中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

程序設計系列

Visual Basic6.X 程序设计

陈俊源 编著

SQL Server7 应用集成篇

Visual Basic6.X程序设计

——SQL Server7应用集成篇

——SQL Server7应用集成篇

陈俊源 编著

中国铁道出版社

1999年·北京

(京)新登字 063 号

北京市版权局著作权合同登记号：01—1999—1989

图书在版编目（CIP）数据

JS234/03

Visual Basic 6.X 程序设计：SQL Server7 应用集成篇/陈俊源 著.—北京：中国铁道出版社，1999.9

ISBN 7-113-03472-1

Ⅰ.V… Ⅱ.陈… Ⅲ.①Basic 语言-程序设计②服务器-应用软件-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（1999）第 38711 号

书　　名：Visual Basic6.X 程序设计——SQL Server 应用集成篇
作　　者：陈俊源
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
责任编辑：苏　茜
特邀编辑：齐　心
封面设计：新创工作室　冯龙彬
印　　刷：北京兴顺印刷厂
开　　本：787×1092　1/16　印张：20.25　字数：490 千
版　　本：1999 年 8 月第 1 版　1999 年 8 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-03472-1/TP•392
定　　价：33.00 元

出版说明

在过去的这几年间，随着企业彼此间的竞争日趋激烈，借助信息技术来获取更低的成本、更有效率的操作方式发挥企业本身最大的竞争力，以适应快速变迁的商业环境逐渐成为不可避免的趋势。

为能够迎合转变快速的信息技术，越来越多的企业倾向于分散式的管理需求，提供分散式主从结构环境的 Microsoft SQL Server 便成为相当重要的核心角色，这个数据库服务器运用了关系型数据库的技术，同时考虑到高效率数据管理的需求，借助标准化的访问环境，提供了和前端应用环境轻松集成的能力，以适应各种企业组织在不同环境下的操作需求。

作为前端应用程序开发环境的 Visual Basic 6，在数据库应用程序开发领域较以往旧版本有着相当大的变化，不论就数据来源的维护、数据访问技术、网际网络和企业内部 Intranet 应用程序等功能，都有令人耳目一新的极佳表现，配合后端服务器的 SQL Server 数据库系统，则更能显示整个开发界面的独到之处。

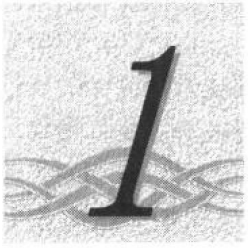
本书内容以主从结构应用集成为原则，分别以数据库的建造与维护、数据访问对象探讨、可视化数据库工具以及 Internet / Intranet 应用等四个主题，引导读者循序渐进地了解 Visual Basic 与 SQL Server 在主从结构应用程序开发方面的强大能力，其中包含了不少精采的专题，例如 DAO、RDO 与 ADO 等对象模型的剖析，以及 Dynamic HTML 应用程序的开发技巧等内容，保证可以让读者一饱眼福。

本书由中国铁道出版社计算机图书项目中心审选，李楠进行了整稿工作，廖康良、孟丽花、颜耳顺、肖志军进行了排版工作。

中国铁道出版社

1999年9月

目录



主从结构系统规划

具有高性能的主从结构应用系统的创建工作，是一门牵涉极广的集成工程，这当中不仅覆盖了有关用户界面开发工具与数据库的设计能力，更由于主从结构分散式数据处理的特性，设计人员必须面对不同的网络软硬件环境而必须对前后端资源运用作妥善的分工。一般而言，系统建造者多半没有足够的时间与资源一一尝试各种方式以找出最佳化匹配，因此如果在系统建造之初便能够具备一些基本的概念，就系统规划程序而言将可获得事半功倍的效果。

本章的内容在于提供有关主从结构系统建造的基本概念与设计原则，协助读者具备足够的判断能力来选择适当的建造方式，借助适当的规划开发出具备高性能的数据库访问应用程序。包含的主题列出如下：

