Prentice Hall PTR Oracle系列 · 关于Oracle的独特见解

KEVIN OWENS

Programming Oracle Triggers and Stored Procedures

Oracle

触发器与存储过程高级编程

(第3版)

- 数据库开发人员的完全教程
- 全面描述 PL/SQL 语言
- 详细介绍 Oracle 实用技术

(美) Kevin Owens 著
欧阳宇 译

清华大学出版社

Oracle 触发器与存储过程高级编程

(第 3 版)

(美) Kevin Owens 著
欧阳宇 译

清华大学出版社
北 京

图书在版编目(CIP)数据

Oracle 触发器与存储过程高级编程(第 3 版)/(美)欧文斯(Owens,K.)著；欧阳宇译.—北京：清华大学出版社，2004.10

书名原文：Programming Oracle Triggers and Stored Procedures

ISBN 7-302-09382-2

Ⅰ.O… Ⅱ.①欧…②欧… Ⅲ. 关系数据库－数据库管理系统，Oracle－程序设计 Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字(2004)第 089353 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **客户服务**：010-62776969
组稿编辑：曹 康
文稿编辑：崔 伟
封面设计：康 博
版式设计：康 博
印 刷 者：北京市通州大中印刷厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 **印张**：20.5 **字数**：525 千字
版 次：2004 年 10 月第 1 版 2004 年 10 月第1 次印刷
书 号：ISBN 7-302-09382-2/TP・6553
印 数：1～4000
定 价：39.80 元

前　言

本书重点阐述了 Oracle 数据库服务器的核心概念：数据库表、索引、表空间、约束、触发器和 PL/SQL。掌握这些核心组件对应用程序开发至关重要。无论您是 Web 开发人员、客户机/服务器(C/S)程序员或者后台程序员，本书都将对您大有裨益。通过本书，您将学会如何用声明性约束和过程性约束在数据库服务器中实现逻辑，以及如何使用 PL/SQL 触发器和存储过程。

本书读者对象

本书针对程序员而编著。掌握一些编程知识将会对学习本书有所帮助，但不是必需的。学习本书之前不需要对 Oracle 有所了解。书中详尽的阐述、众多的图表和示例代码，会使那些从其他数据库转向 Oracle 的开发人员受益匪浅。书中有关触发器和 PL/SQL 的讨论是针对那些想快速地从其他编程范例模型转换到 PL/SQL 模型的程序员而精心设计的。本人曾经参与过和 Oracle 相关的项目，项目中往往有许多非 Oracle 数据库开发人员，因此经常会混淆部分术语。本书的一个重点就是解释基本范例，这对于初学者和有一定经验但需要交叉培训的开发人员来说都会有所帮助。

如果您原来使用非关系数据库，而现在转向 Oracle，那么本书将会助您一臂之力。第 1 章阐述了 SQL 的起源。从事层次数据库和网络数据库的开发人员会发现他们可以用历史的眼光来全方位了解 SQL。在我早年的职业生涯中，曾经有 5 年的时间从事层次数据库，3 年的时间从事网络数据库。我理解为什么需要转向 Oracle，本书将满足那些需要交叉培训的从事非关系数据库的开发人员的需求。

本书的第 5 章深入讨论了 Oracle 数据字典，喜欢钻研的开发人员将从中受益。该章描述了如何从数据字典中抽取约束定义。有兴趣的开发人员可以扩展这一简单技术以便从数据字典中抽取尽可能多的信息。这对那些如饥似渴的初学者大有好处。本书上一版的读者认为这是本书最有用的地方之一，因为它教会他们如何进行独立研究和学习。

开发人员往往需要在开发服务器上进行数据库管理。本书可作为数据库管理的初级教程。开发人员很自然需要和表、表空间等逻辑概念打交道。本书也用了较大的篇幅讨论数据库的物理概念，如表空间数据文件，这对那些需要了解 Oracle 构成的人来说也是一个很好的入门教材。

书中首先强调的是在 Oracle 数据库中建立业务逻辑。本书始终坚持的理念是让数据库为您工作和服务。一个能够很好利用声明性约束的团队能够创建出高度“智能化”的数据库，其中所有的数据都将符合严格的业务规则逻辑。而且，如果该团队将触发器和 PL/SQL 结合使用，则数据库完整性会进一步得到扩展，它将可以代替终端用户执行逻辑操作。包括复杂业务规则的实施和其他操作，如自动 email 通知，这将在第 11 章中讨论。

