本书写于 2001 年秋，其中部分内容可能不够准确，仅供参考

# 第四章　设计数据库应用程序

## 4.1　数据库简介

数据库应用的开发是 Delphi 最重要和最强大的功能之一。

Delphi 能够很方便地开发基于文件数据库的单层数据库应用，基于 RDBMS（Relational Database Menagement System，关系型数据库管理系统）的两层 C/S（Client/Server，客户机/服务器）数据库应用。特别是从 Delphi 3 开始，Delphi 提供了 MIDAS（Multi-tired Distributed Application Services Suite，多层分布式应用服务套件），可以很方便地在 Windows 平台下进行多层分布式的数据库应用开发，在 Delphi 6 里，MIDAS 已被改名为 DataSnap，功能上较 Delphi 5 的 MIDAS 3 有所增强。

另外，对于不同的数据库系统，Delphi 提供了灵活多样的数据库连接方式。包括从 Delphi 1 就开始提供的 BDE（Borland Database Engine，Borland 数据库引擎）以及从 Delphi 5 开始提供的 ADO（ActiveX Data Object，ActiveX 数据对象），IBExpress（专用于 InterBase 数据库的高速访问技术）。特别是 Delphi 6 新增的 dbExpress，这是一种高速、通用、跨平台的数据库访问技术，本书将主要围绕 dbExpress 来介绍 Delphi 6 的数据库应用开发，同时对其它数据库连接访问方式也会作一些简单介绍。

同时，Delphi 提供了丰富的数据感知（Data-Aware）控件，极大地方便了数据库应用程序的界面开发。

### 4.1.1　数据库应用程序的体系结构

随着数据库应用的发展，数据应用程序的体系结构从早期的单层发展到了两层 C/S，特别是随着 Internet 的发展，多层 C/S 和 B/S（Browser/Server，浏览器/服务器）应用也得到越来越广泛的使用。

下面逐一介绍这几种结构。

### 4.1.2　单层数据库应用

单层数据库应用结构如图 1：

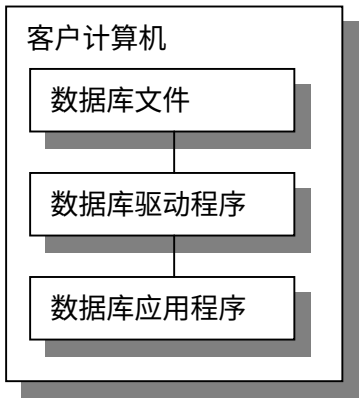



图 1：单层文件型数据库应用结构

单层数据库应用结构的特点是数据库应用程序的运行和数据库数据文件的存放是在同一台计算机中，并且是客户端的计算机。数据库文件的数据供唯一的一个数据库应用程序独占使用。

优点是结构简单；并且由于数据库文件只供一个应用程序使用，不需要处理并发数据访问所必须的数据锁定工作，所以速度通常很快（据有人测试过，在单机中使用 1,500 万条记录的规模下，FoxPro2.5 For DOS 的速度是最快的，较一般 RDBMS 要快数十倍之多）；通过使用网络文件共享的方式可以实现准网络版应用。

缺点是由于数据库文件存放在客户端，用户可能绕过数据库应用程序，直接修改数据库文件内容，即使是准网络版应用，因为数据库文件必须向客户端应用开放共享，情况与本地文件是一样的，所以数据安全性较差；另外，由于数据文件的访问是独占方式，所以即使是准网络版应用，同时也只能有一个客户端访问数据库，不支持并发访问；不支持事务（Transaction），数据库文件的可靠性差，很容易因为意外掉电等原因而出现数据库文件被破坏和数据丢失的情况。

说明：文件型数据库也叫平面文件型数据库(Flat-File Database)，这是 Delphi 专家 Charlie.Calvert 所提出的说法。

### 4.1.3 两层 C/S 数据库应用

两层 C/S 数据库应用结构如图 2：

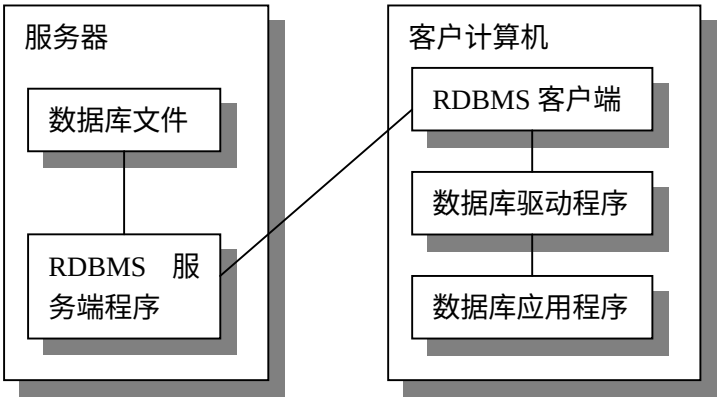


图 2：两层 C/S 型数据库应用结构

其特点是数据库应用程序和数据库数据分别存放于不同的计算机中，数据库数据由单独专门的 RDBMS 服务端程序管理和维护，数据库应用程序通过 RDBMS 的客户端程序和网络通讯与服务端建立数据交流；客户端通过 SQL（Structured Query Language，结构化查询语言）操作服务端的数据，服务端只向客户端返回必要的数据，使得在网络中传输的数据量较单层文件型数据库应用结构的准网络版要少，是一种真正的网络版应用；服务端程序提供了完善的安全机制，没有取得授权的用户无法直接访问到数据库；通常的 RDBMS 都支持标准的 SQL，不过也有少数 RDBMS 不完全支持，例如在 PHP 中常用的 MySQL 数据库；并且提供视图（View）、存储过程（Stored Procedure）和触发器（Trigger）等强大的功能；通常都支持事务。

优点是真正的网络版应用；完善可靠的安全机制；高效的并发访问控制；并且在并发访问增加时性能下降不多，而且很多 RDBMS 支持通过服务器集群技术来改善多用户并发访问时的性能；使用统一的数据库操作语言-SQL；支持事务；由于有专门的服务端程序维护数据库文件，可靠性高；通常提供丰富的管理工具用于数据的备份、恢复、导入、导出等。

缺点是通常较为昂贵，特别是像 Oracle, IBM DB2, Infomix 等企业级数据库系统，但也有免费的产品，如：InterBase 6 免费版，MySQL 等；因为功能强大，所以学习难度也较大，配置使用较为复杂。

注意：InterBase 6 免费版和 MySQL 都只限于在非商业应用的情况下免费，详见相应软件的 License 文件。

### 4.1.4 多层 MIDAS/DataSnap 数据库应用

一般的多层数据库应用结构如图 3：

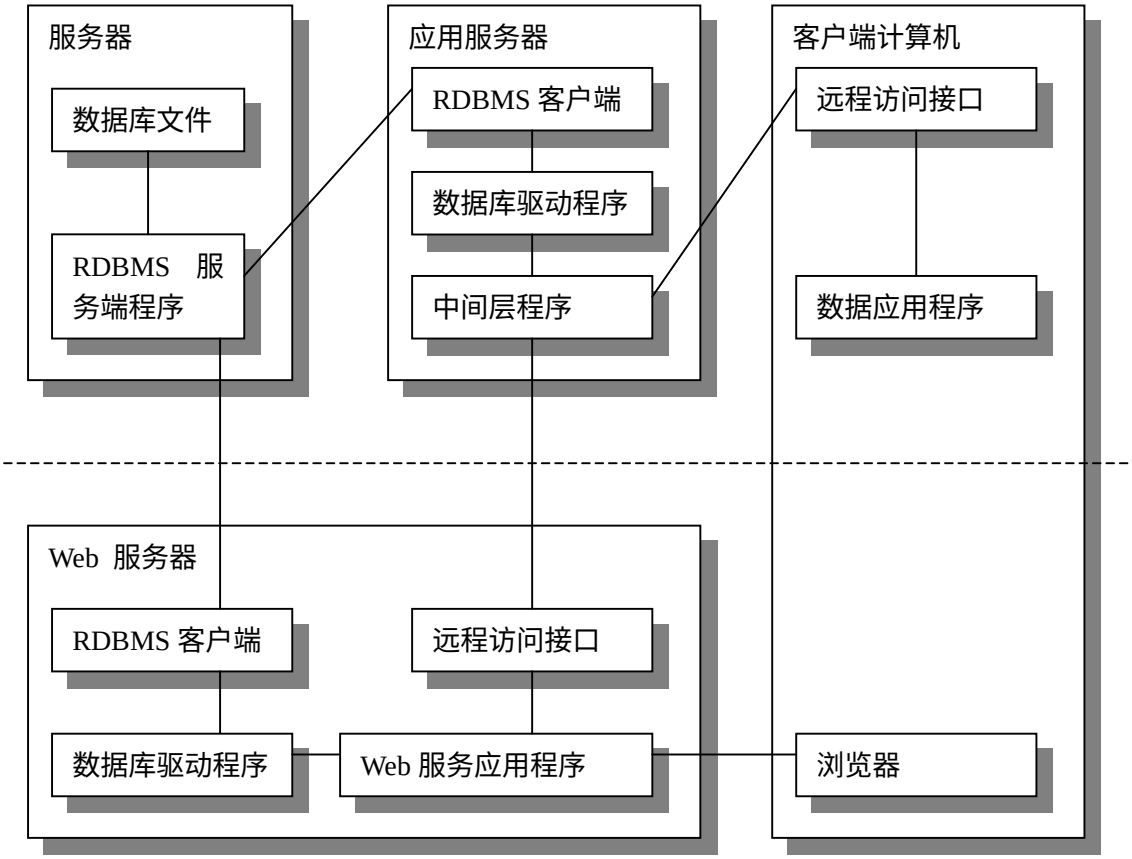



图 3：多层 C/S 和 B/S 型数据库应用结构

多层 C/S 型数据库应用结构（如图 3 中虚线以上部分）的特点是在传统的两层 C/S 型数据库应用结构中的客户端与服务端之间插入一层或几层中间件（Mid-ware）或称为应用服务器（Application Server）；由中间件处理应用系统的业务逻辑，客户端程序只处理界面的显示；由中间件与 RDBMS 通讯，客户端因为不需要与 RDBMS 通讯，所以不需要安装 RDBMS 的客户端程序和数据库驱动程序，可以使客户端程序变得更小，更快；中间件可以有多个并且可以安装在不同的计算机上，将处理工作分散开来，改善性能。

多层 B/S 型数据库应用结构（如图 3 中虚线以下部分）是专门为 Internet 的应用而设计的，其特点是不需要专门的客户端程序，客户端只要有浏览器即可使用；特别适合于使用拨号上网的低速网络；相当于在传统的两层 C/S 型数据库应用结构中的客户端与服务端之间插入一层 Web 服务应用程序（如图 3 中从 Web 服务应用程序到 RDBMS 客户端的数据访问路径），同时也可以利用多层 C/S 型数据库应用结构，只要将 Web 服务应用层作为中间件与客户端之间的一层即可（如图 3 中从 Web 服务应用程序到远程访问接口的数据访问路径）；其它特点与多层 C/S 型数据库应用结构相似。

说明：当然在多层数据库应用结构中，也可将文件型数据库代替 RDBMS 来作为系统的后端数据库，但这种情况下同一时刻只能有一个中间件连接到数据库上，实际上意义不大。

多层数据库模式将数据库应用程序合理地分块。客户端程序专门处理数据显示和用户界面。在理想的情况下，它不需要了解数据是如何被存储及维护的。应用服务器（中间层）协调和处理来自多个客户端的请求和数据更新。它处理了所有定义的数据集的细节以及与数据库的交互。

多层模式的优势包括以下几个方面：

- 把业务逻辑封装在共享的中间层里。不同的客户端都访问相同的中间层。这可以使你减少由于在每个单独的客户端应用中重复你的业务逻辑所造成的冗余（以及相应的维护成本）。
- “瘦”的客户端。你的客户端应用程序可以写得很小，而把大多数工作交给中间层处理。客户端应用程序不仅是变小了，而且还更加的易于发布，因为它们不需要再考虑安装，配置和维护数据库连接软件（例如 Borland 的数据引擎 - - BDE 以及数据服务器的客户端软件）的问题。“瘦”客户端应用程序可以通过 Internet 以更加灵活的方式发布。
- 分布式数据处理。将一个应用系统的工作分布到几台机器上可以改善系统的性能，因为可以提供负载平衡以及用备用的机器去替代发生故障的机器。
- 增强安全性。你可以通过使用不同的访问约束，来分层隔离敏感的功能。这提供了一个灵活的和可配置的安全层。中间层可以限制敏感部分的入口点，使你能更加容易地控制对它的访问。如果你使用 HTTP, CORBA 或是 COM+ ，你还可以同时享受到它们支持的安全模式所带来的优势。

多层数据库应用结构的主要缺点是较为复杂。

基于 **MIDAS/DataSnap** 的多层数据库应用开发：

从 Delphi 3 开始，为多层数据库的应用开发提供了 MIDAS 技术，特别是 Delphi 5 提供的 MIDAS 3，是 Windows 平台下基于 COM（Component Object Model，组件对象模型）技术实现的最好的多层分布式应用开发技术之一，而新的 Delphi 6 将 MIDAS 改名为 DataSnap，是在 MIDAS 3 的基础上增强了用于组件化开发的多 RDM（Remote Data Module，远程数据模块）中间件支持以及对 COM+/SOAP 的支持。

> 说明：SOAP 是指 Simple Object Access Protocol，即简单对象访问协议。这是一种最近由 Microsoft 提出的全新的分布式技术，是一种基于 XML 技术的通用对象访问机制，是 Microsoft 的.Net 计划的核心，并被国际上如 IBM，SUN 等大公司所支持；是一项未来在互联网应用软件开发领域里很有前途的技术。

