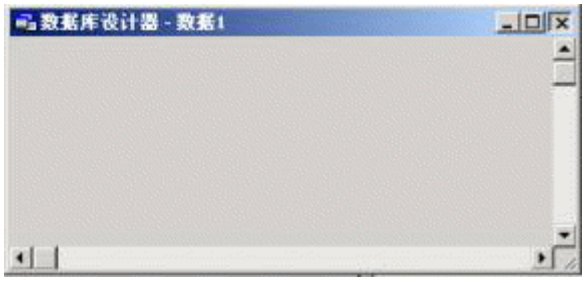


2. 在‘项目管理器’中单击击‘数据’、‘数据库’,单击‘新建’

选好路径,填写名称



单击‘保存’



以上就建好了一个空的 VFP 数据库 例中为‘数据 1’.

3. 单击‘数据库’、‘数据 1’

再单击‘连接’,单击‘新建’.

4. 在‘连接设计器’中选择数据源　下拉列表中将出现 ODBC 数据源名

选中数据源 例中为 bhqtest .

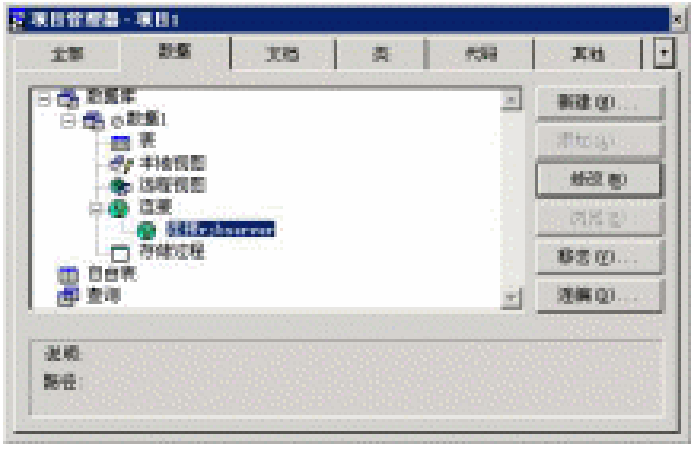
5. 输入用户名、口令后，单击‘验证连接’

如果连接失败，请检查 ODBC 数据源中的设置或网络连通情况 .

6. 连接成功后关闭‘连接设计器’右上角的 x

可修改连接名称 例中为‘连接 rjbserver ’

这时,可在‘项目管理器 ’中看到所建的连接

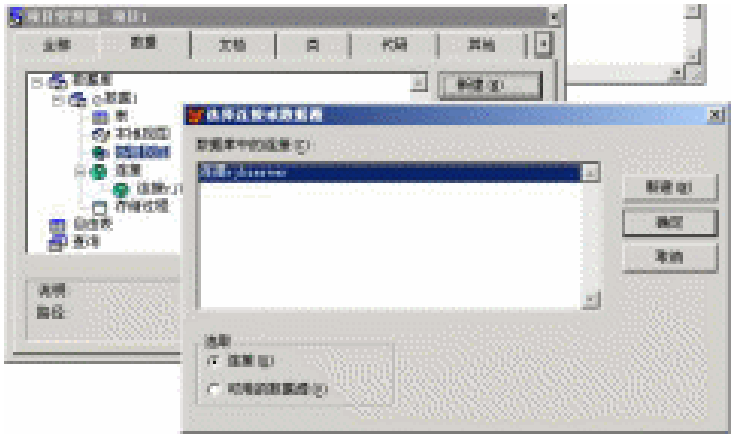


建好的‘连接 ’存放在 VFP 数据库中 例中为‘数据 1 ’ .

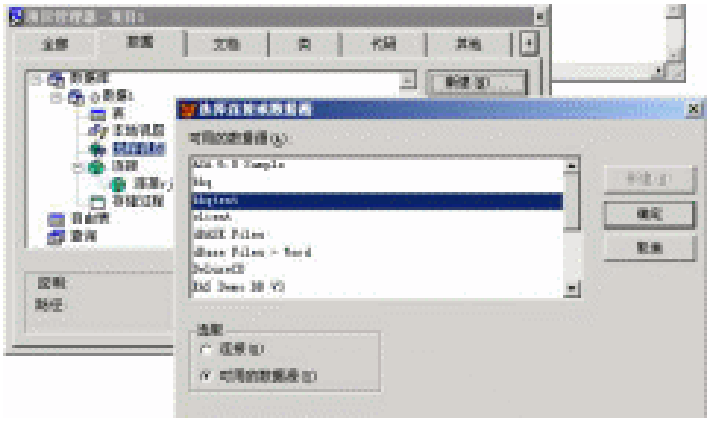
视应用情况,可以建多个‘连接 ’ 连接多个或多种数据库 ,但‘连接名 ’不应相同 .

15.5.3 远程视图的建立

1. 在‘项目管理器 ’中,单击‘远程视图 ’,单击‘新建 ’. 选中所需要的数据源 例中为‘连接 rjbserver ’

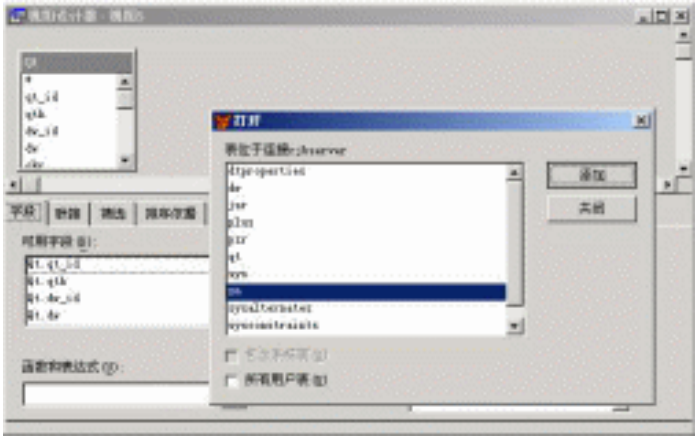


在此处也可直接用 ODBC 数据源 点击‘可用的数据源’



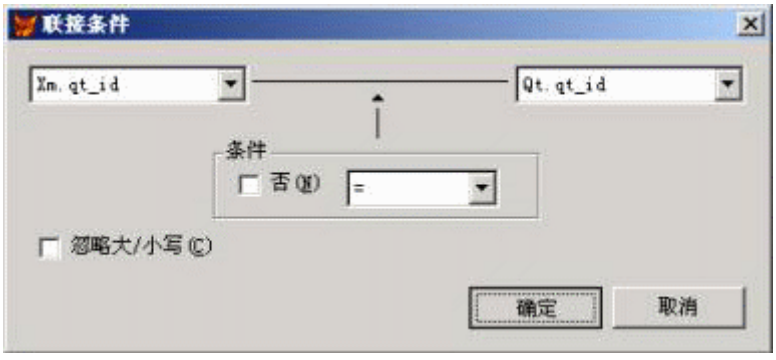
单击‘确定’出现‘视图设计器’.

