分布式 JAVA 2 数据库系统开发指南

Distributed JAVA 2 Platform Database Development

[美] Stewart Birnam 著
孟纯城 译

清华大学出版社

分布式 JAVA 2 数据库系统开发指南

[美] Stewart Birnam 著

孟纯城 译

清 华 大 学 出 版 社

（京）新登字 158 号

内 容 简 介

本书内容包括 JDBC、RMI、Swing、Servlet 和命令行程序的设计，以及如何综合运用这些技术构建一个系统，以便采用 Oracle 数据库和 Linux 操作系统。本书的前几章主要介绍了一些多层系统的背景知识。随后的一些章节讲述具体的编程技术，以及如何复用代码在 Swing 应用程序或 Servlet 中访问数据库。

本书的读者对象包括分布式数据库系统设计与开发人员、IS/IT 管理人员、想学习和研究新的技术的软件开发人员、正在学习编程的人员等。

Distributed Java 2 platform Database Development
Stewart Birnam
Copyright © 2000 by Prentice Hall PTR
Original English language edition published by Prentice Hall PTR / Sun Microsystems Press
All right reserved.
No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher. For sale in the People's Republic of China Only.

本书中文简体版由 Prentice Hall PTR 授权清华大学出版社出版发行，未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号：图字 01-2002-3162 号

版权所有，翻印必究。
本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

书　　名：分布式 JAVA 2 数据库系统开发指南
作　　者：[美] Stewart Birnam
译　　者：孟纯城
责任编辑：王　松
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：世界知识印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×960 1/16　印张：16　字数：344 千字
版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷
书　　号：ISBN 7-302-05755-9/TP・3402
印　　数：0001～4000
定　　价：36.00 元

前　言

本书内容包括 JDBC、RMI、Swing、Servlet 和命令行程序的设计，以及如何综合运用这些技术构建一个系统，以便采用 Oracle 数据库和 Linux 操作系统。本书的前几章主要介绍了一些多层系统的背景知识。随后的一些章节讲述具体的编程技术，以及如何复用代码在 Swing 应用程序或者 Servlet 中访问数据库。

本书中，所有涉及到数据库访问的范例全部采用了 Oracle。其中既有简单的查询范例，也有复杂的连接范例。本书详细阐述了如何插入、更新和删除普通的数据，以及如何将 Web 图形在数据库中保存为 BLOBS 格式。同时阐述了 SQL 语言的 Oracle 专用特性，如 CONNECT BY 语句在分级表中的用法。本书还讲述了如何利用 Swing 的 Jtree 对象，用 GUI（图形用户界面）显示出父－子关系。

本书中的所有范例代码都是在“小红帽”（Red Hat）Linux 6.1 平台上开发的。然而，这些代码实际可以在支持 Java 1.1 的任何平台上运行（目前不支持 Java 1.1 的平台很少）。为了更易于配置和开发，本书特别介绍了如何运用外壳脚本和 makefiles 工具。本书采用了在 Linux 环境中运行的 Oracle 8 数据库。此外，还提供了一些针对 Unix 命令进行系统调用的范例，以及一些构建 sysadmin 远程监视工具的范例。

简介

现在市场上有许多关于 Java 技术的图书，其中大部分定位在 Java 语言的基础教学上。其他图书则对一些特殊的应用程序编程接口（API）进行了深入的探讨。但是，这些图书很少介绍如何综合运用这些应用编程接口和技术，实际构建一个可以运行的系统。讨论解决特殊问题的图书就更少，类似如何访问 Oracle 数据库等等。实际上，我在市面上根本找不到任何一本这样的书，即使我找得非常非常辛苦。

尽管没有现成的参考书，但问题总要解决。为此，我着手一个大型研究项目，找出并积累构建应用程序所需的全部信息和技术。完成后，为了和大家共享我从中获得的知识，我便着手编写本书，以帮助那些被同样问题困扰的人们。

本书的全部内容均围绕采用 Java 技术构建系统而展开。与单纯设计一个有效的算法或者想出某些更显“智慧”的编码技巧相比，在现实世界中成功构建一套系统要涉及的东西多得多！这意味着需要构建易于扩展的、高效的、易于维护的代码。这些代码可能（或不

可能）结合一个现有的系统或软件包，如 Oracle。伴随着互联网的流行，您的软件必定需要采用互联网及其相关协议。另外，还需要采用适当的方法，有效的管理、配置和归档系统。没有好的管理或配置，系统就会毫无用处，而且会给人们留下不能稳定运行的印象。

