Delphi5 深入编程

张子瑜　郑宏兴　编著

东南大学出版社

• 南 京 •

内容提要

本书介绍 Delphi5 的新功能及其开发实用的 Windows 程序的方法，让读者快速领会 Delphi5 的新功能，以及掌握使用 Delphi5 开发的实用技术等等。

全书分为 8 章，主要包括 Delphi5 新增功能及其编程实例、多媒体开发技术、多线程、COM、CORBA 与 MIDAS 技术编程、Web 编程技术与分布式应用、数据库编程、MTS、ADO、网络数据库编程等内容。

本书适合于软件开发人员和 Delphi5 学习者使用，也可作为高等院校广大师生和科研人员进行软件开发的参考书。

图书在版编目（CIP）数据

Dilphi5 深入编程/张子瑜，郑宏兴编著. —南京：东南大学出版社, 1999.8(2000.7 重印)
ISBN 7-81050-510-6

I. D… II. ①张…②郑… III. Delphi 语言-程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2000）第 28507 号

东南大学出版社出版发行
（南京四牌楼 2 号 邮编 210096）
出版人：宋增民
江苏省新华书店经销 江浦第二印刷厂印刷
开本：787mm×1092mm 1/16 印张：19 字数：510 千字
1999 年 8 月第 1 版 2000 年 7 月第 2 次印刷
定价：30.00 元

前　言

Delphi 软件开发工具是 Borland 公司推出的一个完全导向的可视化系统开发环境的工具，被评为美国最优秀的软件之一。Delphi 结合了可视化技术、面向对象技术、数据库技术、网络开发技术等多种先进的软件编程技术和思想，成为创建功能丰富、界面友好的 Windows 应用软件的工具之一。

Delphi5 是 Inprise 公司 1999 年 8 月推出的可视化 Windows 应用程序开发工具，它是对 Delphi4 的扩展和完善，加入了许多新的编程方法和工具。Delphi5 加强了数据库与网络程序开发的功能，如 InterBase 数据库编程、基于 Web 的多层分布式应用编程等，并增加和完善了许多新技术，如 CORBA、DHTML 等。

我们在使用 Delphi5 应用开发经验的基础上，结合一些最新资料和大量开发实例，深入浅出地介绍 Delphi5 实用开发技术。本书注重编程实用性和技巧性，通过大量的编程实例，循序渐进地介绍了从 Delphi5 基本程序编写到图形图像数据库及多媒体和网络应用程序开发，使读者能逐步地掌握 Delphi5 编程技巧及其实用开发技术。书中的许多例子都是从我们开发的程序和其他实际应用程序中部分摘选来的，因而具有较强的实用性。

本书特别适用于广大计算机软件开发人员和计算机爱好者学习和使用。

在本书即将出版之际，我们要感谢东南大学出版社张煦同志的大力支持和帮助。由于时间和水平有限，不妥之处在所难免，敬请广大读者斧正，不胜感激。

编　者

2000 年 2 月

第 1 章　Delphi5 新增功能

Delphi 软件开发工具是一个完全导向的可视化系统开发工具，具有功能强大、运行速度快、易于使用以及开发速度快等特点，一经推出就受到广大用户的喜爱，被评为美国最优秀软件之一。Delphi 结合了可视化技术、面向对象编程、数据库和分布式应用技术等多种先进的软件编程技术和思想，并使用了全特征的代码编辑器，成为创建功能丰富、界面友好的 Windows 应用软件工具之一。

Delphi5 是 Inprise 公司于 1999 年 8 月推出的可视化 Windows 应用程序开发工具，它是对 Delphi4 的扩展与完善，加入了许多新的编程方法和工具，如 InterBas 数据库编程，基于 Web 的多层分布式应用编程以及 CORBA、DHTML 等新技术。Delphi5 以其优良、稳定的性能，强大的数据库支持，更快的开发速度以及全面支持分布式应用等，成为万众瞩目的焦点。

Delphi5 简化了 Windows 应用程序和浏览器、Web 服务器、中间件以及后端大型数据库系统的集成，并支持 XML 和 HTML4，加快了 Internet 和分布式计算应用系统的设计开发速度，一经推出就被《InfoWorld》杂志评为最佳 Web 及 Windows 应用程序开发工具。

