内附光

Delphi 5 深度历险

多层分布式数据库实战

王涛 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

Delphi 5 深度历险

多层分布式数据库实战

王 涛 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

多层分布式应用系统是在大型应用中被广泛采纳的结构方式，无论从灵活性、可扩展性还是运行的高效性几个方面上看，都是值得用户去研究的，尤其在未来的大型数据库应用中，会有越来越多的系统采取多层分布式结构。

本书详细介绍了 Delphi 基于多层分布式应用系统的开发，更结合 Delphi 5 中令人振奋的 Internet Express，为更多对电子商务感兴趣的用户提供了必要的参考。本书语言浅显、示例丰富，适用于使用 Delphi 开发多层分布式应用系统的各级用户阅读。

书　　名：多层分布式数据库实战
作　　者：王涛
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
　　http://www.tup.tsinghua.edu.cn
印 刷 者：北京市丰华印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　印张：13.25　字数：314 千字
版　　次：2000 年 8 月第 1 版　2000 年 8 月第 1 次印刷
书　　号：ISBN 7-900625-99-2
印　　数：0001~5000
定　　价：33.00 元（含光盘）

前　言

Delphi 3 的推出，为数据库应用开辟了新的设计空间，多层分布式应用结构开始被国内用户采纳并推广，新版 Delphi 4 对此又赋予了更深的内涵，但系统介绍此类技术的书籍少之又少，使得在这方面的应用还处于一个比较初级的阶段，好在台湾著名的 Inprise 程序设计师李维先生编撰《精通 Delphi 4 实战篇 1》一书面市，才打破这一局面。笔者自 1997 年开始开发基于三层结构的应用，并在拜读李维先生之大作后对 MIDAS 的理解更加深入，所以本书也算是对几年来实际工作和学习心得的总结。

Delphi 5 是在接受 Microsoft 投资后的产物，秉承了 Delphi 4 这个最具战略意义的产品的所有优点，并在与 Microsoft 产品的整合程度及基于 Internet 的应用上加以补充。Delphi 5 的出现，将加快各个领域应用系统的开发速度。在本书完稿之前，笔者拿到了 Delphi 5 的测试版本，在经过对比后，觉得和 Delphi 4 还是有一些差别，所以把大量的例程用 Delphi 5 重新编写组织，在必要的地方，还有针对性地进行了比较，这么做的目的是为了给更多使用 Delphi 4 的用户提供同样的亲和力，在阅读的时候节省比较和分析的时间。

多层分布式应用作为大型数据库应用中最为关注的系统结构，被越来越多的系统设计者采用。Borland 在并购了 Inprise 后，不懈地致力于这方面的研究，最终以 MIDAS(Multi-Tier Diotributed Application Services Suite，多层分布式应用服务套件)为标准，形成了全系列的多层分布式系统结构，提供了大量的解决方案。Internet 上越来越多的第三方支持程序出现，使得 Delphi 在这个领域的地位更加稳固，开发环境更加强壮，应用条件更加成熟。

Delphi 在广大程序员心中的地位是众所周知的，但作为实现多层分布式应用最方便的编程工具，它影响着一代程序设计人员，这应该说是 Borland 的成就。

本书共分 7 章，主要内容如下所示：

第 1 章是 Delphi 多层分布式数据库应用初步，从数据库开发的基本概念和 Delphi 数据库开发的体系结构着手，阐明单层、两层和多层体系结构的演变过程、特征、优缺点和应用范围，进而提出 Delphi 多层分布式数据库应用。从分析 Delphi 中存取数据库的方式和数据集控件体系着手，从数据库存取方式组态这个独特的视角来阐明单层、两层和多层体系结构的演变过程、特征、优缺点和应用范围，进而提出 Delphi 多层分布式数据库应用。

