HOPE

Delphi 3.0 从入门到精通

一本實用的深入 Delphi 技術書籍

李维 著
希望图书创作室 改编

宇航出版社

FLAG

Delphi 3.0 从入门到精通

李　维　著
希望图书创作室　改编
王　真　审订

宇航出版社

图书在版编目(CIP)数据

Delphi 3.0 从入门到精通/李维著；希望图书创作室改编. —北京：宇航出版社，1998.8

ISBN 7-80144-121-4

I. D… II. 李… III. 程序语言，Delphi 3.0-程序设计-基本知识 IV. TP312

中国版本图书馆 CIP 数据核字(98)第 09168 号

宇航出版社出版发行

北京市和平里滨河路 1 号(100013)

发行部地址：北京阜成路 8 号(100830)

北京科普印刷厂 印刷

新华书店经销

1998 年 8 月第 1 版　　1998 年 8 月第一次印刷

开本：787×1092 1/16　　印张：27　　字数：672 千字

印数：1—5000 册　　定价：40.00元

目　录

第 1 章　认识 Delphi

本章重点：

- Delphi 的项目结构
- Delphi 如何访问数据库
- Delphi 集成开发环境的使用
- Delphi 应用程序的执行流程

在这一章中，读者可以好好地认识一下 Delphi。如果是从 Delphi 1.x 版升级来的读者，或是已经使用过 Delphi 2.0 的用户可能会觉得奇怪，我们已经知道 Delphi 是什么了，为什么还需要花费一章的内容来介绍所谓的“认识 Delphi”呢？

这一章中的一些内容可能是读者已经知道的，但是也有许多是读者不太熟悉的内容。如果读者是第一次接触 Delphi，那么本章将会让你彻底认识 Delphi，为以后的学习过程打下一个坚实的基础。本章将要讨论下面的主题，这些主题在使用 Delphi 开发应用程序的过程中有着非常重要的地位。其中有一些主题与应用程序开发有着密切的关系，有一些是在使用 Delphi 本身开发一些好用的自动工具或是组件时必备的知识，而另外有一些是解释一个 Delphi 的应用程序的整体结构的流程。相信只要对这些主题有一个清楚的认识，就可以牢牢地掌握 Delphi 这个开发工具。

下面列出的就是本章要讨论的主题：

- Delphi 项目的组成元素
- Delphi 如何访问数据库
- Delphi 集成开发环境的使用
- Delphi 应用程序的执行流程

在下面的章节将要逐一讨论这些主题。

1.1　Delphi 项目的组成元素

当使用 Delphi 开发应用程序时，Delphi 会产生许多不同的文件。而这些文件都有着不同的意义。身为一个 Delphi 的开发者，当然需要了解每一个文件所代表的意义，以及每一个文件所负责的工作。当 Delphi 项目出了一些意外情况时，只有对每一个文件有了基本的认识，才有机会修复它。

通常一个 Delphi 的项目会有下列表格中所列出的不同文件：

*.DPR	Delphi 的项目文件，这个文件包含了 Delphi 的程序进入点程序代码，并且指明这个项目中有哪一些窗体或者程序单元在那一个文件中
*.DFM	Delphi 每一个窗体保存的文件。这个文件指明了每一个窗体中有哪些组件，以及每一个组件的属性值。例如，组件的位置、名称、使用的字形等
*.PAS	这个文件是每一个程序单元的原始 Object Pascal 文件
*.RES	这个文件是 Delphi 项目所使用的资源文件。在这个资源文件中指明了这个项目所使用的位图(Bitmap)、图标(Icon)、字符串以及版本资源等
*.DOF	这个文件指明了该 Delphi 项目所使用的各种编译器的开关。例如，该项目使用的堆栈(Stack)大小、堆积(Heap)大小、编译器最优化的设置以及链接器的设置等
*.～??	该文件是其他文件的备份

