本书附光盘一张

三层次 Client/Server 应用开发指南

侯云峰　刘　睿　杨正洪　张立平　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.com.cn

内容简介

本书系统地介绍了作为主流核心系统标准的体系结构——三层次客户机服务器结构的技术及应用。全书共分6章,内容包括三层次客户机服务器结构的系统结构、计算机体系结构的变迁、三层次结构的优点、CICS如何构建一个优秀的三层次结构中间件;如何使用CICS编程应用并使用VB、Delphi、C++ Builder、PowerBuilder、Java和C进行前台界面开发;CICS实验环境的安装与配置,搭建用于实现事务处理的开发环境;CICS高级编程;对CICS进行性能调优方法及参考数据;DB2、SYBASE、ORACLE、INFORMAX等数据库的嵌入式SQL编程等。

在书中附有光盘1张,包括使用CICS在Windows NT平台上搭建三层次结构的所有软件以及书中的所有示范程序。

本书内容丰富、实用,可供从事构建、开发、使用三层次Client/Server结构的工程技术人员阅读,也可供有关专业的大学生、研究生参考。

图书在版编目(CIP)数据

三层次Client/Server应用开发指南/侯云峰等编著. -北京:电子工业出版社,2000.6
ISBN 7-5053-5954-1

Ⅰ.三… Ⅱ.侯… Ⅲ.计算机网络-关系数据库-数据库管理系统-手册 Ⅳ.TP311.132.3-62

中国版本图书馆CIP数据核字(2000)第60642号

书　　名:三层次Client/Server应用开发指南
编 著 者:侯云峰　刘　睿　杨正洪　张立平
责任编辑:龚兰方
特约编辑:陈　琼
排版制作:电子工业出版社计算机排版室
印 刷 者:北京大中印刷厂
装 订 者:三河市金马印装有限公司
出版发行:电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编100036
经　　销:各地新华书店
开　　本:787×1092　1/16　印张:24.25　字数:621千字　附光盘:1张
版　　次:2000年6月第1版　2000年6月第1次印刷
书　　号:ISBN 7-5053-5954-1/TP·3119
印　　数:3000册　　定价:56.00元(含光盘)

前　言

计算机体系结构经历了从主机集中的终端方式、C/S 结构以及现在越来越普遍的三层次客户机服务器结构。在当今中国，从银行、电信，到保险、证券的各个行业，还有不容忽视的电子商务、普遍运算，都越来越多地使用三层次结构作为核心系统的标准体系结构。但是，由于三层次结构提高了开发的起点，加上具体介绍三层次应用开发的书籍并不像介绍 C/S 的书籍那样普遍，对于大多数开发人员来说，三层次结构依然是一种理想，而不是一个能够立即在现实中使用的方法。甚至，由于缺乏对工具的了解，有不少开发人员自行开发了简易的中间件来满足其对三层次结构的需求。正因为如此，给三层次结构开发的普及带来了很大的障碍。

本书在第 1 章中详细介绍了三层次客户机服务器结构的系统结构，从计算机体系结构的变迁，三层次结构的优点，一些常见概念，如数据的一致性、两阶段提交、分布式事务处理、事务处理器以及 XA 规范的系统说明，到 CICS 是如何构造以成为一个优秀的三层次结构中间件。

如果你对于三层次结构的理论已经有所了解，可以在第 2 章中学习如何使用 CICS 编程应用，并且使用 VB、Delphi、C + + Builder、PowerBuilder、Java 和 C 进行前台界面的开发。第 3 章介绍 CICS 实验环境的安装与配置，着重于搭建一个能够用于实现事务处理的最简开发环境。如果要进一步使用 CICS 的底层功能，第 4 章的内容是非常有用的。

在第 5 章中，介绍了对 CICS 进行性能调优的方法以及一些参考数据，可以根据系统来选择合适的硬件环境，满足对性能的需求。

在 CICS 应用服务器端，采用嵌入式 SQL(E-SQL)编程和各种数据库进行连接。第 6 章详细地介绍了 DB2、SYBASE、ORACLE、INFORMIX 等数据库的嵌入式 SQL 编程，即使你不使用 CICS，这也是一个很好的参考。

特别要说明的是，本书所附光盘包含使用 CICS 在 Windows NT 平台上搭建三层次结构的所有软件，以及本书提及的所有示例程序。根据第 4 章的步骤，可以自己建立一个实验环境，亲自体验开发三层次结构应用的感觉。