第 2 章为应用程序服务器设计基础，从实例中揭开 Delphi 中应用服务器的神秘面纱，使用户在深入理解应用服务器的本质及特征后，能快速掌握并实做应用服务器。

第 3 章为客户端应用程序设计基础，详述客户端应用的基础，针对不同的中间层协议，逐一举例说明与应用服务器连接的要点和特点。

第 4 章深入介绍 TClientDataSet，作为多层分布式应用的关键控件，TClientDataSet 起

着决定性作用，深入了解 TClientDataSet，更有助于开发出更加灵活高效的应用系统。本章凝聚了笔者对 TClientDataSet 几个月源代码跟踪的宝贵经验和开发体会。

第 5 章主要介绍电子商务的利器——Internet Express，Internet Express 是 Delphi 5 中提供的最有用的功能之一，含有一系列 Internet 应用的解决方案和一整套 XML 的详细介绍，是本章的最大特点。

第 6 章为高级进阶，详细叙述了多层分布式应用中用户比较关心的的问题，如负载平衡、任务调度、数据容错等，并提出了有价值的建议和实现方法。

第 7 章主要介绍多层分布式应用的发布与系统配置，通过大量的实例图片，阐述了各种应用的发布和系统配置。

目 录

第 1 章　Delphi 多层分布式数据库应用初步

几年前，笔者在利用 Delphi 开发数据库应用系统时，碰到了一个非常棘手的问题，近百个客户分散在 6 个城市，用户没有更多的资金来投资优化系统结构，一台服务器要承担客户端每天数十万笔的交易。对于如此庞大的系统来讲，建立分布式系统是十分必要的，这样既可以降低系统维护成本，又可以提升数据库服务器的运行效率。在当时的应用水平下，笔者只能利用 Socket 协议，非常艰苦地做了一个数据库访问服务系统(就是现在的应用程序服务器)，用于完成相当客户数目的数据调度和交互，虽然勉强应付了用户的需要，其结果是让自己陷入了非常尴尬的境地，因为这样做的局限性是显而易见的。

自 Delphi 3.0 版本以来，在数据库应用开发中引入了一个全新的概念——Multi-Tier，这是一个分布式应用的集合解决方案，笔者有幸成为较早接触和应用这项技术的程序员之一。

1.1　数据库应用的结构模型概述

作为较早的数据库应用系统程序员，笔者经历了漫长的从单机到网络结构、从网络到客户/服务器结构的成长阶段。在这个时期，传统的数据库应用模型在程序员的脑子里根深蒂固。Multi-Tier 的出现，使这种模型得以丰富和升华，将数据库应用提高到一个全新的境界。

正确理解 Multi-Tier 的含义，是应用它的重要前提。附录就是在 Borland 公司的网站上找到的一篇系统、准确地定义 Multi-Tier 的文章，这篇文章由浅入深地阐述了 Multi-Tier 的丰富内涵。为保证理解的准确性，笔者将原文附于附录中。

1.1.1　数据库应用的任务切割

任何数据库应用都可以概括为用户界面、商业逻辑和数据访问 3 种基本任务。这 3 种不同的任务既可在一个程序中实现，也可在多个程序中实现。为方便理解，这里分别称之为界面层、逻辑层和数据层。

1.　界面层

界面层是一个人机接口，它提供给用户一个视觉上的认识，完成操作者和应用程序

的交流。用户通过界面层完成获取数据、输入数据、修改数据以及删除数据等一系列操作。与此同时，界面层还包含了一定的安全机制，确保用户根据授权范围，有针对性地控制数据和机密信息。这一层的实现一般都是由程序员来完成的。

2. 逻辑层

逻辑层是界面层和数据层的桥梁，标准形式位于界面层和数据层中间，是数据的接口，它响应界面层的用户请求，执行任务并从数据层提取数据，然后将必要的数据传送给界面层，从而实现界面层与数据层之间的交互。