本书的目的就是综合运用开放标准以及 Java 技术解决这些问题，同时也处理一些实际的应用问题。为了举例说明这些概念、技术和用法，本书将构建一个真正的“分布式数据库应用系统”，并在其中使用一个 Oracle 数据库。

由于近期 Linux 行业非常红火，所以我也很想知道所有这些技术是否都能在 Linux 下正常使用。现在，我非常高兴地告诉大家，在 Linux 操作系统中，Java 的运行非常正常稳定。目前，Oracle 也为 Linux 开发了一个版本。近几个月中，我们解决了在 Linux 操作系统下配置 Java 的所有问题。在 Linux 下运行 Java 程序和在其他 Unix 操作平台上运行没有什么区别。

什么是分布式数据库应用系统

随着互联网和低端 PC 机的发展，分布式计算已变得非常流行。毫无疑问，互联网改变了我们的生活。但影响我们对互联网产生兴趣的一个方面是，计算机往往处于空闲状态，等待我们点击网页或按键。此外，也不可能允许如此昂贵的计算机仅仅做这么一点点工作。采用网络可以将所有计算机连接到一起，目前可能需要重点考虑在各种不同的计算机间进行分布的问题。

数据库应用系统依据自然地理分布进行开发，而且几乎所有的公司和组织机构都有数据库访问的需求。数据库是一种希望大家共享的东西。分享数据库的人员数目和企业数目可能会有变化，但多数情况是需要通过各种系统传递和访问信息。现在，即便要实现一个两层的数据库模型，也需要将表示层从服务器移到客户机，从而实现工作的“分布”。让客户机处理更多的工作是很有吸引力的想法。这样一来，服务器就可把大多数工作量放到客户机，从而处理更多的请求。

为了集成大量的 API 和技术，需要传递一种可运行的分布式数据库应用系统调用。分布式应用程序的重要性无需过多解释，但要真正实现它，却需要具备联网知识、数据库编程知识、系统管理知识、GUI 编程知识、系统编程知识和网络编程知识——在所有这些领域中没有通用的知识。某一领域的专业人员可能对另一领域一无所知。组建一个了解所有这些领域的开发团队，是一件非常困难的事情。将所有人的开发成果有效地集成为一个最终系统，则是一件更困难的事情。

Java 提供了一个内容全面的 API 集合，可满足所有这些领域的要求。尽管其他语言也提供了类似的功能，但我想没有任何一种语言能把大量有用、齐全的类作为 API 核心程序的一部分来提供。

此为试读，需要完整PDF请访问：www.ertongbook.com

公司和组织机构跨越不同距离和不同种类操作系统访问数据的需求日益增强。电子商务的需求是在不同的企业间进行通信，而且无需识别硬件或者软件。由于 Java 技术在互联网编程和交叉平台上具有“编写一次、到处运行”的特性，所以非常适合解决此类问题。另外，借助于全球范围的协作，科学和技术运算也可以通过此技术获益。

现在，市场需要易于扩展的应用系统，以便对企业和客户快速变化的需求做出反应。目前，技术和应用系统的发展速度可以说是一日千里。显然，为了维护各种生产环境，通常需要将新技术和旧技术、新操作系统和已有的操作系统融合起来。另外，不仅应用系统会随着技术原因快速改变，而且处理事务的方法也会随之快速改变。为了适应这种变化，应用系统也需要采用同样的方式设计。面向对象的程序设计和 Java 技术能够协助构建此类应用系统。

我们可能仍需在已有的系统和资源间进行权衡。现在，许多公司和企业仍然需要早期昂贵的系统，或者由于垂直申请以及（或者）巨大的培训投资等原因，在一些相关部门仍然存在早期的应用程序。通常也会因为政治方面的原因，采用一种技术来替代另一种，而且这种技术可能会一个部门、一个部门地改变。企业员工利用丰富的电子表格、电子邮件、文本文件以及数据库，基于个人需要和技术构建自己的应用程序。

通常在这种环境中，可能会有一些分散在企业中的程序员，他们在各种平台上用不同的语言编写代码。早期让信息从一个部门流向另一个部门的努力，可能会让一些拼凑的系统获得发展。这样虽能维持运行，但当企业的需求改变时，由于害怕破坏旧系统，新系统的实现将会越来越困难。更不要说由于喜欢别人的系统就把他或她的系统降低（没人会感到舒适），这样做可能会失去他们原先拥有的控制权。