本书第 1 章由侯云峰、杨正洪、张立平共同编写，第 2 章、第 4 章和第 5 章由刘睿编写，第 3 章由张立平编写，第 6 章由杨正洪编写，最后由侯云峰统稿。

目　录

第 1 章　三层次 Client/Server 介绍

1.1　计算机体系结构变迁

从计算机诞生一直到今天，计算机网络从无到有发展起来。从 20 世纪 50 年代到 20 世纪 70 年代初期，基本上是独立专用的大型机系统的一统天下；20 世纪 70 年代初期到 20 世纪 80 年代中期，开始有小型机系统与自有网络或 APPANET 连接；20 世纪 80 年代中期到 20 世纪 90 年代初期，开始出现服务器与 PC 客户机通过局域网互联；20 世纪 90 年代初期至今，服务器与“瘦客户机”通过局域网、广域网或 Internet 相连。

1.1.1　终端方式

很长一段时间，大型机是商业计算的核心：工资、档案、库存、账户全都由一个大型中央计算机处理。专门的计算机 MIS 部门负责维护、备份、二次开发和升级等。主机的价格非常贵，只能用于重要的、非交互任务。

在 20 世纪 60 年代后期和 20 世纪 70 年代初期开始有厂商生产小型机。小型机比大型机便宜，有交互能力。小型机主要优点是可以为公司或部门服务，而不仅仅是计算中心。其系统模式往往是大型机和小型机各存储一部分数据，大约 20%的备份和维护的工作量移到了部门中，而主要的维护工作仍处于集中状态，总的拥有成本仍然很高，或许比纯大型机环境还要高。但是，小型机仍有其积极意义。它将计算机带入部门级应用，提供可交互终端和通信软件，E-mail 和电子备忘录开始改造传统工作文化，营造了一个虚拟工作世界。

与此同时，小型机的出现引起了 MIS 中心与其他部门的纷争。早期的 MIS 中心垄断了整个公司的计算，其他部门往往对此不满，小型机第一次给他们提供了向“专制”挑战的机会。有远见的部门看到了自己掌握计算能力的好处，借助小型机，他们开始雇用自己的程序员做自主开发和部分维护工作。随着部门中越来越多的人在学校接受计算机教育，这种转变得到了进一步推动。

1.1.2　Client/Server

在 20 世纪 80 年代中后期，一些新发展又引起计算模式变化。首先，PC 逐渐表现出优势。PC 可以安置在每个办公桌上，有充足的应用软件，包括字处理、电子函件、账户管理、项目管理以及简单的数据库等。局域网的兴起为独立的 PC 机互联提供了可能，更促进了 PC 机的普及。其次，“迷你小型机”——工作站出现了。工作站的性能在很多场合足以替代小型机，而价格则接近 PC 机。很多部门既拥有工作站，又保持以前的支持队伍，做自开发和支持。工作站与 PC 机运行在同一网络上，有足够的性能为部门级应用开发提供环境。由于 PC 机价格飞跌，到了 20 世纪 90 年代初期，个人开始纷纷购买 PC 机在家中工作。刚开始是通过软盘交换数据，后来随着 Modem 的普及，可以通过电话线实现高速数据传输。数据

计算第一次从集中处理转向了本地处理。随着网络速率从 20 世纪 80 年代初的 56kbps 发展为 20 世纪 90 年代的 10Mbps，在部门中与部门间实现大量数据共享逐渐成为可能。从而，出现了客户/服务器系统时代，即把软件安装到每一台机器上。

1.1.3 三层次结构

传统的客户/服务器应用软件模式大都是基于“肥客户机”结构下的两层结构应用软件。客户机方软件一般由应用程序及相应的数据库连接程序组成，服务器方软件一般是某种数据库系统。它面临的一个主要问题是系统的可伸缩性差和安装维护困难。多层结构应用软件与传统的 C/S 模式下的两层结构应用软件相比，有着可伸缩性好、可管理性强、安全性高、软件重用性好以及节省开发时间等诸多优点。在 Internet/Intranet 环境下，这些优点显得更加突出。很多公司也提出了多层应用软件体系结构。

三层次结构的客户/服务器模型是一种先进的协同应用程序开发模型，这种方案将客户/服务器系统中各种各样的部件划分为三“层”服务，它们共同组成一个应用程序，这三层次服务包括:

1) 客户端服务程序;

2) 业务服务和其他“中间层”服务程序;

3) 数据服务(数据库)。

这些层并不一定与网络上的具体物理位置相对应，它们只是概念上的层，借助这些概念可以开发出强大的应用程序。使用这种方法设计应用程序，开发人员在网络上部署进程及数据时可以有相当大的灵活性，从而有利于实现最佳的性能、更好的安全性以及更方便的维护。中间层中包括提供业务服务和其他中间服务的部件，是联系用户服务和数据服务的桥梁，它们响应用户(或其他业务服务)发来的请求，执行某种业务任务，并对相应的数据进行处理。用户不需要直接与数据库打交道。在实际应用过程中，中间层部件通常可分为两个以上的层次。因此，该应用模型也被称为多层次结构。