3. 数据层

数据层一般用于实现数据存取和操作管理工作，有些还负责维护数据的完整性和安全性。它响应逻辑层的请求，产生数据请求结果，然后逻辑层对此结果进行有针对性的处理。这一层通常由大型数据库服务器实现，如 Oracle 、DB2、Sybase 以及 MS SQL Server 等，也可以由本地数据库实现，如 FoxBASE 、Paradox 等。

1.1.2 单层、两层和三层的数据库应用

进行到这里，就该引入 Multi-Tier 的概念。到底什么是 Multi-Tier 呢？不妨先针对上面的叙述，狭义地划分一下单层、两层和多层结构。

从表象和实质上讲，单层结构是将界面层、逻辑层以及数据层合并在一起，并运行于同一台计算机上，就是以前所谓的单机数据库应用系统。最典型的应用是基于 FoxBASE、Paradox 等小型数据库的。笔者刚刚接触 Delphi 1.0 时，正是看好了其开发单层应用的综合能力。

两层结构有两种：一种是将界面层和逻辑层合为一层来实现客户端，用于处理与用户的数据交互，并通过逻辑层完成数据与数据库的交互，数据层是另外一层，用于实现数据库服务，根据客户端的需求进行必要的数据操作，这种结构通常称为客户/服务器结构；另一种是将逻辑层和数据层合为一层，直接完成复杂的数据操作，界面层是另一层，实现用户的数据交互，这种结构通常称为瘦客户/服务器结构。

三层结构是 Multi-Tier 的最基本方式，本书以后的讨论都将以此为基础，即所提到的大部分多层结构例子都将以三层为代表。这种结构最重要的一点就是将这几层分离处理，首先分离的是应用程序，将界面层、逻辑层、数据层分别用不同的应用程序来完成，客户机上运行的是简单的应用程序，完成用户的数据交互。应用程序服务器 AppServer 用于实现逻辑层的功能，接受来自客户机的数据请求并提交数据库服务器端，然后将数据请求的结果反馈回客户机，数据库服务器则根据应用程序服务器的请求进行数据操作。在此基础上，相应地分离运行应用程序的设备，以形成标准的三层结构应用。在特定的环境下，可以建立多个 AppServer，以分解逻辑层的功能，这就形成了 Multi-Tier 结构。在实际中会把负载平衡独立出来，建立一个负载调度服务器，使得结构更加清晰合理，系统中的数据阻塞得到进一步缓解。当然，Multi-Tier 的应用不是一成不变的，在实际应

用中可以有许多方案分解逻辑层。

1.1.3　迎接多层分布式应用的挑战

在客户/服务器结构开始普及时，的确让所有数据库应用程序员振奋不已，但几经风霜后，这种结构的弊端又开始严重影响着大型数据库应用系统的整合性能和执行效能。主要体现在以下几个方面：

- 执行效率无法满足日益膨胀的客户数据请求。这是一个关于系统伸缩能力的问题，相对于数据库服务器而言，客户端基本上是一种有状态的连接，客户端数量的增加，会造成数据库服务器端执行效率的下降，且在吞噬网络资源的同时，数据库服务器端的资源也在大量耗散。
- 过于庞大的系统在负载平衡能力上显得力不从心。尽管目前许多大型的数据库支持分布式应用，但仍然需要客户端应用来实现负载能力的均衡，当数据的连接成为系统负担时，负载平衡问题就显得十分重要了。
- 维护成本偏高。一个标准的客户/服务器应用，其维护成本往往与单个计算机设备的维护成本不成比例，为运行系统而投入的资金，随着应用的不断深入而迅速增加。
- 客户端硬件投资比例大。在经过若干次系统升级后，客户端的投资开始突出。为使系统的运行效率更高，大规模地改善客户端性能是一种行之有效的方法。这意味着大量资金将流向硬件投资。
- 系统扩充工作量大。从系统的开发开始，系统的工作量就以客户机为主，一切设计都是围绕客户机展开的。它同时兼顾着界面层和逻辑层双重任务，这两个任务的整合体需要在一台计算机上安全稳定地运行。每一次系统的扩充，不仅要考虑系统的先进性，更要考虑它的兼容性。