本书的关键在于，确保应用系统能够对企业需求的改变快速做出反应。应用系统在结构上的设计也应该体现这一点。许多组织（包括新组建的、非盈利的、媒体的、科学的/工程的组织，以及其他组织）也许会有预算方面的限制，而且/或者由于是非传统的商业模式，无法采用即拆即用的解决方案，或者需要非常昂贵的解决方式。此外，系统可能需要综合多种类型的数据存储方法，将各部门的信息集中到信息中心。

Java 技术可以帮我构建系统吗

本书可以回答这个问题，而且将要设计一个新系统，它可以满足各种用户的需求，同时保持很好的灵活性。我们可以利用基于分布式计算模型的 Java 技术，通过构建和设计商业应用程序和系统实现这个新系统。作为一种网络编程语言，Java 的功能非常强大，它的跨平台特性和天生可以与各种供应商数据库对接的能力，可以让实现此系统成为可能。此外，为 Java 环境日常添加的 API 程序，可以让采用 Java 语言比采用其他语言更具吸引力。不夸张地说，Java 涵盖了计算技术的方方面面，从 XML 到 3D、从声音到 Jini。从来还没有一种编程语言能够像 Java 这样得到广泛的应用和 API 支持，更不用说这些 API 的源程序

都是可以获得的。

本书的读者对象

您是否：

- 想要设计一个分布式数据库系统？
- 在管理一个 IS/IT 部门？
- 是一名软件开发人员，正在深入研究一些新的技术，以有助于解决一个特殊的问题？
- 是一名正在学习编程的人，而且厌倦了简单的范例程序，希望了解如何将各种程序集成到一个简单的实用系统中？

如果您属于上述任何一种情况，那么阅读本书将会对您有所帮助。

如果您是软件开发人员，本书可以作为构建分布式系统的指南。如果您是系统管理员，本书将协助您理解系统管理和系统维护究竟是什么。如果您是一个 IT/IS 经理，就能对此类系统的需求有深入的了解，为制定项目的开发策略以及人员/预算需求提供帮助。

无论关注的是哪一点，阅读本书将有助于您理解这些问题错综复杂的关系，以便构建一个系统。即使您不愿意亲自开发核心程序，宁愿用集成的一体化方案来代替，看完本书后，也可以从另一个角度评估这些系统，甚至可以亲自编写一段“胶水”程序将它们黏合在一起。

本书的结构

每章的开头都有一个内容概要，以便读者快速了解本章内容。有几章包含有技术概览、编程、管理、支持以及配置等小节。如果您觉得不重要，可以放心地跳过这些内容。

对于编程人员，在浏览不同的 API 说明前，假定您对核心 API 已经非常熟悉。也就是说，您已经读过相关书籍，并厌倦了某些范例，但对如何将 API 组合在一起有疑问。为了这个目标，您需要通读全书，了解如何将不同的 Java API 结合到一起，以便创建实际程序。在这种情况下，即便您不希望亲自实现完整的应用系统，可能也会发现本书非常有价值。理解了不同组件协同工作的基础知识，在评估第三方解决方案及（或者）开发库文件时，具有很大的帮助。

另外，本书还将介绍如何将构建的软件集成到现有的、著名的、开放的源程序和商业系统中。读完本书后，您可获得对决策有帮助的所有必要的信息，以决定系统中的哪些部分需要构建，哪些部分可以采用集成的方式。为了充分利用本书，您应该熟悉以下内容：

- 关系型数据库
- SQL（结构化查询语言——一种用于获取数据库信息的非过程语言）

- JDBC（Java 数据库连接——Java1.1 版本及核心 API 的一部分）
- HTTP 协议
- NFS（网络文件系统——Unix 环境下的通用文件共享模式）
- FTP 文件传输协议
- Samba（运行在 Unix 服务器上的一个 SMB 系统，为 Win32 客户机提供文件共享）
- SMTP（简单邮件传输协议——Internet 电子邮件的基础协议）

本书的内容

第 1 章，介绍了完全采用 Java 技术实现的分布式数据库解决方案；概括介绍服务器的配置，代理服务器的工作原理，以及各种客户机如何与系统交互。本章的重点是硬件无关性，并为判断系统的性能价格比提供一系列准则。