Delphi5 在开发 Internet 应用、数据库系统等方面的性能有了很大提高，同时在提高开发人员的效率、方便开发人员的使用方面也有了很多改进。

1.1　Delphi5 新增功能

下面介绍 Delphi5 的新增功能。

1) Object Pascal 语言扩展

Delphi5 扩展了 Object Pascal 语言的一些功能，包括动态数组定义、方法重载和缺省参数设置等等。

2) 工程管理功能

Delphi5 提供了一个新的工程管理器。该工程管理器允许开发人员将多个工程组合在一个工程组中进行处理，这种功能就可以帮助开发人员组织和操作多个相互关联的工程(如多列应用的单列处理，或 DLLs，以及可执行文件等)一起工作。

3) 类实现代码探测器、模块导向和代码浏览器

代码探测器(Code Explorer)使创建类变得非常容易，其通过多个步骤自动创建一个或多个类，并在接口部分定义方法原型和在实现部分创建类实现框架代码。导向功能通过单元文件可以在接口和实现部分之间实现快速定位。使用 Tooltip 符号暗含功能浏览任何标识符的说明信息，然后使用代码浏览器跳到其说明位置。

4) 可连接的工具窗

Delphi5 提供的 IDE 是一个多配置的工具窗，它可以连接成代码编辑器。用户可以将工具窗拖放到任何的地方。代码探测器和工程管理器也可以进行连接与分解。

5) 改进的调试器

Delphi5 提供的集成调试器具有许多新特征，包括远程和多处理器调试、CPU 浏览与监视、断点加强、调试器子菜单以及可分离窗等。Delphi5 的代码调试器包含以下一些新特点：

- 各种调试窗口和编辑器之间紧密集成，拥有一致的用户界面，支持剪贴板和拖动操作；
- 增加了断点提示条(Breakpoint ToolTips)，以加快浏览和定位源代码；
- 新增的断点动作(Breakpoint Actions)功能，加强了对调试进程的控制；
- 断点集合(Breakpoint Groups)方便了对多条断点的控制；
- 在 DCUs 和调试符号查找路径的帮助下，调试工作变得更简单；
- FPU 调试窗口支持 MMX 指令，从而增强了底层调试的能力；
- 进行多进程调试时，增加了临时进程选项，并支持调试子进程；
- 支持跨越进程边界的调试；
- 能够挂接到正在运行的进程上进行调用。

6) MTS 支持

Delphi5 完全支持 MTS。MTS 支持多数据库应用。另外，一个新的 Wizard 将使创建 MTS 服务器对象变得更容易。

7) 增强的 ActiveX

Delphi5 提供扩展的 ActiveX 和 ActiveForm 支持。

8) 增强 VCL 功能

Delphi5 加强了 Delphi 的对象层次功能，包括面向 NT 服务应用的新组件。另外，新的列表组件(在 Standard 组件板页)允许开发人员对菜单和工具条按钮命令进行居中操作。VCL 控件加上了支持拖放功能，并提供对窗体布局的附加控制等等。在 Delphi5 中，用户可以利用面向对象设计的强大功能开发出稳定、可靠、高效的程序，可以利用现有的面向对象的控件创建自己的控件。

Delphi5 企业版的 VCL 中一共包含有 200 多个控件，利用这些控件，用户可以奇迹般地加快应用程序从开始开发到推向市场的速度。无论是开发 Windows 程序还是 Internet 应用，Delphi5 基于控件的开发模式都能大大降低开发任务的难度。

9) RTL 支持 2000 年表示

TwoDigitYearCenturyWindow 变量用于 StrToDate 和 StrToDateTime 函数中控制在进行日期转换时的两位数字的解释，可以避免 2000 年问题。

10) 支持 CORBA

Delphi5 的客户机与服务器(Client/Server)和 Enterprise 版本的均包含对 CORBA 客户机和服务器应用的支持。Delphi5 提供的导向使创建一个 CORBA 服务器变得非常容易，而且动态请求接口(Dynamic Invocation Interface，简称 DII)允许开发人员为存在的 CORBA 服务器写客户机。CORBA 也内建在多数据库支持中，用户可以建立同时处理 COM 和 CORBA 客户机的服务器。