而这些问题在 Multi-Tier 结构下都得到了完善和提升，正是因为这些功能和性能的改善，使得 Multi-Tier 作为新的数据库应用结构，被越来越多的开发商支持和应用推广。以下是 Multi-Tier 结构的主要特点：

- 多层结构属于瘦客户模式。从某种意义上讲，瘦客户端是一种趋势，它象征着系统应用更加趋于成熟、稳定和简洁，更主要的是，客户端的应用变得非常轻松，大大提高了系统的工作效能。
- 多层结构可以更好地支持分布式计算环境。许多数据库服务器都支持分布计算，而多层结构可以把这种特性发挥到极至。
- 多层结构可以实现更好的安全性。对于所有的数据库应用系统而言，其系统安全性都被放在很重要的位置上，多层结构采用安全认证多级划分的分布式安全管理机制，使系统和数据得到更加安全的保护。
- 更好地支持协同开发，有利于开发大型、复杂的系统。随着我国软件开发水平已经逐渐赶上和超过西方科技发达国家，团队开发和协同开发方式被越来越多的软件开发商所使用，参与开发的各个分支在一个作业指导书(设计方案)的约

束下，可以实现背靠背地全封闭编程。而传统系统结构利用这种开发方式就会显得难以控制。

多层分布式应用的主要目的之一，是创建容纳访问数据所需规则和逻辑的所有服务器，建立一个复杂的服务器系统，使各服务器协同处理，为客户的数据操作提供高效服务。这种方法将网络转变为一种资源，资源内部是可以代表企业整体结构的智能对象，而不是一个简单的传输媒介。为了减小客户端应用程序所占空间，需将智能对象放在中间层。

多层分布式应用的另一个优点是，降低网络资源负担。客户端在访问服务器时，大量的数据可以在客户端生成，这样就减少了数据从服务器上下载的机会，从而增加、编辑、删除各个层次的记录不会增加网络的负担。

对于数据库系统而言，系统的约束一直是非常重要的环节之一，基于对层分布式的数据库应用可以很好地解决这个问题。当数据从服务器下载时，与数据同时下载的是一系列强制赋予的系统约束，这些约束包含了用户的身份验证和有效的安全数据。

公文包模型是多层分布式应用的一种特殊形态。通过这种模型，客户端可以从网络上断开，但仍保持访问数据能力。事实上，这种模型的主要工作过程是这样的：客户端有一个数据库的子集，和服务器断开连接时，客户端程序访问的是本地数据；客户端重新进入网络后，本地的数据将针对服务器中的数据进行复制和更新，这就保证了数据的一致性。

1.2　Delphi 数据库开发概述

Delphi 数据库应用程序的一般模式为：在窗体(Form)上用一般界面控件、数据敏感控件(Data Aware)和设计用户界面来表现数据；在数据模块(Data Module)中用数据存取控件(Data Access)访问数据库中的数据以控制数据，如图 1.1 所示。可以看出，在 Delphi 中，数据库应用设计模式正好符合前面讨论的数据库应用任务切割。

