Oracle 专家高级编程

[美] Thomas Kyte 著

袁勤勇 张玉魁 等译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2001-4306

内 容 简 介

本书是一本关于使用 Oracle 成功开发应用程序的工具手册，由 Oracle 公司的资深开发人员 Tomas Kyte 集自己多年开发经验编写。学习本书能帮助读者彻底理解 Oracle 的工作原理，并将 Oracle 作为一个强大的计算环境来使用，书中内容可以迅速解决大多数信息管理问题。书中选择了最重要的特性和技术，并结合实际范例进行讲解，不仅阐述了这些特性，而且还讨论了如何使用它们开发软件，并指出了潜在的缺陷。

本书内容包括 Oracle 数据库的基本概念；Oracle 数据库结构和实用程序；Oracle 数据库性能优化；高级 SQL 特性；用 interMedia、基于 C 的外部过程、Java 存储过程和对象关系特性实现 Oracle 数据库功能的扩展；Oracle 数据库安全管理的实现方式等。

本书适用于使用 Oracle 进行数据库开发的人员，包括有经验的 Oracle 开发人员、DBA、Oracle 项目管理人员等。

Thomas Kyte：expert one-on-one Oracle
EISBN：1-861004-82-6

书　　名：Oracle 专家高级编程
作　　者：[美] Thomas Kyte　著　袁勤勇　张玉魁　等译
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
责任编辑：郭东青
印 刷 者：北京大中印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：79.75　**字数**：2040 千字
版　　次：2002 年 4 月第 1 版　2002 年 4 月第 1 次印刷
书　　号：ISBN 7-302-05334-0/TP・3133
印　　数：0001～5000
定　　价：138.00 元

出版者的话

近年来，国内计算机类图书出版业得到了空前的发展，面向初级用户的应用类软件图书铺天盖地，但是真正有深度和内涵的高端图书不多。已经掌握计算机和网络基础知识的人们，尤其是 IT 专业人士迫切需要“阳春白雪”。IT 图书市场呼唤精品！

为了满足这种市场需求，清华大学出版社从世界出版业知名品牌 Wrox 出版公司引进了受到无数 IT 专业人士青睐，被奉为 IT 出版界经典之作的 Professional 系列丛书。这套讲述最新编程技术与开发环境的高级编程丛书，从头到尾都贯穿了 Wrox 出版公司“由程序员为程序员而著(Programmer to Programmer)”的出版理念，每一本书无不是出自软件大师之手。实际上，Wrox 公司的图书作者都是世界顶级 IT 公司(如 Microsoft，IBM，Oracle 以及 HP 等)的资深程序员，他们的作品既深入研究编程机理，传授最新编程技术，又站在程序员的角度，指导程序员拓展编程思路，学习实用开发技巧，从而风靡世界各地，被 IT 专业人士和程序员视为职业生涯中的必读之作。

作为国内 IT 出版社中最知名品牌，清华大学出版社与 Wrox 公司合作引进了这套 Professional 系列，然后迅速组织了一批相关领域的知名专家学者进行翻译，经过编辑人员认真细致的加工后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码并获得相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

前　言

本书中材料的灵感来自于我开发 Oracle 软件的经验，以及与 Oracle 开发人员一起工作，帮助他们构建可靠的、健壮的、基于 Oracle 数据库应用程序的经验。本书的内容反映了我每天的工作内容和人们每天所遇到的问题。

本书包括了我认为最相关的内容，也就是 Oracle 数据库及其体系结构。我可以写一本类似主题的书，来解释如何使用特定语言和体系结构来开发应用程序。例如，一个使用 Java Server Pages 与 Enterprise Java Beans 通信的应用程序，该应用程序使用 JDBC 与 Oracle 进行通信。但是，为了成功地构建此类应用程序，您真正需要的是理解本书的主题。本书介绍我认为对使用 Oracle 进行成功开发所应该普遍了解的内容，而不管您是使用 ODBC 的 Visual Basic 程序员，还是使用 EJB 和 JDBC 的 Java 程序员，或者是使用 DBI Perl 的 Perl 程序员。本书不支持特定的应用程序体系结构；它不会比较 3 层结构和客户-服务器结构。相反，本书介绍数据库可以完成的工作，以及您必须了解的数据库工作方式。由于数据库是应用程序体系结构的核心，因此本书应该拥有广泛的读者群。

