第16章 用Delphi创建多层数据库程序

- 多层数据库结构是在客户/服务器结构基础上出现的一种新型的数据库技术。作为数据库编程中一项新兴的技术，多层数据库具有它自身的优点。

  - 16.1 多层数据库技术概述
  - 16.2 MIDAS技术
  - 16.3 创建一个多层数据库应用程序的基本过程

16.1 多层数据库技术概述

- 多层数据库技术是在单层数据库（即本地数据库）技术、双层数据库（即客户/服务器结构）技术基础上发展起来的。首先介绍单层、双层数据库技术，然后介绍使用多层数据库的优点，接下来介绍多层数据库编程中使用的技术。

  - 16.1.1 单/双层数据库程序
  - 16.1.2 多层数据库应用程序简介
  - 16.1.3 采用多层数据库结构的优点

16.1.1 单/双层数据库程序

- 单层数据库即本地数据库，双层数据库即客户/服务器结构。单/双层数据库应用程序通过用户接口管理数据信息，因此在没有别的应用程序共享数据信息时使用单/双层数据库结构是比较合适的。即使有别的应用程序共享数据信息，只要这个数据库的规模比较小，单/双层的数据库结构仍然是合适的。

- 掌握单/双层数据库应用程序的编写对于以后掌握多层数据库应用程序的编写方法也是大有裨益的。Delphi中有两种单层的数据库应用程序：一种是基于BDE的数据库应用程序，另一种是FLAT-FILE类型的数据库应用程序。

16.1.2 多层数据库应用程序简介

- 多层数据库应用程序可以分成几个部分运行在不同的机器上。使用多层数据库结构的应用程序可以通过局域网甚至Internet和其他用户共事数据、网络信息等。这就带大许多好处，比如集中的信息处理机制、简化了的客户端程序。
- 在比较复杂的多层数据库应用程序中，客户端程序和远程数据库服务器往往还有附加的数据操作设备。比如可以有一个保证数据安全的设备专门用于管理Internet事务，可以有一个数据连接设备专门用于与桌面数据库的连接。
- Delphi之所以能够支持多层数据库结构主要是依靠MIDAS技术。MIDAS是Multitier Distributed Application Services Suite的简写，即多层分布式应用程序服务组件。掌据了MIDAS技术以及运用MIDAS创建的多层数据库应用程序的结构特点，就能创建多层数据库应用程序，就能根据需要创建附加的数据服务器层。

16.1.3 采用多层数据库结构的优点

- 采用多层数据库结构就是要把一个数据库应用程序分解成几个逻辑部分。客户端程序集中处理数据显示和用户与数据之间的交互作用，而不涉及数据的存储过程、数据的维护等等。应用程序服务器（即中间层）协调各个用户之间的请求，并且掌握着数据集定义的全部细节以及和远程数据库服务器进行数据通信。

- 使用多层数据库结构的优点如下：

  ① 将数据处理及通信功能封装在一个共享的中间层里。不同的客户端程序都能访问这个中间层，这样就避免了为每个客户程序复制数据处理功能而产生的冗余。

  ② 缩小了客户端程序的规模。使得客户端程序更容易进行开发。这都是因为不需要安装、配置和维护数据库连接软件，比如BDE（即Borland数据库引擎）。

  ③ 采用分布式的数据处理过程。将一个应用程序要处理的任务分在几台机器上进行处理，从而提高了程序执行的性能。

  ④ 提高了数据的安全性。将不同的数据功能部分封装成一定的中间层，并且授予不同的访问权限，这样就能保证对数据的访问限制。使用Delphi中的MTS或者COBRA技术支持这项功能。

16.2 MIDAS技术

- MIDAS技术提供客户端程序和应用程序服务器与数据信息进行通信的结构。使用MIDAS技术需要DBCLIENT.DLL文件，客户端程序和应用程序服务器都使用这个文件管理对数据的存储过程。MIDAS中包含有SQL资源管理器，有助于数据库管理；同时支持将服务器管理加入到数据字典（可以参考数据库工具中有关数据库字典的章节）中，这样可以对多层数据库应用程序进行各种层次的检查。同时MIDAS技术支持OLE，这样就能在程序中加入基于COM的数据管理服务。