利用 MIDAS/DataSnap，可以很方便快速地开发出基于现有的各种分布式技术实现的多层数据库应用系统，包括：DCOM（Distributed COM，分布式 COM）、TCP（Transfer Control Protocol，传输控制协议，MIDAS/DataSnap 是通过一种被称为 Socket 的网络编程接口实现的）、HTTP（HyperText Transfer Protocol，超文本传输协议）、MTS/COM+（Microsoft Transaction Server，微软事务服务）、CORBA（Common Object Request Broker Architecture，公用对象请求代理体系）、SOAP。

MIDAS/DataSnap 的结构如图 4：

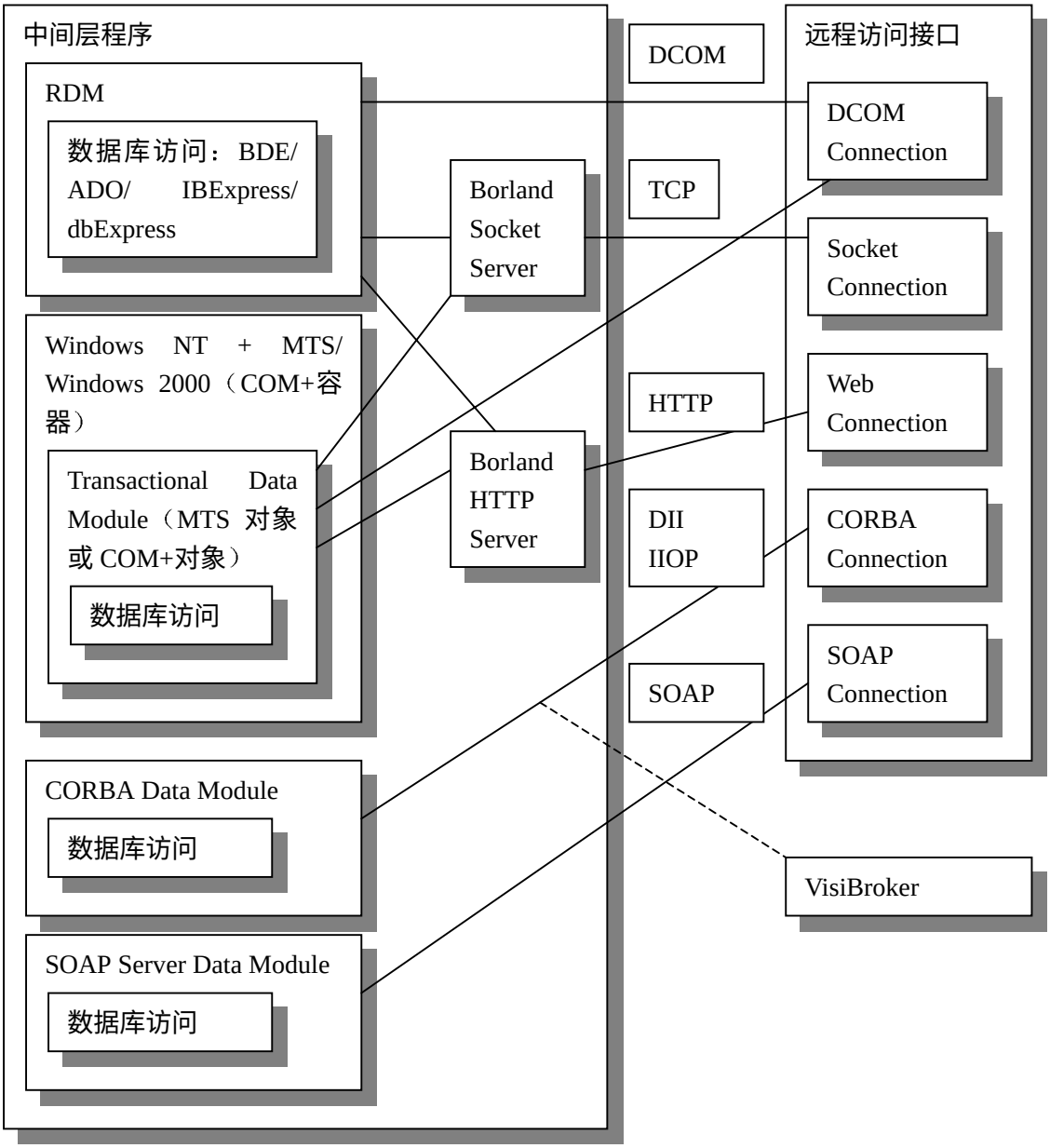



图 4：MIDAS/DataSnap 的结构

在图 4 中，除了 CORBA 和 SOAP 以外，其它的 MIDAS/DataSnap 连接方式都是通过 COM 技术实现的，其中 RDM 是独立运行的 COM 服务器，而 TDM（Transactional Data Module 事务数据模块）是一种 MTS/COM+组件，它和 RDM 的区别是需要在相应的 Container（容器）中运行。在 Windows NT 平台下的容器是 MTS；而在 Windows 2000 下，其本身就是 COM+的容器；采用这种形式可以借助于容器所提供的强大的事务处理、Pooling（一种缓冲机制，可以改善性能，及减少数据库服务端的并发访问量）、集成安全机制等。RDM 和 TDM 的默认通讯方式是 DCOM，Borland 另外还提供了 Socket Server 和 HTTP Server，可以将 DCOM 通讯转化为 TCP 或 HTTP 通讯，使得在用 DCOM 无法连通的情况下，也能使用 MIDAS/DataSnap。

Socket Server 是一个独立运行的服务端程序，客户端可以通过

SocketConnection 连接到中间件，通常在需要绕开 DCOM 的安全机制或跨网段访问等的情况下使用 SocketConnection。

HTTP Server 是一个 ISAPI（Internet Server Application Programming Interface，Internet 服务应用程序编程接口，是一种嵌入到 Web 服务器中运行的高性能的 Web 应用程序类型）程序，通常可以用在 Internet 中，特别是对于通过防火墙（Firewall）或代理服务器（Proxy）连接到 Internet 的情况下，用 SocketConnection 连接不通时，但它同时需要一个 Web 服务器。

CORBA 是一种跨平台、跨语言的分布式技术，通过 IIOP（Internet InterORB Protocol，Internet ORB 间协议，ORB 是指 Object Request Broker，对象请求代理，是 CORBA 的服务协调软件，如 Borland 的 VisiBroker 是目前市场应用最多的 CORBA 服务产品，其中的 SmartAgent 就是一个 ORB 服务程序），Delphi 是通过 DII（Dynamic Invocation Interface，动态调用接口）的方法来访问 CORBA 对象的。使用 CORBA 作为 MIDAS/DataSnap 的通讯方式时需要在网络中运行 VisiBroker 的 SmartAgent。

SOAP 是一种基于 XML 的对象访问协议，它将对象间的访问请求及其响应内容用 XML 表示，可以通过任何已有的协议来传输此 XML 文档，最通常的应用是通过 HTTP 协议，所以 SOAP 服务也被称为 Web Services。每个 Web Services 都有相应的接口，这个接口也是用 XML 描述的，称为 WSDL（Web Services Description Language，Web 服务描述语言）。客户端程序通过从服务端导出一个 WSDL 文档来取得服务端提供的接口，并通过此接口，用 SOAP 来访问服务端。

由于 MIDAS/DataSnap 本身的复杂性，本书不打算作进一步的介绍，有兴趣的读者请参考专门介绍这方面技术的有关书籍。

## 4.2 数据库连接方式

针对于不同的数据库系统，Delphi 6 提供了灵活多样的数据库连接方式。包括支持多种文件型数据库和多种 RDBMS 的 BDE、Microsoft 的 ADO（特别适合用于 Microsoft 的数据库产品，如 MS Access 和 MS SQL Server）、专用于 InterBase 数据库的 IBExpress、高速、通用、跨平台的 dbExpress。

接下来将逐一介绍这几种数据库连接方式：

### 4.2.1 BDE 介绍

BDE 原来也叫 IDAPI，采用 BDE 连接的数据库应用程序结构如图 5：

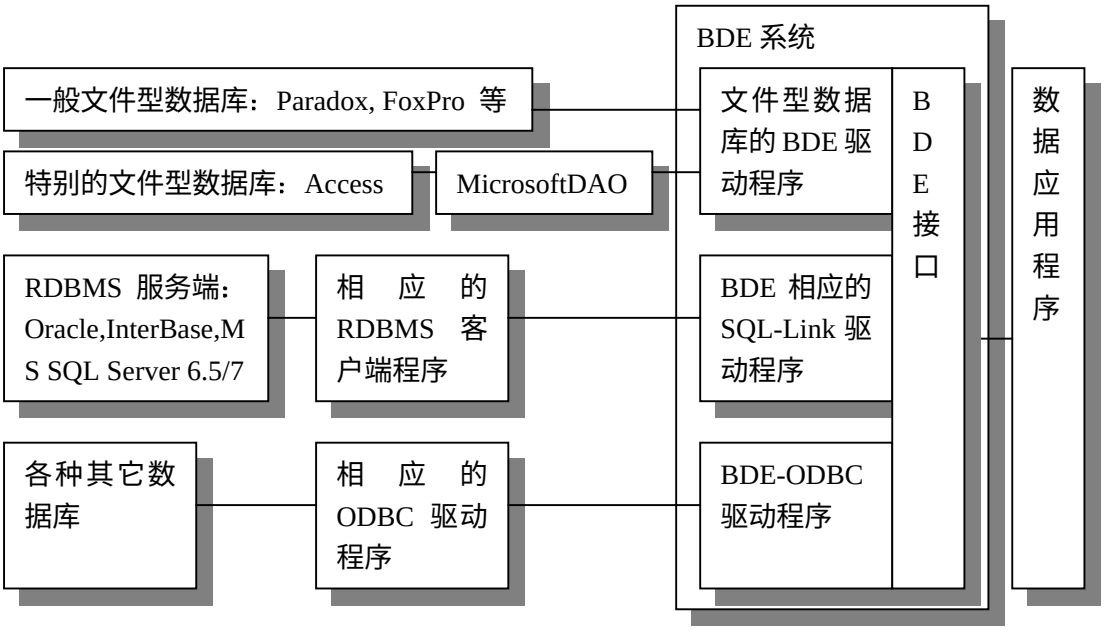



图 5：采用 BDE 连接的数据库应用程序结构

BDE 系统通过三类驱动程序将文件型数据库，RDBMS 和 ODBC（Open Database Connective，开放数据库连接，用于连接到其它 BDE 不能直接支持的数据库）统一到同样的 BDE 接口下。

对于一般的文件型数据库，BDE 的驱动程序能够直接操作相应的数据库文件，实现数据库的基本功能。但其中 Microsoft 的 Access 数据库是个例外，因为它是一种 COM 类型的数据库，它只能通过 Microsoft 所提供的 MS-Jet（Microsoft 用于驱动 Access 数据库的一种专用驱动程序，据说新版本的 MS-Jet 也可以驱动 Excel 文件）接口来访问，而 MS-Jet 只提供了 DAO（Data Access Object，数据访问对象），ADO，ODBC 等几种访问方式，其中以 DAO 最快，所以 BDE 使用 MS-Jet 的 DAO 来驱动 Access 数据库，这也使得使用 Access 数据库的应用程序系统中必须包括 MS-Jet 的 DAO（通常在安装了 MS Office 的机器上就有 DAO）。另外 BDE 的文件型数据库驱动程序为这些文件型数据库提供了一种 SQL 支持，被称为 Local SQL。

对于 RDBMS，BDE 提供了一种被称为 SQL-Link 类型的驱动程序，用于将各种 RDBMS 的客户端接口统一到 BDE 的接口下。目前 BDE 已支持市场上绝大多数的 RDBMS，包括：Oracle, IBM DB2, Sybase, Infomix, Microsoft SQL Server, Borland InterBase 等。由于 Microsoft 在其 SQL Server 2000 中不再提供基于 API（Application Programming Interface，应用程序编程接口）方式的客户端程序，而 BDE 正是通过客户端 API 的方式访问 MS SQL Server 的，所以 BDE 不能直接驱动 MS SQL Server 2000；对于 MS SQL Server 7.0，其客户端 API 也不支持完整的功能， BDE 虽然可以连接，但也不能实现 MS SQL Server 7.0 的全部功能。Microsoft 只在 ADO 连接中提供了对 SQL Server 7.0 和 2000 的全部功能的支持，所以从 7.0 版开始，建议使用 ADO 来连接 Microsoft SQL Server，但对于 MS SQL Server 6.5 及以前版本，BDE 都可以完全支持。

对于 BDE 不能直接支持的数据库，如 MySQL 等，只能通过 BDE-ODBC

驱动程序访问。ODBC 是一种由 Microsoft 提出的通用数据库连接方式，几乎所有的数据库系统都提供了相应的 ODBC 驱动程序，所以，通过 BDE-ODBC 驱动程序，BDE 可以间接地访问这些数据库系统。但由于 ODBC 为了保证做到足够通用，在性能上作出一些牺牲，所以用 ODBC 访问数据库的速度会比较慢一些。另外，对于 BDE 能够直接驱动的数据库系统也可以用 BDE-ODBC 驱动，但性能会较 BDE 直接驱动要差一些，如果不是在不得已的情况下（例如在使用 Access 数据库却没有 DAO 时），不建议使用。

最后要注意的一点是 BDE 有一点限制就是在一台机器上最多只能同时有 48 个 Session，而每个 Session 最多只能有 256 个连接。这一点在进行多层数据库应用开发时要注意，不过对于两层 C/S 结构来说这不是问题，因为每个客户端都是通过各自的 BDE 连接的。

配置和使用 BDE：

BDE 是通过别名（Alias）来访问不同的数据库的。BDE 的别名有两种：BDE 原生（Native）别名和 BDE-ODBC 别名。BDE 原生别名是指使用 BDE 原生驱动程序的别名，BDE 的原生驱动程序包括文件型数据库驱动程序和 SQL-Link 驱动程序；BDE-ODBC 别名包括 BDE 自动从 ODBC 的 DSN（Data Source Name，数据源名）导入的别名和在 BDE 中建立的使用 BDE-ODBC 驱动程序的别名。