本书内容

拥有许多开发选项的一个问题是指出哪一个选项是满足特定需要的最佳选择。每个人都希望拥有尽可能多的灵活性(拥有尽可能多的选项)，但是他们也希望事情简洁明了，换句话说就是容易。Oracle 为开发人员提供了几乎无限的选择。不曾有人说过："您不能在 Oracle 中做这个工作。"他们说："在 Oracle 中有多少种不同方式供您选择来完成这个工作"。我希望本书将帮助您作出正确的选择。

本书写给这样的读者，他欣赏选择，但也喜欢关于 Oracle 特性和功能的一些指南和实际的实现细节。例如，Oracle 拥有一个高效的特性，称为虚拟专用数据库(virtual private database)。Oracle 文档告诉您如何使用该特性，以及该特性能够做什么。Oracle 文档没有指出什么时候应该使用该特性，以及什么时候不能使用该特性，后者也许更为重要。它不会总告诉您其实现细节，如果您不清楚这些实现细节，就会被迷惑(我不是在谈 bug，而是指此特性的工作方式，及其真正的用途)。

本书读者对象

本书的读者对象是使用 Oracle 作为数据库后端开发应用程序的任何人。本书写给需要了解如何使用数据库进行工作的专业 Oracle 开发人员。本书特点决定了本书的很多部分也是 DBA(数据库管理员)所感兴趣的。本书的大多数范例使用 SQL*PLUS 来演示关键特性，因此您

不会找到如何开发真正“酷”的GUI，但您会学习Oracle数据库工作原理，其关键特性的作用，什么时候应该(不应该)使用它们。

本书写给希望事半功倍地使用Oracle的人，写给希望使用新方法使用现有特性的人，写给希望如何把这些特性应用到现实世界(不仅仅是如何使用特性的例子，而且还有为什么这些特性首先是相关的)的人。对本书感兴趣的另一类人，是负责管理Oracle项目开发人员的技术经理，从某种角度讲，他们理解为什么懂得数据库对于成功是至关重要的，这一点也相当重要。本书能够为要以正确的技术培训员工的经理提供必备的参考。

为了充分利用本书，您必须具有下列知识：

- SQL的知识。您不必是最好的SQL编程员，但良好的使用SQL的知识对您是有帮助的。
- 对PL/SQL的理解。这不是前提条件，但将帮助您“理解”范例。例如，本书将不会讲授如何编写一个for循环或声明一个记录类型，这些内容在Oracle文档和很多书籍中都有介绍。但是这并不是说，通过阅读本书您不会学到有关PL/SQL的知识。实际上，您将熟悉PL/SQL的很多特性，了解以前不知道的现有的程序包和特性。
- 熟悉第三代语言，例如C或Java。能够读写第三代语言代码的人将能够成功地阅读并理解本书中的范例。
- 熟悉Oracle服务器概念手册。

关于上述最后一点的说明：由于Oracle文档集的篇幅庞大，很多人发现Oracle文档在某种程度令人生畏。如果您刚刚起步，或根本没有阅读过Oracle文档，那么最好先从Oracle 8i Concept手册学起。该书大约有800页，涵盖了您需要知道的很多主要的Oracle概念。它可能没有给出每一个技术细节(这要用10 000～20 000页文档介绍)，但是它介绍了所有重要的概念。该手册涵盖下列主题：

- 数据库结构，如何组织、存储数据。
- 分布式处理技术。
- Oracle的内存体系结构。
- Oracle的进程体系结构。
- 将使用的模式对象(表、索引、聚簇等等)。
- 内嵌的数据类型和用户定义的数据类型。
- SQL存储过程。
- 事务工作原理。
- 优化器。
- 数据完整性。
- 并行性控制。

本书在需要的时候，会对这些主题进行反复讲解。它们是基础，如果没有这些知识，您的Oracle应用程序很容易失败。这里建议您通读该书，并理解这些主题。

本书结构

为了帮助您阅读本书，本书组织为6个相对独立的部分(参见下面的描述)。这不是严格的

划分，它们将帮助您迅速找到最需要的内容。本书共有 23 章，每一章实质上都是一本微型书，即一个独立部分。有时候，我会在其他章中引用范例或特性(尤其是“安全性”部分，它更多地依赖在以前章节中建立的范例和概念)，但是您可以从本书中选择任何一章，并单独阅读。例如，读者不必为了阅读或使用第 14 章而首先阅读第 10 章。