16.2.1 基于MIDAS技术的多层数据库应用程序

- 基于MIDAS技术的多层数据库应用程序执行的主要步骤如下所示：

① 用户启动客户端程序，客户端程序与应用程序服务器产生连接（连接可以在程序设计阶段或者执行期间指定），这是在首先启动应用程序服务器的情况下进行的。这样，客户端从应用程序服务器获得一个数据供应接口（即Provider Interface）。

② 用户通过应用程序服务器提出数据请求。这样的请求可以是针对全部数据的，通过session组件可以实现。

③ 应用程序服务器接收到数据后，首先建立一个有关数据的通信，并且按照一定规则将数据封装打包，然后以数据包的形式将数据传输到客户端。有关数据格式、定义规则的额外信息也存放在数据包里。这种将数据封装成数据包的过程称为数据供应。

16.2.1 基于MIDAS技术的多层数据库应用程序

④ 客户端程序接收到数据包后，进行数据包的分解，然后将数据显示在用户面前。

⑤ 用户通过与客户端程序的交互操作来实现对数据的更新（比如记录的增加、删除、修改等）。这些对数据的修改都被客户端程序存储在一个数据变更日志里。

⑥ 最后客户端程序向应用程序服务器提交对数据的更新作为对用户请求的响应信息。为了提交这些更新信息，客户端程序将这些数据变更日志也封装成相应的数据包，然后传输给应用程序服务器。

⑦ 应用程序服务器将接收到的数据包分解，然后将这些数据更新加入到服务器的事务处理目录中。如果有一条记录不能进行更新，应用程序服务器将用当前数据作为更新后的数据，或者把不能进行更新的数据存储下来。这样的情况一般发生在这些时候：在用户提出请求以后和数据更新以前，其它的应用程序已经对记录进行了修改。这样的提交数据更新信息和保存有问题记录的过程叫做数据更新。

16.2.1 基于MIDAS技术的多层数据库应用程序

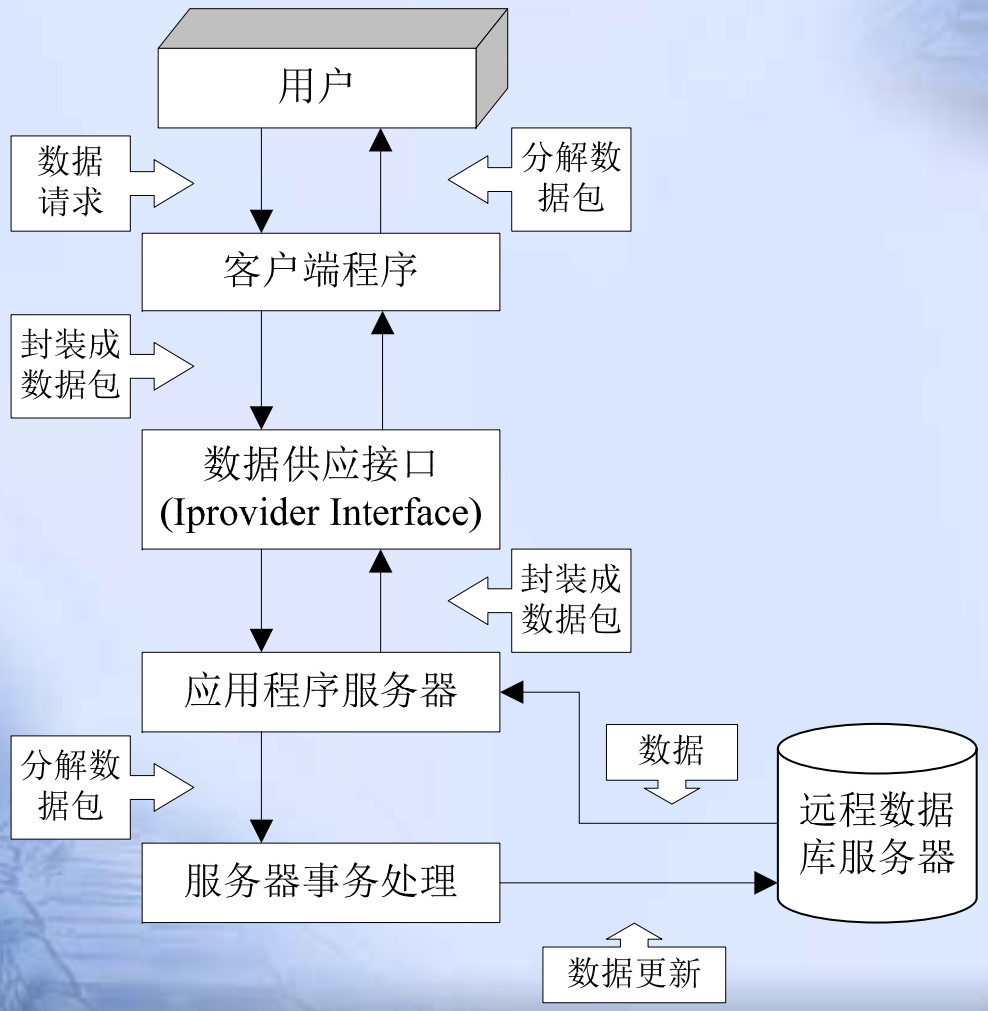
⑧ 应用程序服务器完成数据更新的过程后，它将向客户端返回不能更新的记录以便用户采取更进一步的解决方法。