第 2 章，详细描述了数据库 API 的构建方法，为应用程序开发人员归纳出数据库的具体细节。本章还研究了将 SQL 嵌入 API 的方法。

第 3 章，主要介绍了 RMI 服务器的结构及其管理技巧。

第 4 章，通过构建一个采用数据库 API 的 Swing 应用程序，着手讨论客户端的开发。本章不仅讨论了使用远程对象和 Swing 软件包的常见问题，还着重讨论了例外处理。本章提供了几个特殊范例，利用数据库数据（目录、表单、树及组合框）填充 GUI 组件。

第 5 章，介绍了如何通过 Servlet 访问同一个数据库 API，让我们体验了重复使用软件的美妙感觉。并从管理和开发的角度重点介绍了 Apache JServ 和 Java 的 Web 服务器。

第 6 章，通过构建一个命令行客户程序，为脚本编程人员和网络开发人员提供访问数据库 API 的简单方法，从而利用程序的复用性满足系统的需要。

第 7 章，介绍了如何简化软件部署，以避免部署人员在各计算机间奔波之苦。这些任务可以采用一些著名的服务（HTTP、FTP、NFS 和 Samba）、巧妙的系统结构和脚本来实现。

第 8 章，讲述了如何在数据库中读取和写入 BLOB（大型二进制对象）。特别值得关注的是，本章提供了图像和其他多媒体内容——从 BLOB 到 Servlet 和应用程序。

第 9 章，举例说明了如何构建监视工具，协助进行管理和支持，还介绍了如何通过 RMI 访问系统调用。

运行示范代码

本书的所有范例均在 Red Hat Linux 6.1 环境下开发和测试。所用数据库是 Oracle 8 的 Linux 版本。所用 JVM 来自 IBM 的 Linux 1.1.8 版本。这些范例还采用了 *Blackdown.org* 站点的 1.1.6v5 JDK 和 Sun/Blackdown 1.2.2 Release Candidate 2 进行测试。

可以从下述站点下载 Linux 环境下的 JDK 工具：

- *www.ibm.com/java*——JDK 1.1.8 和 JDK 1.3
- *www.blackdown.org*——JDK 1.1.6v5（最好在 Oracle 中运行）
- *java.sun.com*——JDK 1.2.2

当然，这些范例在 Windows NT、Windows 98 和 Solaris 操作系统上也能顺利运行。Oracle 软件本身也支持 NT、Solaris 以及其他版本的 Unix。我们可以通过 *technet.oracle.com* 网站下载 Linux 或其他测试版本。

Oracle 也为 Windows 95 和 Windows 98 用户开发了一个版本，称为 Oracle Lite（如果希望测试该平台的范例，可以利用此版本）。Oracle Lite 的 JDBC 驱动程序与普通 Oracle 数据库不同。确保阅读有关文档。

如果你使用的是 JVM 1.1，需要分别下载 Swing 库文件。这些文件可在 *java.sun.com/jfc* 站点找到。

这些 Servlet 范例是针对 Apache JServ 1.1b2 开发的，需要和 Linux 操作平台的 Apache Web Server 1.3.9 一起运行，Apache JServ 是 Java 2.0 版本 Servlet 规范的实现。Sun 正在和 Apache 团体合作，以支持 Jakarta 2.2 规范。详情请访问 *jakarta.apache.org* 站点或 Sun 公司的站点 *java.sun.com*。这些站点有简单描述的 README 文件，可以引导我们运行范例程序。

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章　分布式数据库应用程序设计

本章主要内容

- 技术概览
- 数据库应用程序模型
- 两层模型
- N 层模型（3 层或更多）
- 两种模型的对比
- 移植到多层结构

本章介绍了多层系统及其体系结构的背景知识，目的是为程序员和项目管理人员提供一个理论基础，以便理解分布式系统。如果您已经掌握了这些内容，可以跳过本章，直接学习第 2 章。

1.1　技 术 概 览

1.1.1　理解系统

构建一个分布式数据库应用系统比开发简单的应用程序所涉及的东西要多得多。这意味着需要了解系统间如何通过网络进行交互；意味着需要弄清数据库理想运行的条件；意味着了解所有系统组件的优点和缺点，在某种程度上，尝试将问题分布到这些系统，以便发挥最佳性能。

因此，除非从头开始构建系统，否则就要进行大量的分析，找出现有系统值得保留的部分和系统的瓶颈所在。另外，成本和现有系统的要求可能会对如何构建系统产生影响。有时候，需要优先考虑系统的扩展能力；而有时候却需要考虑怎样才能在获得最大灵活性的同时，发挥出最佳性能。