接下来，让我们详细讨论一下每一种文件的意义。

1.1.1　*.DPR 的文件

*.DPR 这种文件是一个 Delphi 项目的主轴。在该文件中指明了这个项目所使用的每一个文件，同时也指明了每一个文件所在的目录。例如，下面的程序代码是 Delphi 的一个范例 BKQUERY 的项目文件，在这个文件中指明了这个项目使用了三个.PAS 的文件，而且这些.PAS 的文件是位于和这个项目相同的目录下。

```
program BkQuery;

uses
  Forms,
  QueryFrm in 'QueryFrm.pas' {AdhocForm},
  ResltFrm in 'ResltFrm.pas' {QueryForm},
  SaveQAs in 'SaveQAs.pas' {SaveQueryAs};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TAdhocForm, AdhocForm);
  Application.Run;
end.
```

← 这个项目使用了三个其他的文件，它们分别在上述的三个.PAS 的文件中，而且这些.PAS 的文件是位于和这些项目相同的目录下

如果在使用如图 1.1 的项目管理器加入一个其他目录的窗体或是程序单元时，

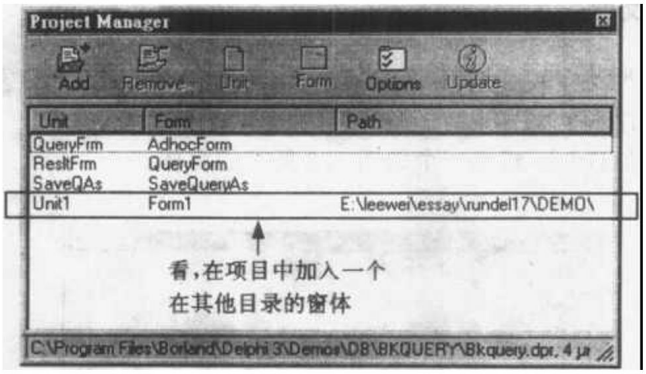

图 1.1　使用 Delphi 的项目管理器加入其他的窗体使用

那么，这个文件就会在上述的 uses 句子中变成如下的程序代码。

```
uses
  Forms,
  QueryFrm in 'QueryFrm.pas' {AdhocForm},
  ResltFrm in 'ResltFrm.pas' {QueryForm},
  SaveQAs in 'SaveQAs.pas' {SaveQueryAs},
  Unit1 in 'E:\leewei\essay\rundel17\DEMO\Unit1.pas' {Form1};
```

指明使用的文件在磁盘的位置

如果移动了这些在其他目录的文件时，例如把这些文件复制到该项目的目录之下，那么你必须修改这个项目文件的该行程序代码，或者使用 Delphi 的项目管理器先把这个文件删除，再从它的新目录中加入，否则在装入这个文件时会看到如图 1.2 所示的错误。

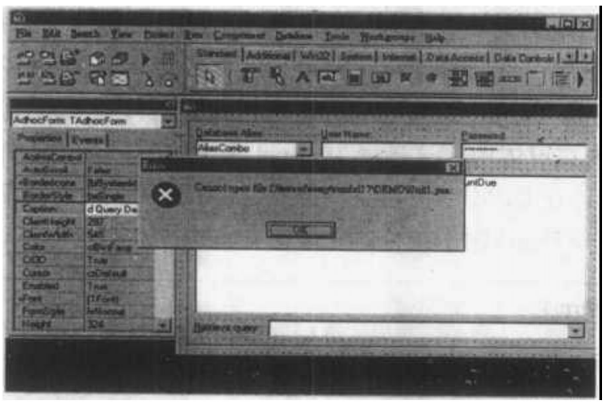

图 1.2　改变项目使用的文件目录会产生错误

现在，应该对这个.DPR 文件有了一定的了解。由于.DPR 文件不是一个简单的文本文件，所以如果该文件毁坏或是不小心被删除了，你是无法使用一般的文本编辑器编辑该文件的。不过，可以直接在 Delphi 中建立一个新的项目，再加入所有原先项目使用的窗体和程序单元。最后记住把这个新项目的主窗体删除，再设置原项目的主窗体即可。