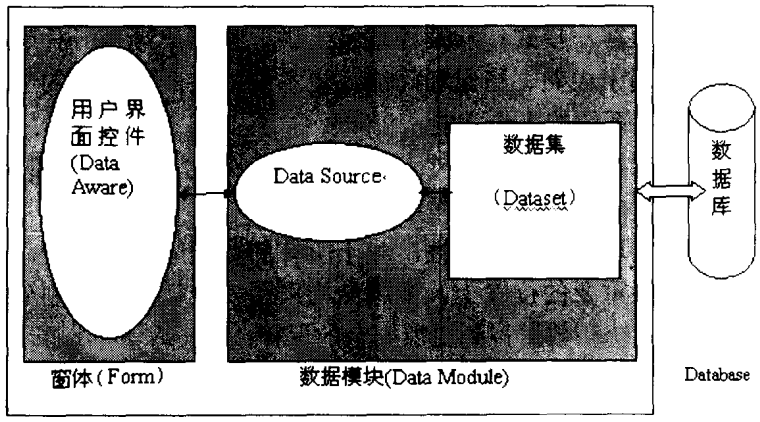

图1.1　Delphi数据库应用程序的一般模式

1.2.1　数据库存取方式

在 Delphi 中，开发应用存取数据库的方式有如下 3 种。

1.　通过BDE存取数据库

使用 Delphi 的程序员都知道，BDE 是 Borland 公司独有的通用数据库引擎，其功能绝不亚于 MS ODBC。在早期的数据库应用中，大都利用 BDE 来完成和数据库的接口。作为 Delphi 的主要数据库存取方式，它的应用十分广泛。客户应用程序通过 BDE 直接访问数据库的应用程序类型，称为基于 BDE 的数据库应用(BDE-based Applications)。

在使用 BDE 时，最令用户头痛的莫过于程序的分发，每一个客户端都需要安装配置 BDE，这也给软件的维护带来了较大的麻烦。

2.　通过DBClient.DLL存取CDS格式数据库(以下简称通过DBClient存取)

DBClient.DLL 是 Borland 公司在 Delphi 3 中引进的一个动态链接库。通过这个动态链接库，可以存取 Delphi 的一种专用数据库格式——CDS 数据库格式。这种方式摆脱 BDE 的牵制，利用 DBClient.DLL 完成对专用数据库格式的存取操作，且基于这种方式的应用程序在分发和维护时显得比较简单。事实上，DBClient.DLL 是 Multi-Tier 应用的结构支持部件，CDS 数据库只是这个部件的功能之一，它难以胜任类似于 SQL 的结构化查询，所以在大型应用中，其地位不是那么明显。不过，在小型桌面数据库应用中，它具有很强的生命力，它的 Filter 功能几乎可以与 SQL 媲美，这一点是值得称道的。对于 CDS 数据库的应用，将在第 4 章专门探讨。通过 DBClient.DLL 存取 CDS 格式数据库的应用程序称为 Flat-file Database Application。

这种数据库应用程序的运行只需很少的附带程序(DBClient.DLL)，通常称为“瘦”应用。

在这个书稿全面完工前，笔者得到了刚刚发布的 Delphi 5，于是不得不对本书中大部分章节进行适应性调整，增加了许多 Delphi 5 与本书相关的新功能，以后本书中凡涉及到 Delphi 5 的内容，都将有特殊标记。

Delphi 5

DBClient.DLL 在 Delphi 5 中已经不存在了，取而代之的是功能更加强大、集成了所有多层分布应用结构支持的 MIDAS.DLL。

3.　通过第三方工具

随着 Delphi 应用的不断深入，第三方数据库存取工具层出不穷，这就从某种意义上支持了 Delphi 的数据库开发、应用和推广，但它们并非本书讨论的重点，所以本书只着重讨论前面两种方式。

1.2.2 Delphi 中数据库存取方式的组态与单层、两层和三层数据库应用

仅从单层、两层以及三层的定义上看，是很难深刻理解它们的异同点及优缺点的下面就从数据库存取方式的组态方面来说明这个问题。Delphi 自带的 BDE 和 DBClient 数据库存取方式通常有如下 3 种组合：

1. 直接用BDE存取

用 Delphi 开发的普通桌面数据库应用程序，大都是通过 BDE 访问本地文件型数据库，桌面、逻辑、数据库在一个应用中完成并运行于一台计算机上，符合单层结构的定义，所以这是单层结构的应用。

最常用的客户/服务器结构，通过 BDE 来访问服务器型数据库，界面层和逻辑层在客户端的一个应用里运行，数据库在数据库服务器上运行，符合两层结构的定义，属于两层结构的应用。