建立 BDE 别名可以通过使用 Delphi 所带的工具：BDE Administrator 或 SQL Explorer（在专业版中叫做 Database Explorer）。可以在“开始”菜单的 Delphi 中找到，其中 BDE Administrator 也可以在“控制面板”里找到，而 Delphi 里的【Database】|【Explorer】菜单项也可以启动 SQL Explorer。用这两个工具建立 BDE 别名的方法基本相同，下面以 SQL Explorer 为例，介绍如何建立上面所说的两类四种 BDE 别名。

SQL Explorer 运行后的界面如图 6：

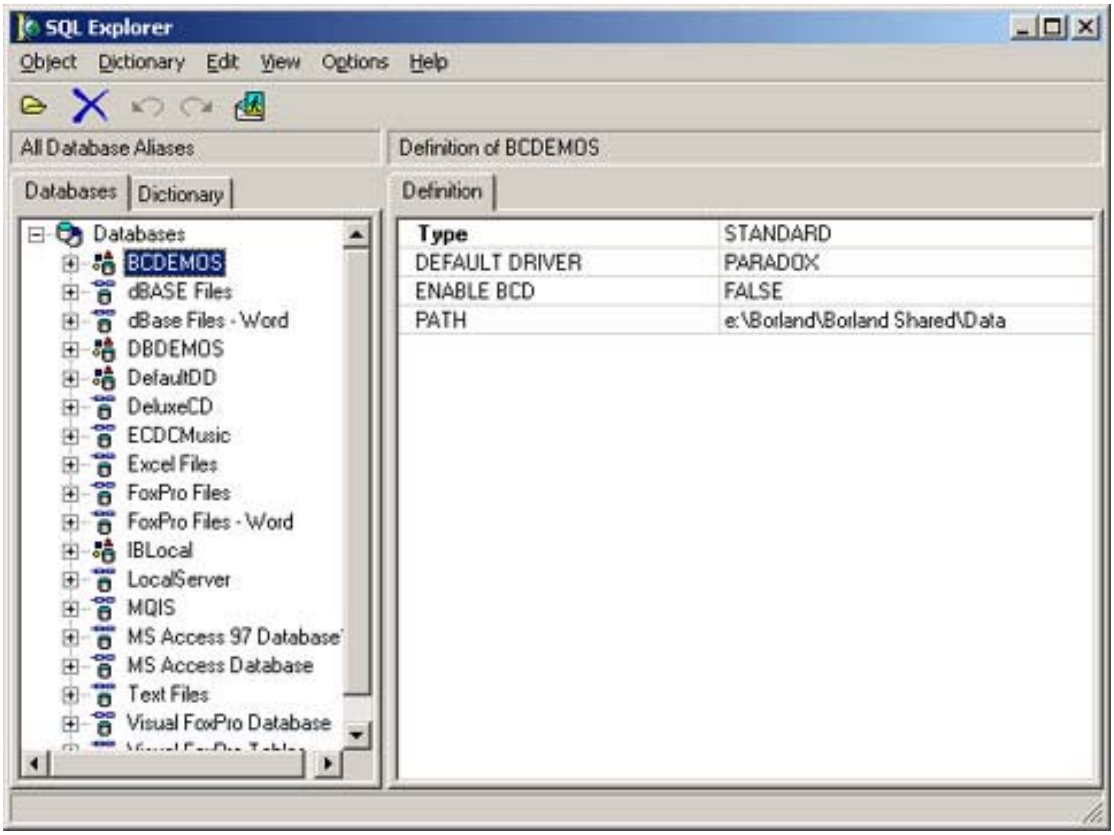



图 6：SQL Explorer

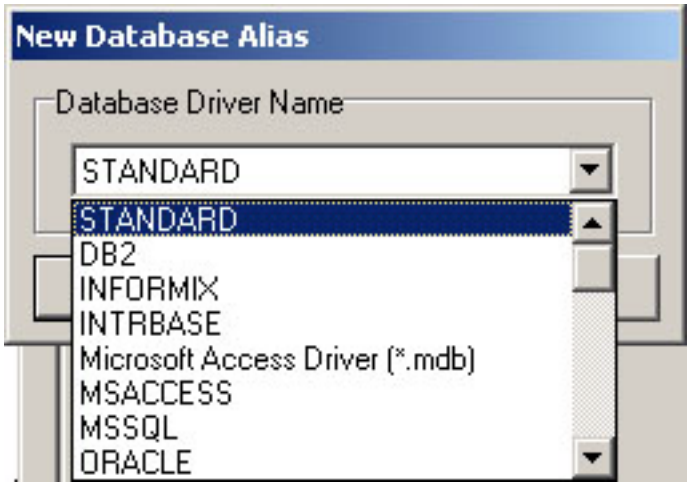



图 7

建立文件数据库型原生别名：在图 6 的左边，Databases 页中的 Databases 节点上右击鼠标，在弹出的菜单上选择【New】，在弹出的对话框（如图 7）中选择相应的驱动程序，其中只有 STANDARD 和 MSACCESS 是文件型数据库的驱动程序，分别用于一般文件数据库和 MS Access 数据库。以 STANDARD 为例，确定后在 SQL Explorer 的左边会新增一个叫做 STANDARD1 的别名，可以将它改为其它你需要的名字，在它左边还有一个绿色的小箭头，表示此新建别名尚未保存。在右边的 Definition 页中设置别名所必要的参数：

1. DEFAULT DRIVER(默认驱动程序)：可以选择 PARADOX，DBASE，FOXPRO，ASCIIDRV 四种，分别用于 Paradox,dBase,FoxPro,文本四种文件型数据库。

2. ENABLE BCD（允许用 BCD 表示浮点数）：用于指示 BDE 是否用 BCD 方式记录浮点数，在需要精确表示浮点数的情况（如记录金额的情况）下应该将此参数设为 TRUE。
3. PATH（路径）：表示数据库文件存放的路径。

设置完以上参数后，就可以按工具栏上的 Apply 按钮（那个向右的蓝色箭头按钮）保存此别名，确定保存后，此别名左边的绿色小箭头就没有了。

建立 SQL-Link 型原生别名：建立方法与文件型数据库原生别名类似，只是所用的驱动程序不同，BDE 原生的 SQL-Link 驱动程序有：DB2, INFOMIX, INTERBASE, MSSQL, ORACLE, SYBASE。因为对不同的 RDBMS，对应的 BDE 别名参数差别较大，在此不再一一介绍，请参考相应数据库的有关文档设置。

建立 ODBC DSN 别名：在“控制面板”中打开“ODBC 数据源管理器”在其中的“用户 DSN”或“系统 DSN”中增加你需要的数据源名并配置好，BDE 就会自动将其转为一个 BDE-ODBC 别名，可以像一般 BDE 别名一样使用。如果在 SQL Explorer 里看不到刚加的 ODBC DSN 别名，可以刷新一下。不过 BDE 不支持“文件 DSN”。

建立 BDE-ODBC 别名：建立方法与文件型数据库原生别名类似，所不同的只是选择相应的 ODBC 驱动程序，在驱动程序列表中，除了上面所说的原生 BDE 驱动程序以外的，都是 ODBC 驱动程序，根据需要选择相应的驱动程序即可。

### 4.2.2 ADO 介绍

采用 ADO 连接的数据库应用程序结构如图 8：

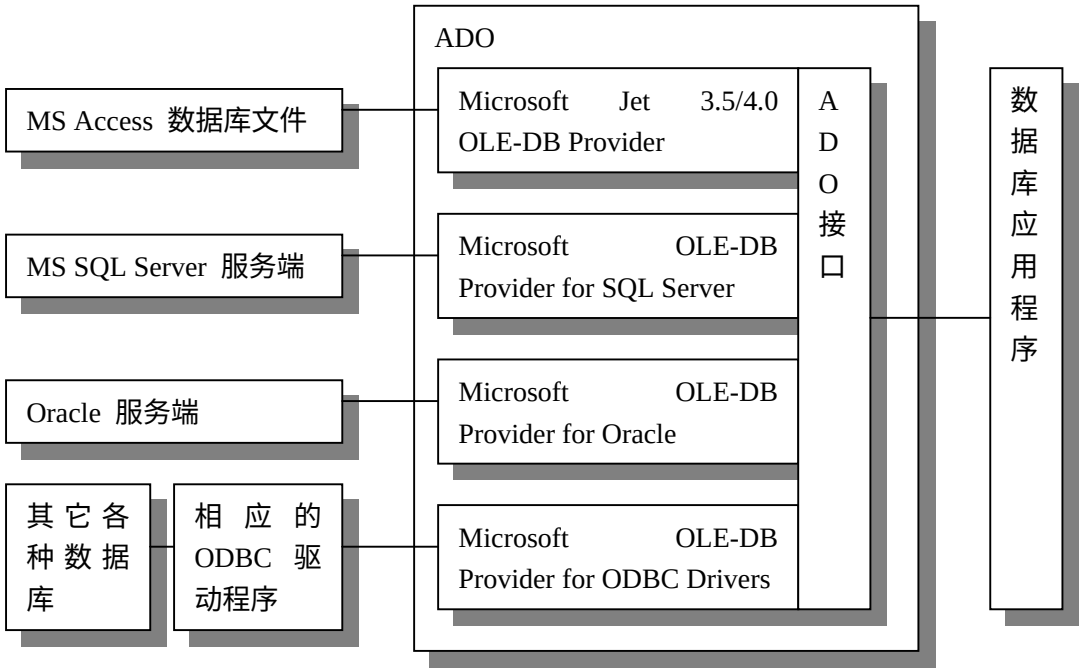



图 8：采用 ADO 连接的数据库应用程序结构

ADO 是通过一种称为 OLE-DB Provider 的驱动程序来访问各种数据库的，

OLE-DB 是 Microsoft 提出的一种基于 COM 技术的数据库访问方式，由于此技术过于复杂，Microsoft 对它进行了封装，简化了对它的使用，这就是现在的 ADO。

由于对 MS SQL Server 来说，它的 OLE-DB Provider 驱动包括了 RDBMS 客户端的内容（甚至更多），所以不需要另外再安装 RDBMS 的客户端，使得对文件型数据库的访问与对 RDBMS 的访问是一样的。对于未提供相应的 OLE-DB Provider 驱动的数据库系统，只能通过 ADO-ODBC 驱动（即 OLE-DB Provider for ODBC Drivers）来访问。

由于目前提供了全部功能的 OLE-DB Provider 的数据库只有 Microsoft 的 Access 和 SQL Server，虽然 Microsoft 也提供了 Oracle 的 OLE-DB Provider 驱动，但由于不是 Oracle 提供的，所以不能支持 Oracle 的全部功能，此外，还有一些第三方的厂商提供部分其它数据库系统的 OLE-DB Provider 驱动，但同样由于不是数据库厂商提供的，支持也不是十分全面，而且使用第三方的东西需要冒一定的风险。通常 ADO 最适合用于连接 Microsoft 的数据库系统。

ADO 对连接的数量没有限制。但是在进行多线程开发时要注意一些与 COM 技术有关的问题，具体请参考有关 COM 技术的资料。

### 4.2.3 IBExpress 介绍

采用 IBExpress 连接的数据库应用程序结构如图 9：

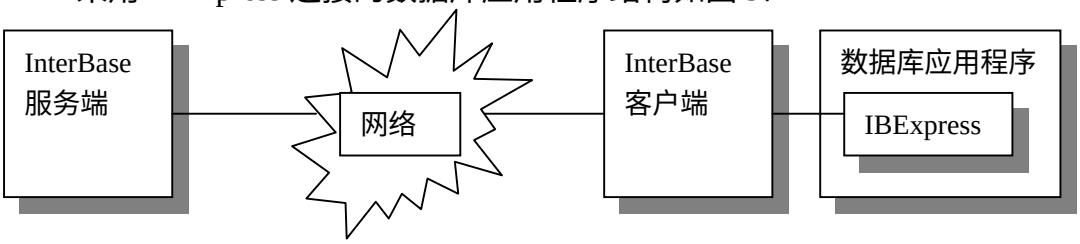



图 9：采用 IBExpress 连接的数据库应用程序结构

IBExpress 是一组专门用于 InterBase 数据库的数据访问控件。由于它是通过直接使用 InterBase 客户端 API 访问 InterBase 数据的，所以它具有很高的数据库访问速度，并且除了 InterBase 客户端程序外，不需要其它任何驱动程序。

虽然目前 Kylix 暂不支持 IBExpress，但因为 InterBase 是一种跨平台数据库，而 Kylix 现在暂时还只提供 dbExpress 一种数据库连接方式，所以未来 Borland 很可能会为以后版本的 Kylix 提供 IBExpress 的支持。

> 说明：Kylix 是 Borland 在 2001 年推出的最重要的几个产品之一，是 Linux 平台下的 Delphi。Borland 专门为 Kylix 开发了一种与 VCL 很相似的，但是是跨平台的应用程序框架（Application Frameworks）--CLX（Component Library Cross-Platform，跨平台组件库，读作[‘kliks]）。Delphi 6 也支持 CLX 应用的开发。通过 CLX，Delphi 6 可以与 Kylix 共享源程序，并实现跨平台开发。
>
> 在 2001 年第四季度，Borland 推出了全新的 Kylix2，全面支持 Delphi6 的 SOAP/WebSnap 等技术，支持完整的 CORBA 开发能力。Kylix2 的发布将 Linux 平台下的各种应用开发带入了一个全新的领域。
>
> 关于 CLX 的更多内容，请参考关于 Kylix 的资料。