⑨ 当客户瑞程序的数据更新请求不能得到服务器的满足时，可以通过其它途径进行解决。比如用户可以更改使得数据更新请求不能通过的外部环境，或者干脆取消对数据更新的请求。如果导致用户请求不能通过的外部环境得到更正，用户可以再次提交数据更新请求。

⑩ 用户提出的数据更新请求在应用程序服务器上得到实现。

- 程序执行过程中各个步骤之间的关系如图所示。

16.2.1 基于MIDAS技术的多层数据库应用程序



16.2.2 客户端程序的结构

- 对于终端的用户，多层数据库应用程序中的客户端程序执行的操作看起来和传统双层数据库应用程序一样。在结构上，客户端程序和单层数据库应用程序一样。用户通过数据访问组件执行对数据的操作，数据显示通过客户端的数据集组件来实现。

- 在多层数据库应用程序中客户端的数据集通过应用程序服务器的一个用户接口获得数据，通过这个用户接口还能提交数据更新。在大多数情况里，这个用户接口是IProvider接口。客户端程序通过使用一个通信组件可以获得这样的用户接口。

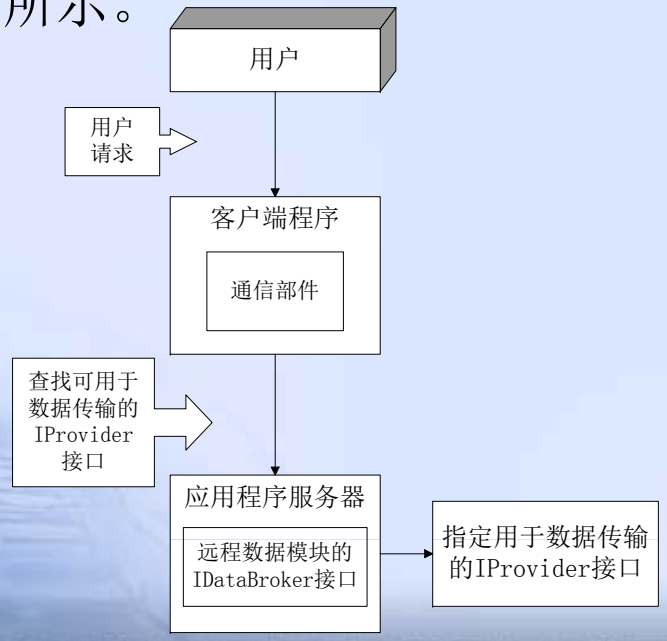
16.2.2 客户端程序的结构

- 通信组件建立起客户端程序和应用程序服务器之间的联系。针对不同的通信协议可以选用不同的通信组件，通信协议和通信组件之间的关系如表所示。

通信组件	通信协议
TDCOMConnection	DCOM协议
TSocketConnection	TCP/IP协议
TOLEnterpriseConnection	RPCs协议
TCorbaConnection	UDP协议

16.2.2 客户端程序的结构

- 对于TRemoteServer和TMIDASConnection组件，具有相互兼容的能力。一旦建立起客户端程序和应用程序服务器之间的联系，通信组件使用应用程序服务器的IDataBroker接口获得远程数据模块中的IProvider接口来进行数据联系（远程数据模块存在于应用程序服务器上）。整个客户端程序执行的过程如图所示。