本书组织结构

第 1、2 章

第 1、2 章讨论 Oracle 表、列类型、如何使用 SQL*Plus 与数据库进行交互。描述了 SQL*Plus 十分有用的功能，包括用于显示 SQL 执行计划及其统计信息的 SQL*Plus 命令。这两章的重点是术语，解释了 SQL、SQL*Plus 和 PL/SQL 等术语的直观含义。

第 3、4、5 章

第 3、4、5 章全面探讨了声明性约束，包括从 Oracle 数据字典中抽取约束定义。这些章节包括：(a)将会告诉您如何创建具有高度完整性的数据库——这也是每个客户所期望的；(b)让您能够从数据字典中抽取众多信息——这是每个人都希望的本领；(c)让您能够深入洞悉声明性约束，并帮助您了解其工作机制。

第 6、7、8 章

第 6、7、8 章讨论了数据库触发器，也称为过程性约束。第 6、7 两章通过大量的图表来描述表行和语句级触发器，让您有一个直观的了解。这个直观了解对您选择触发器类型很关键。第 8 章描述了如何使用数据库触发器和 PL/SQL 来进行复杂规则的实施。

第 9 章

第 9 章有助于您迅速掌握 PL/SQL 程序的编译。建议您最好是边学习边编码。本章将帮助您了解 PL/SQL 编程环境。学完本章的内容之后，您将会基本了解如何进行编码、编译和执行 PL/SQL 过程。有关 PL/SQL 的技术性内容将在下面两章介绍，本章只介绍编码和测试的方法。

第 10 章

第 10 章将从软件工程的角度讨论 PL/SQL。仅了解 PL/SQL 的语法并不够。在实际应用之前，我们需要了解 PL/SQL 范例和 PL/SQL 程序单元(过程、函数和程序包)模型。本章将从概念上向您介绍 PL/SQL 程序单元，并鼓励您在设计阶段思考在何时、如何使用这些程序单元。

第 11 章

第 11 章讨论了 PL/SQL 语言的功能。重点放在了 PL/SQL 结构和内置 SQL 函数上，并强调使用简单易懂的算法。本章最后一节描述了一个应用程序，该应用程序通过 Oracle 内置的 Alerts 程序包用一个数据库触发器发送 email 通知。

本书的所有程序脚本都在 Oracle 9i Release 2 和 10g Version 10.1.0 上运行通过。

本书约定

本书内容严格区分大小写形式，旨在增强其可读性和条理性。书中的 SQL 和 PL/SQL 代码除了关键字用大写外，其余均为小写。下面这行代码是从第 3 章选出来的，说明了这种行文

风格。

```
CREATE SEQUENCE student_vehicles_pk_seq;
```

表达式和 Oracle 内置函数通常都用大写。下面这行代码选自第 11 章，它用到的两个 Oracle 内置函数(NEWTIME 和 SYSDATE)都用大写。

```
pst_date := NEW_TIME(SYSDATE, 'EST','PST');
```

Oracle 有许多内置程序包。书中示例代码所引用的内置程序包都用小写。这样做一方面可以进一步增强代码的可读性，另一方面可愉悦读者的双眼。我们知道过度使用大写对视觉没有好处。下面一行代码选自第 11 章，说明当引用 Oracle 的内置程序包 DBMS_ALERT 时用了小写。

dbms_alert.register('OUT_OF_STOCK_ITEM');

PL/SQL 示例代码中经常会含有计数器变量的声明；这些通常用大写形式。下面一行代码选自第 11 章，它声明了一个变量 ID。虽然它不是关键字，但是用大写更便于阅读。

ID VARCHAR2(10),

最后，对于非代码文本，也就是正文中的段落文本，在引用代码项时一律用大写。我们在段落文本中可能会引用一个约束名称或者一个 PL/SQL 变量名称。总之，段落中出现的代码标识符总是用大写。

致谢

感谢技术审阅和编辑小组成员：Pat Starr、Kathryn Tyser 和 Linda Owens。感谢 Prentice Hall 公司的 Jeffrey Pepper 所付出的辛勤工作，感谢 Pine Tree Composition 的文稿编辑 Laura Burgess 和 Jessica Balch。

Kevin Owens

目　录

第 1 章　关系数据库表

1.1　关系表

1969 年，阿波罗号宇航员 Neil Armstrong 和 Buzz Aldrin 在月球上留下了人类的第一串足迹。而就在这一年，哲学博士 Edgar Frank Codd 提出了关系数据库系统的理论。之后不久，Codd