2. 首次出现的‘视图设计器’有个‘打开表’的窗口 其它时间可按右键选‘添加表’.

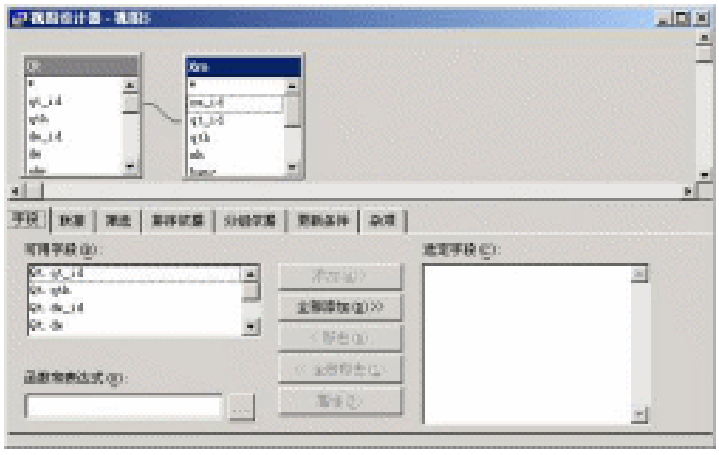


选中 qt 表,再选中 xm 表

自动出现这两个表的连接条件



看准对应条件 可另选 后,按‘确定’



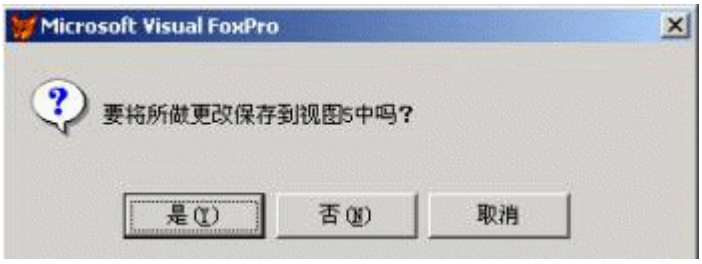
3. 添加字段于视图中 在‘可用字段’中选中字段并单击‘添加’即会在‘选定字段’中出现

上图列出了部分已选好的字段.

现在可以浏览数据了 在‘视图设计器’中按右键选‘运行查询’.

4. 浏览的部分数据

5. 关闭‘视图设计器’右上角的 x



可修改视图名 例中为‘视图欠条’

按‘确定’即可在‘项目管理器’的‘远程视图’中看到‘视图欠条’.

第 16 章　FoxPro 扩展教程

16.1　Visual Foxpro 的数据一致性设计

如何保证数据库系统的数据一致性长期以来一直是人们所关心的问题，传统的 Xbase 关系型数据库系统在此问题的处理上存在明显的局限性。VisualFoxPro 作为一个面向对象的数据库系统，成功地将一套控制信息存储和合法检验规则的数据字典引入了 Xbase 领域，从而使数据的一致性限制由庞大的程序代码变成了数据库的一部分，极大地增强 VFP 的数据处理能力。

一致性　这个术语描述了系统中存储的数据资源的正确状态，VFP 从数据有效性、触发器和引用完整性这三个方面来维持数据的这种一致性状态。在此，我们以一个销售系统为例，阐述在 VFP 中数据一致性的设计方法。

1. 数据的一致性要求

VFP 的数据字典只对数据库有效，而非针对自由表。就一个销售管理系统而言，其核心是销售发票，因此，我们就可以设计一个销售数据库（Sales. DBC），其中主要包含以下几个相互关联的数据表：

商品信息表（goods. dbf）：存储商品的有关信息。包含商品编码、商品名称、商品规格、计量单位、税率、参考单价等字段，其中 商品编码 为 主关键字。

客户信息表（customer. dbf）存储有业务关系的客户情况。包含了客户编码、名称、地址、开户银行、帐号、纳税人登记号等字段，其中 客户编码 为主关键字。

销售发票表（invoice. dbf）：存储所开发票的信息。包含了发票号码、开票日期、客户编码、备注、经办人等字段，其中 发票号码 为主关键字，并通过 客户编码 与客户信息表形成多对一的关联。

销售明细表（details. dbf）：存储与销售发票对应的商品的明细信息。包含了发票号码、商品编码、数量、单价、金额、税金等字段，通过 发票号码 与销售发票表形成多对一的关联，同时通过 商品编码 与商品信息表保持多对一的关联。

为确保数据的正确性，要求有以下的一致性限制：

开票日期的缺省值为当天日期

销售明细表的单价必须大于 0

销售明细表的金额 = 数量 * 单价

客户信息表中的开户行、帐号和税号字段均不能为空

当删除销售发票表的一条记录时，给出提示信息，确认后删除该记录，并且对应的销 售明细信息自动删除

当客户信息表的客户编号被修改时，销售发票表的对应客户编号随之改变

当商品信息表的商品编号被修改时，销售明细表的对应商品编号随之改变

若在销售发票表中存在某客户编号，则禁止删除该客户信息

若在销售明细表中存在某商品编号，则禁止删除该商品信息

现在我们就可以通过 VFP 的数据字典来实现这些 一致性要求。

2. 数据有效性

数据有效性分字段级有效性和记录级有效性，它们分别在 Fields Properties 字段属性 和 Table Properties (表属性) 中进行定义。

在 Table Designer (表设计器) 的左下部分可以看到字段属性分组，其中包含了用于字段有效性检测的四个域 Validation Rule (有效性规则)，Validation Text (字段有效性文本)，Default Value (缺省值) 和 Caption (标题)。有效性规则为一个逻辑表达式，当其返回值为 .f. 时，将显示一条内容为字段有效性文本的错误信息。缺省值为产生新记录时该字段的初始化值。

那么，对于开票日期字段，可以将其缺省值属性设为：

Default Value　date

对于单价 price 字段，可设为：

Validation Rule　price> 0

Validation Text　商品单价必须大于 0

Default Value

Caption　单价

字段的有效性规则在焦点离开字段时即被启动，如果需要同时检测多个字段的情况，那么就需要用到记录级的有效性规则。记录级有效性规则和字段有效性规则一样工作，只是它在离开记录时启动，并且只有在项被修改时才被检验。

点击 Table Designer 中右上角的 Table Properties 按钮，你就可以在表属性对话框中创建记录级的有效性规则。对客户信息表，可以建立如下的有效性规则以确保银行、帐号、税号均不为空。