16.2.3 应用程序服务器的结构

- 应用程序服务器包括一个远程数据模块，这个数据模块的IDataBroker接口用来获得可以用于在客户端程序和应用程序服务器之间传输数据的IProvider接口。

- 通常情况下，在远程数据模块中加入一个数据供应组件（即Provider组件）来适应应用程序服务器对数据集的需要。每个数据供应组件为客户端程序和应用程序服务器之间的数据通信提供接口，具体的作用：

1. 数据供应

  - 接收来自客户端的数据请求，从数据库服务器获得用户需要的数据，封装数据、传输数据到客户端数据集。

16.2.3 应用程序服务器的结构

2. 处理客户请求

- 接收来自客户端数据集需要更新的数据，向数据库提交数据更新．将不能实现的数据更新存入日志，向客户端返回不能进行更新操作的记录信息。
- 对于在应用程序服务器中使用BDE的数据集，如果不使用数据供应组件，程序将自动产生一个。明确的添加数据供应组件可以加强对程序的控制，增加对其它类型数据库的支持。

16.2.4 选择合适的通信协议

- 用于连接客户端程序和应用程序服务器的每—种通信协议都有自己的特点。因此在选择通信协议以前，需要考虑程序用户的数量、程序设计以及开发计划的实施。

- 1. 使用DCOM协议
  - DCOM协议提供最直接的通信方法，不需要在服务器上添加额外的程序。但是因为DCOM协议不包括在通常使用的Windows操作系统中，用户的机器可能不能使用DCOM协议。

16.2.4 选择合适的通信协议

- 2. 使用TCP/IP协议
  - 使用TCP/IP协议可以创建低级别的用户。比如当程序设计者将客户端程序作为一个Active Form分布在网络上，这时并不能确定用户的系统是否支持DCOM协议。TCP/IP协议可以提供一种与服务器最普通、最低级别的连接方式，同时这种方式又是适用而最广的。有别于采用DCOM协议时在远程数据模块上直接接收用户请求，采用TCP/IP协议将使用一个存在于服务器上的程序接收用户请求（如ScktSrvr.exe或者scktsrvc.exe）。使用这种协议数据库应用程序的安全性不如采用其它协议。

16.2.4 选择合适的通信协议

- 3. 使用RPCs协议
  - 使用这种协议时，允许程序设计者使用Business object Broker来代替在客户端对数据的管理。当使用这种协议时，程序设计者必须同时在客户端的系统和服务器的系统安装RPCs协议。
- 4. 使用UDP协议
  - 使用这种协议允许设计者将多层数据库应用程序集成在一个用CORBA标准规范化的环境里。客户端程序和应用程序服务器可以通过采用其它方式创建的CORBA类型客户和服务器来实现相互操作。

16.3 创建一个多层数据库应用程序的基本过程

- 在前而的内容里介绍了多层数据库的结构特点以及使用的基本技术这里将介绍创建一个多层数据库应用程序的基本过程：

① 创建应用程序服务器；

② 注册或者安装应用程序服务器：

③ 创建客户端程序。

16.3.1 创建应用程序服务器

- 创建应用程序服务器和创建绝大多数数据库应用程序的过程基本相同，主要的区别就在于在应用程序服务器中必须包含某种数据供应组件（即Provider组件）。程序设计者可以使用TProvider组件、TDataSetprovider组件或者继承TDBDataSet组件的Provider属性。

- 如果要创建COM类型的自动操作的服务器，需要选用远程数据模块（即Remote Data Module），客户端是采用DCOM、TCP/IP和RPCs协议和这种服务器进行通信的。如果要创建DLL类型的服务器，需要选用MTS数据模块（即MTS Data Module），客户端与服务器进行通信采用的协议和前一种情况相同。

- 如果要创建CORBA类型的服务器，需要选用CORBA数据模块（即CORBA Data Module）。远程数据模块的作用远远超出了数据模块的范围：CORBA远程数据模块在程序中起到了一个CORBA服务器的作用，其它种类的远程数据模块则是起到COM自动操作服务器的作用。

16.3.1 创建应用程序服务器

- 在设置或者执行应用程序服务器时，服务器并不会建立和客户端程序的任何数据通信。这种联系是客户瑞程序使用它的通信组件建立的。一旦建立了应用程序服务器和客户端程序之间的联系，客户端程序会选择一个合适的数据供应接口。所有这些过程都是在程序执行过程中自动生成的，并不需要编写任何的代码。