每一章的格式和风格实际上是相同的：

- 对特性或功能的简单描述。
- 为什么(或为什么不)要使用它。概括介绍考虑使用该特性的时间，以及什么时候不要使用该特性。
- 如何使用该特性。这不仅仅是 SQL 参考的简单副本，而且是分步骤详细描述该特性。这些内容正是读者所需要掌握的，这些内容也是您必须完成的操作，这些内容又是您必须掌握的关键知识。这部分的内容包括：
  - 如何实现它
  - 范例
  - 调试特性
  - 使用该特性的警告
  - 事先处理错误
- 小结

本书有很多范例和大量的代码，所有这些都可以在 http://www.wrox.com上下载。下面是对每一部分内容的详细介绍。

理解数据库

- **第 1 章“开发成功的 Oracle 应用程序”**。本章陈述了数据库编程的基本方法。所有的数据库都不会相同，为了成功地、按时地开发数据库驱动的应用程序，您必须确切地理解特定的数据库能够完成的工作，它是如何做的。如果您不知道数据库的功能，那么就有可能做那些重复性的工作，即开发数据库已经提供的功能。如果您不知道数据库如何工作，就可能开发性能拙劣、不能按预订方式操作的应用程序。

本章介绍了一些由于缺乏对数据库的基本理解而导致项目失败的实际例子。使用这种举例方法，本章讨论了开发人员需要理解的基本特性和功能。最低要求是您不能把数据库视为简单地返回答案的黑盒子，由数据库自身维护伸缩性和性能。

- **第 2 章“体系结构”**。Oracle 数据库是高度复杂的工具。每次您连接到数据库，或发出一个 UPDATE 命令时，在后台发生许多进程，以保证应用程序运行平稳，数据的完整性得到维护。例如，如果需要，数据库保证拥有足够的信息以便能够把数据恢复到其本来的初始状态。它将高速缓存程序数据，并自动地在合适时重用数据，等等。 在大多数时间里，所有这些操作都是透明的(至少对于开发人员来说)，但是发生故障时，一半的工作是知道在何处找到问题。

本章介绍 Oracle 体系结构的 3 个主要部件——内存结构(尤其是系统全局区域(System Global Area))、物理进程及其文件集(参数文件、重做日志文件等等)。理解 Oracle 体系结构是理解 Oracle 实现特定特性的独特方式及其如何影响应用程序的独特方式的基础。

- **第 3 章“封锁和并行性”**。不同数据库具有不同的运行方式(在 SQL Server 中运行良好，

在 Oracle 中不一定能运行)，理解 Oracle 实现锁定和并行控制的方式，对于应用程序的成功是绝对至关重要的。

本章讨论了 Oracle 解决这些问题的基本方法、能够应用的锁定类型(DML、DDL、锁存器(latches)……)以及没有注意实现封锁时产生的问题(死锁、阻塞、锁定扩大(escalation))。并行性控制部分讨论了 Oracle 提供的控制用户访问并修改数据库的功能。

- **第 4 章“事务”**。事务是所有数据库的基本特性，这是数据库区别于文件系统的特点之一。但是，事务常常被错误理解，许多开发人员甚至不知道他们竟然没有使用事务。本章解释了在 Oracle 中应该如何使用事务，也揭示了在用其他数据库进行开发时的一些“坏习惯”。特别是，强调原子性(atomicity)的含意，及其如何影响 Oracle 语句。接着讨论事务控制语句(COMMIT、SAVEPOINT、ROLLBACK)、完整性约束和分布式事务(两阶段提交)。最后，探讨有关使用事务的一些实际问题：如何记录事务以及重做(redo)和撤消(undo)的作用。

数据库结构和实用程序

- **第 5 章“重做和回滚”**。可以说，与 DBA 相比，一般开发人员没有必要理解同样多的重做和回滚(rollback)的细节，但开发人员也需要理解重做和回滚在数据库中的作用。在第一次定义重做以后，考查了 COMMIT 命令的确切功能。同时也考虑了诸如产生多少重做、日志的关闭、重做的分析等等问题。

在本章有关回滚的部分，在介绍 Set transaction SQL 语句之前，首先介绍是什么毫无例外地产生撤消操作。这一般用于选取一个大回滚段进行一些大型操作。然后，详细介绍臭名昭著的“ORA-01555 snapshot too old”错误，探讨其原因及其解决方案。

- **第 6 章“数据库表”**。Oracle 支持很多类型的表。本章介绍各种类型的表：堆组织表(heap organized) (默认情况下的普通表)、索引组织(index organized)表、索引聚簇(index clustered)表、散列聚簇(hash clustered)表、嵌套(nested)表、临时(temporary)表和对象(object)表，并讨论了什么时候、如何、为什么使用它们。在大多数时候，堆组织表就足够使用了，但是您必须能够意识到什么时候使用其他类型的表更为合适。