Validation Rule　!empty　khyh　.and. !empty　zh　.and. !empty　sh

Validation Text　开户行、帐号和税号均不能为空

3. 触发器

就本质而言，对于表格里的一个记录，我们仅能做三件事情：删除、修改或建立，与之对应的三种触发器使我们能有效地控制这些事件发生，它们通常都使用一个函数，当返回的逻辑值为 .t. 就继续执行，否则就退出。

在表属性对话框中，除了记录有效性规则之外，还有三种触发器：INSERT Trigger、UPDATE Trigger 和 DELETE Trigger。如果想限制某一事件的发生，可直接在其域中填入 .f.，例如：在 UPDATE Trigger 中填 .f.，则该表不允许修改。对于本文例示中的销售发票表，我们可以事先在数据库的 Stored Procedure 中自定义一个函数 Inv_del，然后在删除触发器中填入：

```
DELETE Trigger Inv_del
```

自定义函数 Inv_del 的代码如下：

```
function Inv_del
*
* INVOICE. DBF Delete Trigger
*
local lnRrt
lnRet = MESSAGEBOX　是否确认删除 本记录　4 + 32 + 256　提示
do case
case lnRet = 6
select DETAILS
```

```
delete for invno = INVOICE.invno
select INVOICE
return .t.
case lnRet = 7
return .f.
endcase
```

这样,当你确认删除销售发票表的一条记录时,对应的销售明细信息将被自动删除。也许你已经注意到,本例中所示的触发器,其目的是为了维持数据的完整性,其实,如果不需要额外的确认提示,那么在 VFP 中维护数据的完整性有更方便的手段,那就是引用完整性构造器。

4. 引用完整性

保证所有子记录有一个有效的记录的过程被称为引用完整性,这是确保父子表之间关系的关键。

在 Database Designer 中右击鼠标打开菜单,选择 Referential Integrity,进入引用完整性构造器。该构造器的上部分为一个七列的网格,前两列包含所有父表和子表的名字,中间三列分别显示对更新、删除和插入操作的引用完整性类型,最后两列为关联表的关键字。你可以在网格中选择一行(即选中一种关联),然后为更新、删除和插入操作指定引用完整性的类型。

构造器下部分的页框给出了分别对应更新、删除 和插入操作的三个页面,每个页面均提供了三种引用完整性类型(Cascade、Restrict 或 Ignore)的单选按钮。通俗地讲,这三种引用完整性类型的效果分别为:

Cascade:当父表发生变化时,在所有子记录中作相同的改变。

Restrict:如果有子记录,不允许改变。

Ignore:忽略子记录,而只在父记录中修改。

至此,我们可以极其方便地解决本文开始所提的后五项数据一致性限制要求。例如:当客户信息表的客户编号被修改时,销售发票表的对应客户编号随之改变

从构造器上部网格中选择父表为 Customer 并且子表为 Invoice 关联行的 Update 列,并在 Rules for Updating 页面中选中 Cascade 选项,此时,网格中 Update 列中的内容应该为 Cascade,连续确认后即可。

其它几项要求的解决方法类似。

5. 几点有益的提示

虽然 VFP 的数据字典的功能极强,但它也并非是万能的,因此有必要注意以下几个方面的问题。

1 有效性检测的顺序

记录有效性规则适所有的记录,并且无论何时增加或修改记录时都被调用。若使用了字段有效性和触发器,则记录有效性规则必须返回一个逻辑值。记录有效性的调用在字段有效性之后,在触发器之前。

2 缺省值表达式不能有字段参与

字段属性中的缺省值表达式在记录创建时调用,而不是在进入字段时。对于 金额=数量*单价 这样简单的限制,你不能简单地定义金额字段的 Default Value 为 数量*单价 ,这样你得到的缺省值将永远是 0。

3 更新触发器不能改变启动它的记录值

同样是 金额 = 数量 * 单价 的问题,我们可能会认为此时用触发器是理想的时机,但千万别这么做,触发器不允许改变它自己记录的一个值,因为这种改变将导致更新触发器的再次调用,进而递归,形成死锁。其实,这个简单的金额问题根本就不能通过数据字典来解决。

4 触发器中不能移动当前记录指针

当触发器的验证规则试图移动当前工作区中的指针时,你会很不幸地看到出错信息 Illegal recursion in rule 。

5 自定义函数应事先存储

当自定义函数在验证规则编辑框中被引用时,如果出现了 函数 * * * 没找到 的出错信息,通常有两种情形引起这个错误,一是在创建该函数前就输入该函数的调用,二是虽然在存储过程中创建了该函数,但没有保存该存储过程。所以一定要保存存储过程。

以对象方式处理记录

RMH

以对象方式处理记录

在命令窗口中打入以下代码并注意结果 .

```
create table 客户　客户名 c 6
insert into 客户　客户名　values　张三
insert into 客户　客户名　values　李四
go 1
scatter name 第一个
go 2
scatter name 第二个
  第一个 . 客户名 && 显示 张三
  第二个 . 客户名 && 显示 李四
  type　第一个　&& 显示 O, 表明变量"第一个"是一个对象
  type　第二个　&& 显示 O 表明变量"第二个"是一个对象
第一个 . 客户名 = 第二个 . 客户名
  第一个 . 客户名 && 显示 李四 表明 :对象"第一个"的属性"客户名"已被替换为 :对象"第二个"的"客户名"属性的值。
go 1
  客户 . 客户名 && 显示 张三 :表明对象属性值的替换并未影响表中记录的值
gather name 第一个
  客户 . 客户名 && 显示 李四　:表明 对象 "第一个"和 "第二个"没有方法 但在很多情况下确非常有用 .
```

在 SCAN. . . ENDSCAN 中重新选择主表不是必须的

RMH

在 SCAN. . . ENDSCAN 中重新选择主表不是必须的

注意以下代码

```
select invoices && 选择发票表
locate
scan
```

select customers && 选择客户表

custm = invoices. custcode

seek custm && 在客户表中搜索相关发票和客户

insert into cursorx custname amount values customers. name invoices. amount

&& 在通常情况下 此处都要加上一句

&& select invoices

&& 但是这并不是必须的

endscan

&& 以上代码对于发票表中的每一个客户都能处理得很好.

&& 另外 SCAN WHILE NOT EOF 中的 WHILE NOT EOF 也不是必须的.

如何添加已存在的表单到表单集

RMH

最后检查日期 1995. 10. 23

文号 Q138044

本文信息应用于

Microsoft Visual FoxPro for Windows version 3. 0

概要

创建表单集时 没有明确的办法添加已存在的表单到表单集. 本文向你展示如何添加已存在的表单到表单集.

更多信息

假设要添加表单 MyForm 到表单集 按以下步骤

1 打开 MyForm.

2 在文件菜单中 单击另存为类 并单击保存表单.