- 在建立应用程序服务器和客户端程序的数据通信后，这两者之间的接口将可以被程序设计者或者用户利用。数据通信的过程是客户端程序来维护和管理。

- 创建了一个新的远程数据模块后，必须指定服务器怎样响应客户请求。而这取决于对数据模块的属性设置。

16.3.1 创建应用程序服务器

- 远程数据模块必须指定的内容：

Threading Modal取值	含义和用途
Single	选用这种取值后，在一个时刻COM程序只能为一个客户的请求提供服务．这样就不用为多个用户请求相互干扰担心
Apartment	选择这个取值后，COM程序将提供一个实例来响应一个用户请求。在为Apartment线程的库编写代码时，如果程序设计者要使用全局变量或者不包含在数据模块中的对象，必须防止各个线程之间可能发生的冲突
Free	选用这个取值，程序可以在多个线程同时接收用户请求．但是需要确保线程的安全。因为多个用户可以在同一时间访问数据模块，程序设计者必须象对待全局变量一样关注实例的数据。因为这种模式中没有内置支持IProvider接口的线程，不推荐使用这种取值
Both	选择这种取值和Free基本相同，唯一的区别在于：所有对客户端的响应信息对于程序没计者都是连续的

Instancing取值	含义及用途
SingIe Instance	每个客户端的通信过程都执行自己的实例。这个过程说明对应于每个客户端的通信过程，都有自己唯一的实例
Multip1e Instance	程序的一个实例为程序中所有远程数据模块服务。每个远程数据模块都有一个特定的客户端通信过程，但是执行这种过程的空间是共享的
Internal Instancing	只有在客户端程序的代码是通信过程中的一部分时选择这个取值

16.3.2 注册、安装应用程序服务器

- 1. 注册应用程序服务器为自动操作服务器
  - 如果应用程序服务器使用DCOM、TCP/IP和RPCs作为通信协议，服务器将执行自动操作服务器的各项功能，并且必须向ActiveX或者COM服务器进行注册。
  - 自动操作服务器可以分为三种：In-Process服务器、Out-of-Process服务器和Remote服务器。这三种服务器有着各自不同的特点。

16.3.2 注册、安装应用程序服务器

自动操作服务器种类	特点
In-Process服务器	以DLL的形式运行在和客户端同样的程序运行空间里。比如在一个运行在Internet Exploror或者Netscape上的网页其中植入一个ActiveX组件。这个ActiveX组件下载到客户端的机器上，并且在浏览器浏览这个网页时调用这个组件。客户端和In-Process服务器通信直接通过COM接口实现
Out-of-Process服务器（即本地服务器）	以EXE的形式和客户端运行在同一台机器的不同程序运行空间里。比如有一个植入Excel电子数据表的word文档，这就是同时有两个应用程序运行在同一台机器上。本地服务器使用COM和客户端进行通信
Remote服务器	以DLL或者其它形式运行在和客户端不同的机器上。比如一个数据库应用程序连接到网络中另一台机器上的应用程序服务器。在这种情况下，远程服务器和应用程序服务器使用COM或者DCOM接口进行通信

16.3.2 注册、安装应用程序服务器

- 2. 将MTS对象安装到MTS数据包中
  - 如果使用了MTS技术，应用程序服务器将以动态链接库（即DLL）的形式存在。因为所有来自服务器的响应都是通过MTS的代理机制来传输，不需要注册应用程序服务器。但是需要将应用程序服务器安装到MTS的数据包中。

- 3. 注册应用程序服务器的接口
  - 当在应用程序服务器中使用CORBA技术时，需要注册应用程序服务器的接口。如果想要使客户端程序获得与服务器接口的连接，必须将服务器的接口安装到专门设定的接口集中。如果允许在应用程序服务器没有运行时通过客户端程序启动应用程序服务器，必须用OAD（即Object Activation Daemon）对服务器接口进行注册。

16.3.3 创建客户端程序

- 从绝大多数角度看，创建一个多层数据库应用程序和创建一个传统的双层数据库应用程序是类似的。两者之间最主要的区别在于：多层数据库应用程序使用数据通信组件建立到数据库服务器的数据传输管道，使用一个或者多个TClientDataSet组件连接应用程序服务器下的数据接口，客户端中的数据控制组件通过数据集组件连接到相应的客户数据集。