- **第 7 章“索引”**。索引是应用程序的一个至关重要的方面。数据库程序的正确实现需要深入了解数据及其如何分布、如何使用的知识。索引常常被视为应用程序开发中事后规划的内容，从而使性能受到损害。

本章详细介绍不同类型的索引，其中包括 B*Tree、位图、基于函数和应用程序域索引，并讨论什么地方应该或不应该使用它们。还将回答在“常见问题”部分中的问题，例如“索引在视图上能运行吗？”、“为什么没用使用索引？”。

- **第 8 章“导入和导出”**。导入和导出是 Oracle 提供的两个最老的工具，它们用于从一个 Oracle 实例中抽取表、模式(schema)或整个数据库定义，并导入到另一个实例或模式中，但是很多开发人员并不知道如何使用这两种工具。这里介绍诸如大型导出、构建子集和传输数据并使用它们作为备份或重组工具等主题。本章最后介绍使用这些工具时潜在的缺陷和问题。

- **第 9 章“数据装载”**。本章重点介绍 SQLLDR，并介绍使用该工具装载并修改数据库中数据的各种方法。 本章内容包括如何装载定界的数据，如何更新现有行、插入新行，如何卸载数据，如何在存储过程中调用 SQLLDR。SQLLDR 是一个已建立的关键工具，但就实际使用而

言，它还会导致许多问题。

性能

● **第 10 章“优化策略和工具”**。在本章中将详细介绍优化 Oracle 应用程序的方法，然后提供了使用优化工具的高度实用的指南和技巧。开放部分集中介绍应用程序优化，包括下列专题：绑定变量(bind variable)和分析、SQL_TRACE、TIMED_STATISTICS 和 TKPROF、DBMS_PROFILER，以及对应用程序进行日志操作的重要性。在应用程序完成彻底的优化以后，注意力应该转向数据库，尤其是在优化中使用的 StatsPack 实用程序组和 V$表。

● **第 11 章“优化器方案稳定性”**。使用 Oracle 8i(及其更高版本)的开发人员现在可以存储一组“服务器提示”，它称为优化器方案，详细描述如何在数据库中执行特定的 SQL 语句。显然，这有利于性能的提高，本章详细介绍如何生成大纲，如何管理大纲。

高级 SQL 特性

● **第 12 章“分析函数”**。有时候，针对数据库可以非常有规律地提出某些问题，但是直接使用 SQL 难于编写回答它们的查询，该查询也无法总能迅速地执行。Oracle 8.1.6 引入了分析函数。这些函数扩展了 SQL 语言，容易编码此类查询，并通过功能类似的直接 SQL 查询显著提高性能。本章阐述了分析函数运行方式、完整的语法(包括函数、分区、窗口子句)，然后给出了一个使用这些函数的完整的实际例子。

● **第 13 章“物化视图”**。某些“聚集”查询为了产生答案，必须处理数以千兆的数据。性能的含意很清晰，尤其是进行常见查询的时候，就是每当询问该问题时，将处理大量的数据。利用这个特性，我们简单地预先做一些工作。即在物化视图中汇总回答特定查询所需的数据，后续的查询重定向到该汇总数据。此外，数据库能够识别使用汇总数据的类似查询，并自动地重写查询。本章讨论了它的工作原理，建立物化视图的方法，包括约束、维数和 DBMS_OLAP 软件包的使用。

● **第 14 章“分区”**。分区旨在简化大型表和索引的管理。它是通过“分割-占据”逻辑实现的，也就是把表和索引分成更小更易管理的小片。这是 DBA 和开发人员必须合作，以使应用程序的可用性和性能最大化的领域。本章介绍表分区和索引分区。还将介绍使用局部索引和全局索引的分区。前者常见于数据仓库，后者常见于 OLTP 系统。

● **第 15 章“自治事务”**。利用该特性，可以创建子事务，子事务可以独立于父事务提交或回滚变化。还将介绍自治事务适用的场合，例如审核修改安全信息的“非法”企图，以避免使表发生突变，或作为在触发器中执行 DDL 的方法。本章还将讨论诸如事务控制、作用域、结束自治事务和存储点等。