3 为新类取一个类名和要保存的类库文件名 .VCX 文件 .

4 关闭 MyForm.

5 进入表单设计器 打开你的表单集 单击表单设计工具条中的 查看类 然后单击添加.

6 选择你保存 MyForm 类的类库文件 .VCX 并单击打开.

7 选择工具条中的 MyForm 并将该表单类拖放到表单集中.

这样就把 MyForm 类的一个实例添加到了表单集中.

16.2 如何以编程方式添加数据环境到表单

概述

在用表单设计器创建表时 数据环境对象被自动添加到表单. 要在编程方式建立的表单中模仿该行为 在表单中必须定义一个 DataEnvironment 属性. 然后一个指向数据环境的对象可以保存在表单的 DataEnvironment 属性中.

更多信息

建立一个新的程序并打入以下代码

oform1 = CREATEOBJECT Myform && 实例化表单 oform1. SHOW 0 oform1. refresh READ

```
EVENTS RETURN
    DEFINE CLASS Data1 AS dataenvironment   && 数据环境类
    Name  =   Dataenvironment
    ADD OBJECT   Cursor1   AS MyCursor      && 为数据环境中的各游标
    ENDDEFINE                               && 添加 Cursor 类对象
    DEFINE CLASS MyCursor AS cursor
    Alias  =   customer
    CursorSource  =   customer
    Name  =   Cursor1
    ENDDEFINE
    DEFINE CLASS Myform AS form
    Top  =  0
        Left  =  0
    Height  =  386
    Width  =  587
    DoCreate  =  . T.
    Caption  =   Form1
    Name  =   FORM1
    DataEnvironment =                   && DataEnvironment 是表单的一个属性
    PROCEDURE Destroy
    WAIT WINDOW PROGRAM    TIMEOUT 2
    THISFORM. DATAENVIRONMENT. CLOSETABLES  && 在 destroy 过程的结束处关闭表
    ENDPROC
    PROCEDURE Unload
    CLEAR EVENTS
    ENDPROC
    PROCEDURE Load
    * DataEnvironment 被实例化 且表单的属性
    * DataEnvironment 引用该对象 . DataEnvironment
    * 并没有实际地添加到表单 . THISFORM. DATAENVIRONMENT
    * 只是一个指向 DataEnvironment 对象的指针 .
    THISFORM. DATAENVIRONMENT = CREATEOBJECT   Data1
    * 在 LOAD 开始时打开表 .
    THISFORM. DATAENVIRONMENT. OPENTABLES
    WAIT WINDOW PROGRAM    TIMEOUT 2
    ENDPROC
    ENDDEFINE
    打开调试窗口 .
    在调试窗口中打入以下命令
    _SCREEN. ACTIVEFORM. DATAENVIRONMENT. NAME
```

运行第一步中创建的程序.

16.3 如何在运行时添加表到表单的数据环境

概述

表或视图可以在运行时添加到 Visual FoxPro 表单中. 虽然你可以用 USE 命令打开表 但将其添加到表单的数据环境更好. 如果表单是设置为私有数据工作期 则该表会只添加到当前数据工作期的数据环境中.

逐步演示

用表单设计器创建一个新表.

设置表单的 DataSession 属性为 2 - 私有数据工作期.

添加一个命令按钮到表单 并放入以下代码到它的 Click 事件中

```
WITH THISFORM.DataEnvironment
. ADDOBJECT mycursor cursor
. mycursor.Database = SYS(2004) + samples\testdata.dbc
. mycursor.CursorSource = customer
. CloseTables && 关闭所有的表和与数据环境相关的视图
. OpenTables && 打开所有的表和与数据环境相关的视图
ENDWITH
```

保存表单. 然后运行表单的两个实例.

单击表单的第一个实例上的命令按钮 但不要

单击表单的第二个实例上的按钮.

在调试窗口中打入

```
_SCREEN.ACTIVEFORM.DataEnvironment.mycursor.CursorSource
```

当你在两个表单间切换时 注意调试窗口中的值的改变与活动的数据工作期有关. 表只在表单的第一个实例的数据环境中.

16.4 用应用程序的表单替换 Visual FoxPro 的桌面

概述

在设计 Visual FoxPro 应用程序时可能要求隐藏 Visual FoxPro 桌面并让应用程序自己的表单出现在整个窗口桌面区域.

本文添加必须的步骤来生成应用程序表单桌面. 并用必须的步骤终止应用程序和恢复 Visual FoxPro 桌面.

逐步演示

启动应用程序表单并用 READ EVENTS 命令建立 Visual FoxPro 事件处理器. 在 .prg 文件中的代码继续处理的过程需要 终止应用程序的表单 停止 Visual FoxPro 事件处理器 并返回 Visual FoxPro 桌面. 在这些代码后面的是隐藏 Visual FoxPro 主窗口的函数.

＊＊

```
** Starter.prg 启动应用程序的最少的代码
DO FORM appform.scx
READ EVENTS
PROCEDURE back2vfp
** 应用程序表单已在桌面级作为桌面级表单激活. 事件处理器是在桌面级 并
** 似于 _SCREEN 对象的一个属性相等 因此它必须被 _SCREEN 对象的方法终
** 止. 由于没有可用的 _SCREEN 对象成员 没有命令按钮 文本框等 使用一
** 个 Timer 事件来清除事件. timer 对象是不可视的 且不需要激活.
** 要激活计时器 添加它到 _SCREEN
*
_SCREEN.AddObject oTime MyTimer && MyTimer 是
**                                  当前实例
** Back2vfp 过程结束
**
** 类定义. TIMER 事件两秒钟触发一次
DEFINE CLASS MyTimer AS Timer
Interval = 2000
PROCEDURE Timer
CLEAR EVENTS
_SCREEN.RemoveObject oTime
ENDPROC
ENDDEFINE
**
** Starter.prg 结束 **********************
```

在应用程序表单中 定义了三个表单属性来用于以下函数中的全局引用. 同时放置一个命令按钮或其它对象到表单中 该对象提供一个可以终止应用程序的事件. 例如 这三个属性为 FoxHWND FoxGone 和 ShowState. 该命令按钮名为 Quit. 它的 Click 事件将包括终止表单的代码.

警告 如果使用模板类创建表单 且模板的 DeskTop 属性设置为 .F. 则设置已创建的表单的 DeskTop 属性为 .T. 将不起作用. 在这种情况下 运行表单将造成 Visual FoxPro 主窗口和它内部的表单不可见 这就必须重新启动 Windows 3.x 并可能破坏应用程序相关的文件.