像这类复杂的问题，最适宜用 Java 技术来解决。无论是集成老系统，还是从头构建新系统，采用 Java 技术可在一个相对短暂的开发周期中，构建出稳定而复杂的系统。然而，在正式编写程序前，需要先了解一些分布式应用系统的硬件问题。本章将介绍一些影响应用程序开发的硬件问题，回顾传统的数据库计算模型，让大家了解分布式系统的优越性，并熟练地掌握一些技术术语。

1.1.2 硬件和网络配置

构建系统前，需要了解清楚当前环境中的服务器和工作站，以及它们之间是如何连网的（100BaseT/10BaseT/千兆以太网？LAN/WAN？防火墙？代理？）。同样重要的是，在每台计算机上运行的操作系统是什么？服务器和工作站分别运行不同的操作系统是很常见的。不过，基于应用软件的需求，目前越来越多的人为客户机选配“另类”操作系统（非Win32）。特别是随着另类操作系统的发展，现在的工作站并不仅仅局限于 Win32，而是包括了 Solaris、Linux、FreeBSD、Win32 及 MacOS 等多种操作系统。服务器可能会运行 Solaris、True64、AIX、IRIX 等操作系统，甚至可以运行 Linux 和 FreeBSD 操作系统。我相信，随着应用程序需求的不断提高，将迫使计算机环境从同质走向异质，这种百花齐放的情况还会持续下去。但是，不管是否出现这种情况，我们都应提前准备好用 Java 技术来使用它们。编写本书时，我发现支持 Java 1.1 的操作系统至少包括：FreeBSD、Linux（Intel 及 Alpha）、HPUX、AIX、Irix、Tru64 以及 MacOS，当然还有 Solaris 和 Win32。

1.1.3 系统瓶颈

一旦从掌握的情况中归纳出要点，就能清楚当前操作环境中有几处瓶颈很难轻易解决，或者需要马上修改预算。了解这些情况，有助于判断是否需要尝试用软件解决这些问题。

1.1.4 为明天做好准备

我们还需要为将来做计划，设计的系统需要有足够的灵活性以应付未来的变化。现在使用的 Linux 服务器是否会在明年被 Solaris 服务器取代？是否会有一定比例的 Win32 操作系统要被 Linux 操作系统所取代？如果系统的任何一部分依赖于具体的平台，日后必然会带来问题。例如，如果采用了一种第三方应用程序服务器或者其他中间件，就会受限于供应商所支持的操作系统。即便您决定使用 Java 2，也只能局限于当前支持它的操作平台和其他软件。

例如，为了使用服务器端的应用程序，Oracle 8i 数据库有一个内置的 Java 虚拟机（JVM）。这种 Java 虚拟机为数据库作了优化，而且可以提高在线事务处理应用程序的性能。Java 虚拟机基于 Java 1.1.6 版。如果计划采用这种技术，在应用程序框架中，应该考虑维护两个不同 Java 版本所涉及的潜在问题。

刚开始编写本书时，只有 Java 1.1 能在各平台通用。Java 2 的许多 alpha 和 beta 版本可以用于 Win32 和 Solaris 外的其他平台，但 Java 2 还需要进一步完善。目前，如果想开发可靠的、易于配置的软件，必须采用 Java 1.1。幸运的是，服务器端并没有过多地限制，而且 Java 2 的许多特性（如 Swing 和 JNDI）都有独立的软件包，以便 Java 1.1 用户使用。

1.1.5　将系统连接到一起

把系统连接到一起需要两样东西：网络和 Java 技术。首先，确保可以访问所有计划连接的计算机。例如，可能需要通过代理与其他系统通信。也许可以自由访问一些系统，但由于这些系统过于繁忙，需要以对通信速度影响尽可能低的方式与它们通信。早期发现这些问题有助于设计出更好的系统。

1.2　数据库应用程序模型

让我们快速回顾一下目前流行的各种数据库应用程序模型，以便清楚地了解即将面临的一些系统问题。许多数据库开发人员共有的缺点，就是将硬件和系统的规划留给系统管理员。这样一来，在协调软件问题和硬件问题时，就会留下一个无人管理的巨大空间。开发人员有责任为管理员构建一个合理的数据库，因此需要从根本上了解系统是如何设计的。