2. 直接用DBClient存取

现在用户做小型桌面数据库应用时，比较喜欢通过 DBClient 访问本地 CDS 文件型数据库，尽管它利用 Delphi 多层结构支持部件来实现数据库的访问，但仍然符合单层结构的定义，也是单层结构的应用。

3. 用BDE和DBClient组合分阶段存取

通过这种方式存取数据库有两个阶段：一个阶段是通过 BDE 取得数据库中的数据，另一个阶段是由 DBClient 取得 BDE 中的数据。如果这两个阶段分别在不同的应用程序(Application)中完成，则无论这两个阶段的应用程序是运行于一台计算机上还是分布在不同的计算机中，构成的都是典型的三层数据库应用。当然，还要真正发挥三层结构的优势，使这两个阶段的应用程序分布在不同的计算机中，为应用程序带来极大的伸缩性。后面还将详细解释这个问题。

不难发现，“1+2=3”的简单原理在这里同样适用，一个基于 DBClient 的单层应用加上一个基于 BDE 的两层应用正好可以构成一个三层应用。当然，用户只是可以这样理解，但千万不要认为“3”一定等于“1+2”。

1.2.3 Delphi 的数据集控件

数据集(Dataset)是应用程序存取数据库中数据的代表，是在 Delphi 中开发数据库应用程序的核心部件，在设计数据库应用程序中占有极为重要的位置。不同的数据库存取方式取决于不同的数据集控件。如图 1.2 所示为 Delphi 数据集的继承关系。

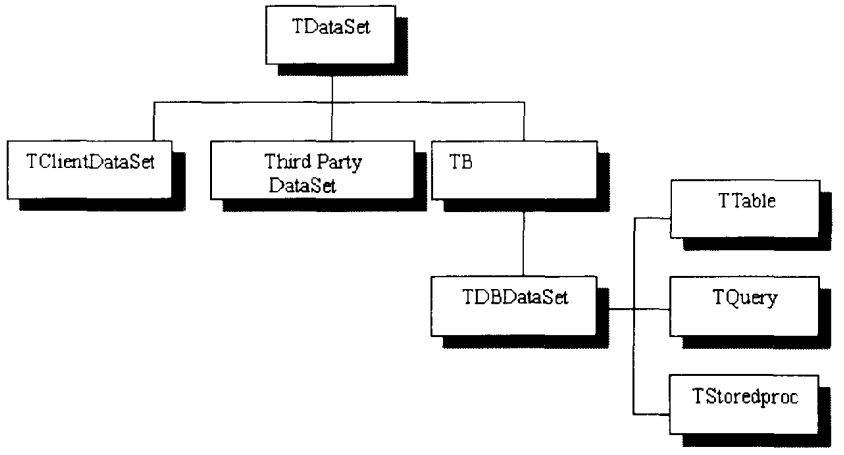

图1.2　Delphi数据集的继承关系

从图 1.2 可以看出，常用的数据集控件(如 TTable、TQuery、TStoredproc 等)都是由 TDBDataSet 继承而来的，而 TDataSet 则是所有数据集控件的抽象基类，这个基类的大部分方法是虚拟抽象的。一方面这些方法可以被派生类重载，另一方面这些方法只有声明，没有定义。派生类必须定义后才可以调用这些方法，不同的派生类有不同的定义。由于其包含抽象的方法，使得它的实例无法直接创建。

关于 3 个数据集的讨论不属于本书的范畴，这里就不再深入阐述了。

1.2.4　Delphi 中典型的单层、两层和三层设计模式

为了更加详细地阐述单层、两层和三层数据库应用系统的结构，本书简单地做了两个小程序来进行说明。

- 单层应用

一般数据库应用从数据模块开始，在数据模块中放置需要的 Table 控件和 DataSource 控件，以实现与数据库的连接。为了获得数据库的连接状态，首先应该在 Main 单元中加入与数据模块的关联关系，通常用 uses DBS(数据模块单元名称)。然后在主窗体中放置一个 DBGrid 和 DBNavigator 控件，并建立与数据模块的连接，编译并运行创建的程序，从而实现一个简单的数据库应用程序。