11) 增强客户机数据集

Delphi5 客户机数据集支持过滤表达式变量和维护总量，以及允许面向对象的域类型。

12) MIDAS3 支持

Delphi5 提供了许多对多层应用的控制，包括 refresh/resync 支持和对数据包中包含

的数据和更新如何应用等方面的控制。一个新类 TDataSetProvider 允许用户使用 nested 表从一个支持 Master/detail 的数据集中取得数据或发送数据到它们之中。另外，用户数据集的增强功能使参数传递到应用服务器或保存用户信息到数据包中更为方便。

利用 Delphi5 中的 MIDAS 3(Multi－tier Distributed Application Services Suite)，用户可以创建高效灵活的 Delphi 程序，以满足日益增长的事务量和用户数的需要。MIDAS 3 支持所有的分布式计算标准，如 CORBA、COM 和 MTS 等，可以实现现有系统和电子商务程序之间的无缝集成。它的特点表现为：没有界限的数据请求在移动或窄带的环境中可实现更多控制；WebConnection 组件能够在保证安全性的前提下帮助用户把程序移到防火墙外；服务对象库增强了扩展性。

13) 增强数据库功能

Delphi5 的数据连接组件允许开发人员调整数据模型，以便建立数据模块或窗体。修改的数据连接组件和 Borland Database Engine(BDE)允许用户使用新的数据服务器类型，包括 Access'97 和 Oracle8 到 SQL 的新扩展，其中包括 ADTs (Abstract Data Types，抽象数据类型)、数据、索引和 nested 表。全新的 InterBase Express 数据库组件使得 InterBase 数据库连接和服务器的事务控制更方便快捷。可视化 Query Builder 已被 SQL Builder 替代，这是一个新的智能查询创建器。

14) 强大的 COM 技术支持

Delphi5 的 DAX 技术有力地将 COM 技术容入任何的网络开发中，使其更方便地支持 MTS、ActiveX、MIDAS 和自动化对象与服务器开发。

15) 支持 XML

Delphi5 提供了对 XML 的支持，从而简化了数据分布，优化了数据交换。用户可以方便地创建高效率的 Internet 程序，将数据迅速发布到 Internet 上。新版 Delphi 对 XML 数据的支持使得开发人员能够快速建立起具有可移植性和扩展性的系统，以迎接 Internet 正面临着的又一次冲击波——电子商务。

16) 支持 HTML4

Delphi5 支持 HTML4，使用户能在 Web 应用中创建各种类型的动态瘦客户端程序。把 HTML4 和 XML 结合起来使用，从而创建动态的极瘦客户端程序，用来满足 Internet 上各种任务的需求。

17) 支持 ADO

ADO(ActiveX Data Objects)是微软提供的一项技术。通过 ADO，可以方便地访问各种类型的数据库，特别是 OleDB 数据库。ADO 已成为访问数据库的新的标准接口。Delphi 5 增添了对 ADO 的支持，是为了让用户能迅速实现对终端用户用来做商业决策的数据的一致性访问，结合 Delphi 本身的开放式数据元件结构，用户可以很快地建立应用程序，用来把自己的商业数据通过 Internet 发送给客户、最终用户以及整个销售环节。通过 ADO，Delphi 5 能让用户快速访问关系型或非关系型数据库以及 E－Mail 和文件系统。

18) InterBase Express

利用 InterBase Express，系统集成商和独立软件商可以快速开发高性能的应用系统。把 Delphi5 和 InterBase Express 结合起来使用，便可以在只需少量维护工作且只占用很少系统资源的数据库上进行高效率、高性能的开发。有了 InterBase Express 后，用户不需要使用 BDE(Borland Database Engine)便可以创建一个优秀的嵌入式系统。

19) TeamSource

Delphi5 集成了一个与微软的 SourceSafe 功能相似的源代码管理工具——TeamSource，这样，工作组一同进行的开发工作会合作得更好更快，工作更有效率。TeamSource 建立在现有的源代码管理工具和版本引擎的基础上，并把它们扩展成一种工作流程的模式，从而简化了工作组开发的源代码管理工作。