当企业信息系统从客户/服务器模式向多层分布式应用模式转变时，需要应用服务器(ApplicationServer)的支持，以便将不同的应用技术集成在一起，使多层分布式应用的开发、分发、管理变得更加容易。现在已经有很多企业采用了应用服务器技术，极大地增强了企业应用的性能。目前，企业级应用服务器主要分为以下两类:

1) 基于中间件的应用服务器

基于中间件的应用服务器通过与现有系统的集成，可以为企业提供更强大的功能，包括事务处理、安全管理、容错、负载平衡等。

2) 基于 Web 的应用服务器

20 世纪 80 年代末，WWW 开始为人所知。但在 Netscape 使之大众化之前，Web 站点与应用的发展缓慢。随着浏览器的发展，研究三层次结构与客户/服务器应用的人员马上意识到浏览器就可以作为“瘦客户机”，Web 服务器作为应用服务器。Java 为所有浏览器提供统一用户接口，而 Web 服务器端的 CGI 完成数据处理。即：将运行在客户端的应用软件移植到服务器端。客户端将不再需要应用程序，它们完全集中在服务器端。这意味着用户完全可以通过浏览器来执行应用。在这种体系结构下，Web 应用服务器通常运行在 WebServer 上，负责处理客户请求，与后台数据库的连接一般采用 ODBC 和 JDBC 技术。这种类型的应用服务器易于使用，并且支持基于 EJB（EnterpriseJavaBeans）的服务器应用程序开发。但这

种应用服务器也存在不支持事务处理、安全性差、对已有交易系统支持有限以及性能较低等缺陷。B/S 结构解决了各种分布式应用和跨平台应用，扩展了业务范围；在 B/S 结构下，整个系统的管理、资源分配、数据库操作、业务逻辑部件的管理及动态加载等工作集中于应用服务器，容易部署和管理。为实现 B/S 结构的应用，我们认为解决之道是采用 Java 技术，面向 Internet 的数据库。

下面，我们深入探讨一下这种新型对象 Web 的三层次客户/服务器体系结构。

1) 客户端

第一层是属于以传统的 Web 浏览器和 Web 为中心的新的桌面范畴，它与现在静态的 Web 页不同，新的内容将具有更接近于现实世界中真实对象的观感。这种非常生动的动态内容是由 JavaBeans 组合提供的。JavaBeans 被嵌到可移动容器(HTML 页或 Jars 等)中，用户用拖放操作来和这些对象进行交互。客户端 Beans 可以和容器内的其他客户端 Beans 以及服务器 Beans 进行交互，服务器 Beans 使用 CORBA 事件和回叫启动客户端 Beans 上的方法。IIOP 和 HTTP 可在同一个网络上工作，HTTP 用于 Web 页、Jars、图像的下载，CORBA 用于 Java 客户端和服务器间的双向通信。

2) 中间层

第二层是一些服务器，这些服务器能为 HTTP 和 CORBA 客户端提供服务。UNIX、NT、OS/2、Netware、MacOS、OS/400、MVS 以及 TandemNonStopKernel 等所有的操作系统平台都支持这种 CORBA/HTTP 组合。CORBA 对象是作为中间层的应用服务器来运行的，这些对象通过 CORBA/IIOP 与客户端 JavaBeans 进行交互。在可缩放性要求较低的应用中，通过在 HTML 服务器页内运行的脚本也可以调用这些对象，例如，Netscape 的 Web 应用接口(WAI)就提供这样的功能。

另外，第二层也必须提供服务器端的组件协调程序，这就是众所周知的对象 TP 监视器。组件协调程序不是管理远程过程，而是管理对象。组件协调程序启动对象池分散负荷，提供抗故障能力，协调多重组件进行事务处理。如果没有这些组件协调程序，ObjectWeb 就不能管理数目庞大的服务器端对象。

在 CORBA/JavaObjectWeb 中，第二层还有存储组件标题、HTML 页和可移动位置的功能。这些组件标题、HTML 页和可移动位置可以存储在由 DBMS 或 ODBMS 所管理的、可移动的 JavaJars 内。

3) 后端

第三层次包含所有 CORBA 对象能访问的内容，包括过程的 TP 监视器、面向消息的中间件、DBMS、ODBMS、Lotus Notes 和电子函件等。这样，CORBA 业务对象将置换中间层的 CGI 应用。这是一件大好的事情，正是 CORBA 语言发挥通信作用的用武之地。

在开发大型分布式应用系统中表现出强大的生命力，并形成了 4 项具有代表性的主流技术，即 IBM 公司的 CICS 和 BEA 公司的 TUXEDO、OMG 的 CORBA (Common ObjectRequest Broker Architecture)、Microsoft 的 ActiveX/DCOM (Distributed Compound Object Model)和 SUN 公司的 Java/RMI。IBM 公司的 CICS 产品在中间件的市场占有率很高，也支持 CORBA 技术。