16.5 获取一些重要的系统路径

在数据库应用程序设计中，经常需要获得一些重要的系统路径，如 Windows 系统的安装目录，SYSTEM 目录的所在路径，当前的工作目录等。以下函数通过调用 Windows API 函数实现了这些功能。

```
DECLARE LONG GetSystemDirectory IN WIN32API
STRING @ lcSysDir LONG
DECLARE LONG GetWindowsDirectory IN WIN32API
```

```
STRING @ lcWinDir  LONG
DECLARE LONG GetCurrentDirectory IN WIN32API
LONG  STRING @lcCurDir
LOCAL lcSysDir  lcWinDir  lcCurDir  lnStringLen
lcSysDir = SPACE 200  + CHR 0
lcWinDir = SPACE 200  + CHR 0
lcCurDir = SPACE 200  + CHR 0
lnStringLen = GetSystemDirectory @lcSysDir  200
lcSysDir = LEFT lcSysDir lnStringLen
lnStringLen = GetWindowsDirectory @lcWinDir  200
lcWinDir = LEFT lcWinDir lnStringLen
lnStringLen = GetCurrentDirectory 200  @lcCurDir
lcCurDir = LEFT lcCurDir lnStringLen
  lcSysDir
  lcWinDir
  lcCurDir
```

16.6 获得当前 Windows 的缺省语言版本号

在程序设计、尤其是大型程序设计中,我们经常需要获取 Widnows 的缺省语言版本号。这本身并不是一个困难的问题,困难的是获得该版本号以后,如何与其含义相对应。本文给出了一个语言版本号及其含义列表以供朋友们参考。

```
DECLARE Integer GetSystemDefaultLangID IN Win32Api
nLangId = GetSystemDefaultLangID
 TRANSFORM nLangId  @0
Below are the Language ID s
0x0000 Language Neutral
0x0400 Process Default Language
0x0401 Arabic  Saudi Arabia
0x0801 Arabic  Iraq
0x0c01 Arabic  Egypt
0x1001 Arabic  Libya
0x1401 Arabic  Algeria
0x1801 Arabic  Morocco
0x1c01 Arabic  Tunisia
0x2001 Arabic  Oman
0x2401 Arabic  Yemen
0x2801 Arabic  Syria
0x2c01 Arabic  Jordan
```

0x3001 Arabic　Lebanon
0x3401 Arabic　Kuwait
0x3801 Arabic　U. A. E.
0x3c01 Arabic　Bahrain
0x4001 Arabic　Qatar
0x0402 Bulgarian
0x0403 Catalan
0x0404 Chinese　Taiwan Region
0x0804 Chinese　PRC
0x0c04 Chinese　Hong Kong SAR　PRC
0x1004 Chinese　Singapore
0x0405 Czech
0x0406 Danish
0x0407 German　Standard
0x0807 German　Swiss
0x0c07 German　Austrian
0x1007 German　Luxembourg
0x1407 German　Liechtenstein
0x0408 Greek
0x0409 English　United States
0x0809 English　United Kingdom
0x0c09 English　Australian
0x1009 English　Canadian
0x1409 English　New Zealand
0x1809 English　Ireland
0x1c09 English　South Africa
0x2009 English　Jamaica
0x2409 English　Caribbean
0x2809 English　Belize
0x2c09 English　Trinidad
0x040a Spanish　Traditional Sort
0x080a Spanish　Mexican
0x0c0a Spanish　Modern Sort
0x100a Spanish　Guatemala
0x140a Spanish　Costa Rica
0x180a Spanish　Panama
0x1c0a Spanish　Dominican Republic
0x200a Spanish　Venezuela
0x240a Spanish　Colombia
0x280a Spanish　Peru

0x2c0a Spanish Argentina
0x300a Spanish Ecuador
0x340a Spanish Chile
0x380a Spanish Uruguay
0x3c0a Spanish Paraguay
0x400a Spanish Bolivia
0x440a Spanish El Salvador
0x480a Spanish Honduras
0x4c0a Spanish Nicaragua
0x500a Spanish Puerto Rico
0x040b Finnish
0x040c French Standard
0x080c French Belgian
0x0c0c French Canadian
0x100c French Swiss
0x140c French Luxembourg
0x040d Hebrew
0x040e Hungarian
0x040f Icelandic
0x0410 Italian Standard
0x0810 Italian Swiss
0x0411 Japanese
0x0412 Korean
0x0812 Korean Johab
0x0413 Dutch Standard
0x0813 Dutch Belgian
0x0414 Norwegian Bokmal
0x0814 Norwegian Nynorsk
0x0415 Polish
0x0416 Portuguese Brazilian
0x0816 Portuguese Standard
0x0418 Romanian
0x0419 Russian
0x041a Croatian
0x081a Serbian Latin
0x0c1a Serbian Cyrillic
0x041b Slovak
0x041c Albanian
0x041d Swedish
0x081d Swedish Finland

0x041e Thai
0x041f Turkish
0x0421 Indonesian
0x0422 Ukrainian
0x0423 Belarusian
0x0424 Slovenian
0x0425 Estonian
0x0426 Latvian
0x0427 Lithuanian
0x0429 Farsi
0x042a Vietnamese
0x042d Basque
0x0436 Afrikaans
0x0438 Faeroese

16.7　怎样在 Visual FoxPro 中增加与去除网络联接

概述

在 FoxPro for Windows 2.x 中用 Foxtools.fll 和在 Visual FoxPro 中用 DECLARE DLL 命令定义相关的 Windows API 应用程序编程接口 函数　可以在 FoxPro 中增加与去除网络联接.

仅管 Visual FoxPro 仍然支持 FOXTOOLS 库作为向后兼容　DECLARE 命令是调用 DLL 函数的更好的方法.

更多信息

以下章节包括 FoxPro 2.x 和 Visual FoxPro 用法约定.

FoxPro 2.x

在 FoxPro 2.x 中用 FOXTOOLS 库　按以下步骤添加和移除网络连接.

用以下命令载入库

SET LIBRARY TO SYS 2004 + FOXTOOLS.FLL ADDITIVE

注册你要调用的 Windows API 函数. 在目前情况下我们要用到的是 WNetAddConnection　和 WNetCancelConnection　.

addconn = RegFn　WNetAddConnection　CCC　I

delconn = RegFn　WNetCancelConnection　CI　I

要连接到网络设备　发布以下命令

= CallFn addconn　hSERVER HARE　password　<drive>

要断开网络连接　发布以下命令

= CallFn delconn　<drive>　0

Visual FoxPro

使用 DECLARE DLL 命令定义要调用的 DLL 函数

＊＊ - - DLL 定义

Declare integer WNetAddConnection in WIN32API string string string Declare integer Declare integer WNetCancelConnection in WIN32API String integer

＊＊- - 添加网络连接

=WNetAddConnection hSERVERHARE DriveLetter

＊＊- - 移除网络连接

=WNetCancelConnection DriveLetter 0

以下信息提供了这两个 API 调用的附加的参考材料.