关于 InterBase 数据库，将在本章接下来的内容中作进一步介绍。

### 4.2.4 dbExpress 介绍

采用 dbExpress 连接的数据库应用程序结构如图 10：

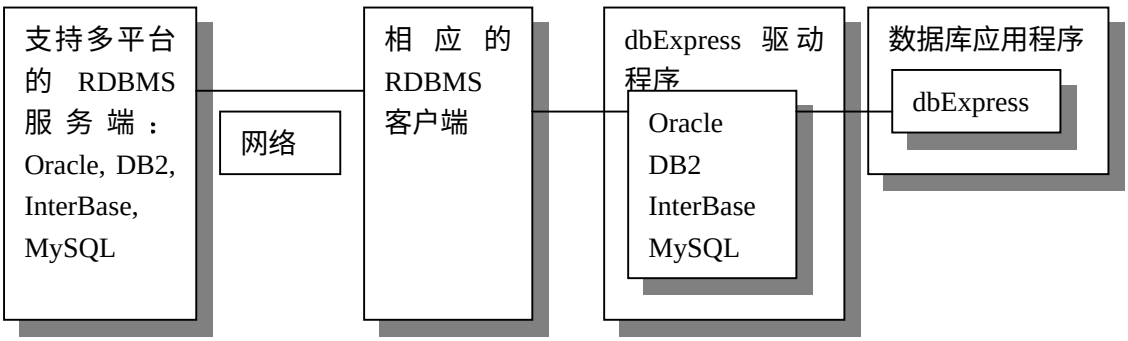



图 10：采用 dbExpress 连接的数据库应用程序结构

dbExpress 是 Borland 首先在 Kylix 中实现的一种高性能的，通用跨平台数据库连接方式。它将成为 Borland 继 BDE 之后又一项重要的数据库访问技术。

目前在 Delphi 6 中，dbExpress 同时支持 VCL 和 CLX。

dbExpress 的一大特点是只实现了数据库操作的最基本功能，如：不作数据缓冲，只能按单向顺序访问数据等，所以性能很好，并且驱动程序很简单（目前支持的四种数据库的 dbExpress 驱动程序都只有一个 DLL 文件—指 Delphi 6 用的 Windows 版本）。但同样因为这个原因，访问数据会不太方便，例如只能单向访问就不能直接使用数据敏感控件，但这可以通过借助于 Delphi 6 的 DataSnap 功能中的 DataSetProvider+ClientDataSet 的方法来为其增加数据缓冲和双向数据访问能力。

使用 dbExpress+DataSnap 的技术还有一个附加的好处，就是它与多层的 MIDAS/DataSnap 的结构极为相似，如图 11：

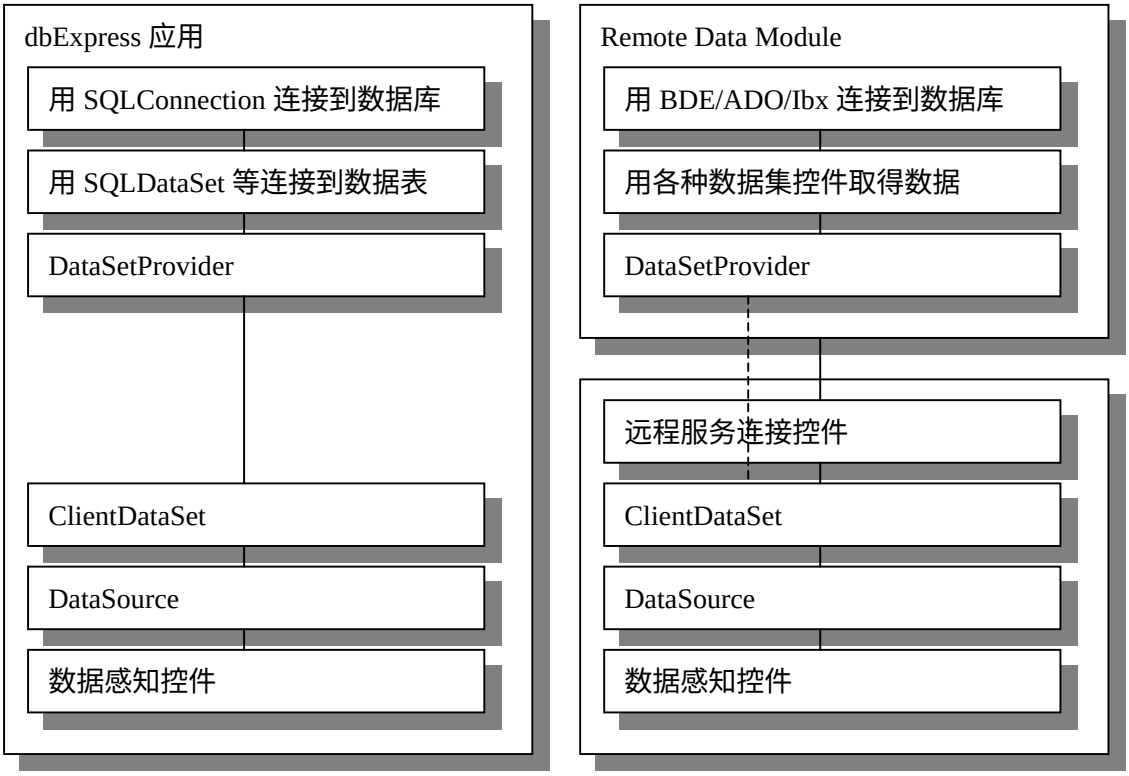



图 11：dbExpress 与 MIDAS/DataSnap 结构的比较

这使得用 dbExpress 的两层 C/S 结构可以很容易地改成多层的数据库应用系统，这也就意味着这样系统具有很好的灵活性和弹性。

### 4.2.5 不同连接方式的比较及选择

总结以上介绍的四种数据库连接方式，并将结果以列表形式表达，如表 1：

表 1：四种数据库连接方式的比较

| | BDE | ADO | IBExpress | dbExpress |
|---|---|---|---|---|
| 支持的数据库 | 支持绝大多数据文件型数据库和 RDBMS（不包括 MS SQL Server 2000），支持通过 ODBC 访问其它数据库 | 支持 Microsoft 的文件型数据库 Access 和 RDBMS 产品 SQL Server，部分支持 Oracle 数据库，支持通过 ODBC 访问其它数据库 | 只支持 InterBase 数据库 | 只支持跨平台的数据库，目前已提供四种驱动程序，未来可能增加 |

| | | | | |
|---|---|---|---|---|
| 应用程序发布 | 需要同时发布BDE（较大），RDBMS 还需要相应的客户端 | 需要客户安装MDAC（Microsoft Data Access Component，其中包括 ADO 和主要的 OLE-DB Provider 驱动程序），对 SQL Server 来说不需要客户端 | 只需要客户端即可 | 只需要一个 DLL 文件的驱动程序和相应的客户端即可 |
| 跨平台 | 否 | 否 | 否（未来可能提供） | 是 |

从上表可见，四种数据库连接方式各有所长，那么在实现应用中应该如何选择呢？

首先应当确定系统将要使用什么数据库（关于数据库的选择，将在本章接下来的内容中作简要介绍），然后才能确定用什么方式连接。

一般来说，使用除 Microsoft Access 以外的其它文件型数据库时建议使用 BDE，因为只有 BDE 和 ADO 支持文件型数据库，而 ADO 除了能直接驱动 Access 数据库以外，其它的文件型数据库都只能通过性能较差的 ODBC。

对于 Microsoft 的数据库产品，不论是文件型的 Access 还是 RDBMS 的 SQL Server，建议最好还是用 ADO。因为 BDE 只能间接地通过 DAO 来访问 Access 数据库；因为 BDE 是通过早期（6.5 及以前版本）的 SQL Server 所提供的 DB-Lib 的应用程序接口来访问 SQL Server 的；而 SQL Server 7 几乎重写了 6.5 的全部代码，它所提供的 DB-Lib 只包含了 SQL Server 7 的部分功能，所以 BDE 不能完全支持；而 Microsoft 对 SQL Server 2000 及以后的版本不再提供相应的 DB-Lib 接口，所以 BDE 是不支持 SQL Server 2000 及以后的版本（除非通过 ODBC），这一点要注意。

对于 InterBase 数据库，建议使用 IBExpress，因为它是最直接的访问 InterBase 的方法。用 dbExpress 也不错。当然，用 BDE 也还是不错的，毕竟它已经很成熟了，不过要注意的是，BDE 在这三种方式中相对性能较差（详见本章末的比较结果）。

对于其它 RDBMS，建议采用 dbExpress 或 BDE，因为这两种连接方式支持市场上的大多数 RDBMS 产品。

在必须使用 ODBC 的情况下，可以根据需要选择用 BDE-ODBC 或是用 ADO-ODBC，因为瓶颈在 ODBC 上，所以这两种连接方式在性能上相差不多。

将以上的结果整理如表 2：

表 2：不同的数据库的建议连接方式

| 数据库 | 建议连接方式（优先选择排在前面的） |
|---|---|
| Paradox, dBase, FoxPro | BDE |
| Access | ADO |
| MySQL | dbExpress |
| InterBase | IBExpress, dbExpress, BDE |
| MS SQL Server | ADO(7.0 及以后版本), BDE(7.0 以前版本) |
| Sybase, Infomix | BDE |
| IBM DB2 | dbExpress, BDE |
| Oracle | dbExpress, BDE, ADO |
| ODBC | BDE-ODBC, ADO-ODBC（任选其一） |

注意：几乎所有数据库都可以通过 ODBC 连接，但这是不得已的选择，所以表中没有列出。大多数情况下 BDE 的性能比 ADO 好一些，在两者均可的情况下（如 Oracle）优先考虑 BDE。

### 4.2.6 数据库系统的比较与选择

Delphi 6 可以支持很多的数据库系统，而在实际应用中也常常有几种数据库可供选择，如何才能从中确定最适合的数据库呢？我们将在本小节中对市场上常见的数据库作一个比较，便于读者在实际工作中选择需要的数据库。

通常按照不同的标准来划分，数据库系统可以划分为不同的级别。一般可以按数据库系统的在实际应用中的规模将其分为如下个级别：桌面应用级、部门级、企业级三个级别。

各级别适用的情况及相应的数据库系统如表 3：

表 3：数据库系统的分级

| 数据库系统的级别 | 适用情况 | 数据库 |
|---|---|---|
| 桌面应用级 | 单机或少量机器通过网络文件共享方式访问，不存在并发访问的情况，通常数据量也较少，一般记录数在 $10^4$ 数量级，大多数库文件一般在几 M 以内，少数可能达到几十甚至上百 M。但也有例外的情况，由于所用的都是文件型数据库，且不存在并发访问，不需要额外的数据锁定开销，所以对大数据量的数据操作速度可能比 RDBMS 更快，记录数也可能很多，甚至可能达到 $10^7$ 数量级 | Paradox, MS Access, FoxPro, dBase 等 |

| | | |
|---|---|---|
| 部门级 | 数十台至上百台机器通过网络以 C/S 方式访问，存在一些并发访问的情况，通常数据量较大，记录数一般在 $10^5$ 数量级以上，库文件一般在百 M 以上甚至 GB 级。在要求不高的情况下，也可以用于 Internet 及多层分布式应用。 | InterBase, MS SQL Server, Sybase 等 |
| 企业级 | 数百台甚至更多的机器通过网络或 Internet 以 C/S 或多层分布方式访问，存在大量并发访问的情况，通常数据量极大，记录数可能达到 $10^7$ 以上甚至更多，库文件则可能达到天文数字的 TB 级。一般企业级数据库都支持分布式数据库，能够通过分布方式处理巨型数据库。 | Oracle, IBM DB2, Infomix 等 |

上述的划分只是数据库系统的一种划分方式，对数据库系统的划分并不是那么严格，特别是部门级数据库，有时也可以在桌面级或企业级情况下使用；而桌面级有时也可以用于部门级；如果不考虑成本的关系，同样可以将企业级数据库用于部门级的情况；这些都要结合实际情况考虑。

下面具体介绍一下上面列出的几种数据库。

dBase：

最早的 PC 下的数据库系统，由美国 Ashton-Tate 公司发表。后因 Ashton-Tate 公司被 Borland 公司收购而成为 Borland 的产品。作为文件型数据库的典型代表，获得最广泛的支持，但毕竟是早期的数据库产品，已经不适合现在的应用情况，建议不是为了与老系统接口，不应该再使用这种数据库。

FoxBase/FoxPro：

FoxBase 是一种从 dBase 基础上发展而来的，与 dBase 完全兼容的数据库系统，由美国 Fox 公司发表。后来 Fox 公司在 FoxBase 基础上又开发出了 FoxPro，在 Fox 公司被 Microsoft 收购后发表了 FoxPro 2.5，成为 DOS 平台下应用最广泛的数据库系统之一。后来 Microsoft 又将 FoxPro 移值到 Windows 平台下，在 Windows 95 发布后，Microsoft 又将 FoxPro 并入 Visual Studio 开发工具集中，推出了 Visual FoxPro，达到了 FoxPro 的光辉顶点，曾经是 Windows 平台下应用最广泛的数据库系统之一。但 FoxPro 毕竟是从 dBase 发展而来，存在很多早期数据库系统的通病，如没有数据锁定机制，不安全的数据库文件结构等，所以 Microsoft 已经不打算再对 FoxPro 作进一步的大的发展，建议如果不是为了与老系统接口，不应该再使用这种数据库。

Paradox：

是由 Borland 公司开发的一种文件型数据库，在安全性，性能等方面较 dBase 有很大改进，提供了基于数据表的锁定机制，曾经在 DOS 平台和早期 Windows 平台获得广泛应用。专门为 Delphi 提供的 Paradox 7 是 Paradox 最后一个版本，是 BDE 支持得最好的一个文件型数据库系统。建议在采用 BDE 连接的文件型数据库系统中首选 Paradox。