1.1.2　*.DFM 的文件

该文件记录了 Delphi 每一个窗体的信息。当在 Delphi 中建立一个新的窗体并且保存时，Delphi 就会把这个窗体保存在一个.DFM 文件中。

例如，现在如果在 Delphi 建立一个如图 1.3 所示的窗体。

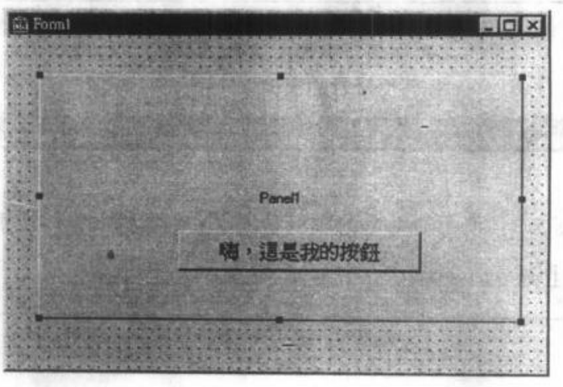

图 1.3　我们建立的新窗体

在这个窗体上有一个 Panel 组件以及一个按钮组件。当保存该窗体时，Delphi 会保存一个如下的.DFM 文件。从下面的程序代码中看到，在保存窗体组件、Panel 组件和按钮组件时，都会保存这些组件的一些属性值。而这些被保存的属性值，必须是那些与该组件设置的缺省属性质不同的属性值以及一些必要的属性值。例如，按钮组件保存了 Top，Left，Width 和 Height 属性值，是因为每一个组件在窗体中的位置不一样，所以 Delphi 必须保存这些属性值，这样才能在下次装入这个窗体时，可以正确地把每一个组件放到特定的位置。而 Caption 这个属性值也会被保存，因为每一个组件都有不同的 Caption 属性值。但是按钮的 Style 属性值就没有保存，因为我们并没有改变按钮的 Style 属性值。如果你改变了按钮的 Style 属性值，那么该属性值也会被保存。因为它已经和缺省的属性值不一样了。

另外，必须注意的是，每一个组件在保存时会内嵌在另外一个组件中。例如，按钮内嵌在 Panel 中，而 Panel 内嵌在窗体中。这是因为 Panel 是窗体的一个子组件，在设计该窗体时把按钮放在 Panel 中，所以在 DFM 中按钮是内嵌在 Panel 中的。这个代表 Panel 是按钮的父代(Parent)组件，而按钮是 Panel 的子组件。

```
object Form1: TForm1
  Left = 168
  Top = 105
  Width = 435
  Height = 300
  Caption = 'Form1'
  Font.Color = clWindowText
  Font.Height = -12
```

```
  Font.Name = 'Times New Roman'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 15
  object Panel1: TPanel          ←这是 Panel 组件
    Left = 8
    Top = 16
    Width = 401
    Height = 161
    Caption = 'Panel1'                    这是按钮组件。请注意,由于这个
    TabOrder = 0                          按钮是位于窗体的 Panel 组件中,
    object BitBtn1: TBitBtn             ←所以它的定义是内嵌在 Panel 中
      Left = 112
      Top = 104
      Width = 201
      Height = 33
      Caption = '嘻,这是我的按钮'
      Font.Charset = CHINESEBIG5_CHARSET
      Font.Color = clWindowText
      Font.Height = -11
      Font.Name = 'System'
      Font.Style = [fsBold]
      ParentFont = False
      TabOrder = 0
    end
  end
end
```

虽然.DFM 文件不是一个纯文本文件,但是我们可以在 Delphi 中使用编辑器直接修改它。如果这样做的话,你必须对可能发生的后果负责。现在让我们做一个小实验:把上面的程序代码中按钮的定义移到 Panel 组件外,让它与 Panel 组件并列,并且修改这两个组件的 Top 和 Left 属性值。

```
object Form1: TForm1
  object Form1: TPanel
    Left = 8
    Top = 50
    Width = 401          ←把 Panel 组件往下移
    Height = 161
    Caption = 'Panel1'
    TabOrder = 0
    end
    object BitBtn1: TBitBtn
      Left = 0
      Top = 0            ←把按钮移到窗体的左上角
      Width = 201
      Height = 33
      Caption = '嗨,这是我的按钮'
      TabOrder = 0
      Font.Charset = CHINESEBIG5_CHARSET
      Font.Color = clWindowText
      Font.Height = -11
      Font.Name = 'System'
      Font.Style = [fsBold]
      ParentFont = False
      end
end
```