在美国计算机学会(the Association of Computing Machinery，ACM)的刊物 *Communications of the ACM* 上发表了“A Relational Model of Data for Large Shared Data Banks”这样一篇论文。Codd 提出的模型后来成为一种原型的基础，该原型就是 IBM 内部开发的一个名为 System/R 的关系数据库项目。IBM San Jose 研究中心(现在是 Almaden 研究中心)对此作了一些研究，到了 1974 年，IBM 已经有了一个可以运行的原型关系数据库系统。该系统建立在多表查询和多用户访问的基础上。在 System/R 中访问数据的方法叫做 Structured English Query Language(SEQUEL)。

继 System/R 的研究和开发之后，IBM 就把 System/R 作为原型发布了。麻省理工学院斯隆管理学院(MIT Sloan School of Management)和制造行业、库存部门的商业组织使用了 System/R。该项目表明 Codd 的理论可以在真实世界里应用和实现，但在 1979 年 Codd 的理论走到了尽头。

System/R 的成功不像阿波罗号宇航员在 1969 年 7 月 20 日那天那样引人注目。但是，System/R 的成功却在信息技术领域开创了一种新的范例——一种直至今天仍然统治信息时代的范例。

Codd 相信程序员应该能够控制所使用的数据来联结和构造对多个独立表的查询。而在现有的层次和网络数据库中，这是不成立的。对于这些其他的技术，程序员要编写一个数据库导航算法，根据预先定义的导航路径访问数据。当软件需求出现变化时，程序员常常需要为这些路径而修改数据库，并且需要定义新的导航路径。这样一来，软件开发过程变得非常缓慢，并且因为要频繁地满足数据访问需求而受到影响。Codd 的模型对软件开发过程有着深远的影响。

System/R 是由一群极具才华并受过良好教育的软件工程师实现的。他们证明了大型信息系统可以建立在 Codd 的多个独立表以及结构化查询语言的基础之上。就这样，SEQUEL 诞生了。在整个 System/R 项目中，工程师们坚持将他们的工作公布在技术刊物上以及国际论坛上。人们认为这种开放发布的策略是 System/R 获得成功的一个因素，它导致了很多关系数据库(例如 DB2 和 Oracle)的创立。

IBM 将对 System/R 结构化查询语言的控制权转交给了美国国家标准协会(American National Standards Institute，ANSI)，这是一家非盈利性的私有机构，掌管着美国的自愿技术。由于商标法的缘故，将该语言由 SEQUER 重命名为 SQL，即“Structured Query Language(结构化查询语言)”。如今，SQL 已经是一项 ANSI 标准，并且也是一项国际标准化组织(International Standards Organization，ISO)标准。

1997 年，一群一直关注 System/R 项目(主要是通过出版物)的工程师认识到它的潜力，并成立了一家公司。这家公司就是 Software Development Laboratories，后来更名为 Relational Software。这家新公司开发了第一套商业版的关系数据库管理系统，并将之推向了市场，该系统建立在 ANSI SQL 标准的基础之上。他们将这套产品命名为 Oracle。

当 System/R 项目在 1979 年结束时，许多信息系统还是依赖于层次或者网络数据库的体系结构。到了 20 世纪 80 年代，很多开发项目便转移到了关系技术上来。如今，大部分新型信息系统都是关系的。不过，层次和网络数据库产品在市场上仍然占有一席之地。

IMS/DB 是 IBM 的旗舰层次数据库。Rockwell International 和 IBM 在 1969 年开发了 IMS，美国航空航天局使用它来管理“阿波罗”计划的数据。就是在今天，IMS/DB 仍在国际上稳占一席之地。IMS 是面向全行业的，目前大约有 2 亿终端用户。IMS/DB 数据库曾为包括“阿波罗 II”在内的“阿波罗”计划服务，“阿波罗 II”曾将 Armstrong、Aldrin 和飞行员 Mike Collins 送上了月球。

Computer Associate 公司的网络数据库产品是 CA-IDMS，这是一种被广泛使用的数据库，

它为全球 2,000 个站点提供服务。虽然该产品已更名和易主，但是它仍然支持运行在 IBM OS/390 平台上的大型公司数据系统。

IDMS 有一段有趣的历史。John J. Cullinane 在 1968 年成立了 Cullinet Software 公司。他从 B.F. Goodrich 那里买下了 IDMS 的所有权，IDMS 是一种 CODASYL 数据库。CODASYL 是 Conference on Data System Languages(数据库系统语言会议)的首字母缩写形式，该组织是 1960 年美国国防部为标准化语言和软件应用程序而建立的。Cullinet 是第一家在纽约证券交易所榜上有名的软件公司。1989 年 9 月，Computer Associate 公司收购了 Cullinet，并将产品更名为 CA-IDMS。