● **第 16 章“动态 SQL”**。在本章中，对在程序中使用 SQL 语句的两种方法进行了比较：“标准”的静态 SQL 方法和动态 SQL 方法。动态 SQL 是在运行时执行的 SQL 语句，而在编译时并不知道它们。还将介绍在程序中使用动态 SQL 的两种方法，也就是使用所提供的内置程序包 DBMS_SQL 和本机动态 SQL。本机动态 SQL 是在 PL/SQL 中使用的声明方法。选择其中一种的理由很多，例如，在编译时是否知道绑定变量，是否知道结果，给定语句是否在会话中执行一次或多次，等等。本章将详细讨论这些问题。

可扩展性

- **第 17 章“interMedia”**。本章重点介绍 interMedia Text。本章没有详细介绍如何使用 interMedia Text，而是介绍什么是 interMedia Text，它提供的功能，启用该功能的数据库特性。本章介绍如何搜索文本，管理各种文档，索引来自多种数据源的文本，搜索 XML 应用程序。本章最后给出了 interMedia 的说明，包括索引和数据库外的索引信息的同步化。
- **第 18 章“基于 C 的外部过程”**。Oracle 8.0 中引入了在数据库服务器上实现过程的能力，这种过程可以用不同于 PL/SQL 的语言，例如 C 或 Java 编写。它们被称为外部过程。在本章中，将从体系结构的角度介绍基于 C 的过程。还将演示如何配置服务器以使用这些过程，测试安装，并建立一个传递、处理各种类型变量的范例过程。同时研究了 LOB to File (LOB_IO)外部过程，该过程把 CLOB、BLOB 和 BFILE 写到硬盘上。
- **第 19 章“Java 存储过程”**。通过明智地应用少量 Java，可以获得大量的有用功能，而这些功能超出了 PL/SQL 所能实现的功能。在本章中，将介绍一个实际例子，该例子说明了该功能有用的场合，如获得目录列表或运行一个操作系统命令。最后，还将提供在试图使用该特性时可能遇到的问题以及一些解决方案。
- **第 20 章“使用对象关系特性”**。在数据库中，对象关系特性可用性极大地扩充了开发人员可以使用的数据类型集。但是，什么时候应该使用它？同样，什么时候不应该使用它？在本章中，将说明在系统中加入新数据类型(这里创建了一个新的 PL/SQL 数据类型)的方法，并介绍集合的独特用法。最后，将介绍对象关系视图，该部分内容适用于想使用对象关系特性，但仍然为应用程序提供关系视图的读者。

安全性

- **第 21 章“精细存取控制”(Fine Grained Access Control)**。该特性允许您在运行时把谓词附加到发给数据库的所有查询。该特性在服务器上实现，意味着能够访问数据库的任何应用程序都可以使用该特性。使用该特性的深层理由包括易维护和以 ASP 方式宿主应用程序的能力。通过测试一对范例您会弄清它的工作原理。其中一个范例基于安全策略的实现，另一个范例使用应用程序上下文。本章的最后是警告信息，其中包括参照完整性、导入导出问题，以及错误信息。
- **第 22 章“多层身份验证”**。在本章中，将讨论 Web 的作用，它导致了客户在真正访问数据库之前向中间层应用程序服务器提供证书。将解释这个特性是如何实现的，以及如何工作的。还将介绍如何授权，如何审核代理账户。
- **第 23 章“调用者和定义者权限”**。从 Oracle 8i 开始，可以为一个存储过程的不同用户，授予不同的权限。利用调用者权限，可以开发存储过程，使存储过程能够在运行时以调用者特权执行。还将探讨为什么该特性很有用，例如开发通用的实用程序和数据字典应用程序时很有用；也将探讨为什么在大多数情况下，定义者权限仍然是正确的选择。在“如何工作”部分，将介绍在编译定义者和调用者权限过程时实际发生的操作。

附录

- **附录 A“必须提供的软件包”**。开发过程中可能会忽略其中的很多软件包，或者说它们

的目的没有真正被理解。在此，这里将讲解如何使用并扩展它们，帮助您搞清这些疑惑。

客户技术支持

我们努力使本书尽可能地准确并令读者便于阅读，但是本书的实际效果取决于您的心得体会。读者意见可通过 email：feedback@wrox.com与我们联系，把您的批评、建议告诉我们。