- ✧ 主从结构的内涵
- ✧ 分散式处理的考虑
- ✧ 数据访问的风格

1-1 主从结构的内涵

随着数据的累积与用户数量的增加，大量的数据处理需求便成为大型主机以及区域网络的重大负担。为了让系统能够达到像样的执行效率，主从结构的分工体系逐渐成为数据库访问操作寄望的所在，借助客户端（Client）和服务器端（Server）的工作分配，完成有点复杂又不太复杂的数据处理任务。

分散式处理的需求

常见于学校及研究机构的大型主机系统，提供了多人同时上机的操作环境，每个用户通过监视器和键盘下达命令直接让主机来执行。这种系统虽然有够大的服务主机，但随着上线者人数的增加，造成系统在各个命令间疲于奔命，用户只有排队等着主机按顺序完成所有在线用户所托付的任务。

另外近来风行于企业信息管理的区域网络系统，如当前最普遍的 Novell 网络操作环境，其运作方式便和上述的大型主机有些差异。服务主机此时主要是扮演数据共享的角色，也即

网络上的 PC 都可通过操作系统取用主机本身或其他 PC 的文件资源，而对于命令的执行或数据的处理则依靠该部 PC 自己完成，因此遇到访问大量数据的数据库操作需求时，服务主机也只有在旁边看热闹的份。

上述这两种运作方式，明显地都将工作负荷放在服务器端或客户端 PC 一边上，一旦遇到用户众多或者数据处理量大的情况，系统资源很快便耗尽而拖垮了执行效率。为解决这个问题，分工的概念开始被导入到数据库系统的操作中，借以均衡网络两端的工作负担。

简而言之，分工的方式可以将数据库操作系统安装到服务器端，由服务主机负责数据的处理操作；用户则通过各个客户端的应用程序界面，下达命令执行数据访问的需求。这样分工合作便可避免头重脚轻的现象，这也正是主从结构能够广泛应用于网络系统中的基本内涵。

开放与弹性的精髓

随着各个公司企业的操作体系差异，每个部门常会因为工作特性或者阶段性的需求，而可能具有不同的操作平台与应用程序界面，通过主从结构将可以轻易地集成这些区域网络，使得公司内部 Intranet 形成一个彼此能够相互交流的开放式系统环境。

另一方面，主从结构的技术也使得 Internet 的应用达到无远弗届的境界，例如你我都可以轻易地通过网际网络连接远在太平洋对岸的数据库服务器，迅速从浏览器程序当中取得即时的数据信息。

1-2 分散式处理的考虑

主从结构应用程序开发之初，首先我们要面对的问题将是整个系统的分工方式，考虑到底有哪些工作要交由前端工作站来完成，哪些任务应该在数据库服务器直接处理完毕，甚至还可以另外赋予其他的服务器来一同参与，以增加整体系统执行的效率。下面我们分别针对各种开发结构，讨论每一种结构的特性与创建方式。

数据处理过程

依数据处理的过程来看，主从结构的工作特性可以划分成数据服务（Data Service），商业服务（Business Service）以及操作界面服务（Presentation Service）等三个不同的层次，图 1-1 为整个数据处理过程的示意图，它们所负责的任务详见下图和然后的说明。

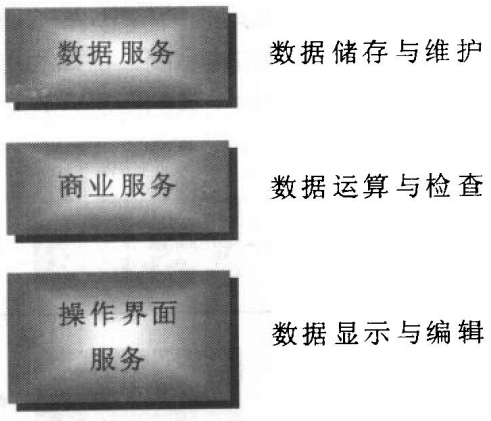


图 1-1 数据处理过程

数据服务

这个部份的工作特性主要在于提供数据储存、对照与数据集成性验证的服务。例如对于一个客户数据库而言，数据服务层次负责存放客户相关数据，并且检查各个相关数据表的对照关系是否正确。