20) Borland Translation Suite

这是 Delphi5 包含的一个国际化和本地化工具，它能帮助用户同时开发多语言版本，管理和维护各种资源的翻译代码，从而加速产品的国际化开发步伐。现在利用 Delphi 5 开发的系统可以在世界范围内应用，帮助用户开拓全新的世界范围内的市场。

21) 改善的 IDE

Delphi5 中的集成开发环境(IDE)有了很大改进，能极大地提高开发效率，它主要依靠简化读写和浏览代码的操作来提高开发效率。代码编辑器让浏览本单元或相关单元的内容变得容易，工程管理器中的拖放支持使得从已打开的工程中或资源管理器的对话框中选择文件并增加到工程的文件中变得更简单，使用工程管理器中多个工程管理的功能让用户能同时编辑多个工程的文件，应用 Delphi5 的可视化窗口设计让用户能够轻松地从模板中选择创建 Internet 程序、分布式计算及 Windows 程序等等。

与 Delphi 4 相比，Delphi 5 的 IDE 新特点主要包括：桌面的用户化设置增强了用户对开发环境的控制；编辑键映射功能可以让编辑器按照用户的习惯来工作；工程浏览可以帮助用户更好地理解代码，操作 VCL；带有树视图和数据图表视图(Data Diagram View)的 DataModule 设计器可以帮助用户充分理解程序中的数据；资源文件与工程管理器和编译器集成在一起；动作列表保存了开发计划；控制面板向导可以帮助用户定制应用程序的属性。

1.2 Delphi5 新问题

下面是更新到 Delphi5 将遇到的兼容问题：

• Delphi5 包的扩展名被改动，由原来的.DPL 替换为.BPL。

• 如果已经创建了自己的设计包，Delphi5 要求用户在安装到 Delphi5 之前必须重建该包。为了能够识别其他包的制造者(如 C++Builder3、Delphi5 等)，内部包格式也做了修改。

• C++Builder4 的.BPL 文件不能在 Delphi5 上使用。如果用户购买了第三方的 Delphi 组件，那么用户必须使用它们提供的源代码。

Delphi5 不能打开 Delphi4 的 DPK 文件，而是自动转换成 Delphi5 包新的源格式。

• 改变所有 SYSDATE 实例到 GetDate。不能使用 Year、Month、Day、Hour、Minute 或 Second 提取时间、日期等。

• OnDragOver 和 OnDragDrop 事件原参数被修改。在 Delphi4 中，OnDragOver 和 OnDragDrop 事件可以使用 TDragControlObject 的源参数执行，不需要知道其内部拖动控制。在 Delphi5 中，事件源参数实际上控制拖动操作的开始。

• DWORD、UINT、HResult、OLE_HANDLE 和 API 处理类型被改变。DWORD、UINT、HResult、OLE_HANDLE 和所有的 API 处理类型被定义为无符号的 2 位 LongWord，它们不再与 Integer 类型兼容。

• 新的 Int64/LongWord 类型。为了进行算术运算，整型结果导致转到 Int64，避免溢

出。因此，某些不适合于 Integer 类型的变量将处理成 Int64(64 位 Integer)/LongWord (无符号 32 为 Integer)类型。详细请看帮助信息。

- Delphi5 提供了 NetManage 的 Internet 组件。
- 与 Oracle8 兼容。在 SELECT 陈述中，关键字 OR UPDATE 会锁住结果集，用户需要将 QLPASSTHRU MODE 设为 SHARED NOAUTOCOMMIT。

第 2 章 线程

本章介绍多线程的概念和编程，重点介绍 Delphi5 的线程对象及其线程在图形和数据库处理中的应用。

2.1 线程原理

线程概念是为了解决多任务和多进程的内存分配以及协调并行的工作而提出来的。我们知道，Windows 可以完成多任务的工作，比如，在前台进行文字处理，在后台可以进行打印、发送传真等任务。多线程是某一个任务同时处理几个工作，比如，同时对三个数据库进行查询等。Delphi5 提供了一个多线程对象，可以允许用户编写多线程处理程序。