Microsoft Access：

是由 Microsoft 公司开发的一种文件型数据库，作为 Microsoft Office 的一部分发表，是一种基于 COM 技术实现的数据库系统，提供了简单的数据表锁定机制，是 VB 的默认文件型数据库，从 Access 97 开始获得广泛应用。由于 Access 是一种 COM 数据库，所以不能像一般文件型数据库那样直接访问库文件，必须通过 Microsoft 提供的数据库驱动程序，早期是通过 DAO 访问 Access 数据库的，较为复杂，后来 Microsoft 提供了更为简单的 ADO 数据库访问技术，并且据称将淘汰 DAO。相比 Paradox，Access 的数据库只有一个文件，而 Paradox 每个表都至少有一个文件，所以 Access 数据库的发布与维护都比 Paradox 要简单，特别是因为它是使用 ADO 的，而从 Windows 98 开始就提供有 ADO（Windows 98 的 ADO 在 PWS 里，并且版本较低），Windows 2000 更是默认提供了 ADO 2.5，可以很方便地访问 Access 数据库。建议建议在使用 ADO 连接的文件型数据库系统中首选 Access 数据库。

InterBase：

是由 Digital 公司开发的数据库系统，后来 Borland 购得了 InterBase，并专门成立了一家名为 InterBase 的子公司，负责此产品，是早期的 RDBMS 之一，其历史与 Oracle 及 DB2 相差无几，是相当成熟的产品。InterBase 的主要特点是弹性好，中等规模，兼有大型数据库和小型数据库的优点，可用于从桌面级到小型企业级，并且对硬件要求低，在硬件条件不高的情况下仍有很好的性能，提供基于记录的锁定机制，完全支持 SQL 标准，完整支持 Transaction, View, Stored Procedure 和 Trigger，支持多种操作系统平台，Delphi 为 InterBase 提供了灵活多样的支持。缺点是不支持日志，只使用单数据库文件（虽然 InterBase 6 支持多数据库文件，但是用主从文件的形式，较单文件的情况改善有限），在大量并发访问时可能出现 IO 瓶颈影响性能，与操作系统结合不够紧密，不支持分布式数据库，相关扩展功能少，不适合大型企业级应用。

Microsoft SQL Server：

最早是由 Microsoft 和 Sybase 共同开发，从 7.0 版开始为 Microsoft 独立开发。早期版本与 Sybase 兼容程度很高，从 7.0 版开始加入许多 Microsoft 特有的功能，如与 Windows 紧密结合，大量采用 COM 技术等。提供了基于数据页的锁定机制，完全支持 SQL 标准，完整支持 Transaction, View, Stored Procedure 和 Trigger，提供简单的日志，扩展了许多与 Microsoft 的其它产品（如 IIS，Exchange 等）结合的功能，有适用于不同情况的多个版本，从 7.0 版开始提供对桌面应用的支持。缺点是只支持 Windows 平台，对企业级的支持很有限，由于数据库本质上仍是单文件（虽然数据库由两种文件组成，其中一种是日志文件，另一种是数据库文件，每种文件都可以有多个，但不是像 Oracle 那样按表空间来分，所以相比单文件改进很有限，这一点与 InterBase 类似），性能一般，但从 SQL Server 2000 版的性能有较大的提高（但在大量并发访问时的 IO 瓶颈问题仍然存在），增加了一些如服务器连接（注意，与真正的服务器集群有区别，要差一些，而且配置较麻烦）等企业级应用的功能，在要求不高时也可以用于中型企业级应用中。

补充说明：Microsoft 为了方便在桌面级应用中使用 SQL Server，特别是为了方便程序的发布，提供了一种称为 MSDE（Microsoft Database Engin）的软件，它基本上相当于最小化安装的 SQL Server Desktop。

Sybase：

由 Sybase 公司开发，是 Microsoft SQL Server 的前身，与 MS SQL Server 的早期版本兼容程序很高，功能上与 MS SQL Server 相当，主要区别是支持多种操作系统平台，提供有多种版本以用于不同情况的需要。有时也可以用于企业级应用。

Oracle：

Oracle 是由 Oracle 公司开发的一种企业级数据库系统，以 Oracle 8 为例，其最主要的特点是在大量并发访问的情况下仍有很好的性能，很好的安全性，支持强大的日志，以表空间为单位支持多文件数据库，可以通过将数据库的表空间分散到多个磁盘中，降低 IO 冲突的可能，进一步提高在大量并发访问情况下的性能，完全支持分布式数据库，与操作系统紧密结合，甚至取代了部分操作系统的功能，Oracle 就有部分内存是由其自已管理的，不受操作系统控制，通常还提供了全方位的开发工具。目前在企业级应用领域占第一位。最新的 Oracle 9i 有更多的改进。

DB2/Infomix：

DB2 是由 IBM 公司开发，Infomix 是由 Infomix 公司开发，不过 Infomix 公司的数据库部门已于 2001 年 4 月被 IBM 公司收购，Infomix 数据库将成为 IBM 的产品，并且在未来可能被并入 IBM 的 DB2 数据库中，所以这里将二者一并说明。DB2 除了具有一般企业级数据库的功能外，还集成了电子商务、商业智能与内容管理等功能，在性能方面也相当好，新的 DB2 UDB（通用数据库，IBM 还有 DB2 的专用版，如 for IBM AS/400 的 DB2/400 等） 7.x 版支持从多种数据库移植到 DB2，提供了对 OLE-DB（即原生 ADO 驱动）的支持，与 Windows 2000 的高度集成等。

> 特别说明：除了上面所说到的以外，T.c.X 公司的 MySQL 也是一种可以用于部门级应用的数据库系统。目前除了用 dbExpress 可以直接访问以外（Delphi 6 需要打补丁 UpdatePack 1 才能使用新版本的 MySQL），BDE 和 ADO 都只能通过 ODBC 访问 MySQL，它不完全支持 SQL 标准（如不支持嵌套查询），不支持 View, Stored Procedure, Trigger，不支持事务。只是因为 PHP 对其支持很好，所以随着 PHP 的流行而应用渐多，但基于上述原因，不推荐使用。

由于 Microsoft 的宣传，许多人常会将 SQL Server 认为是企业级数据库产品，表 4 将 SQL Server 和真正的企业级产品 Oracle 和 DB2 作一比较，相关内容来自网站文章，仅供参考：

表 4：MS SQL Server, Oracle, DB2 的综合比较

| | MS SQL Server | Oracle | DB2 |
|---|---|---|---|
| 开放性 | 只能在 Windows 上运行，没有丝毫的开放性， | 能在所有主流平台上运行（包括 | 能在所有主流平台上运行（包括 Windows）。 |

| | | | |
|---|---|---|---|
| | 操作系统的系统的稳定对数据库是十分重要的。Windows9x 系列产品是偏重于桌面应用，Windows NT server 只适合中小型企业。而且 Windows 平台的可靠性，安全性和伸缩性是非常有限的。它不象 Unix 那样久经考验，尤其是在处理大数据量的关键业务时。 | Windows）。完全支持所有的工业标准。采用完全开放策略。可以使客户选择最适合的解决方案。对开发商全力支持。 | 最适于海量数据。DB2 在企业级的应用最为广泛,在全球的 500 家最大的企业中，几乎 85%以上用 DB2 数据库服务器，而国内到 97 年约占 5%。 |
| 可伸缩性，并行性 | 并行实施和共存模型并不成熟。很难处理日益增多的用户数和数据卷。伸缩性有限。 | 平行服务器通过使一组结点共享同一簇中的工作来扩展 Window NT 的能力，提供高可用性和高伸缩性的簇的解决方案。如果 Windows NT 不能满足需要，用户还可以把数据库移到 UNIX 中。 | 具有很好的并行性。DB2 把数据库管理扩充到了并行的、多节点的环境。数据库分区是数据库的一部分，包含自己的数据、索引、配置文件、和事务日志。数据库分区有时被称为节点或数据库节点。 |
| 安全性 | 没有获得任何安全证书。 | 获得最高认证级别的 ISO 标准认证。 | 获得最高认证级别的 ISO 标准认证。 |
| 性能 | 多用户时性能不佳。 | 性能最高，保持 Windows NT 下的 TPC-D 和 TPC-C 的世界记录。 | 适用于数据仓库和在线事务处理，性能较高。 |
| 客户端支持及应用模式 | C/S 结构，只支持 Windows 客户，可以用 ADO，ODBC 等连接。 | 多层次网络计算，支持多种工业标准，可以用 ODBC，JDBC，OCI 等网络客户连接。 | 跨平台，多层结构，支持 ODBC，JDBC 等客户。 |
| 操作简便性 | 操作简单，但只有图形界面。 | 较复杂，同时提供 GUI 和命令行，在 Windows NT 和 Unix 下操作相同。 | 操作简单，同时提供 GUI 和命令行，在 Windows NT 和 Unix 下操作相同。 |
| 使用风险 | 从 7.0 版起，完全重写代码，经历了长期的测试，不断延迟，许多功能需要时间来证明。并不十分兼容早期产品。使用需要冒一定风险。 | 长时间的开发经验，完全向下兼容。得到广泛的应用。完全没有风险。 | 在巨型企业得到广泛的应用，向下兼容性好。风险小。 |

可以看出 MS SQL Server 的差距还是不小的。

为方便读者在实际工作中选择合适的数据库系统及连接方式，表 5 提供了部分数据库及连接方式的性能数据，由于软硬件情况的不同，可能会有差异，仅供参考：

表 5：不同的数据库及连接方式的性能比较

| 数据库 | 连接方式 | 连接时间 | 数据增加 | 缓冲增加 | 数据查询 | 删除数据 |
|---|---|---|---|---|---|---|
| Paradox 7 | BDE | 20 | 364 | 8392 | 256 | 83 |
| InterBase 6.01 | | 76 | 27543 | 22813 | 451 | 113 |
| MS SQL Server 7（MSDE） | | 258 | 11613 | 18046 | 137 | 95 |
| Oracle 8.0.5 | | 1148 | 31365 | 25971 | 169 | 2036 |
| Microsoft Access 2000 | ADO 2.6 | 10 | 30687 | 24185 | 405 | 38 |
| MS SQL Server 7（MSDE） | | 2 | 17001 | 14411 | 639 | 231 |
| SQL Server 2000 Enterprise | | 0 | 18610 | 14685 | 619 | 167 |
| Oracle 8.0.5 | | 0 | 78743 | 71279 | 497 | 2131 |
| InterBase 6.01 | IBExpress | 28 | 6940 | 3004 | 326 | 2547 |
| InterBase 6.01 | dbExpress | 25 | NA | 10332 | 500 | 113 |
| Oracle 8.0.5 | | 987 | NA | 17696 | 1338 | 1970 |

说明：以上数据为笔者用 Delphi 6 写的一个测试程序，在赛扬 600+128M RAM 的机器上，Windows 2000 Server 平台，每次 5000 条记录，三次重复的平均操作时间。表中 NA 表示不支持，因为 dbExpress 数据集只能通过 Provider/Resolve 方式更新数据（当然直接用 SQL 语句更新也是可以的，不过这就不同了）。连接时间不包括程序运行后第一次连接数据库的时间，通常因为第一次连接要作一些初始化的工作，会很慢的，一般比上表的连接时间要长数十倍，ADO 更是长达数千倍。

对表中的数据作一番分析可以发现一些有趣的现象：因为这个程序是一个典型的桌面应用程序，所以文件型数据库的表现比 RDBMS 好很多，特别是 Oracle 的性能非常的差，但要注意的是 Oracle 并不是一个适合桌面应用的数据库产品，它主要是在大量并发访问的情况下能有很好的性能，将它列出仅供参考。

主要从表中可以看出，在一般情况下，ADO 要比 BDE 慢一些，特别是对 Oracle 数据库，差别极大，这可能跟所用的 ADO 驱动程序是 Microsoft 提供，而不是 Oracle 有关。对于 InterBase 数据库，用 dbExpress 比用 BDE 快，而用 IBExpress 又比 dbExpress 快，这是正常的。

其中也有一些数据比较不正常：首先是用 BDE 连接 Paradox 数据库在打开 CachedUpdates 后，插入数据的速度反而要慢很多，这可能跟 BDE 本地文件处理机制有关。而 SQL Server 7 在用 BDE 连接时的表现比 InterBase 用 BDE 连接的表现还要好，这是因为所用的 SQL Server 7 是 MSDE 有关， MSDE 是 Microsoft 专门为桌面应用而推出的最小化的 SQL Server 7 Desktop 版，与标准版的 SQL Server 7 有所不同，主要是为桌面应用作了优化，但对并发访问的能力影响很大，SQL Server 7 Desktop 版/MSDE 在几个连接的并发访问情况下，性能就会有很大下降。这也是 SQL Server 2000 Enterprise 的表现比它略差的原因。另外，MSDE 也存在用 CachedUpdates 时速度反而慢的情况，估计这也是由于 MSDE 的优化方式与 BDE 的缓冲方式不同而导致的。

## 4.3 使用 dbExpress 的数据库应用

dbExpress 作为一种高性能、跨平台的通用数据库访问技术，将是 Borland 继 BDE 之后的一个新的发展方向。

dbExpress 引擎的观念是提供一层非常简单，有效率的数据存取接口和数据库驱动程序。借着这个接口，程序员能够存取各种不同的数据源，却只需要了解一个相同的接口即可，而在这个接口之后则是实际存取各种具体数据库的驱动程序。目前 Delphi 6 提供了四种数据库驱动程序,分别用于 DB2, InterBase, MySQL 和 Oracle。图 12 所示便是 dbExpress 控件：


图 12：dbExpress 控件

dbExpress 是为了以下目的而设计：

1. 最小的长度和系统资源利用
2. 最快的速度
3. 平台独立性
4. 配置简易
5. 容易开发驱动程序

dbExpress 驱动程序小而快，因为它们只提供很有限的功能。一个 dbExpress 驱动程序只实现了读取元数据、执行 SQL 语句和存储过程，并且返回的结果只是一个单向的只读游标。这种数据存取方式被称为 Firehose（消防水管）。然而，通过使用 DataSetProvider 和 ClientDataSet 等 MIDAS/DataSnap 控件来实现的 Provider/Resolve（提供/获取）数据读取，可以提供一个操纵 SQL 数据库的功能齐全、高效的、高协同性能的解决方案。