源代码和更新代码

在使用本书中的例子时，您可能决定宁愿手工输入所有代码。许多读者这样做的理由是，这是熟悉所使用的编码技巧的好方法。

不管您是否输入代码，我们都把源代码放在我们的 web 站点：

http://www.wrox.com/

如果您要输入代码，也可以使用我们的文件来检查应该得到的结果，如果您认为输入有错误，您可以与站点代码进行比较。如果您不想输入代码，那就从我们的站点下载源代码吧。

无论使用哪种方法，它们都有助于您更新和调试。

勘误表

我们努力使文本和代码没有错误。但是，人难免犯错误，一旦错误被指出并更正，我们就反馈给读者。在http://www.wrox.com上有本书的勘误表。如果您发现了尚未报告的错误，请告诉我们。

我们的 web 站点也提供其他信息和支持，其中包括所有书中的源代码、范例章节、即将出版的书、文章、相关主题意见的预览。

序　言

本部分中，将描述如何建立能运行本书中范例的环境。主要介绍下列内容：

- 怎样建立 SCOTT/TIGER 演示模式(schema)
- 需要建立和运行的环境
- 怎样配置 AUTOTRACE(一个 SQL*PLUS 工具)
- C 编译器的建立
- 我在本书中使用的编码约定

安装 SCOTT/TIGER 模式

SCOTT/TIGER 模式已经多次应用于您的数据库中。它包括在典型安装中，但它不是数据库的强制性。您可以把 SCOTT 例子安装到任何数据库账户中，SCOTT 账户的用法没用任何神秘之处。如果愿意，您可以把 EMP/DEPT 表直接安装到自己的数据库账户中。

本书中很多例子都利用 SCOTT 模式中的表。如果您希望使用这些例子，那么您将需要这些表。如果您在一个共享的数据库上工作，那么把这些表的副本安装到不同于 SCOTT 的账户中，就是明智之举，这样可以避免其他用户使用和修改相同数据所带来的副作用。

为了创建 SCOTT 演示表，需要进行如下的简单操作：

- cd [ORACLE_HOME]/sqlplus/demo
- 作为任何用户连接时运行 demobld.sql

demobld.sql 脚本将创建并安装 5 个表。完成后，它自动退出 SQL*PLUS ，因此在运行完该脚本以后，SQL*PLUS 就会消失，对此不必惊讶，本来就应该这样。

标准演示表包括施加参照完整性的标准约束。有些例子指望它们拥有参照完整性。我们建议，在运行完 demobld.sql 脚本以后，也要执行下列语句：

```
alter table emp add constraint emp_pk primary key(empno);
alter table dept add constraint dept_pk primary key(deptno);
alter table emp add constraint emp_fk_dept
                                    foreign key(deptno) references dept;
alter table emp add constraint emp_fk_emp foreign key(mgr) references emp;
```

这就完成了演示模式的安装。如果您为了清理而要取消该模式，则可以简单地执行 [ORACLE_HOME]/sqlplus/demo/demodrop.sql 脚本。它将取消 5 个表，并退出 SQL*PLUS。

SQL*PLUS 环境

本书中大多数例子设计成能完全在 SQL*PLUS 环境中运行。值得注意的例外是基于 C 的

例子，它们需要与Oracle分开的C编译器(参见后面的C编译器部分)。与之不同的是，SQL*PLUS是需要建立和安装的惟一环境。SQL*PLUS 提供了很多有用的选项和命令，贯穿本书我们将频繁地使用它们。例如，本书中几乎所有的例子都以某种方式使用 DBMS_OUTPUT。为了使DBMS_OUTPUT 能工作，必须使用下列 SQL*PLUS 命令：

```
SQL> set serveroutput on
```

如果您像我一样，您马上就会厌倦每次都要进行输入工作。幸运的是， SQL*PLUS 允许建立脚本文件 login.sql，这个脚本在每次启动 SQL*PLUS 会话时执行。而且，它还允许建立一个环境变量 SQLPATH，从而能够找到启动脚本，而不管存储它的目录。

本书中所有范例使用的 login.sql 脚本如下所示：

```
define _editor=vi

set serveroutput on size 1000000

set trimspool on
set long 5000
set linesize 100
set pagesize 9999 column plan_plus_exp format a80

column global_name new_value gname
set termout off
select lower(user) || '@' ||
decode(global_name, 'ORACLE8.WORLD', '8.0', 'ORA8I.WORLD',
'8i', global_name ) global_name from global_name;
set sqlprompt '&gname> '
set termout on
```