WNetAddConnection

WNetAddConnection 函数重定向指定的本地设备 磁盘或打印端口 为给定的共享设备或远程设备. 它使用以下参数

lpszNetPathName

指向以 null 结尾的字符串 该字符串指定了要连接的网络资源 如 hServehare.

注意 通常 Novell 用户不使用 符号来引用服务器和目录. 例如 不要试着用以下方法来引用一个目录

hserverolume ydirectory.

而应使用以下方法

hserverolumeydirectory

lpszPassword

指向以 null 结尾的字符串 该字符串指定了要用来进行连接操作的口令. 该参数通常是与当前用户相关的口令 如果该参数为 null 那么使用缺省口令. 如果该字符串为空 则不使用口令 使用一介空串作为占位符

=CallFn addconn hSERVERHARE <drive>

lpszLocalName

指向以 null 结尾的字符串 该字符串指定了要被重定向的本地设备. 所有 lpszLocalName 串如 LPT1 是要区分大小写的. 只使用了设备名 A 到 Z 和 设备名 LPT1 到 LPT3.

WNetCancelConnection

WNetCancelConnection 函数取消网络连接. 它使用以下参数

lpszName

指向以 null 结尾的字符串 该字符串指定了重定向的本地设备名 如 LPT1 或 D 或解除连接的远程网络资源。当该参数指定了一个重定向的本地设备,则该指定的设备的重定向被解除。若该参数指定了 一个远程网络资源,那么只有该远程资源的连接而不是设备被解除。

fForce

指定即使在连接上有打开的文件或任务时,是否任进行中断连接。 如果该参数为 FALSE ,那么在有打开的文件或任务时,调用该函数失败.

16.8 通过编程运行拨号网络连接

概述

用 FoxPro 的 RUN 命令 可以以编程方式启动 Windows 95 的拨号网络连接.

更多信息

以下命令可用于启动拨号网络　DUN　连接

RUN　/N Rundll Rnaui. dll　RnaDial　<your DUN connection name>

对于需要检查名字的连接　用以下命令启动连接

RUN　/N Rundll Rnaui. dll　RnaDial Test

其的函数名 RnaDial 和连接名是要区分大小写的 .

注意　仅管 Rnaui. dll 文件内的函数允许你以编程方式拨号　但没有可用的以编程方式断开连接的函数 .

要查看 Rnaui. dll 或任何 . dll 文件中可用的函数　找到该文件　右击该文件　然后选择 QuickView. 就可以从下方列出的输出表中看到该文件中的可用的编程函数 .

如果 QuickView 没有出现在右击菜单中 则需要安装 QuickView. 从控制面板中选择 添加 / 删除程序　然后选择 Windows 95　or Windows NT　安装标签 . 然后单击 Accessories 并选择 Quick-View.

以下是 Rnaui. dll 文件的输出表中列出的可用的函数

输出表

Name　Rnaui. dll

Characteristics　00000000

Time Date Stamp　320c06ce

Version　0. 00

Base　00000001

Number of Functions　00000009

Number of Names　00000009

Ordinal	Entry Point	Name
0000	0000773e	DllCanUnloadNow
0001	00002a80	DllGetClassObject
0002	00007880	Remote_CreateEntry
0003	000029ef	Remote_CreateInstance
0004	00003988	Remote_EditEntry
0005	00003260	Remote_Notify
0006	000031b9	RnaDial
0007	000031da	RnaRunImport
0008	00007a6a	RnaWizard

16. 9　在 VFP 应用程序中调用 MS - DOS 应用程序

在 VFP 数据库应用程序设计中，经常需要调用一些以前在 MS - DOS 环境下运行的应用程序。这时候我们经常会想到 RUN 命令。利用 RUN 命令确实可以运行 MS - DOS 应用程序，但是却会跳出一个令人讨厌的黑屏幕。利用本文介绍的方法调用 MS - DOS 应用程序，就可以解决这个问题了。

```
FUNCTION run
PARAMETER doscmd
DECLARE INTEGER WinExec IN win32api AS run
STRING command  INTEGER param
cmdstart = fullpath  FOXRUN.PIF  +  /C
fullcmd = cmdstart + doscmd
retval = run fullcmd  0
RETURN retval
```

在使用这个程序片段时，必须保证资源文件 FOXRUN.PIF 能够在当前工作目录下找到（可以将其从 VFP 的根目录下拷贝到应用程序目录下，不要对其缺省设置进行任何改动）。在调用 MS-DOS 应用程序时，可以使用如下语句：

```
DO RUN WITH  DosProgramName
```

例如我们需要运行在当前工作目录下的 MS-DOS 应用程序 TEST.EXE，就可以使用如下语句：

```
DO RUN WITH  TEST.EXE
```

16.10 将文本文件调入表单的编辑框的方法

最简单的方法就是利用 FoxPro 的 Append Memo 命令。

```
Create Cursor curFile  TxtFile M
Insert Into curFile Values
Append Memo TxtFile From File.txt
ThisForm.edtBase1.ControlSource = curFile.TxtFile
```

这样调入的文本文件在编辑框中是可编辑的，如果允许改变其中的内容，可以使用如下命令将改变以后的内容写回到原来的文件中。

```
Copy Memo TxtFile To File.txt
```

另外，在 Visual FoxPro 6.0 中提供了两个新的函数 FileToStr 和 StrToFile 这使得以上功能的实现变得更加容易了。

也可以使用低级文件函数来实现以上功能。

```
xnFile = FOPEN  File.txt
xnString = FGETS xnFile
thisform.edtbase1.value = xnString
```

另外，还可以使用 RTF 文本控件来实现以上功能。将该控件放置到表单上，然后设定其文件名属性就可以了。在以下的例子中，允许用户自己指定一个文件名。

```
LOCAL lcFile
lcFile = GETFILE  txt
THISFORM.oleRTFedit.FileName = lcFile
```

16.11　VFP 调用 EXCEL 的补充方法

下面是我使用 VFP CALL EXCEL 的部分例子：

这是本人从书本上抄的片断和我的小小经验，可以让你很方便的在 VFP 中调用 EXCEL，所有的例程我都试验过。

用 Visual Foxpro 设计用 Excel 表格的程序

利用 OLE Automation 设计 Excel 应用程序

Excel 支持的对象说明：

a　VBA 对象：

对象名称 意义

Application Excel 应用程序对象

WorkBooks Excel 活页薄对象

b 所使用的 Method

对象名称 Method 执行意义

Application Cells 设定或传回来某个网格的内容

Range 传回或设定某一个范围的网格

Charts 传回或设定活页簿的单一统计表

Quit 结束 Excel Application

Save 激活存储文件对话框

WorkBooks Add 新增一个工作簿

Charts Add 新增一个统计图

c 所使用的 Property

对象名称 Property 设定意义

Application Visible 是否现实再 SCREEN 上 . T.　. F.