OMG 是一个非营利性国际组织，致力于使 CORBA 成为“无所不在的中间件”。1989 年成立时仅有 8 家公司参与，而今天已经是拥有 900 多个机构成员的“议会式”标准化组织，世界上几乎所有最有影响的计算机公司（如 IBM、Microsoft 和 HP 等）、著名的工商企业（如

Boeing、Citibank、FordMotor 等）和大学研究机构都是这个组织的成员。OMG 所制定的分布对象计算标准规范包括 CORBA/IIOP、对象服务、公共实施和领域接口规范。遵照这些规范开发出的分布计算软件环境可以在几乎所有的主流硬件平台和操作系统上运行。现在，CORBA/IIOP 已成为 Internet 上实现对象互访的技术标准，OMG 的 IIOP 也已成为许多公司(如 Oracle、Netscape、Sun 和 IBM 等) 进行系统集成的基本协议。1995 年以来，基于 CORBA 软件的企业级应用发展迅猛，大有覆盖 DCE 之势。目前世界上有一定影响的 CORBA 软件制造商已有 10 多家。

ActiveX/DCOM 是由 Microsoft 推出的对象构件模型，最初用于集成 Microsoft 的办公软件，目前已发展成为 Microsoft 世界的应用系统集成标准，并集中反映在其产品 ActiveX 中。在分布计算技术上，OMG 的优势比 Microsoft 至少领先 2～3 年。目前，只有 OMG 的技术能够支持异构环境中大型分布式应用的开发，而 Microsoft 的 DCOM 技术尚不能胜任。Microsoft 的优势主要表现在应用和市场能力上。从未来市场策略考虑，Microsoft 决定支持 OMG 提出的 OLE/COM 与 CORBA 的互操作标准，从而使 COM 的对象能够与 CORBA 的对象进行通信。今后 3～5 年内，OMG 和 Microsoft 的分布对象技术将共存，并在许多方面相互渗透。

按照 Sun 和 Javasoft 对 Java 的界定，Java 是一个应用程序开发平台，它提供了可移植、可解释、高性能和面向对象的编程语言及运行环境。RMI(Remote Method Invocation)是分布在网络中的各类 Java 对象之间进行方法调用的 ORB 机制。

从应用架构上看，两次重大的迁移最为引人注目。第一次是从主机终端方式向 Client/Server 计算方式的迁移，这次迁移的积极效果之一是打破了计算方式高度集中的局面，使计算环境向客户靠近了一大步。第二次则是从经典的 Client/Server 计算方式向 Internet 架构下的集成计算方式的过渡与融合，这是一个目前正在进行的过程。

每次迁移都没有也不可能简单地完全扬弃各自的出发点。例如，在 Client/Server 计算方式开始显现其优越性时，虽“抛弃大型主机”的议论甚嚣尘上，但基于大型主机的集中式计算不但依然存在而且在继续发挥作用。现在，在 Internet/Web 的影响越来越被认同的今天，我们也依稀听到了“Client/Server 计算行将消亡”的议论。事实上，无论是今天还是以后，绝不会有任何单一的技术能够解决应用中的一切问题。据 GartnerGroup 的一项调查显示，在世界经济 500 强的跨国大企业中，Client/Server 计算方式现在仍然占据主导地位。

1.2 三层次的必要性

1.2.1 数据的集中 → 分布 → 合理化集中

在整个计算机发展的过程中，计算模式也在不断发生变化。集中主机运算模式在各个阶段都起了非常重要的作用，而且，目前的很多关键运算，例如，银行核心业务、航空订票等还必须依靠主机的高可靠性来维持关键任务的不间断运作。但以廉价的PC机为代表的分布式客户机/服务器将电脑的应用普及到社会的各个方面。在中国，计算机的真正发展可以说就是PC机的发展，在社会的很多部门，根本没有经历大型主机集中式处理的阶段。

客户机/服务器系统在使用的初期有非常明显的好处，特别是因为进入门槛低，可以非常快速地建立一个应用系统，而且，在一定程度上，系统的升级和维护很方便；比起主机系

统，还有一个重要的优势是开发和管理人员的来源充分，这样，就进一步降低了拥有成本。正因为这些原因，我们看到，客户机/服务器系统在相当程度上成为一个计算机应用系统的代名词。在很大程度上，说起建立一个应用系统，就是选择一个合适的数据库，再加上一个好用的前台界面开发工具。

但是，随着客户机/服务器系统的发展，企业规模不断扩大，很多采用此类分散型运算的中国企业突然发现其服务器数量已经达到令人吃惊的地步。由此带来的是复杂的管理模式，运算营运成本失控，关键型应用无法实现，各子系统之间互不往来，而且，完全失去了以客户信息为中心的关注能力。各子系统都是为各自独立的部门服务，各自包含各自的客户信息，但是，却无法得到客户的完整信息，也就无法为客户提供更好的服务，甚至，也无从知道什么才是更好的服务。