←请注意,此时 Panel 组件和按钮组件已经是并列的了,都是窗体的子组件

把这个.DFM 文件保存起来。当 Delphi 再装入这个项目之后,主窗体就会变成图 1.4 的样子。

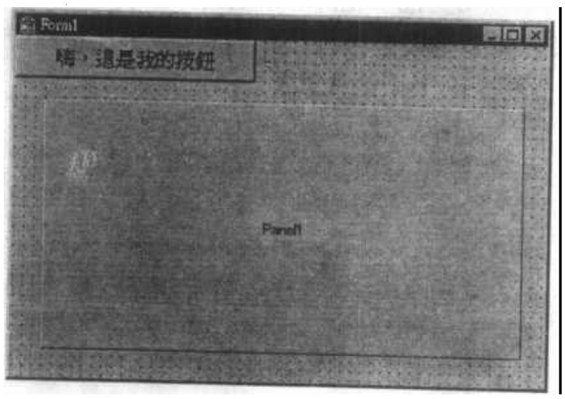

图 1.4　新的窗体以及它的子组件

从上面的图形中可以看到,在.DFM 文件中改变组件的属性值,以及改变这些组件内嵌

关系时，这些改变在项目重新装入时都会体现出来。

事实上，当在 Delphi 的集成开发环境中设计一个窗体并且在窗体上放入各种组件后，保存该窗体时，Delphi 是调用 WriteComponentResFile 这个程序把设计的窗体以. DFM 的文件格式保存在磁盘中。当 Delphi 打开项目中的. DFM 文件时，是调用 ReadComponentResFile 这个程序把窗体显示在 Delphi 的集成开发环境中。

1.1.3　*. PAS 的文件

该文件中保存的是 Object Pascal 的程序代码。不管这个 Object Pascal 程序单元是和一个窗体(. DFM)链接在一起的文件，还是一个独立的 Object Pascal 程序单元，都保存在这种文件中。这种文件只是一般的文本文件，你可以使用一般的文本编辑器来编辑它。但是如果这样做的话，那么必须小心的是，当这个文件是和一个. DFM 文件链接的 Object Pascal 程序单元时，不要修改 Delphi 自动帮我们在这个 Object Pascal 程序单元中加入的程序代码。因为这些自动加入的程序代码是和. DFM 中的对象有关系的，因此如果改变了它们，那么当 Delphi 再装入这两个链接的文件时会产生错误。例如，在 Object Pascal 的文件加入了一个在窗体中没有的组件时，产生了如图 1.5 所示的错误。删除 Object Pascal 中的一个组件时，产生了如图 1.6 所示的错误。

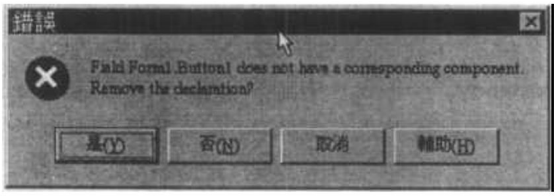

图 1.5　加入组件后产生的错误

图 1.6　删除组件后产生的错误

1.1.4　*. RES 的文件

该文件是 Windows 的资源文件。它包括了应用程序所使用的一些窗口资源。例如，图像、图形和字符串等资源。在 Windows 中，. RES 的文件是资源已经被编译过的格式。通常在 Windows 中，我们必须使用一些所谓的资源编辑器来设计应用程序所需要的资源。设计完之后，这些资源编辑器会把我们使用的资源以 *. RC 文件保存起来。然后 *. RC 的文件经过资源编译器编译成. RES 文件，最后由链接器把这个. RES 的资源文件和应用程序一起编译成. EXE 的可执行文件。

例如，Delphi 中的 Image Editor 虽然听起来似乎只是一个编辑图像的编辑器。但是它事实上是一个资源编辑器，只是它直接产生一个. RES 文件而不产生. RC 文件。而在这个. RES文件中，可以加入许多的资源(如图像、图形和光标等)。例如，图 1.7 就是使用 image editor 编辑的一些资源。