### 4.3.1 跨平台的数据库连接方式和单向游标的数据集访问方式

下面将用一个简单的例子来演示如何使用 dbExpress 连接数据库，以及用单向游标访问数据集。程序的 IDE 如图 13：

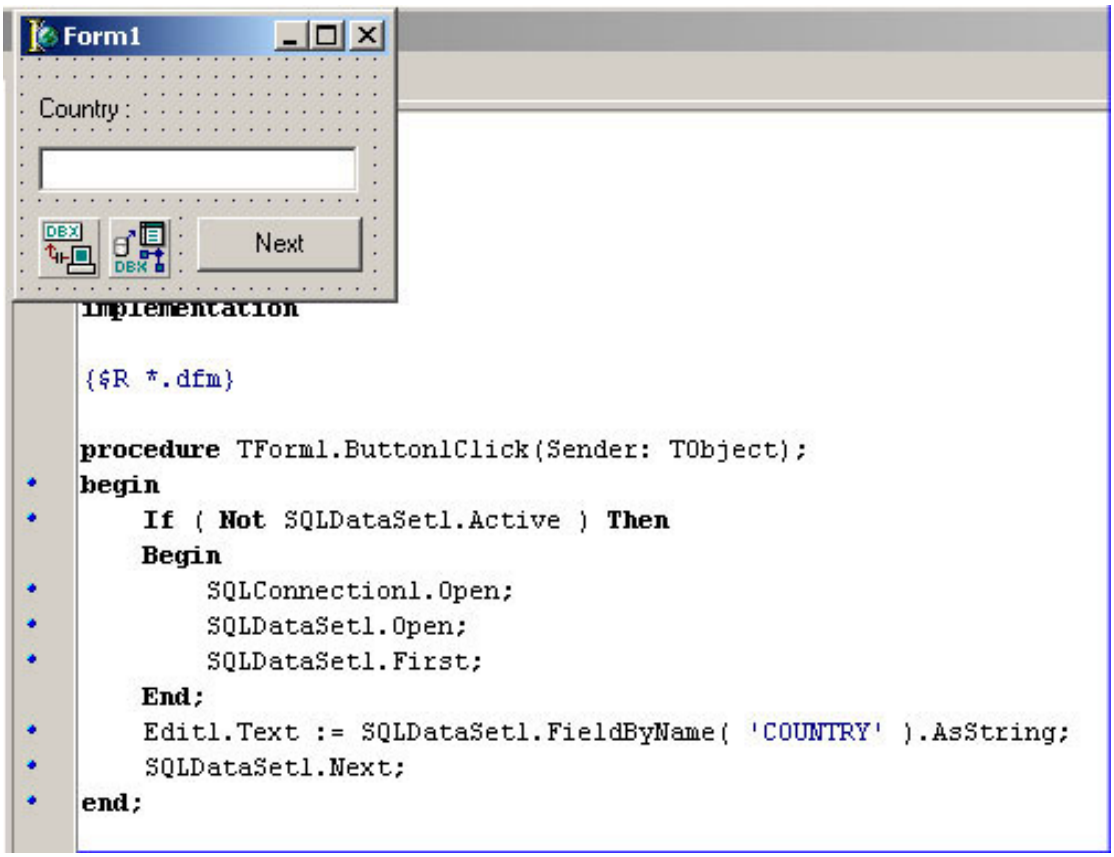



图 13：一个最简单的 dbExpress 应用程序例子

在 Form1 上依次放入几个控件，并设置其相应属性，如表 6：

表 6：最简单的 dbExpress 例子中的控件及属性

| 控件 | 要设置的属性 |
|---|---|
| TSQL Connection | ConnectionName := ‘IBLocal’;<br>LoginPrompt := false;<br>Params 中的 Database :=<br><InterBase 的安装路径><br>\Examples\Database\employee.gdb;<br>这是 InterBase 提供的一个例子库，请根据自己的 InterBase 安装路径设置上面的路径；<br>如果修改过 SYSDBA 的密码，请相应修改 Password |
| TSQL DataSet | SQLConnection := SQLConnection1;<br>CommandText := ‘select COUNTRY from COUNTRY’; |
| TLabel | Caption := ‘Country:’; |
| TEdit | Text := ‘’; |
| TButton | Caption := ‘Next’; |

IBLocal 是一个预定义的连接名，使用预定义连接可以自动设置大多数属性参数，也可以使用自己定义的连接，或者自己增加预定义的连接。

自定义连接也很简单，不用设置 ConnectionName，而是设置 DriverName，现有的四个 DriverName 都有预先设置好的参数，同样也会自动设置很多属性，所有驱动程序的预设置内容记录在 Borland Shared\DBExpress 目录下的 dbxdrivers.ini 文件中，在增加其它数据库的驱动程序后可以修改此文件，以增加对新驱动程序的支持。

增加预定义连接的方法也不复杂。双击 TSQLConnection 控件即可打开预定义连接管理界面，如图 14：

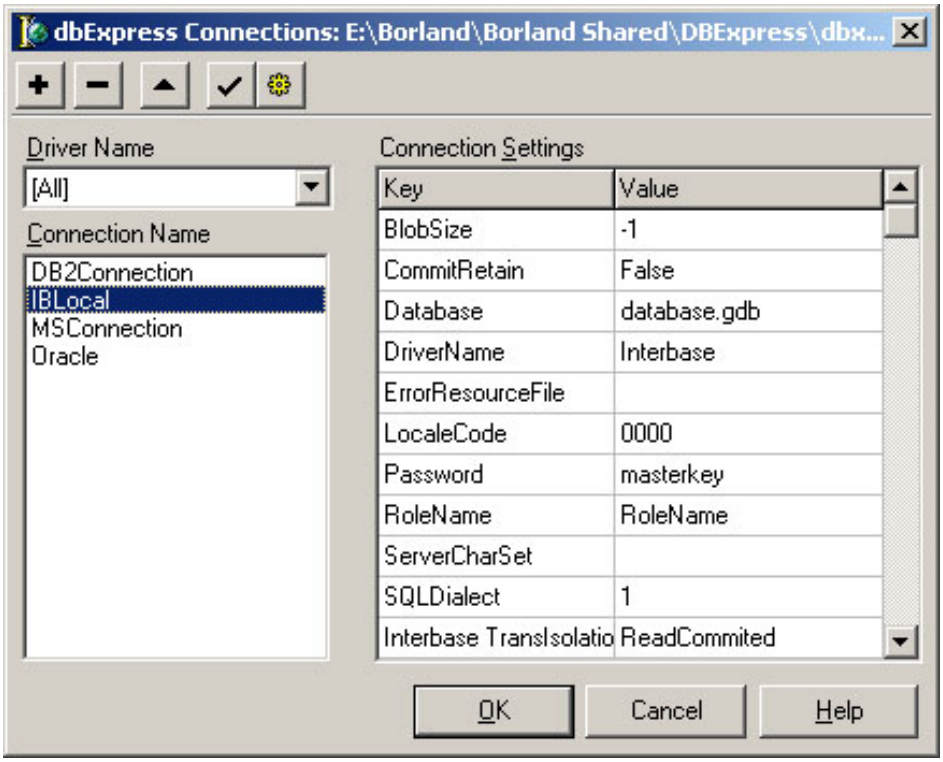



图 14：预定义连接管理

可以很方便地在其中增加自定义的新的连接设置，所有的预定义连接设置保存在 Borland Shared\DBExpress 目录下的 dbxconnections.ini 文件中，也可以通过修改此文件实现。

TSQLConnection 的 Params 属性设置方法如图 15：

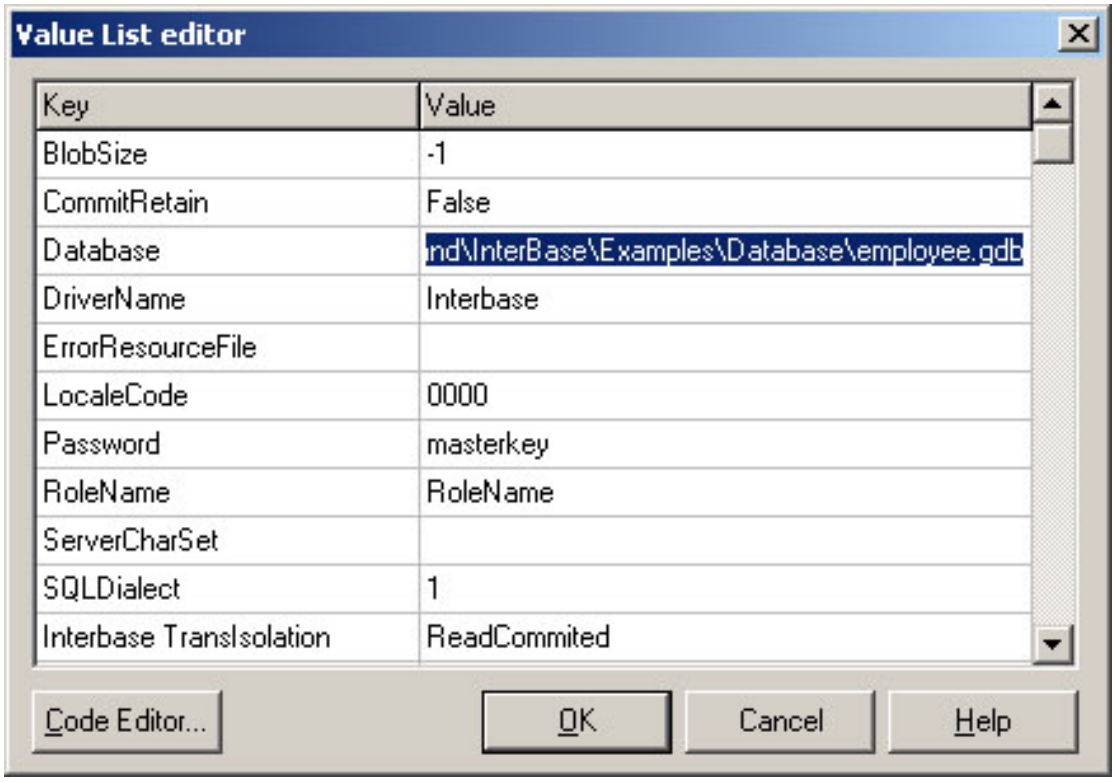



图 15：TSQLConnection 的 Params 属性设置

如表 5 依次设置好各控件属性后，双击按钮，输入下面的事件响应代码：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    If ( Not SQLDataSet1.Active ) Then
    Begin
        SQLConnection1.Open;
        SQLDataSet1.Open;
        SQLDataSet1.First;
    End;
    Edit1.Text := SQLDataSet1.FieldByName( 'COUNTRY' ).AsString;
    SQLDataSet1.Next;
end;
```

确定本机中的 InterBase 服务已启动，即可运行本程序，点击按钮即可依次查看下一个国家名。如图 16：


图 16：最简单的 dbExpress 例子程序

这个程序非常简单，最主要的一句是：