其中：

- DEFINE EDITOR=VI 为 SQL*PLUS 设置默认编辑器。 您可以把它设置为您喜爱的文本编辑器(不是字处理器)，例如 Notepad 或 EMAC。
- SET SERVEROUTPUT ON SIZE 1000000 使 DBMS_OUTPUT 在默认时启用(因此，不必每次都输入它)。它也把默认缓存大小设置得尽可能得大。
- SET TRIMSPOOL ON 当假脱机操作文本时，保证文本行将没有空格，而不是固定宽度。如果把它设置为 off (默认设置)，假脱机行的宽度由 linesize 设置。
- SET LONG 5000 设置在选择 LONG 和 CLOB 列时显示的字节的默认数。
- SET LINESIZE 100 把 SQL*PLUS 显示的行宽设置 100 个字符。
- SET PAGESIZE 9999 把 pagesize 设置为一个很大的数。Pagesize 控制 SQL*PLUS 输出头的数量，每一页都会得到一组头。
- COLUMN PLAN_PLUS_EXP FORMAT A80 设置 EXPLAIN PLAN 输出的默认宽度，使用 AUTOTRACE 可以收到 EXPLAIN PLAN 输出。宽度 a80 通常足够容纳整个计划。

login.sql 下一部分建立 SQL*PLUS 提示符，由下面的行开始：

```
column global_name new_value gname
```

该指令让 SQL*PLUS 取出它所接收的名为 GLOBAL_NAME 列的最新值，并把它放置在取代变量 GNAME 中。然后我们拥有下列查询：

```
select lower(user) || '@' ||
decode(global_name, 'ORACLE8.WORLD', '8.0', 'ORA8I.WORLD',
'8i', global_name ) global_name from global_name;
```

这可以从数据库中选择 GLOBAL_NAME，使用 DECODE 函数为一些更常见的数据库实例指定熟悉的名称，并把它与当前登录所用的名字连接起来。最后，在 SQL*PLUS 提示符中反映这个信息：

```
set sqlprompt '&gname>'
```

这样提示符就会变成：

```
tkyte@TKYTE816>
```

利用这种方式，我知道我的身份和位置。另一个非常方便的脚本是 connect.sql ，它与 login.sql 在同一个目录中：

```
set termout off
connect &1
@login
set termout on
```

在 SQL*PLUS 初始启动时，它只执行 login.sql 脚本。通常，要在每次连接时执行它。使用：

```
tkyte@TKYTE816> @connect scott/tiger
```

而不是 CONNECT SCOTT/TIGER。这样我的提示符总是正确设置，如同其他所有设置一样，如 SERVEROUTPUT。

在 SQL*PLUS 中建立 AUTOTRACE

贯穿本书，我们通过获取 SQL 优化器所使用的执行计划报表以及其他有用的执行统计数据，来监控所执行的查询性能，这是非常有用的。Oracle 提供了一个称为 EXPLAIN PLAN 的工具，该工具使用 EXPLAIN PLAN 命令，允许我们生成执行计划输出。

欲了解有关解释 EXPLAIN PLAN 输出的信息，请参见 Oracle8i Designing and Tuning for Performance guide。

然而，SQL*PLUS 提供了 AUTOTRACE 工具，该工具允许查看已执行查询的执行计划、它们使用的资源，而不必使用 EXPLAIN PLAN 命令。报表在成功执行 SQL DML (即 SELECT、DELETE、UPDATE 和 INSERT)语句后生成。本书广泛地使用了该工具。有多种配置

AUTOTRACE 工具的方法。下面是使用步骤：

- cd [ORACLE_HOME]/rdbms/admin
- log into SQL*PLUS as SYSTEM
- run @utlxplan
- run CREATE PUBLIC SYNONYM PLAN_TABLE FOR PLAN_TABLE;
- run GRANT ALL ON PLAN_TABLE TO PUBLIC;

可以用 GRANT 替代 GRANT…TO PUBLIC 授予特定用户。通过给 PUBLIC 角色授予特权，任何人可使用 SQL*PLUS 进行跟踪。这使得每个用户不必都安装他们所拥有的计划表。该备选方案用于在想使用 AUTOTRACE 工具的每个模式里运行@UTLXPLAN。

下一个步骤是创建并授予 PLUSTRACE 角色：

- cd [ORACLE_HOME]/sqlplus/admin
- log into SQL*PLUS as SYS
- run @plustrce
- run GRANT PLUSTRACE TO PUBLIC;