商业服务

这个部份的工作主要是处理各类商业规则的判断，以及商业运作上所牵涉到的数据处理工作。例如在处理客户的刷卡消费数据时，商业服务层次负责检核该笔金额是否在客户信用额度之内的工作。

操作界面服务

这个部份的服务主要负责用户操作界面的处理，包括数据的输入与显示的方式。例如显示特定客户的所有消费纪录，或是运用报表或数据图表显示数据内容等等。

在使用不同的开发工具产生应用程序时，我们可以有许多做法来让前端工作站或后端数据库服务器负责执行这些过程，下面我们就提出四种实际制作的方式分别作介绍。

- 以前端为主的两层式设计
- 以后端为主的两层式设计
- 三层式处理结构
- Internet 处理结构

以前端为主的两层式设计

当我们在规划应用程序的分工结构时，上述这几种系统建造方式的主要差异，在于商业服务层次的工作位置所在。以往传统的设计多半是以两层的方式来规划，也即将商业服务

的运算检核工作安排到前端的工作站上，在这种结构中，后端服务器只是扮演类似 SQL Server 的数据服务角色。当前市面上常见的主从结构系统，例如利用 Visual Basic 或者 Delphi 所开发出来的应用程序，大多数都是属于这种类型的两层式设计，图 1-2 是这种结构的参考示意图。

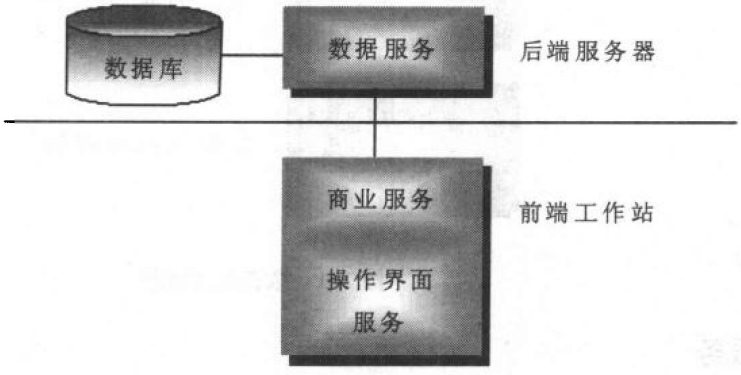


图 1-2　以前端为主的两层式设计

采用这种以前端工作站为主的两层式设计，最大好处在于支持这种结构的开发工具都具备了极佳的工作界面，不论在开发过程或者后面的调试阶段皆有不凡的表现，而且在市场上也充斥许多成熟且稳定的产品，让应用程序开发者能够有较多的选择。

当然这类型的结构也不是毫无缺点，由于数据库服务器存放的数据都必须传送到前端工作站进行处理，造成网络数据流量过大的负荷是其最主要的问题，当网络上用户的人数增加时，数据传送操作所造成的负担也随之加重。

以后端为主的两层式设计

另外一种两层式的结构特性，是以后端服务器为主的应用程序设计，此时商业运作逻辑和数据处理都是在数据库服务器中执行。一般而言，我们可以借助存储过程（Stored Procedure）或触发程序（trigger）的创建，由其执行商业服务所肩负的工作，后面第 4 章内容中将会介绍这两种数据库对象在 SQL Server 的运作特性和建造技巧。图 1-3 是此类型运作结构的示意图。

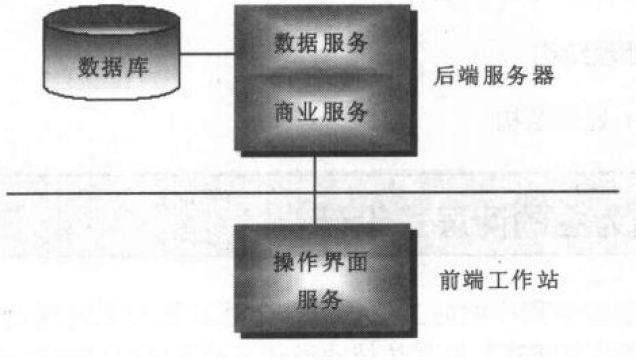


图 1-3　以后端为主的两层式设计

以后端服务器为主的设计方式，由于商业逻辑处理直接在服务器中执行，已经不需要将大量的数据传送到前端工作站，因此可以减少网络间的数据转移动作，高执行性能成为其最大的优势。