Value 传回或者设定存储文件的内容

ActiveSheet 回应 Excel Application 执行工作表对象

实例说明：

启动 Excel：

MyExcel = CreateObject　Excel. Application　&&建立 Excel 对象

MyExcel. Visible = . T.　&&让 Excel 对象再屏幕上显示出来

如何增加工作簿：

MyExcel. WorkBooks. Add &&在 Excel 对象中增加一份工作簿 (WorkBook

如何在工作簿中增加 Sheet 工作表)

MyExcel. Sheets. Add &&增加工作表 (在当前工作簿中)

如何删除工作表

MyExcel. ActiveWorkBooks. Sheets 1 . Delete &&把工作簿中的 BOOK (1) 删除

向指定的工作簿中的工作表 Sheet 中存储数据

＊Excel. application Object

＊Excel. application. ActiveWorkBook Property

＊WorkBOoks Object

＊WorkBOoks Object 的 Add Method.

＊Sheets 对象

＊Sheets Index 对象指定索引工作表

＊Excel. Application 对象的 Cells Method 结合 Value 属性

Example

```
CLEAR ALL
SET PATH TO SYS 2004 + SAMPLES ATE
USE CUSTOMER
MYEXCEL = CREATEOBJEXT EXCEL. APPLICATION
MYEXCEL. VISIBLE = . T.
MYEXCEL. WORKBOOKS. ADD
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS 1 1 . VALUE = 客户编号
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS 1 2 . VALUE = 公司行号
SELECT CUSTOMER
R = 2
C = 1
GOTO TOP
FOR I = 1 TO 20
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS R C . VALUE = CUSTOMER. CUST_ID
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS R C + 1 . VALUE = CUSTOMER. COMPANY
R = R + 1
SKIP
ENDFOR
```

调整单元格宽度：

```
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS 1 1 . columnwidth = 30
```

调整单元格对齐方式：

```
MYEXCEL. ACTIVEWORKBOOK. SHEETS 1 . CELLS 1 1 . horizontalalig = 1
```

1 为默认方式，2 为左对齐，3 为中对齐，4 为右对齐。

如何将数据存储：

```
MYEXCEL. SAVE
```

注 可以用 Save FileName 指定预存储文件名 则可不用激活 SAVE AS 窗口

如何打印表格：

```
MYEXCEL. ActiveWorkBook. PrintOut &&默认打印增个 Sheet
```

如何指定打印表格：

```
MYEXCEL. ActiveWorkBook. PrintOut 1 1 1 . T. &&默认打印增个 Sheet
```

PrintOut 有四个参数

A. 数值 表示指定的工作簿中进行打印的 Sheet 的开始编号

B. 数值　表示指定的工作簿中进行打印的 Sheet 的结束编号

C. 打印分数 .

D. 是否进行 Preview　. T. 预览　. F. 打印

如何产生统计图

```
CURROW = MYEXCEL. ACTIVESHEET. ROWS. COUNT
RANGESTRING =  A1  +  B  + ALLTRIM  STR  CURROW
MYEXCEL. RANGE  RANGESTRING  . SELECT
MYEXCEL. CHARTS. ADD
结束 EXCEL
MYEXCEL. QUIT
EXCEL. ActiveWindow. SelectedSheets. PrintPreview &&预览打印
EXCEL. ActiveWorkbook. SaveAs  C :My Documentook1. xls  &&另存为
EXCEL. ActiveWorkbook. Close &&关闭一个工作表,如果有修改则提示
EXCEL. ActiveWorkbook. Close  . t.  &&提示另存为
EXCEL. ActiveWorkbook. Close  . f.  &&关闭一个工作表不用提示是否存盘
```

16. 12　利用 DDE 进行动态数据交换

在数据库应用程序中,经常需要与外部的数据源进行数据交换。通常我们会考虑利用开放式数据库互连标准 ODBC。ODBC 由驱动程序管理器和一系列把 SQL 作为它们访问语言的驱动程序所组成。当需要时,ODBC 驱动程序翻译各种产品的 SQL 语言以使不同产品间的连接完美无缺,从而使得连接另外一个应用程序上的数据成为可能,甚至可以在 Visual FoxPro 并不直接支持的格式(例如电子表格)中提取和修改数据。鉴于 ODBC 早已经被很多程序员所熟知,我们这里仅仅讨论一种不太常用的方法:DDE。

DDE 是一种动态数据交换机制 (Dynamic Data Exchange, DDE)。使用 DDE 通讯需要两个 Windows 应用程序,其中一个作为服务器处理信息,另外一个作为客户机从服务器获得信息。客户机应用程序向当前所激活的服务器应用程序发送一条消息请求信息,服务器应用程序根据该信息作出应答,从而实现两个程序之间的数据交换。在 Visual FoxPro 中,一共十一个常用的 DDE 函数,它们是:

DDEAbortTrans　删除异步的 DDE 处理

DDEAdvise　建立与服务器应用程序的温或热连接

DDEEnabled　设置或返回 DDE 状态

DDEExecute　向服务器应用程序发送一条执行消息

DDEInitiate　打开到服务器应用程序的 DDE 控制板

DDELastError　返回 DDE 函数引起的最后一条错误信息

DDEPoke　传送数据库到客户机或服务器应用程序

DDERequest　服务器应用程序请求数据

DDESetOption　修改或返回 DDE 设置

DDESetService　添加、删除、修改服务名的状态

DDESetTopic　连接服务名与标题名

DDETerminte　关闭 DDE 控制板

Visual FoxPro 既可以作为 DDE 客户机，也可以作为 DDE 服务器。当 Visual FoxPro 作为客户机时，一个典型的请求从其他应用程序输入数据的 FoxPro 程序由包含以下步骤的代码组成：

调用 DDEInitiate　函数建立与服务器应用程序的连接。

如果成功地建立了连接关系，调用 DDERequest　函数请求从服务器应用程序输入信息。可以重复调用 DDERequest　函数请求输入其他的信息。

完成对数据的请求后，调用 DDETerminate　函数技术与服务器应用程序的连接关系。只有这样做才能够释放系统资源。

当 Visual FoxPro 作为服务器时，一个典型的响应其他应用程序请求数据的 FoxPro 程序由包含以下步骤的代码组成：

调用 DDESetService　函数生成服务过程（建立服务过程名）并且指定服务过程的类型。

调用 DDESetTopic　函数生成服务标题并且指定客户机请求中设定标题时的运行过程。

生成 DDESetTopic　中指定的过程用以接受传递给过程的参量。客户机请求信息调用此过程，此过程执行请求操作并且向客户机应用程序返回所请求的信息。

应用程序之间的 DDE 会话可以使用冷连接、温连接和热连接。当 Visual FoxPro 作为客户机时，所建立的连接为冷连接；当 Visual FoxPro 作为服务器时，可以使用其中任意的一种连接方式。可以使用 DDEAdvise　函数来初始化 DDE 温连接和热连接的会话。传递给 DDEAdvise　函数的参数之一是一个用户自定义函数的名称，服务器应用程序调用该函数通知客户机数据已经改动。因此，在温连接和热连接会话中，当数据源有所改动时，客户机能够根据服务器的通知自动进行数据更新。