TThread 就是一个能够在应用程序中创建互不相干的执行线程的类。每一个 TThread 对象的子类的新实例就是一个新的执行线程。从 TThread 类派生的多个实例就可以是 Delphi 的应用程序完成多个线程的工作。要在应用程序中使用线程，必须从 TThread 中派生一个新类，然后重载其相关的方法。

当一个应用程序运行时，程序就会装载线程到内存中并执行。从这一点来看，程序运行就变成包含一个或多个线程的过程，这些线程包含了运行这个程序的数据、代码和其他系统资源。一个线程执行程序的一部分，并通过操作系统分配 CPU 的时间。这个过程中的所有线程共享同一个地址空间，并使用全局变量处理这个过程。

使用线程可以改进程序以下几个方面的性能：

- 为多个通信设备管理数据的输入。
- 区分多个任务的优先权变化。

例如，一个优先权高的线程可以处理主要的任务，优先权低的则处理其他任务。

下面给出一些使用线程时要注意的问题和建议。

- 跟踪多线程比较耗时，建议在一个单处理器系统中，每一个进程不能超过 16 个线程。
- 在多个线程更新同一个资源时，要保持线程同步以避免冲突。
- 处理一个 VCL 线程和更新一个窗体的方法必须在主 VCL 线程内请求。

创建和使用一个新线程的步骤如下：

- 选择 File | New | Thread 对象，然后单击，系统创建了一个新单元。这个新单元就包含了一个从 TThread 类派生的新的线程对象。
- 定义这个新线程对象的 Create 方法。
- 通过插入执行代码，定义这个新线程对象的 Execute 方法。
- 传递请求使用 VCL 组件的 Synchronize 方法，该方法能够引发主 VCL 线程执行这个请求，从而避免多线程冲突。

例如，使用上述的方法创建一个 DDD 线程对象的单元如下：

```
unit Unit2;
interface
uses   Classes;
type
    ddd = class(TThread)
 private
     { Private declarations }
 protected
     procedure Execute; override;
 end;
 implementation
```

注意：在 VCL 中对象的方法和属性只有使用 Synchronize 调用以后才能启动线程同步执行，例如：

```
Synchronize(UpdateCaption);
```

UpdateCaption 过程与执行过程：

```
procedure ddd.UpdateCaption;
begin
    Form1.Caption := 'Updated in a thread';
end;
procedure ddd.Execute;
begin
        { 这里放置线程代码}
end;
```

2.2　线程属性、方法和事件

下面是线程的重要属性、方法和事件。

- FreeOnTerminate 属性决定当线程终止时线程对象是否自动释放。其语法说明如下：

```
property FreeOnTerminate: Boolean;
```

缺省情况下，FreeOnTerminate 为 False，即在应用程序中，线程对象必须通过代码释放。

- Handle 属性包含了线程的句柄。其说明如下：

```
property Handle: THandle;
```

当线程控制请求 Win32 API 函数时，应使用 Handle 属性。

- Priority 属性决定线程之间的事先规定的优先权。其语法说明如下：

```
type
    TThreadPriority = (tpIdle, tpLowest, tpLower, tpNormal, tpHigher,
            tpHighest, tpTimeCritical);
    property Priority: TThreadPriority;
```

Priority 属性是一个枚举类型，其缺省为 tpNormal，用于调整线程需要的优先权。TThreadPriority 类型定义了 Tthread 对象的 Priority 属性的可能值。Windows 为每一个线程按照优先权安排 CPU 周期，Priority 属性调整一个线程的优先权因子。Priority 属性的可选项有：