这时候，分散型运算的弱点体现在：管理困难、资料分散、运算成本失控、低安全性等因素，而这些弱点在集中主机运算模式中恰恰能够得到很好的解决而变成了优点。这样企业被迫重新考虑再集中的问题。而最近电子商务的商业运作模式的发展，中国企业与国际进一步接轨的需求，使企业与产品迅速扩展到全球各地，对全球市场与金融变化的信息的及时掌握，以及对更快的反应速度的竞争要求，进一步决定了集中计算模式成为目前与未来的运算模式的主流。

与国外企业有一个显著的不同，国外企业可以实现完全意义上的集中计算，建立一个或几个互为备份的超级计算中心，为全国范围甚至全世界范围内这个企业的全部核心业务服务。我国国内由于体制上的原因，往往无法实现一个全国范围内的计算中心，而一般较容易实现地、市级或省级的集中，这已在电脑应用水平最高的几个行业得到体现。

例如，银行，作为我国计算机应用最规范，应用水平相对最高的行业之一，也最先实现数据的集中化，这样，在其基础上的通存通兑，信用卡联网，一卡通，一本通，及至现在的各种费用的代收代缴等吸引客户的服务才得以实现。这些业务无一例外，都属于集中式的应用，都需要一个集中式的数据中心，否则，如果这些数据还是分布在各个地方，则不可能有这样的应用。

中国电信，无疑是中国发展最迅速的一个行业，在快速发展及市场竞争之下，也同样面对这个问题。随着规模的发展，电信的最终分家，如何能为客户提供更好的服务，如何降低成本，如何快速面对市场变化，都要求走向集中。在去年中国电信总局在制定本地电信业务计费账务系统时，明确倡导利用三层次结构来实现。同时，可以看到，不但本地计费系统这样要求，近期的其他系统也纷纷提出相同要求。

在保险行业，根本就是以客户为中心的。为了能够为客户提供更好的服务，中国的保险行业将客户服务中心（Call Centre）作为发展的一个方向，而一个统一的客户服务中心必须有一个集中的数据中心。所以，现在的中国保险行业，也在开始数据的大集中，建立地、市级和省级的数据中心。

其他行业，也是一样，竞争越激烈，发展越迅速，对集中计算要求的迫切性也越高。

1.2.2 三层次结构应用系统的优越性

1. 性能问题

三层次系统要解决的第一个问题是性能问题：

我们先以一个例子来说明系统架构对性能的影响，如表 1-2-1 以电信市话系统的规模为例，表示用户数在特定数量时，需要多少业务终端为其服务。

表1-2-1　市话系统规模与业务终端数

规　模	终 端 数	最大终端数
<20万	50	100
20万～60万	100	200
60万～150万	200	400
>150万	300	1000

如果采用二层结构的话，当应用相对简单、数据访问量不大的情况下可以承受。而在上表，当应用变得复杂、庞大，数据的访问量增大，客户机数目很多，就会带来性能急剧下降的后果。我们来看一看性能问题究竟有几个方面：

1) 数据库并发连接：作为二层应用，客户机在访问数据前，必须与数据库建立连接。而建立一次连接一般都会在数据库服务器端有比较复杂的准备工作，所以，为避免在每次访问数据库时都建立一次连接的额外开销，客户机一般会保持长连接，即使不访问数据，也保持此连接。而现在的开发工具往往在一个应用之内会保持若干个长连接，这在客户机数目有限的情况下，不会有太大的问题，但是，若是在有几百甚至上千个客户机的情况下，这些数据库的并发连接就相当可观。每种数据库的实现方式不同，每个并发连接所需消耗的系统资源也不一样。如果以每个连接 1MByte 内存来计算，1000 个客户机至少要占用服务器 1GByte 的内存，而且，这些内存显然利用率非常低；

2) 远程连接：二层结构的应用一般是在局域网中的应用，不太关心网络开销问题，而在一个复杂应用环境，客户机往往与数据库服务器不在同一个比较高速的网络上，需要通过广域网甚至拨号线路来实现连接。而数据库客户机和服务器之间的通信一般并不是非常有效，一次数据库操作，客户机和服务器之间的交互可以有若干次。而且，由于远程网络环境的不可靠，长连接的可靠性也会有很大的疑问；