```
Edit1.Text := SQLDataSet1.FieldByName( 'COUNTRY' ).AsString;
```

它将数据集中的 COUNTRY 字段内容取出并在 Edit 控件中显示。这个程序与一般 BDE 的简单应用程序最大的区别是没有用到 TDataSource 控件和 Data-Aware 控件，因为 dbExpress 的数据集控件—TSQLDataSet—是一个单向数据集控件，而与 TDataSource 相连的数据集控件必须支持双向访问，所以在这个例子中是无法用 TDataSource 控件的，同样也就无法用 TDBEdit 控件来显示字段内容。另一个特别之处在于：TSQLDataSet 只有 Next 方法，而没有与之相对应的 Prior 方法，这也是单向数据集控件与一般数据集控件的一大区别之处。

### 4.3.2 如何实现缓冲式双向数据访问

从前面的例子可以看出，直接使用单向数据集访问控件是很不方便的，因为不支持 Data-Aware 控件，功能也较弱。为了解决这个问题，接下来将介绍一下如何通过 MIDAS/DataSnap 的 Provider/Resolve 机制进行缓冲双向数据访问。其结构如图 17。

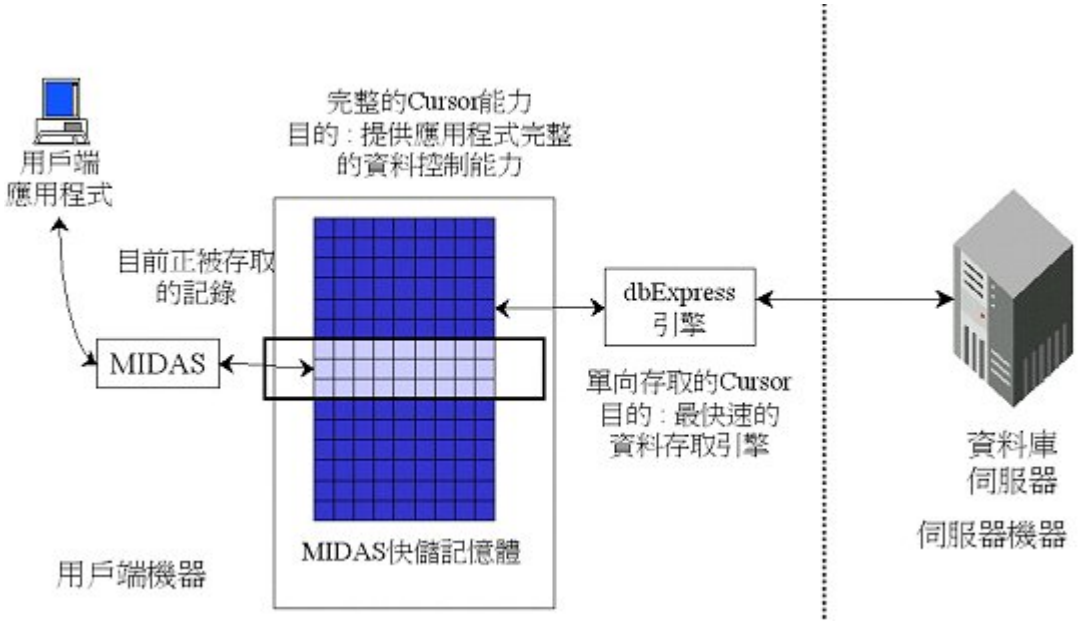



图 17：dbExpress+MIDAS 结构

那么 Provider/Resolve 机制是如何工作的呢？

Provider/Resolve 使用四个控件来进行数据读取和更新等操作。除了前面所说到的两个 dbExpress 控件外，还有两个分别是 TDataSetProvider 和 TClientDataSet。整个 Provider/Resolve 的数据获取和数据更新的工作过程如图 18 和图 19：

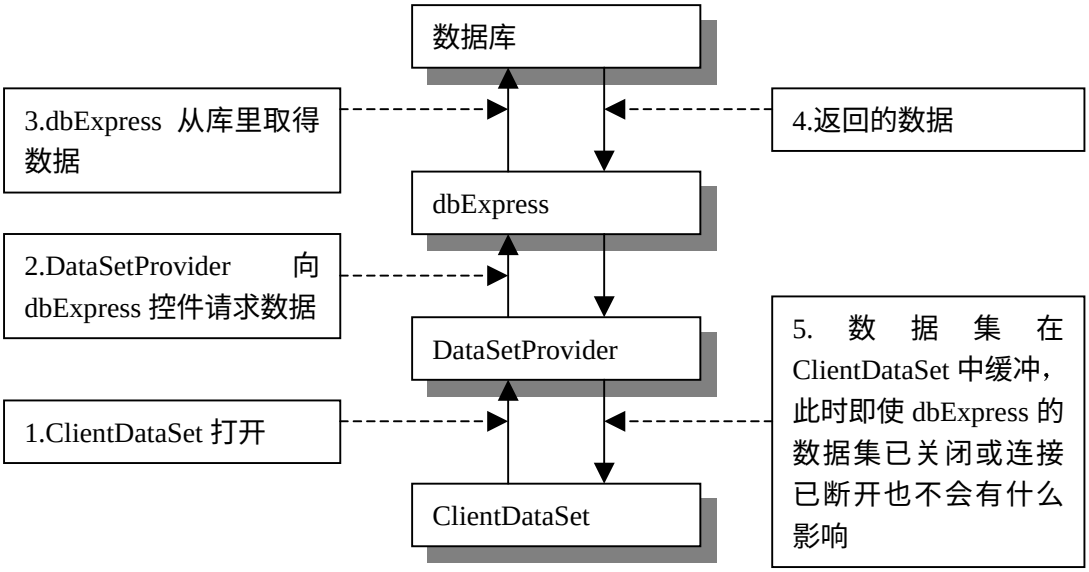



图 18：Provider/Resolve 的数据获取过程

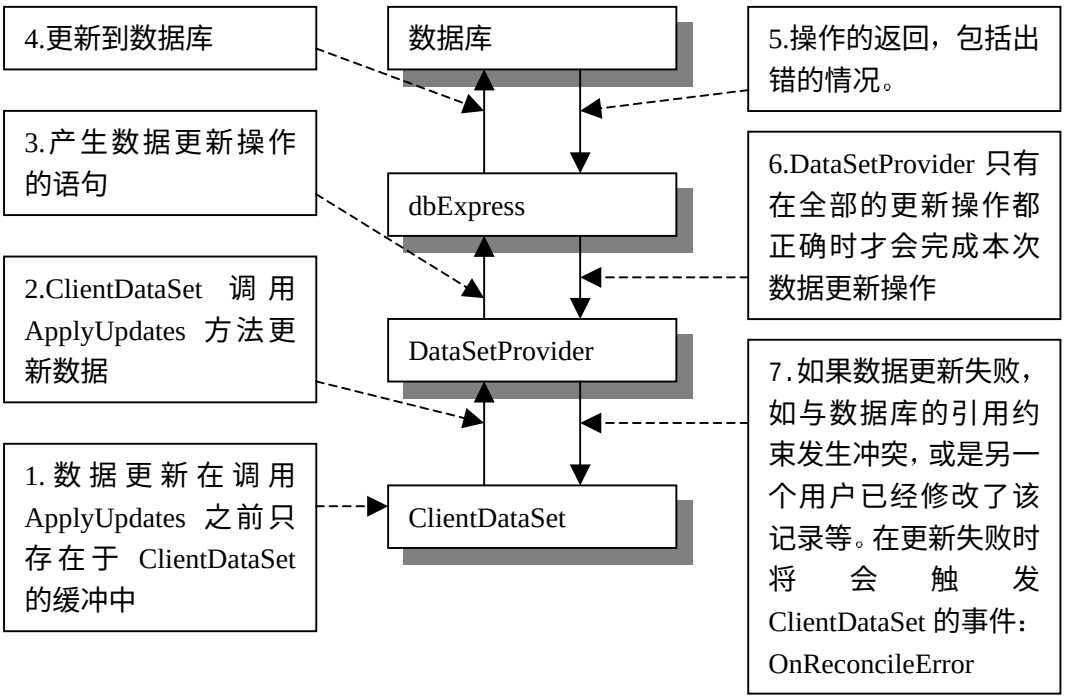



图 19：Provider/Resolve 的数据更新过程

那么，使用 Provider/Resolve 机制有什么好处呢？

1. 数据更改的递交的生命期短

长的递交生命期必然会强制数据库服务器进行数据加锁，这将降低数据的协同性能，消耗服务器资源。通过 Provider/Resolve 架构，在数据读取和更新时的操作都只存在很短的时间，这将大大减少资源的消耗，对于一个很忙的数据库服务器将大大提高性能。

2. 使任意行都可访问

在数据库应用程序中，由多表结合、存储过程和视图返回的行是不能直接编辑的。DataSetProvider 提供三种工具以解决这个问题。

第一是如果记录包含有从单表中得到的字段，例如由存储过程返回的记录，那么唯一的问题就是如何确定表的名字。解决的办法是在 DataSetProvider 中响应一个 OnGetTableName 事件并在其中取得数据表的名字。

第二个可能是有一个多表联合，但只更新其中部分表的一些字段。那么，设置这些字段的 ProviderFlags 属性以声明该字段是将更新的。然后响应一个 OnGetTableName 事件并在其中取得表名，DataSetProvider 会自动创建相应的 SQL 命令的。

如果你对记录的每一个表都要更新，那么要在 DataSetProvider 响应一个 BeforeUpdateRecord 事件，在该事件中为每个表建立 SQL 命令并执行。

3. 快速排序和查找

既然 ClientDataSet 将记录放在内存中，它们能很快地排序。如果内存中的排序还是慢，你可以为 ClientDataSet 的数据在程序设计和运行时建立索引。这些内存中的索引可以让你不使用数据库的索

引而快速地改变记录的显示顺序或是查找记录。

4. 自动摘要信息

   ClientDataSet 能够自动处理你自己定义的诸如 Sum(Price)、Sum(Cost)的复杂的摘要计算。你可以将这些摘要以任意字段或字段的组合进行分组，以得到每组的汇总。你也可以使用 Min、Max、Count、Avg 等计算汇总。

5. 浏览数据的子集

   通过使用 SQL Where 语法的过滤表达式可以使你显示 ClientDataSet 的数据子集而不用在数据库服务器上执行另一个查询。

6. 数据的多并发视图

   克隆 ClientDataSet 游标使你可以同时查看数据集的多个不同视图，也可以查看相同数据的不同排序。

7. 可计算的字段

   你可以在程序设计阶段将可计算字段加入到 ClientDataSet，使可计算字段成为内存中数据集的一部分。由于相关计算是通过已编译的对象化的 Pascal 代码实现的，所以速度很快，比 SQL 命令中声明的的可计算字段或是触发器中的计算复杂得多，而且不会给数据库服务器增加存储和计算的负担。

8. 并不存在的限制

   将记录保存在内存中看起来会限制记录的数目。然而，传统的 C/S 模式的数据库应用程序为了减少网络传输量和服务器的负载也是选择很少的记录数。即使你需要处理一个比较大的记录的集合，比如说 10000 条记录，每条记录占 100 个字节，也只占用一兆内存。就算是你要处理一个很大的记录集合，ClientDataSet 和 DataSetProvider 构件都包括了有关的属性和方法，可以使你分组从内存中读取、编辑和删除记录，然后处理下一组数据。

9. 配置简易

   一个使用 dbExpress 开发的数据库应用程序只要求有两个动态链接库。一个是 dbExpress 驱动程序，比如说 DBEXPINT.DLL，是 InterBase 的驱动程序；另一个是 MIDAS.DLL，是 ClientDataSet 控件的支持库。这减小了应用程序的大小并简化了安装。

10. 容易创建驱动程序

    dbExpress 驱动程序只要求实现在线帮助中所描述的五个接口就可以了。各数据库提供商可以轻松地创建高效稳定的驱动程序。如果你在使用某个少见或是古老的数据库系统而没有相应的驱动程序，你甚至可以自己创建一个。关于这方面的资料可以访问 Borland 的网站，目前已经有一个免费并开放源代码的 ODBC dbExpress 驱动程序发表在 Borland Community 上了。

接下来将用一个例子来介绍如何使用 Provider/Resolve 机制来实现缓冲式双向数据访问。程序的 IDE 如图 20：

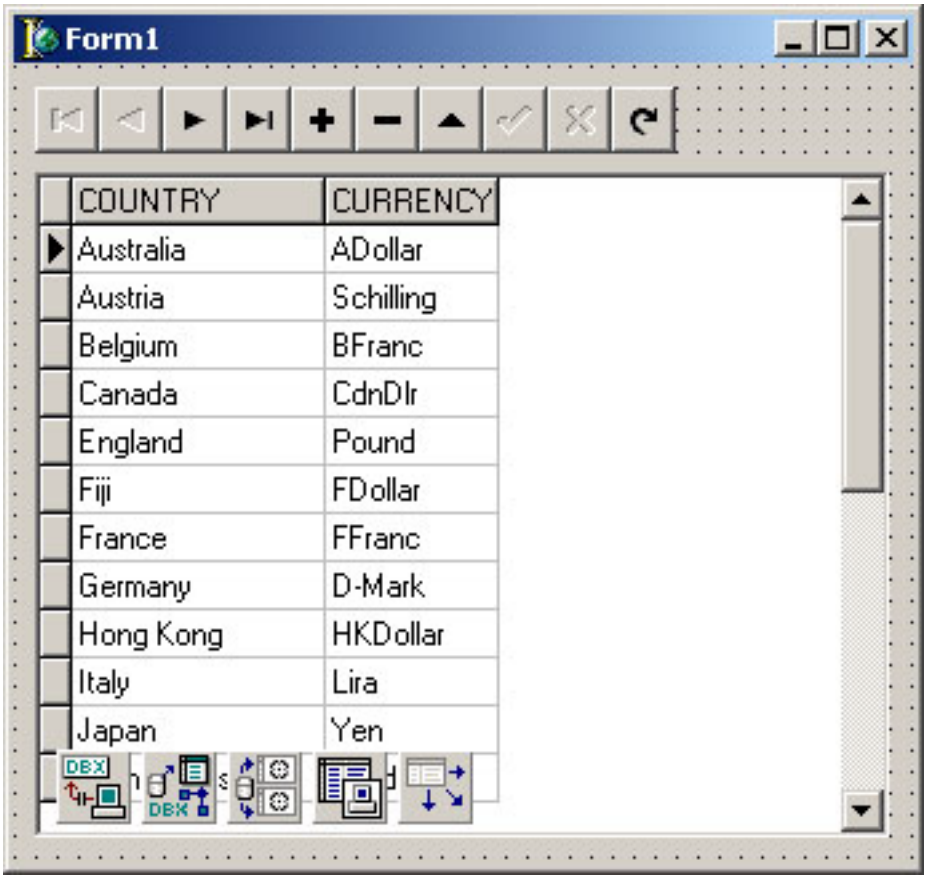



图 20：一个用了 Provider/Resolve 的 dbExpress 应用程序例子

按表 7 依次放上几个控件，并设置其相应的属性。

表 7：一个用 Provider/Resolve 的 dbExpress 例子中的控件及其属性

| 控件 | 要设置的属性 |
|---|---|
| TSQL Connection | ConnectionName := ‘IBLocal’;<br>LoginPrompt := false;<br>Params 中的 Database := <InterBase 的安装路径>\Examples\Database\employee.gdb; |
| TSQL DataSet | SQLConnection := SQLConnection1;<br>CommandText := ‘select COUNTRY, CURRENCY from COUNTRY’; |
| TData SetProvider | DataSet := SQLDataSet1; |
| TClient DataSet | ProviderName := DataSetProvider1; |
| TData Source | DataSet := ClientDataSet1; |
| TDB Navigator | DataSource := DataSource1; |

| | |
|---|---|
| TDBGrid | DataSource := DataSource1; |

完成以上设置后，确定本机的 InterBase 服务已经运行，将 ClientDataSet 的 Active 属性设置为 true，即可如图 20 那样显示出数据来。

运行此程序，即可在 DBGrid 中自由操作数据，包括前后移动，增加，修改，删除等。

### 4.3.3 用 TSQLMonitor 控件监视数据库访问过程

为了监控 dbExpress 的数据库访问过程，了解通过 TSQLConnection 对数据库所作的操作，dbExpress 提供了 TSQLMonitor 控件来跟踪数据库访问过程，此功能在程序调试期间特别有用，此外，借助它还可以分析数据库的访问为优化程序提供依据。

接下来的例子演示了如何在程序中使用 TSQLMonitor。程序的 IDE 界面如图 21：

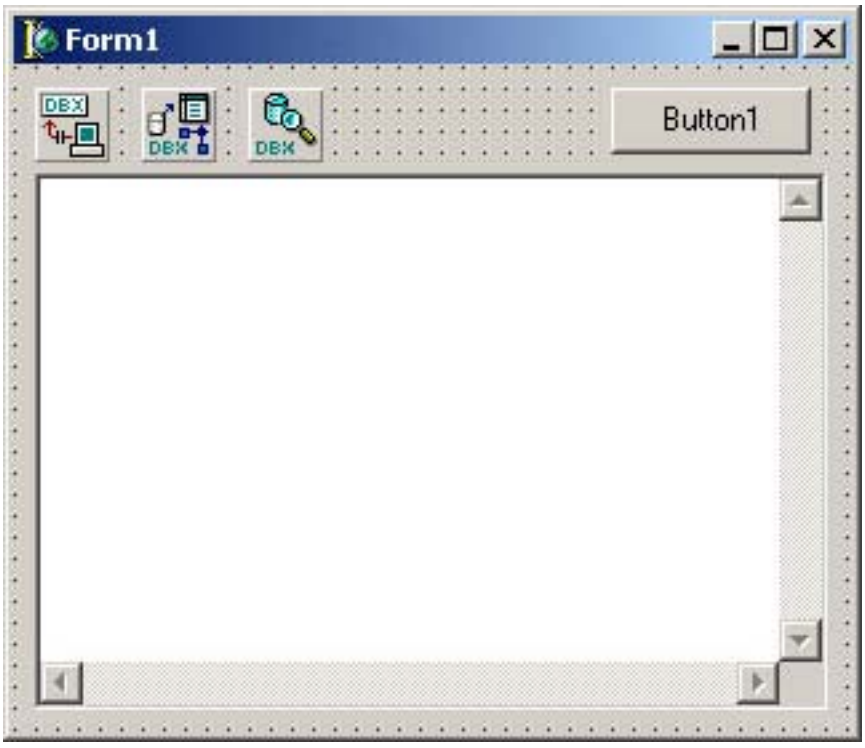



图 21：用 SQLMonitor 监控数据库操作的例子

按表 8 依次放上几个控件，并设置其相应的属性。

表 8：一个用 SQLMonitor 监控数据库操作的例子中的控件及其属性

| 控件 | 要设置的属性 |
|---|---|
| TSQL Connection | ConnectionName := ‘IBLocal’;<br>LoginPrompt := false;<br>Params 中的 Database := <InterBase 的安装路径> |

| | |
|---|---|
| | \Examples\Database\employee.gdb; |
| TSQL<br>DataSet | SQLConnection := SQLConnection1;<br>CommandText := ‘select COUNTRY, CURRENCY from COUNTRY’; |
| TSQL<br>Monitor | Active := true;<br>SQLConnection := SQLConnection1; |
| TButton | 全部默认值 |
| TMemo | Lines := ‘’;<br>ScrollBars := ssBoth; |

完成以上设置后，双击按钮控件，输入下面的事件响应代码：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    SQLConnection1.Open;
    SQLDataSet1.Open;
    SQLDataSet1.Close;
    SQLConnection1.Close;
    Memo1.Lines.AddStrings( SQLMonitor1.TraceList );
end;
```