tpIdle：当系统空闲时执行线程。Windows 不能中断其他线程来执行一个具有 tpIdle 级的线程。

tpLowest：线程优先权低于 normal 二级。

tpLower：线程优先权低于 normal 一级。

tpNormal：线程优先权具有 normal 级。

tpHigher：线程优先权高于 normal 一级。

tpHighest：线程优先权高于 normal 二级。

tpTimeCritical：线程优先权级最高。

• ReturnValue 属性等效于一个函数的返回变量值。其语法说明如下：

```
property ReturnValue: Integer;
```

ReturnValue 报告线程的执行是成功还是失败。WaitFor 方法返回 ReturnValue 值。

• Suspended 属性显示一个线程是否被挂起。其语法说明如下：

```
property Suspended: Boolean;
```

挂起的线程被重新启动才能继续执行。设置 Suspended 为 True，则可挂起一个线程；设为 False 则使线程继续执行。

• Terminated 属性显示线程是否被要求终止。Terminate 方法自动设置 Terminated 属性为 True。

线程的 Execute 方法和请求 Execute 的方法均定期地检查 Terminated 的值，并当 Terminated 为 True 时会自动终止线程的执行。

• ThreadID 属性用于识别系统中的线程过程。ThreadID 不同于线程的 Handle 属性。其语法说明如下：

```
property ThreadID: THandle;
```

ThreadID 对于请求 Win32 API 函数来控制线程是非常有用的。

• Synchonize 方法完成同步执行存在于主 VCL 线程中的方法。其说明如下：

```
TThreadMethod = procedure of object;
procedure Synchronize(Method: TThreadMethod);
```

Synchronize 方法在主 VCL 线程内部执行特定的方法，从而避免了多线程冲突。请求处理一个 VCL 线程的方法和在主 VCL 线程内部更新一个窗体时，它们都会传递到 Synchronize。

注意：当 Synchronize 方法执行一个 VCL 线程时，其他线程的执行将被挂起。

• DoTerminate 方法引发线程的 Terminate 事件过程，其说明如下：.

```
procedure DoTerminate; virtual;
```

DoTerminate 请求 OnTerminate 事件句柄，但不终止这个线程。

• Resume 方法重新启动被挂起的线程。

• Suspend 方法暂停一个运行的线程。

• Create 方法创建了一个线程对象的实例。其语法说明如下：

```
constructor Create(CreateSuspended: Boolean);
```

如果 CreateSuspended 为 False，则 Execute 立即被请求；如果 CreateSuspended 为 True，则 Execute 直到 Resume 被请求后才能被请求。

• Execute 方法是一个必须被重载的方法，它包含线程开始执行时的代码。其说明如下：

```
procedure Execute; virtual; abstract;
```

当 Create 方法被请求时，一个线程就开始执行，并将 CreateSuspended 设为 False。当 Create

被请求并将 CreateSuspended 设为 True 时，线程自动请求 Resume 方法。线程开始执行时，将重载 Resume 方法和插入执行代码。Execute 负责检查 Terminated 属性值，并决定线程是否需要退出。

• Destroy 方法破坏线程对象和释放内存。其说明如下：

```
destructor Destroy; override;
```

• Terminate 方法终止线程的执行，它通过自动设置 Terminated 属性为 True 来实现。

线程的 Execute 方法和请求 Execute 的方法定期地检查 Terminated 属性值，且当 Terminated 值为 True 时退出。

• WaitFor 方法等待线程终止，然后返回 ReturnValue 的属性值。其说明如下：

```
function WaitFor: Integer;
```

WaitFor 直到线程终止时才返回值，因此，线程必须通过 Execute 方法或当 Terminated 属性值为 True 时退出。如果线程使用了 Synchronize 方法，就不要在主 VCL 线程中使用 WaitFor，这样做可能导致死机。Synchronize 在试图进行同步执行之前等待主 VCL 线程进入信息链。如果主 VCL 线程已经请求 WaitFor，那么它将不进入信息链，Synchronize 也永远不返回线程信息。主 VCL 线程只有引发 Ethread 才能终止。当 WaitFor 被请求时，如果 Synchronize 在主 VCL 线程中处于等待状态，那么线程对象不能介入，导致程序死锁。

• OnTerminate 事件是在线程的 Execute 方法执行完成之后和线程被破坏之前引发。其语法说明如下：

```
TNotifyEvent = procedure (Sender: TObject) of object;
property OnTerminate: TNotifyEvent;
```

OnTerminate 事件句柄在主 VCL 线程中被请求，这意味着在 OnTerminate 事件中可以使用 VCL 方法和属性，线程对象也可在事件过程中释放。