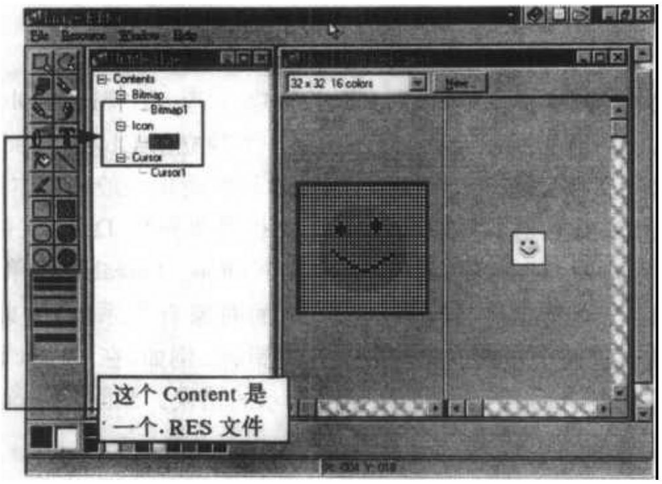

图 1.7　使用 image editor 编辑资源并且产生. RES 的文件

图 1.8 显示的是 Borland C++ 5.0 中的资源编辑器，可以看到它产生了其他各种的资源。

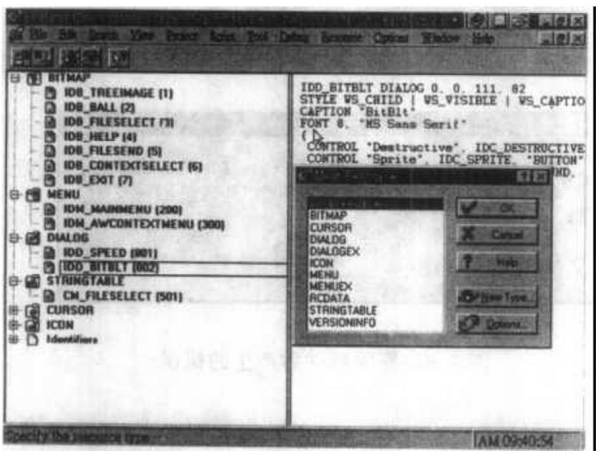

图 1.8　Borland C++的资源编辑器

在使用 Delphi 开发应用程序时，Delphi 会自动帮我们产生一个和项目同名的资源文件，并且在应用程序的主程序处使用这个资源文件。例如，在下面的程序代码中，请注意 Delphi 使用编译器指令 $R *. RES 资源文件。

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.RES}          ←使用资源文件
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

由于 Delphi 内置的 Image Editor 并没有编辑字符串资源的功能，你必须自己使用 Delphi 的命令栏工具把字符串资源链接到 EXE 文件中。据我所知，台湾 Borland 公司准备在 Delphi 3.0 的中文组件中附送一个可以处理 DBCS 的 Borland C++资源编辑器，让用户可以使用这个资源编辑器编辑任何资源，以弥补 Delphi Image Editor 的不足。

1.1.5　*.DOF 的文件

*.DOF 文件是 Delphi 产生的编译器以及链接器开关的文件，通常情况下这个文件会自动产生。如果有兴趣的话，可以看看这个文件的内容。由于这个文件是选择性的，所以它存不存在并没有太大的关系，可以删除这个文件。如果这样做的话，那么 Delphi 在编译以及链接应用程序时会使用缺省的设置。如果改变了一个 Delphi 应用程序的任何编译器和链接器的设置时，Delphi 会把这些信息保存在.DOF 文件中。如果 Delphi 发现没有.DOF 文件，它会自动产生一个.DOF 文件。

1.1.6　*.~?? 的文件

*.~??文件是当在 Delphi 中保存项目或是任何文件时，Delphi 会把前一版本的文件保存成这种文件。例如，如果改变了一个程序单元 XXX.PAS 的文件，保存该文件时，Delphi 会把原先 XXX.PAS 文件保存成 XXX.~PA 文件，再把新的内容保存成 XXX.PAS 文件。所以如果发生了任何严重的错误，可以把 *.~?? 文件复制回来。当然，也可以在有把握的情况下删除所有 *.~?? 文件以节省磁盘空间。

以上讨论了组成一个 Delphi 项目的所有文件元素，希望在阅读完了之后对这些文件有更深一层的认识。在使用 Delphi 开发应用程序时，应注意保护重要的文件，并且在发生错误时能够减少损失。

1.2　Delphi 如何访问数据库

Delphi 是一个非常优秀的 RAD 工具，也是一个非常好的客户/服务器数据库开发工具。通常，Delphi 是使用 Borland 的 BDE/IDAPI 这个数据库引擎来访问数据库的。这个数据库引擎再配合 Borland 的 SQL Links 或是 ODBC 来访问后台的数据库。可以使用图 1.9 来表