这种结构的缺点在于所能够使用的开发工具受到了限制，由于这些存储过程都必须以数据库所支持的语言如 T-SQL 语法来撰写，以往因为开发界面与调试工具的缺乏，设计人员得耗用不少资源在应用程序的反复测试工作上。所幸 Visual Basic 从 5.0 版开始，提供了处理 SQL Server 存储过程的调试程序，让开发人员可以逐步查看 Transact-SQL 的程序代码，例如我们可以通过 T-SQL Debugger 窗口，运用中断点的设置以及查看区域变量的内容逐步查看 T-SQL 语句。

三层式处理结构

在三层式结构的设计中，商业服务是在独立的执行空间中运作，但是并不一定要存放在不同的服务器上，它也可能和数据库服务器运行在同一台电脑上，只是此时这三个层次在执行时都是在不同的执行空间中，例如 Visual Basic 6 便可以在服务器中开发应用程序，负责执行商业服务应有的功能，图 1-4 是这类型结构的示意图。

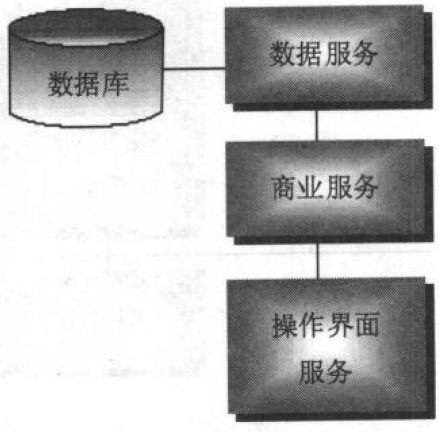


图 1-4 三层式处理应用

建造这种三层式主从结构的好处是可以分别减轻前后端的工作负荷。一方面使数据库服务器只需要与服务器应用程序创建单一连接，从而能够专心地执行其数据处理的工作；另一方面对前端工作站而言，则不需在每一部 PC 中安装访问界面软件，只要负责用户界面的执行动作即可。

Internet 处理结构

Internet 的出现引发了另一股三层式设计的旋风，这种三层式结构将用户界面服务分割到浏览程序（如 Internet Explorer）和 Web 服务器上。Web 服务器主要的工作在于创建用户浏览页面的格式，然后传送到客户端浏览程序中显示出信息画面，并且视需求下载其他

必要的数据或部件。而在 Web 服务器和数据库之间，仍然存在安排商业逻辑的不同选择。

一般在 Internet 上的设计手段，是同时在 Web 服务器上进行用户界面及商业处理的服务，某些产品甚至可以将商业服务工作集成到 Web 服务器的程序中执行，减少彼此程序交替所形成的系统负荷。其中的一个例子是 Microsoft Internet Information Server（IIS）所提供的数据处理服务 ——Microsoft Internet Database Connector（IDC）。IDC 可以连接任何 ODBC 数据来源 —— 当然包括 SQL Server，进行数据库数据的访问处理并产生对应的 HTML 页面，直接送给前端的浏览程序显示，图 1-5 是这类型结构的相关示意图。

Internet 结构由于可以通过浏览程序进行数据访问的工作，因此最大好处在于提供了跨操作平台的能力。也即同样一个应用程序，可以通过浏览程序在 Windows、Apple Macintosh、OS/2 以及 UNIX 等操作系统上执行，所有前端工作站所必须具备的功能都可以通过标准的 Web 浏览程序来完成。此外容易管理则是 Internet 处理结构的另一个特点，因为一旦我们更新了 Web 服务器上的设计，所有连接服务器的前端工作站也将同步完成更新的操作。