3) 数据库的瓶颈效应：在整个系统中，由于任何一个操作都必须通过数据库，数据库服务器往往是最先成为瓶颈的地方。在二层结构中，每增加一个客户机，就又加重服务器的负担。 因此，在完成了 CPU 升级，增加内存之后，我们对数据库服务器能力的增加也到头了，而一台主机的升级能力总是有限的。 同时我们知道，在线事务处理 (OLTP)环境中，并发数据库提高数据库性能的能力也是有限的，不像在线分析处理系统 (OLAP)可以采用并行度很高的数据库，几十个，上百个节点的并行数据库可以大有用武之地。这是因为分析操作一般用户不多，数据库操作也不多，但都是复杂的、需要大量时间的查询操作，可以将其分散在各个节点独立完成后将总体结果得到。在线事务处理的操作都不太复杂，但非常多，而且往往都有写操作，在并行系统中，为保证事务的完整性，实际在每次操作时，都必须有一次各节点的两阶段提交，复杂性随节点的增加有呈指数上升的趋势。所以，一般在线事务处理系统中，即使采用并行数据库，节点数往往是 2 个、4 个，更多的节点不但无法提高系统的性能，甚至还会使速度变慢。

这些问题，二层结构（图 1-2-1）无法得到满意的解决，而三层次结构正是解决问题的关键。与二层结构不同，三层次结构无须保持客户机与服务器之间的长连接，通常采用了无连接或短连接(Sessionless、Connectionless)的方法。客户机和应用服务器之间没有复杂的上

下文关系，可以在每次请求时建立连接，服务器返回结果后可以马上断开与客户机的连接，这样，可以充分共享服务器端系统资源，为更多的并发用户服务。事实上，Web 服务器采用的 HTTP 协议就是一种无连接的协议，浏览器只有在请求时才和 Web 服务器连接，取到结果后马上结束此连接。只有采取这种无连接模式，才可能同时为几百、几万甚至更大的并发请求服务。

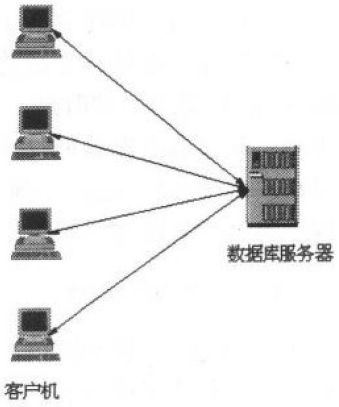

图 1-2-1　二层客户机/服务器系统

1) 减少数据库并发用户：两层 C/S 结构无法满足成百上千的终端用户同时联机的需求，是因为大量的数据库并发连接消耗了很多宝贵的数据库服务器资源，但其实这些并发连接的使用率非常低。三层次结构可以通过共享数据库连接的方式，来明显地减少数据库并发连接数。以下，举例说明如何减少并发用户数。

假设，此时客户机数目有 100 个，而此时，采用三层次结构，系统为图 1-2-2。

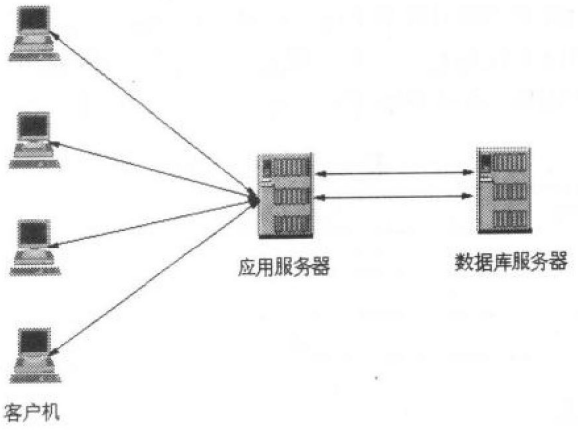

图 1-2-2　三层次客户机/服务器系统

每个交易在客户机端和服务器端所需时间是不一样的。例如，以银行信用卡取款为例，对银行的营业员来说，这个服务必须包括接受用户的信用卡、取款单、身份证、输入用户卡号、得到授权、要求用户签名，最后，交还用户卡、证及所取款项。而相应的操作，在服务器端，只是几个 SELECT、UPDATE、INSERT 语句，两者完成时间相差很大。为简化，我们假设每个交易在客户机端需要 10 秒完成，而在服务器端，只需 1 秒即可完成。这样，在二层结构中，在同一个交易内，这个服务器连接只在 1/10 时间内工作，而在 9/10 的时间中都是空闲无法利用。在三层次结构中，所有客户机不直接和数据库连接，而是作为应用服务器的客户，向应用服务器发出请求。同样，我们可以看到，如果所有客户机都在进行同一交易，从平均意义上说（即，排除所有的交易都同时开始，同时结束，而是平均在任意时间段），同时只有 1/10 的客户在向服务器请求服务。应用服务器只要保持 1/10 的客户机总数的数据库并发连接，同时保证这些连接的满负荷工作，就能和保持全部连接的二层系统一样，满足所有客户机对服务器的请求。就是说，只要有 10 个数据库并发连接，就可以满足 100 个客户机。这样，由于同一交易在终端机完成所需时间与此交易在服务器端完成所需时间的不匹