很多人在做简单应用时，习惯把 Table 控件和 DataSource 控件放在窗体里，虽然这样更简洁，但笔者不提倡这样做，因为在将来扩充的时候会带来一些不必要的麻烦。在数据模块中创建数据关联控件是比较明智的做法。

下面是创建程序的基本步骤和源代码。在本书配套光盘中的 CH1\EX1 文件中包含这个程序的所有代码。

(1)　单击 File | New 命令新建数据模块，弹出的对话框如图 1.3 所示。

(2)　在图 1.3 选中数据模块——Data Modules，然后单击 OK 按钮。接下来在数据模块中放置数据关联控件，如图 1.4 所示。

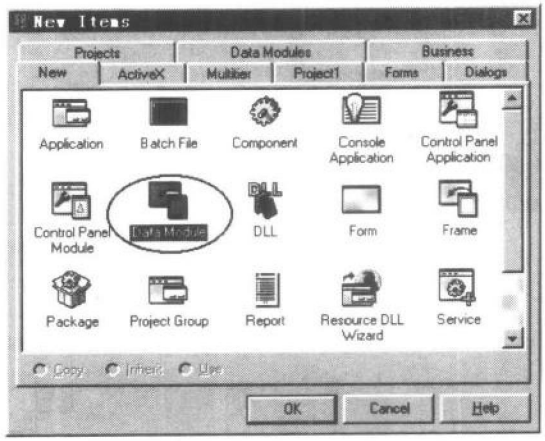

图1.3　新建对话框

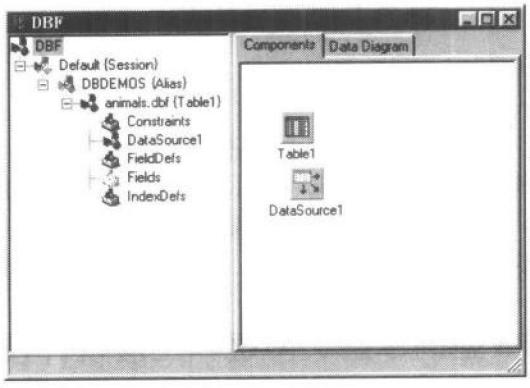

图1.4　在数据模块中放置数据关联控件

(3) 在数据模块中放置数据关联控件，使 Table 指向数据集，DataSource 指向对应的 Table。然后将 Table 的 Active 属性设为 True，如图 1.5 所示，再打开数据库。

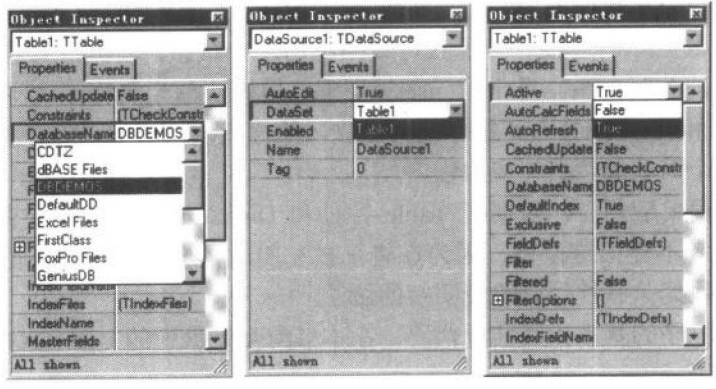

图1.5　Table指向数据集，DataSource指向对应的Table

(4) 在主窗体中放置 DBGrid 和 DBNavigator 控件，并设置其关联关系，如图 1.6 所示。