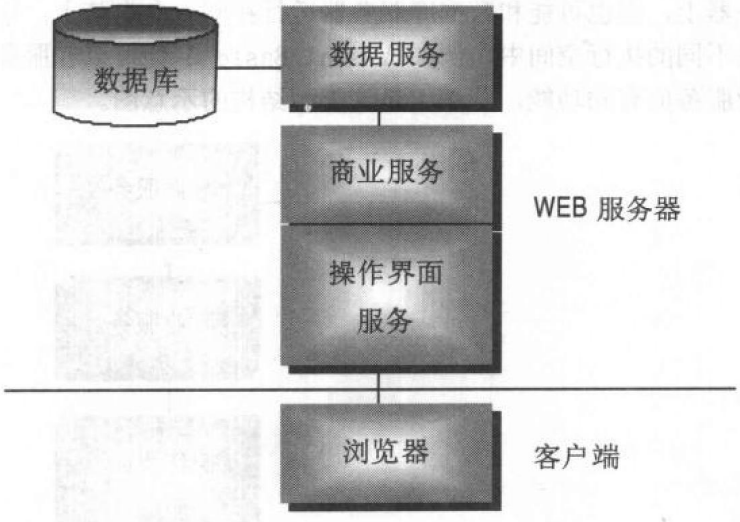


图 1-5　Internet 应用结构

1-3　数据访问的风格

设计高性能的关联式数据库应用程序并不能单纯从服务器的角度进行改善，前端程序的设计对于执行性能而言也具有举足轻重的份量。所以在整个性能的改善中，对于整个开发结构的熟悉程度，前端工作站所扮演的角色也具有决定性的影响。

因此除了适当的分工技巧外，规划主从结构的另一个要素则是系统集成的工作，换言之，如何处理应用程序和数据库沟通的方式也是影响执行性能的关键。应用程序可以只把数据库视为储存记录的场所，绝大多数的数据处理工作则交给应用程序进行；或者是将数据库视做数据管理的环境，让数据库进行大多数数据处理的动作。下面内容是有关数据访问应用的相关讨论。

善用 SQL 语法

由于 SQL Server 所使用的数据处理语言是 SQL（Structured Query Language）语法，当应用程序通过网络尝试和数据库沟通时，直接运用 SQL 语法进行数据访问动作是最适合且有效率的方式，因为这个交互过程中将不会牵涉到任何叙述转换的过程，使得 SQL Server 能够快速的进行最佳化的处理动作。如果使用的是前端应用程序所提供的特殊语法来进行数据访问工作，执行时将会多一道将语法转译成对应 SQL 语句的过程，造成时间上的浪费。

另一方面，若能够很有效率的运用 SQL 语句，应用程序将可以直接要求后端的数据库服务器直接运算出所需的结果，减少数据来回传送的频率，由于网络的沟通次数常常是构成性能不佳的因素之一，这样对于系统的执行效率而言便自然提升了。

缩短数据事务过程

在前后端数据事务的过程中，也即从执行 BEGIN TRANSACTION 到认可事务（COMMIT TRANSACTION 叙述）的期间，服务器必须保持前端工作站相关状态的数据，例如用户当前所取用的数据集内容，这些数据多半是存放在一些临时的储存空间，以便维持事务过程中各个数据的一致性。如果前端工作站的数量一多，随着个别的事务时间拉长，相对的就会消耗掉服务器上较多的资源。

数据访问的适当策略，应该是让前端应用程序快速的提出要求并取得数据结果，使得服务器不必长时间保留前端工作站的状态而可以很快解除锁定。由于 SQL 语句是以集合的运算为基础，在此情况下前端应用程序可以要求服务器一次更新所有符合条件的数据集合，避免数据库服务器执行过多的数据检核以及锁定的动作，因此使用 SQL 语法能够符合缩短数据事务过程以加快系统操作效率。

数据库正规化

数据库系统的功能即在于将数据整理得有条不紊，随着不同设计者规划的差异，创建的数据库结构可能并不相同。如何能使整理出来的数据表尽量不发生数据重复的情况，对于数据库大小及访问效率所造成的影响相当显著。为了达到这个目的，通常必须将数据分成多个独立的数据表，这种数据分离的技巧我们称之为正规化程序。