TNotifyEvent 类型是一个无参数事件类型，这些事件仅仅关注特定事件发生的组件对象。

2.3 多线程查询数据库

本节介绍一个使用多线程的数据库查询的实例。该实例通过使用多线程在后台进行多个数据库查询并显示查询结果。这是一个综合性的数据库应用程序，包括汇编程序的调用、初始化文件的使用以及多线程使用等多方面的编程问题。

首先建立一个数据库查询显示窗体，该窗体上有一个 Query 组件、一个 DataSource1 组件、一个 DBGrid 组件、一个 Database 组件和一个 Session 组件。设置 DataSource 组件的 DataSet 属性为 Query，设置 DBGrid 组件的 DataSource 属性为 DataSource1。然后放置两个 Label 组件 QueryLabel 和 StatusLine，将 QueryLabel 组件的 Caption 属性设为 Query Text，将 StatusLine 组件的 Caption 属性设为 Opening...。整个窗体布局如图 2-1 所示。

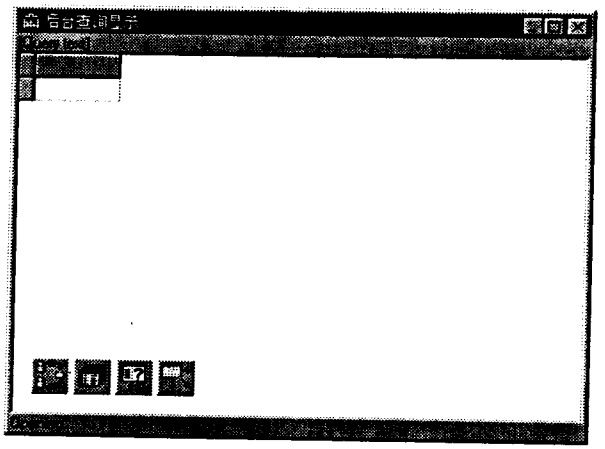

图 2-1 数据库查询显示窗体

系统建立了一个窗体对象 QueryForm：

```
type
  TQueryForm = class(TForm)
    QueryLabel: TLabel;
    DBGrid1: TDBGrid;
    DataSource: TDataSource;
    Query: TQuery;
    Session: TSession;
    StatusLine: TLabel;
    Database: TDatabase;
end;
```

下面基于这个窗体建立一个线程对象：

```
type
  TQueryThread = class(TThread)
private
  QueryForm: TQueryForm;
  MessageText: string;
  procedure ConnectQuery;
  procedure DisplayMessage;
protected
  procedure Execute; override;
public
  constructor Create(AQueryForm: TQueryForm);
end;
```

创建线程过程：

```
constructor TQueryThread.Create(AQueryForm: TQueryForm);
```

```
begin
    QueryForm := AQueryForm;
    FreeOnTerminate := True;
    inherited Create(False);
end;
```

线程执行过程：

```
procedure TQueryThread.Execute;
var
  UniqueNumber: Integer;
begin
  try
    with QueryForm do
    begin
      UniqueNumber := GetUniqueNumber;      {获得线程号}
        {获取数据库信息}
      Session.SessionName := Format('%s%x', [Session.Name,
            UniqueNumber]);
      Database.SessionName := Session.SessionName;
      Database.DatabaseName := Format('%s%x', [Database.Name,
                        UniqueNumber]);
      Query.SessionName := Database.SessionName;
      Query.DatabaseName := Database.DatabaseName;
      Query.Open;      { 打开 Query }
      Synchronize(ConnectQuery); { 连接 Query 到 DBGrid1 上}
        {更新状态行信息}
      MessageText := 'Query openned';
      Synchronize(DisplayMessage);
    end;
    except
     on E: Exception do
      begin
     { 显示线程出错信息}
        MessageText := Format('%s: %s.', [E.ClassName, E.Message]);
        Synchronize(DisplayMessage);
     end;
    end;
  end;
```

使用 GetUniqueNumber 函数获取一个惟一线程号，以确保 Query 一个惟一的 Session 和 Database。每一个线程需要一个惟一的 Session，因为 Database 是特定的 Session，所以它必