附录　Visual FoxPro6. 0 的相关附表

附表 1　VisualFoxPro6. 0 文件类型

扩展名	文件类型
. act	向导操作图的文档
. app	生成的应用程序或 ActiveDocument
. cdx	复合索引
. chm	编译的 HTMLHelp
. dbc	数据库
. dbf	表
. dbg	调试器配置
. dct	数据库备注
. dcx	数据库索引
. dep	相关文件 由“安装向导”创建
. dll	Windows 动态链接库
. err	编译错误
. esl	VisualFoxPro 支持的库
. exe	可执行程序
. fky	宏
. fll	FoxPro 动态链接库
. FMT	格式文件
扩展名	文件类型
. FPT	表备注
. FRT	报表备注
. frx	报表
. fxp	编译后的程序
. h	文件 VisualFoxPro 或 C / C + + 程序需要包含的
. hlp	WinHelp
. htm	HTML
. idx	索引,压缩索引
. lbt	标签备注
. lbx	标签
. log	代码范围日志
. lst	向导列表的文档

续 表

.mem	内存变量保存
.mnt	菜单备注
.mnx	菜单
.mpr	生成的菜单程序
.mpx	编译后的菜单程序
.ocx	ActiveX 控件
.pjt	项目备注
.pjx	项目
.prg	程序
.qpr	生成的查询程序
.qpx	编译后的查询程序
.sct	表单备注
.scx	表单
.tbk	备注备份
.txt	文本
.vct	可视类库备注
.vcx	可视类库
.win	窗口文件

附表 2　　VisualFoxPro6.0 的数据类型

数据类型	说明	示例
字符型 Character	字母、数字型文本	商品名称、编号
货币型 Currency	货币单位	价格
数值型 Numeric	整数或小数	重量
浮点型 Float	整数或小数 数值较大时	重量
日期型 Date	年,月,日	进货日期
日期时间型 Datetime	年,月,日,时,分,秒	员工上班的时间
双精度型 Double	双精度数值	要求高精度数据
整型 Integer	不带小数点的数值	数量
逻辑型 Logical	真或假	货物是否到
备注型 Memo	不定长的字母数字文本	备注文字
通用型 General	OLE 对象链接与嵌入	Excel 电子表格
字符型 二进制 Character Binary	字母、数字型文本	代码页更改时字符值不变
备注型 二进制 Memo Binary	不定长的字母数字文本	代码页更改时备注不变

附表 3　VisualFoxPro6.0 的设计器

设计器名称	设计器功能
表设计器 TableDesigner	创建表及其索引
数据库设计器 DatabaseDesigner	建立数据库和表的关系
数据库环境设计器 DataEnvironment	创建表单、报表等的数据环境
查询设计器 QueryDesigner	在本地表中创建查询
视图设计器 ViewDesigner	创建可更新的查询
表单设计器 FormDesigner	创建表单
报表设计器 RepertDesigner	建立用于打印数据的报表
标签设计器 LabelDesigner	建立用于打印数据的标签
类设计器 ClassDesigner	可视化地创建并修改类
菜单设计器 MenuDesigner	创建菜单和菜单项
连接设计器 ConnectionDesigner	为远程视图创建一个连接

注：通过设计器可以迅速地创建表、表单、类、报表、菜单、查询以及视图。

附表 4　VisualFoxPro6.0 的生成器

生成器名称	生成器功能
组合框生成器 ComboBoxBuilder	生成组合框
编辑框生成器 EditBoxBuilder	生成编辑框
列表框生成器 ListBoxBulider	生成列表框
文本框生成器 TextBoxBuilder	生成文本框
命令组生成器 CommandGroupBuilder	生成命令组
表格生成器 GridBuilder	生成表格
选项组生成器 OptionGroupBuilder	生成选项组
自动格式生成器 AutofprmatBuilder	格式化控件组
表单生成器 FormBuilder	生成表单
参照完整性生成器 ReferentialIntegrity	在数据库表间创建参照完整性

注：生成器用于简化创建和修改表单、复杂控件和参照完整性符号的程序。

附表 5　VisualFoxPro6.0 的向导

向导名称	向导功能
表向导 Table	根据典型的表结构创建表
表单向志 Form	由单个表创建用于输入数据的表单
一对多表单向导 One - To - MaryForm	由两个相关表创建一个表单

续 表

一对多报表向导 One - To - MaryReport	由两个相关表创建一个报表
报表向导 Report	由单个表创建经过格式编排的报表
标签向导 Label	为表生成一个标签
查询向导 Query	基于指定的规则创建选定记录的视图
本地视图向导 LocalView	使用本地数据创建视图
远程视图向导 RemodeView	使用远程数据创建视图
应用程序向导 Applicating	创建一个有各种功能的应用程序
文档向导 Documenting	将项目和程序中的代码生成文本文件
交叉表向导 Cross - Tab	以电子表格显示数据
安装向导 Setup	根据发布树中的文件创建发布磁盘
图形向导 Graph	在 Microsoftgraph 中生成一个图形,显示来自 visualfoxpro 中的数据
分组 / 总计向志	创建一个总结报表,可以对成组的记录分组汇总
导入向导 Import	从其它文件格式中输入数据到 Visualfoxpro 表中
Oracle 升迁向导 OracleUpsizing	建立 Oracle 服务器数据库
SqlServer 升迁向导 SqlServerUpsizi	建立 SQL 数据库
数据透视表向导 Pivaltable	生成一个数据透视表,数据透视表是使用户能根据已有的表汇总及分析数据的交互式工作图表,您可以将它保存在 Microsoft Excel 中,也可以作为对象添加到表单上。要创建数据透视表,计算机上必须带有 MicrosoftQuery 的 MicrosoftExcel
Wed 搜索页向导 WedPubishing	创建一个 Web 页,允许 Web 页的访问者从 VisualFoxPro 表中搜索和检索记录
邮件合并向导 MailMenge	为 MixrosoftWord 合并文档创建一个数据源或任何字处理器能使用的文本文件

注 :向导可以为您创建功能齐全的数据库、表、表单、报表、查询,而您无需编写任何代码。