以一个简单的选课数据库为例，现有 100 位学生要在 4 个学科中选修一门课程，想要制作这个数据库内容，不假思索便可以按表 1-1 的格式创建单一数据表结构。

表 1-1

编 号	姓 名	性 别	选修课程	授课老师	学分
1	章文立	男	语 文	马立功	4
2	徐瑞琪	女	心理学	王天才	3
3	王小琳	女	语 文	马立功	4
⋮	⋮	⋮	⋮	⋮	⋮
98	戴号安	男	现代史	史 正	3
99	林丽丽	女	古典文学	陈通达	2
100	李士豪	男	心理学	王天才	3

由于每个学生选修的课程可能相同，因此从上表中您可以发现与课程有关字段的数据一直重复出现，整个数据库因此而虚胖不少；另一种创建的方式可以通过正规化程序，将上述数据表分为两个数据表，一个纪录有关学生的数据；另一则存放与课程相关的数据如表 1-2 和表 1-3 所示，整个数据库因此而简化许多。

表 1-2

编 号	姓 名	性 别	课程编号
1	章文立	男	3
2	徐瑞琪	女	2
3	王小琳	女	3
⋮	⋮	⋮	⋮
99	林丽丽	女	4
100	李士豪	男	2

表 1-3

课程编号	课程名称	授课老师	学 分
1	现代史	史 正	3
2	心理学	王天才	3
3	语 文	马立功	4
4	古典文学	陈通达	2

两个数据表之间只需以“课程编号”字段来创建关系，即可成为一个完整的数据库结构。正规化程序不仅达到简化的目的，当字段的数据要修改时，例如上述想要修改某个课程的授课老师或课程描述，更可见到正规化程序的优点。

采取必要的数据库正规化步骤将有助于数据访问效率的提升，如上述例子经过正规化的数据库最大特点是每一个数据表的字段数量变少了，应用程序可以通过关联对照的方式来取得一些复杂的数据结果。数据库正规化所获得的好处分述如下：

- 因为每个数据表的字段都不多，相对的每条记录的数据量都较小，因此可以加速数据排序和索引的创建过程。
- 由于数据表经过分割过程，相对的数据表数量也增加，形成更多的 clustered indexes 得以加快数据访问动作。
- 每个数据表上的索引较少，将有助于数据更新的效率。
- 数据表中出现较少的 NULL 值，重复性的数据也变少，可以提升数据库的访问效率。

如果遇到数据库的基本结构已经定型了，此时要谈任何正规化动作似乎为时已晚，然而我们仍然可以进行部份或权宜的正规化工作，将造成访问瓶颈的大型数据表作适当调整。例如前端应用程序是通过存储过程执行数据的访问操作，那么这个结构改变的过程对前端程序并不会有什么影响；如果不是这种处理方式，我们则可以运用创建视图（View）的方式，产生与数据库正规化的相同效果，同时也可以将更改前端应用程序的程度降到最小。

充分运用数据检核能力

不管数据库的结构如何规划，能够充分运用 SQL Server 的数据检核功能，让数据库在数据事务过程中自动依据定好的规则筛选特定数据内容，对于后面数据库维护工作而言将可以省却不少麻烦。在 SQL Server 中所支持的数据检核功能包括下列各项：

- CHECK 约束 —— 这个约束可以检查数据的值是否合理。
- DEFAULT 以及 NOT NULL 约束 —— 这些约束可以避免因为字段值遗漏所造成的程序错误或是其他复杂的影响。
- PRIMARY KEY 及 UNIQUE 约束 —— 能够强迫检查字段数据的唯一性，同时创建必要的索引来维护这些数据。
- FOREIGN KEY 约束 —— 能够确保相关联的数据表中有对应的记录存在，而不至于产生数据不一致的情况。
- IDENTITY 字段 —— 能够自行产生唯一的标识码。
- TIMESTAMP 字段 —— 确保多用户更新数据时，能够检查彼此间的时间差。
- 用户自定义类型 —— 可以确保跨数据库或数据表时，字段的类型能够保持一致。

如果我们能够善用这些功能，直接在数据库上创建必要的处理规则，所有的前端工作站用户访问这些数据时自然都要按照这些规则，程序设计者便不需要辛苦的通过应用程序来达到这样的要求，执行性能也可以因此而提升。