配(可以达到一个数量级或更大)，应用服务器可以维持较少的数据库并发连接而为大量终端用户服务。

实际上，所需共享连接数不但和客户端、服务器端完成时间的比值有关，也和用户需要的响应时间有关。还是以上面的例子为例，完成一个交易的实际时间为 11 秒，即客户机端处理时间+服务器端处理时间，这时，由于可以认为客户机端的处理时间为常数，响应时间的变化就只和服务器端处理时间有关。但是，如果并不需要有这样的响应速度，而可以接受服务器端平均 2 秒的响应，我们可以进一步减少共享连接数目到 5 个，因为，这个时候，在服务器端每个交易都可以等待 1 秒才得到服务。因此，共享连接数可以用如下公式计算

$$N=(T_C/T_S)/(T_R/T_S)$$

其中，N代表共享连接数，T_C代表交易在客户机端所用时间，T_S 代表交易在服务器端所用时间，T_R代表要求交易在服务器端的响应时间。

注：某些数据库有连接共享的功能，实际并不为所有并发连接在物理上分配系统资源，也可以减少大量并发用户对服务器的消耗。

2) 减少网络开销：同一次交易，应用服务器与终端客户只有一次交互，而如果是两层结构，就会有多次交互。仍以上面的例子为例，假设每次服务器与客户机的交互时间是相同的，且客户机与服务器的每次交互为T_C，而应用服务器和数据库服务器之间的交互每次为T_S。而且，由于三层次结构接受用户请求会将结果一次返回，所以网络开销有如表1-2-2所示的结果。

表 1-2-2 网络开销

交互次数	二层结构耗时	三层次结构耗时
1	T_C	T_C+T_S
2	$2*T_C$	T_C+2*T_S
n	$n*T_C$	T_C+n*T_S

3) 消除数据库瓶颈：

• 我们可以看到，由于客户机和服务器之间往往不在局域网上，而应用服务器和数据库服务器往往在高速局域网，甚至是同一台主机，因此T_C远大于T_S。这样，我们有结论，在只有一次交互时，三层次结构和二层结构所需网络开销大致相同，但如果一个交易有若干个数据库操作时，三层次结构的优势就非常明显。即使只有一个数据库操作，其实常常也是两个数据库操作，如

```
SELECT C1 FROM TEST WHERE ... ;
COMMIT;
```

这就代表两次数据库操作，会有两次客户机和服务器的交互。注意，使用存储过程也可以减少数据库和客户机的交互，但即使在这种情况下，客户机和服务器也会有多次交互的情况，例如，要在一个存储过程中返回多个结果集时，两者必然要进行多次交互；或者一个交易根本无法在一个存储过程内完成，而需要调用几个存储过程时，也必须有多次交互。

• 应用服务器负载平衡：虽然数据库的并行系统不能有很大的并发度，应用服务器却无此限制。当应用服务器成为瓶颈时，可以任意增加应用服务器数目，由多台应用服务器同时为终端客户服务，实现平衡负载，同时提高系统的整体可靠性。而且，在数据库瓶颈不可逾

越时，可以由应用服务器上的应用来实现将分类过的数据访问不同的数据库的形式，由多个数据库实现应用级的一个逻辑数据库，一定程度上，消除数据库服务器的瓶颈。

• 分担数据库部分工作：现在的二层结构的应用大量采用存储过程，完成商业逻辑，同时，也可以减少与客户机的交互，提高性能。但是，在用户并发连接很多的时候，数据库本身已经是瓶颈，还要其完成包括复杂商业逻辑的存储过程，就无法集中所有资源来处理 INSERT，UPDATE，SELECT 等必须由数据库完成的操作。而且，由于存储过程是一种高级功能性语言，缺乏编程语言的一些特性，所以，往往会借助临时表等技术来存放中间结果，这样，就进一步加深数据库的负担。因此，将存储过程中的商业逻辑前移到应用服务器，就可以分担原来数据库的一部分工作，使数据库资源的利用更有效。

2. 互联问题

第二个问题就是系统互联问题，从连接的范围要求上说，可以分为：

• 同一系统内应用交互：在同一公司，同一部门，局域网环境，甚至在共用主机系统或数据库系统的环境下，将各个子系统连接在一起。这种互联由于通信、数据信息、相互接口容易控制，一般比较容易实现。但是，如果子系统很复杂，各自采用不同技术平台，就需要一个统一的接口标准共同遵循。

• 异种系统间应用交互：跨公司，跨部门应用之间的互联，各互联系统之间相互保持独立，即对外是黑盒子，有特定的接口与外界交互，外部系统通过此接口调用其公布的功能，而无从得知其内部结构、实现、数据格式等。而且，只要保证此接口不变，系统可以作任何形式的改变。