示这个结构。

从图 1.9 中可以看出，当 Delphi 访问 xBase 的数据库时，是直接由 BDE/IDAPI 处理的，而不再使用任何其他的文件。但是对于其他的数据库，BDE/IDAPI 必须再通过 SQL Links 或是 ODBC 才能够访问它们。

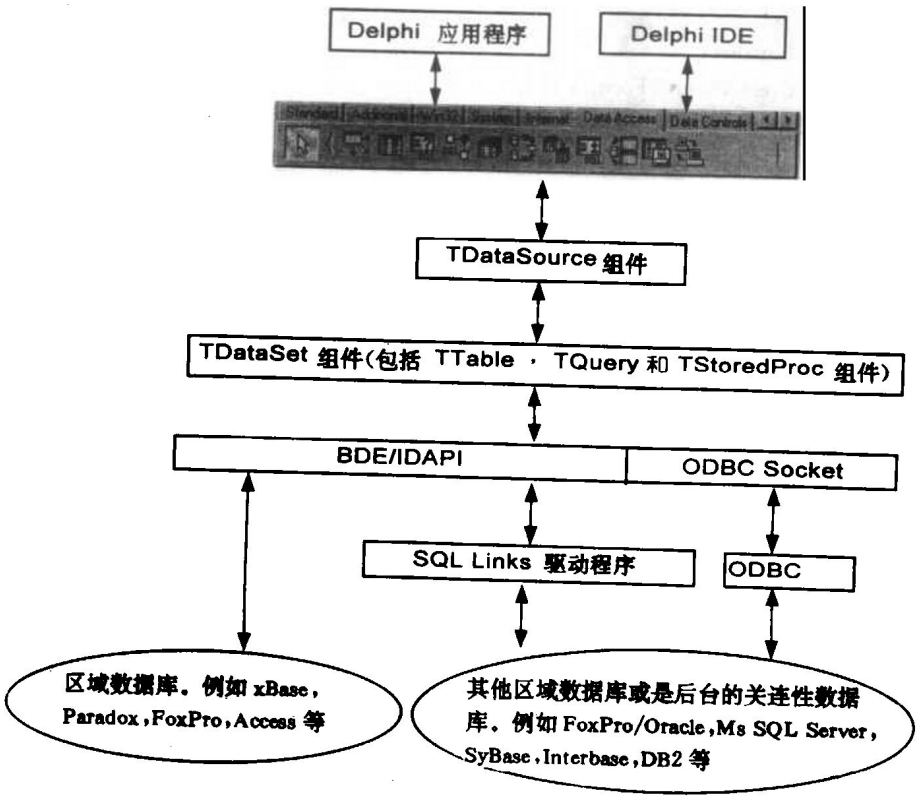

图 1.9

事实上，上面的图形是 Delphi 如何访问数据库的简化概述。用户可以详细地看看 Delphi 访问数据库的过程。一个 Delphi 的应用程序在使用数据库的功能时，一定要在程序单元的 uses 句子中加入 DB. PAS 这个程序单元。因为这个程序单元只装入初始化 BDE/IDAPI 的进入点。此外，应用程序访问数据库之前，必须使用 BDE Configuration 这个应用程序来设置数据库的别名。事实上，这个应用程序在输入数据库的别名后，这些信息被保存在一个. CFG 的文件中。所以一个 Delphi 或是 Delphi 的应用程序使用如图 1.10 所示的结构来访问数据库。

从这个图形中，可以了解 Delphi 以及它的应用程序如何访问数据库了。由于 SQL Links 是一个本地(native)的数据库访问驱动程序，所以在 Delphi 3.0 中都有指明，通过 SQL Links 可以访问哪一个数据库的哪一个版本。例如，Delphi 3.0 可以访问 MS SQL Server 6.0 ～6.5.x 的数据库，并且修改了许多 BDE/IDAPI 3.5.x 的错误。限制访问数据库的版本是使用本地驱动程序的缺点，不过使用本地驱动程序可以使应用程序有着良好的数据访问效率。但是，一旦使用的数据库版本和本机驱动程序不一样时，只能期望 Borland 能够尽快地