为此，我们需要回顾一下常见的数据库应用程序模型。介绍了基础的硬件和网络问题，希望开发人员能够清楚地构建系统，并将其作为系统设计的参考。

1.2.1　两层模型

许多数据库应用系统使用两层模型。实际上，目前许多商业系统都采用这种模型，因为这种模型比较容易实现和设计。该模型的基本结构如图 1.1 所示。

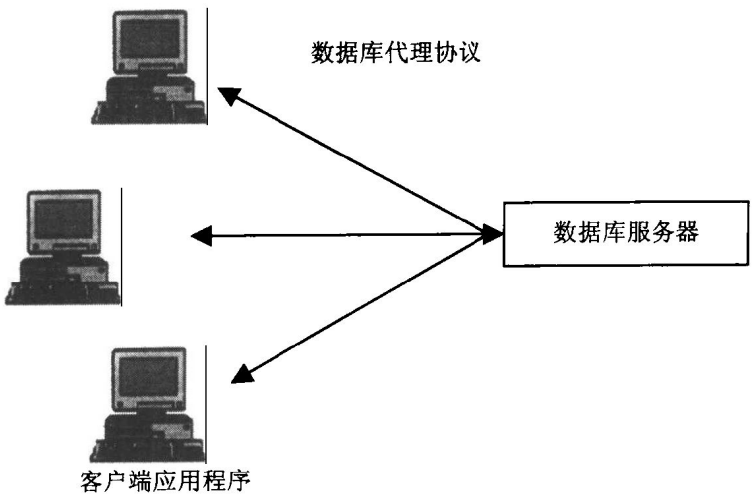

图 1.1　两层数据库模型

1. 硬件配置

大多数采用此模型的系统都有中型或大型的数据库服务器，通常是具有多处理器的计

算机，并运行数据库供应商支持的 Unix 版本。一般来说，此类系统往往需要投入大量的时间和金钱（多 CPU 计算机一般相当昂贵）。公司决定：必须升级计算机，并且每周 7 天、每天 24 小时运行。这是尝试提高性能的第一步，因而系统的运行会非常出色——至少我们希望如此。很可能是由于增加了系统的成本，它才会如此出色；因此，我们的任务是在不多花钱的前提下，让系统更加有效地工作。

提示 如果还有余钱，最好的改善方式就是购买内存。这样能显著地提高数据库应用程序的性能（假定您是一位熟练的数据库管理员，能利用所增加的内存修改数据库）。

2. 网络拓扑

在两层数据库模型中，当运行应用程序时，客户机程序会与数据库维持一种直接的固定连接。无论用户是否使用应用程序，此连接都会处于激活状态。连接一般由供应商协议组成（如 Oracle 的 SQLNet），并依赖于顶级传输协议（如 TCP/IP 或 DECNet）。这要在客户机和服务器间，建立一个复杂的连接。无论是否提供了传输协议，供应商协议会不停地进行错误校验和连接监视。对于大多数数据库开发人员来说，这是一个合理的、符合期望的情况，无需任何担心。然而，如您所见，仅仅是维护系统连接，就需要相当多的系统消耗。如果一个标准用户同时运行几个网络应用程序（Netscape、电子邮件等等），这对系统资源的消耗，将会影响工作站的运行。在某些操作系统上，甚至会引起工作站的崩溃。

3. 管理

为了配置采用两层数据库应用程序模型，每一台客户机均需安装数据库供应商提供的客户端软件，以及客户操作平台的编译库。安装客户端软件可能不太容易。例如，Oracle 的客户端软件需要配置几个用户环境变量，还需要编辑一些系统配置文件，以便指向主机的数据库。

若工作站运行了多个数据库应用程序，则可能会因安装了多个版本的客户端软件而出问题，有时是因为疏忽，有时是由于需要。

4. 支持——一个案例

数据库用户许可证一般根据可能同时使用数据库的用户数量来定义。因此，假设一个数据库定义了 5 个用户许可证。早上第一件事，财务部的用户启动了数据库应用系统并开始浏览。一个用户输入了一些数据，然后去吃午饭、开会、或者坐下来看 1 小时的邮件。突然，市场部产生了新业务，需要修改数据库，他们需要更新客户信息，或往数据库输入新账户。当他们尝试连接数据库时，由于数据库服务器统计到已有 5 个用户在线，所以数据库服务器拒绝了他们的连接请求。

此为试读，需要完整PDF请访问：www.ertongbook.com