从时间意义上说，可以分为：

• 实时或准实时交互： 实时互联需要即时的响应，一般响应时间在秒级，就是说，必须在几秒时间内返回结果。但由于在和其他系统互联时，往往无法预测交互在其他系统内的运行时间，所以对即时响应的要求会有所放宽，我们将这种情况称为准实时交互。

• 异步交互：无须即时响应，一般用于系统之间批处理，如数据采集、日终对账等。这种交互，对时间的要求并不很高，从几分钟，几小时或更长，但一般需要信息可靠地传送。

系统互联不但需要一个好的架构以便与将来应用互联，也需要帮助已有系统甚至未知系统的顺利连入。二层结构，由于以数据库为中心，除非不同系统之间能够共享数据库，即，不同的应用之间能将自己的数据库公开，否则，必须采用其他方式来解决，譬如，数据库的复制等。总之，二层结构的应用要互联，一定是在同一系统环境下，往往还要满足非常苛刻的条件，而且，异种系统互联根本不可能。这是因为，直接对数据源的引用有如下弊端：

1) 所有客户机必须安装需要连接的所有不同种数据库的客户软件，这增加了维护复杂性；

2) 所有客户机必须有所有数据库的访问权限，对于一个分布式环境，不但管理困难，而且容易造成安全问题；

3) 增加数据库的并发连接，加重数据库负担；

4) 数据模式对外公开，降低对其修改的自由度。

这样，互联时的耦合度太高，影响系统的独立性。相反，三层次结构提供事务级别的调用而非数据源的引用。客户机看到的就是和业务逻辑相关的事务，这样，与其他系统互联时，只要提供这些被外部调用事务的调用规范，不用做其他特殊的工作，自然形成三层次甚至多层 C/S 结构。而且，在异种系统互联时，为屏蔽内部系统和外部应用，保证本地系统的安全，

往往会提供和外界相连的网关、前置机，本身就是多层次的表现。

三层次结构的应用服务器可以使用多种数据源为其提供数据服务，而且，应用服务器中的事务也可以作为一种资源，被本地或远程应用服务器的其他事务来调用。例如，用户要查询账户信息，向本地应用服务器发出请求，账户信息可以在本地，也可以在远程。所有的应用服务器都有一个事物 QRY1，接受用户输入的账户号，并返回查询结果。

```
QRY1
接受用户输入账户号 ACC;
IF ACC 是本地账户 THEN
    SELECT ACC 信息 FROM 本地数据库;
    返回 ACC 信息;
    结束;
ELSE
    判断 ACC 在应用服务器 SVRN 上;
    CALL QRY1 ON SVRN WITH ACC;
    返回 ACC 信息;
    结束;
END;
```

那么，完成一个账户查询的操作，流程可能如图1-2-3所示。

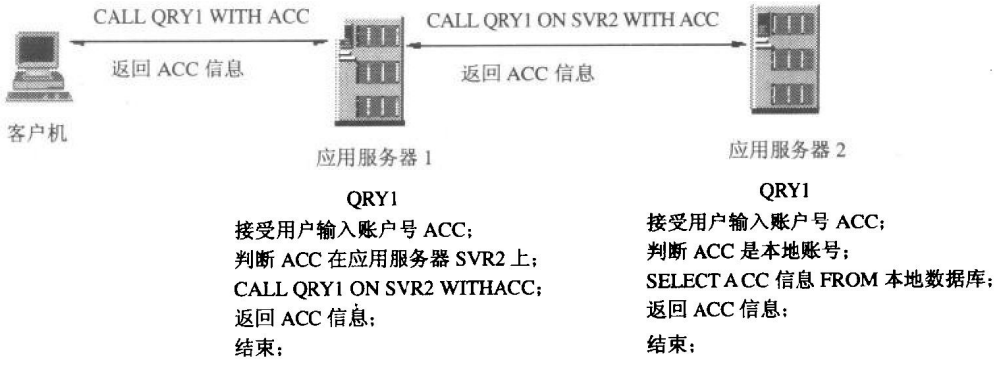

图 1-2-3　账户查询流程

从中可以看出，三层次系统的互联在结构上已经得到保证，可以满足用户的需求。

3. 安全问题

在二层结构中，安全问题也有很大的缺陷。安全，分为两大部分：

• 权限控制：控制不同的角色在系统中能够访问不同的资源。对于二层系统来说，就是控制用户对数据库资源的访问，对于不同的用户、组、角色，赋予对不同的表格、视图以及只读、插入、修改、删除等权限。在权限控制中，有一条非常重要的原则：权限最小原则，即必须保证每个人得到的权限能够满足其担当角色正常工作的要求，并且赋予的权限只能完成其指定的工作，而不应该拥有其他任何无关权限。