须惟一。

因为设定 Query 到 DataSource 将会改变 DBGrid1 的内容，而且 DBGrid1 的内容改变必须在主 VCL 线程内完成，所以 Query 与 DBGrid1 的连接必须通过 Synchronize 函数同步化。由于 StatusLine 是一个 VCL 控制组件，所以它也必须在主 VCL 线程中同步进行。

这个过程使用 try...except...end 结构处理线程运行的出错情况。

下面完成线程数据集连接：

```
procedure TQueryThread.ConnectQuery;
begin
  with QueryForm do DataSource.Dataset := Query;
end;
```

下面的过程显示线程信息：

```
procedure TQueryThread.DisplayMessage;
begin
  with QueryForm do StatusLine.Caption := MessageText;
end;
```

最后给出查询显示的后台进行过程：

```
procedure BackgroundQuery(const QueryName, Alias, User, Password,
    QueryText: string);
var
  QueryForm: TQueryForm;
begin
  QueryForm := TQueryForm.Create(Application);  {创建数据查询窗体}
  with QueryForm, Database do
  begin
    Caption := QueryName;
    QueryLabel.Caption := QueryText;
    Show;          {显示数据查询窗体}
    {得到数据库别名和口令以及查询指令}
    AliasName := Alias;
    Params.Values['USER'] := User;
    Params.Values['PASSWORD'] := Password;
    Query.Sql.Text := QueryText;
  end;
  {创建后台线程执行查询}
  TQueryThread.Create(QueryForm);
end;
```

由于线程本身可以随时终止，所以我们不需要一直监控它，只要创建它就可以了。这个查询窗体程序代码如下：

```
unit ResltFrm;
interface
uses
```

```
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    ExtCtrls, DB, DBTables, Grids, DBGrids, StdCtrls;
  type
    TQueryForm = class(TForm)
      QueryLabel: TLabel;
      DBGrid1: TDBGrid;
      DataSource: TDataSource;
      Query: TQuery;
      Session: TSession;
      StatusLine: TLabel;
      Database: TDatabase;
  end;
  procedure BackgroundQuery(const QueryName, Alias, User, Password,
    QueryText: string);
  implementation
  {$R *.DFM}
  { 查询线程}
  type
    TQueryThread = class(TThread)
    private
      QueryForm: TQueryForm;
      MessageText: string;
      procedure ConnectQuery;
      procedure DisplayMessage;
    protected
      procedure Execute; override;
    public
      constructor Create(AQueryForm: TQueryForm);
    end;
  constructor TQueryThread.Create(AQueryForm: TQueryForm);
  begin
    QueryForm := AQueryForm;
    FreeOnTerminate := True;
    inherited Create(False);
  end;
```

调用汇编程序跟踪线程号：

```
  var
    Guard, Numbers: Integer;
  function GetUniqueNumber: Integer;
  asm
```

```
@@1:    MOV     EDX,1
        XCHG    Guard,EDX
        OR      EDX,EDX
        JNZ     @@2
        MOV     EAX,Numbers
        INC     EAX
        MOV     Numbers,EAX
        MOV     Guard,EDX
        RET
@@2:    PUSH    0
        CALL    Sleep
        JMP     @@1
end;

procedure TQueryThread.Execute;
var  UniqueNumber: Integer;
begin
  try
    with QueryForm do
    begin
      UniqueNumber := GetUniqueNumber;
      Session.SessionName := Format('%s%x', [Session.Name, UniqueNumber]);
      Database.SessionName := Session.SessionName;
      Database.DatabaseName := Format('%s%x', [Database.Name,
                  UniqueNumber]);
      Query.SessionName := Database.SessionName;
      Query.DatabaseName := Database.DatabaseName;
      Query.Open;
      Synchronize(ConnectQuery);
      MessageText := 'Query openned';
      Synchronize(DisplayMessage);
    end;
  except
    on E: Exception do
    begin
      MessageText := Format('%s: %s.', [E.ClassName, E.Message]);
      Synchronize(DisplayMessage);
    end;
  end;
end;
procedure TQueryThread.ConnectQuery;
```