确定本机的 InterBase 服务已经运行，再运行此程序，按 Button1 即可 Memo1 中看到 SQLConnection 与数据库服务之间的通讯内容，其中包括客户端发出的 SQL 请求语句。Memo1 中的内容如下：

```
INTERBASE - isc_attach_database
INTERBASE - isc_dsql_allocate_statement
INTERBASE - isc_start_transaction
select COUNTRY, CURRENCY from COUNTRY
INTERBASE - isc_dsql_prepare
INTERBASE - isc_dsql_describe_bind
INTERBASE - isc_dsql_execute
INTERBASE - isc_dsql_allocate_statement
SELECT 0, '', '', A.RDB$RELATION_NAME, A.RDB$INDEX_NAME,
B.RDB$FIELD_NAME, B.RDB$FIELD_POSITION, '', 0, A.RDB$INDEX_TYPE, '',
A.RDB$UNIQUE_FLAG, C.RDB$CONSTRAINT_NAME,
C.RDB$CONSTRAINT_TYPE FROM RDB$INDICES A,
RDB$INDEX_SEGMENTS B FULL OUTER JOIN
RDB$RELATION_CONSTRAINTS C ON A.RDB$RELATION_NAME =
C.RDB$RELATION_NAME AND C.RDB$CONSTRAINT_TYPE = 'PRIMARY
KEY' WHERE (A.RDB$SYSTEM_FLAG <> 1 OR A.RDB$SYSTEM_FLAG IS
NULL) AND (A.RDB$INDEX_NAME = B.RDB$INDEX_NAME) AND
(A.RDB$RELATION_NAME = UPPER('COUNTRY')) ORDER BY
RDB$INDEX_NAME
```

INTERBASE - isc_dsql_prepare
INTERBASE - isc_dsql_describe_bind
INTERBASE - isc_dsql_execute
INTERBASE - isc_dsql_fetch
INTERBASE - isc_dsql_fetch
INTERBASE - isc_commit_retaining
INTERBASE - isc_dsql_free_statement
INTERBASE - isc_dsql_free_statement
INTERBASE - isc_dsql_fetch
INTERBASE - isc_commit_retaining
INTERBASE - isc_dsql_free_statement
INTERBASE - isc_dsql_free_statement

程序中只是连接到一个数据库，并打开一个数据集，然后关闭，但实际上需要做这么多的工作。其运行结果如图 22 所示：

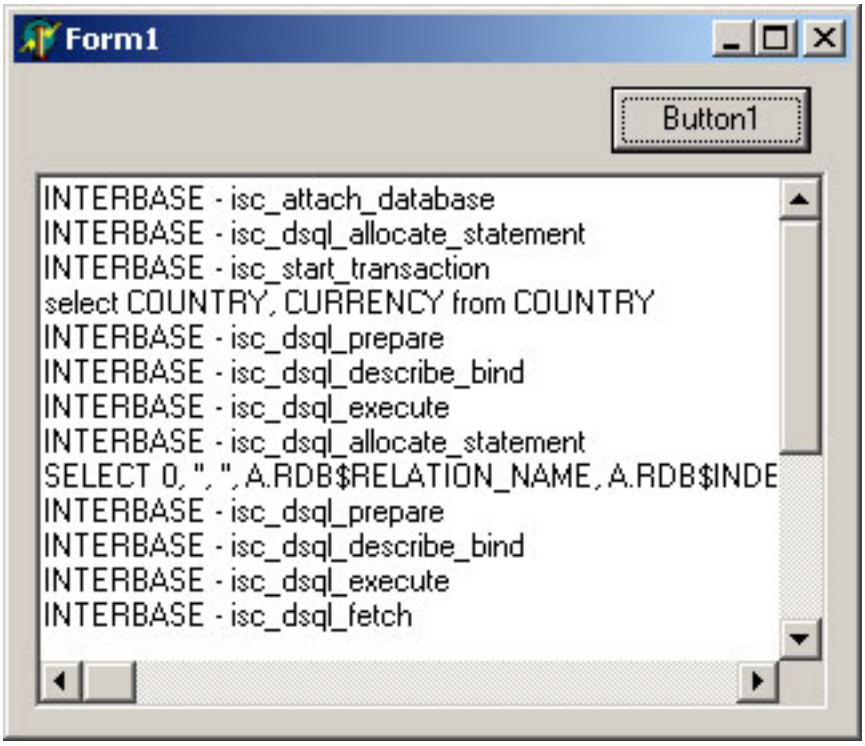



图 22：用 SQLMonitor 监控数据库操作的例子的执行结果

### 4.3.4 dbExpress 的性能

因为 dbExpress 的核心是由几个简单的接口组成，这些接口是：

- 跟数据库系统沟通的 ISQLDriver；
- 连接数据库的 ISQLConnection；
- 发送 SQL 命令的 ISQLCommand；
- 控制游标的 ISQLCursor；
- 存取数据库 MetaData（元数据）的 ISQLMetaData

这些接口定义的目标就是简易，有效率，这跟 Java 的 JDBC 的观念非常类似，但 Borland 又提供了 MIDAS/DataSnap 的 Provider/Resolve 机制，这又比 JDBC 要强得多。虽然 dbExpress 是一项新的数据库访问技术，但是它的执行速度和老资格的 BDE 相比丝毫不逊色，甚至还要好一些。

下表 8 是 BDE 与 dbExpress 在各方面的性能比较（此数据来源于 Borland 台湾分公司的工程师、著名 DELPHI 专家—李维的文章）：

表 8：BDE 与 dbExpress 的性能比较

| | BDE | 优化的 BDE | dbExpress | 优化的 dbExpress |
|---|---|---|---|---|
| 连接数据库 | 1.831 | | 1.467 | |
| 增加 10 条记录 | 0.036 | | 0.052 | 0.047 |
| 增加 100 条 | 0.342 | | 0.334 | 0.206 |
| 增加 1000 条 | 3.421 | | 3.186 | 1.190 |
| 增加 2000 条 | 6.732 | | 6.514 | 2.686 |
| 增加 10000 条 | 36.109 | | 37.992 | 17.472 |
| 随机数据查询 | 4.826 | 3.171 | 1.215 | |

图 23 是用 BDE 和 dbExpress 分别连接 InterBase 时所需时间的比较，从表 8 和图 23 中可以看出 dbExpress 比 BDE 还要好些：

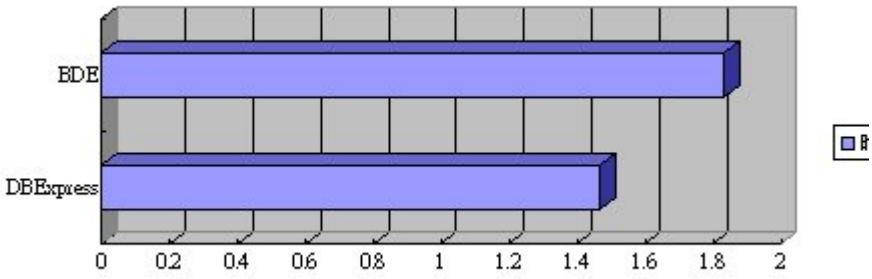



图 23：BDE 与 dbExpress 连接 InterBase 所需时间的比较

图 24 是用一些测试程序让 BDE 和 dbExpress 随机生成一些数据并更新到数据库中时的性能比较。从图 24 和表 8 的数据来看，dbExpress 与 BDE 相差不多。

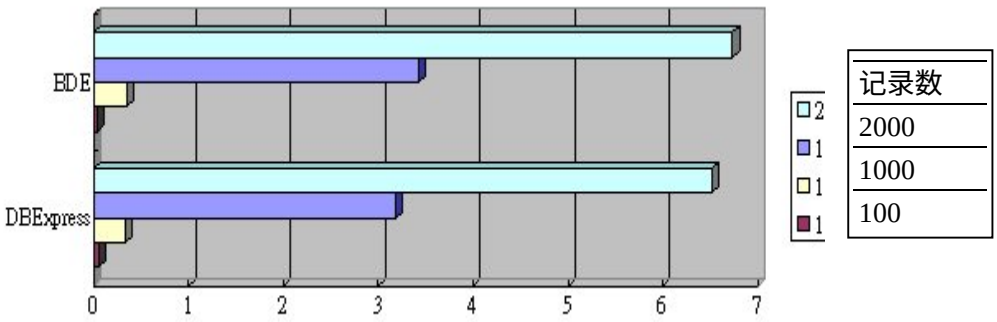
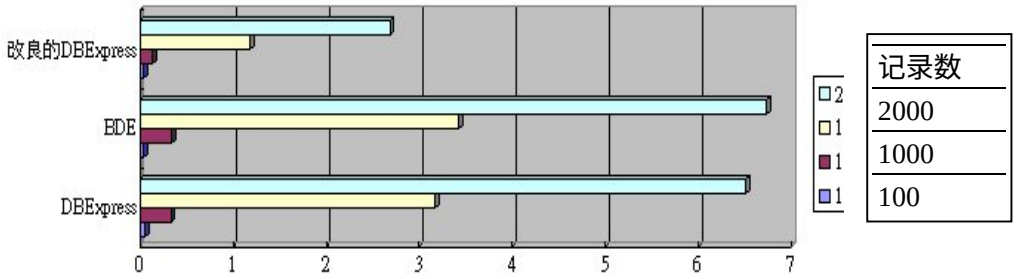



图 24：BDE 与 dbExpress 插入数据所需时间的比较

图 25 是对 dbExpress 进行一些优化后的性能比较。这是 dbExpress 更吸引




人的地方，优化过后它几乎可以以闪电般的速度处理数据。从图 25 和表 8 的数据中可以看出，优化后的 dbExpress 比优化前差不多快了 3 倍，远远超过 BDE。

图 25：BDE、dbExpress 与优化过的 dbExpress 插入数据所需时间的比较

对数据库应用来说，最多的操作应该是数据查询，而 dbExpress 除了在数据更新的速度上超过 BDE 以外，在数据查询方面也是大幅领先于 BDE。图 26 是 BDE 与 dbExpress 在查询大量随机数据时的性能比较。在这一方面，dbExpress 也大约是 BDE 的 3 倍快。

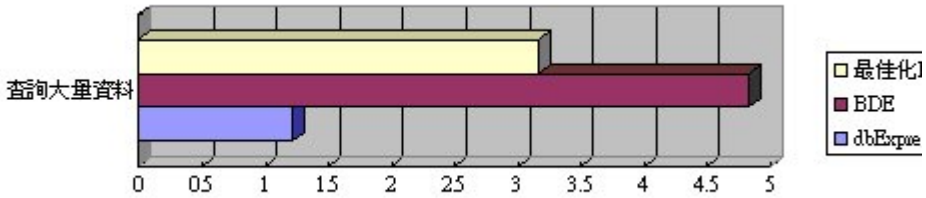



图 26：BDE、优化的 BDE 和 dbExpress 查询数据所需时间的比较

从表 8 和上述各图的数据显示，dbExpress 在数据库操作的各个方面都比经过多年发展的 BDE 要强得多。