再次重申，如果您愿意，您可以在 GRANT 语句中用特定用户替代 PUBLIC。

控制执行计划报表

可以通过设置 AUTOTRACE 系统变量来控制在执行计划报表中显示的信息(见表 1)。

表 1

列　名	描　述
SET AUTOTRACE OFF	不产生 AUTOTRACE 报表。这是默认值
SET AUTOTRACE ON EXPLAIN	AUTOTRACE 报表只显示优化器执行路径
SET AUTOTRACE ON STATISTICS	AUTOTRACE 报表只显示 SQL 语句执行统计运算
SET AUTOTRACE ON	AUTOTRACE 报表包括优化器执行路径和 SQL 语句执行统计运算
SET AUTOTRACE TRACEONLY	类似于 SET AUTOTRACE ON，但是限制用户查询输出的打印结果

解释执行计划

执行计划显示 SQL 优化器的查询执行路径。执行计划的每一行代码都有连续的行号。SQL*PLUS 也显示父操作的行号。

执行计划包含以表 2 所示的顺序显示的 4 列。

表 2

列　名	描　述
ID_PLUS_EXP	显示每一个执行步骤的行号
PARENT_ID_PLUS_EXP	显示每一个步骤与其父操作之间的关系。对于大型报表，该列非常有用
PLAN_PLUS_EXP	显示报表的每个步骤
OBJECT_NODE_PLUS_EXP	显示数据库链接或所使用的并行查询服务器

使用 COLUMN 命令可以改变列的格式。例如，要停止执行将要显示的 PARENT_ID_PLUS_EXP 列，就输入：

```
SQL> column parent_id_plus_exp noprint
```

C 编译器

依据操作系统不同，Oracle 支持的编译器也不同。在 Microsoft Windows 上，使用 Microsoft Visual C/C++，且只使用该工具的命令行部分(nmake 和 cl)。没有一个例子使用 GUI 开发环境。但是，如果您愿意，它们也可以在 GUI 环境中开发。配置合适的 include 文件，链接合适的库，这些都是您的职责。本书的每一个 makefile 都很小，而且简单，很容易就会明白需要哪些 include 文件和库。

在 Sun Solaris 系统上，所支持的 C 编译器是 Sun SparcsWorks 编译器。我只使用命令行工具：make 和 cc 来编译脚本。

编码约定

本书中使用了一些编码约定，其中必须明确指出的是在 PL/SQL 中命名变量的方法。例如，考虑如下所示的代码：

```
create or replace package body my_pkg
as
    g_variable varchar2(25);

    procedure p( p_variable in varchar2 )
    is
        l_variable varchar2(25);
    begin
        null;
    end;
end;
/
```

其中有 3 个变量：一个全局包变量 G_VARIABLE，另一个过程的正规参数 P_VARIABLE，第三个是局部变量 L_VARIABLE。按照其作用域命名变量，所有的全局变量都以 G_开头，所有的参数都以 P_开头，局部变量都以 L_开头。 这样做的主要原因是把 PL/SQL 变量与数据库表中的列区分开。例如下面的过程：

```
create procedure p( ENAME in varchar2 )
as
begin
    for x in ( select * from emp where ename = ENAME ) loop
```

```
        Dbms_output.put_line( x.empno );
    end loop;
end;
```

它将输出 EMP 表中的每一列。SQL 看到 ename = ENAME 语句，比较 ename 列和它本身。我们可能看到 ename = P.ENAME ，即用过程名限定对 PL/SQL 变量的应用，但它太易于忘记，从而导致错误。

用作用域来命名变量，这样就能容易地把参数与局部变量和全局变量区分开。另外，这样也消除了列名和变量名的模糊性。

其他问题

本书中的每一章都是自成体系的。在每一章的开始，我取消我的测试账户，并重新创建账户。也就是说，每一章都开始于空模式——没有对象。如果您从头到尾学习每章中的例子，则必须执行上述操作。在我为了弄清作为一个命令的副作用而产生的对象而查询数据字典或其他数据时，常常会迷惑：对象是否是其他例子留下的。 同样，我总是倾向重用表名(尤其是表 T)，因此如果不是每章使用新模式，可能遇到冲突。

此外，如果您试图手工取消例子中创建的对象(与之相反的是，通过使用 drop user USERNAME cascade 来取消用户，然后重新创建)，那么您必须意识到 Java 对象是在不同的情况下创建，因此如果运行第 19 章的范例，如下所示。

```
tkyte@TKYTE816> create or replace and compile
  2  java source named "demo"
  3  as
  4  import java.sql.SQLException;
...
```

为了取消对象，必须运行：

```
tkyte@TKYTE816> drop java source "demo"

Java dropped.
```

记住要用双引号把 Java 对象的标识符括起来，因为这些对象是在不同情况下创建和保存的。

目　录