虽然 CA-IDMS 建立在数据库网络体系结构的基础之上，但是该产品还包括一个遵从 ANSI 的 SQL Option。这个选项允许应用程序使用 SQL 访问 IDMS 数据。该产品提供了一套丰富的工具，包括企业监控代理(agents for enterprise monitoring)、使用 Parallel Sysplex Cluster 技术的并行事务处理、对 Java 的 JDBC 支持，以及很多其他的电子商务技术。IDMS 应用程序的大部分都是用 COBOL 编写的。Oracle 在关系企业服务器(Relational Enterprise Server)方面有着绝对的知名度，它还有其他两个数据库产品：(a) 一个名为 DBMS 的网络数据库产品，这是一个 CODASYL 数据库；(b) 一个名为 Rdb 的关系数据库产品。Rdb 和 DBMS 这两个产品都是在 DEC(Digital Equipment Corporation，数字设备公司)被 Compaq 收购之前从 DEC 买进的。

Oracle 是在从 DEC 购买 Rdb 产品家族时得到 DBMS 的。Ken Olsen，毕业于麻省理工学院(MIT)，他的兄弟 Stan，以及 Harlan Anderson 共同创办了 DEC。除了诸如 PDP-11、VAX 和 Alpha 之类的硬件体系结构之外，DEC 还开发了 Rdb(一个关系数据库)，以及 DBMS(一个网络数据库)。这个网络数据库在 Alpha AXP 平台上发布 V5.0 版本时从 VAX/DBMS 更名为 DEC DBMS。

Oracle 的 CODASYL DBMS 是一种多用户网络数据库，像 Rdb 一样，它经过优化，可用于 Alpha 或 VAX 体系结构平台上的 Compaq OpenVMS 操作系统。DBMS 非常适合于制造业系统和车间系统，这些系统都要求有一个稳定的环境，该环境中数据库信息都是相当静态的。Consillium 公司的 WorkStream 是半导体和电子行业最广泛安装的制造执行系统(Manufacturing Execution System，MES)。WorkStream 在 VAX/DBMS 上运行过很长一段时期，如今它主要运行在使用 Oracle CODASYL DBMS 的 VAX 和 Alpha 服务器上。

Rdb7 是 Oracle 首次发布的 Rdb 产品，从构思和设计来看它都是一个 Oracle 产品。Oracle 的 Rdb7 是一种企业关系数据库，它经过了优化，可用于数字平台。也就是说，可用于 Hewlett-Packard 公司的 OpenVMS 操作系统和 Compaq Digital UNIX。Compaq 收购 DEC 的时候，废除了 DEC 的 UNIX 版本(Ultrix)，而采用名为 DIGITAL UNIX 的产品——一个基于 64 位体系结构、运行在 Alpha AXP 平台上的操作系统。

总而言之，数据库市场已经明显地被关系技术所统治。但是，层次和网络数据库依然可以满足企业网络计算环境的需要。IBM IMS/DB 是目前主流的层次数据库，可服务于高事务率的系统，并且有很大的用户基础。两种网络数据库—— CA-IDMS(Computer Associate 许可)和 DBMS(Oracle 许可)，在对高事务企业级系统的支持中扮演着关键角色。除了 Oracle Enterprise Server 以外，Oracle 公司还拥有 Rdb7(一种关系数据库)以及 DBMS(一种网络数据库)，这两种产品都经过了优化，可用于 Compaq 平台，包括 Alpha AXP。如今，Oracle 数据库服务器解决了小型和大型企业在线事务处理(OLTP)系统、决策支持系统(DSS)以及电子商务解决方案在信息处理方面的需要。这些类型的应用程序可以在小型、独立的服务器上运行，也可以在分布式、高可用性的体系结构中运行。

1.2 SQL

SQL 的含义非常专一，不像“数据库”那样在日常使用中有多种意义和解释。本节将 SQL 语言作为一个标准进行详细的阐述。这一节的主题包括 SQL 实现、嵌入式 SQL、Direct SQL 以及作为 Direct SQL 的实现的 SQL*PLUS。

1.2.1 ANSI 标准

SQL 是目前 ANSI 标准中定义的一种语言(上一节讨论了 SQL 的历史，它是由 IBM 转交给 ANSI 的)。如果您正在编写使用 SQL 的 Oracle 应用程序，那么就要编写 SQL 语句，这种语句的语法遵从 ANSI-X3.135-1999，“Database Language SQL”。

用户可以通过 ANSI 网上商店得到这种 ANSI 标准。可以访问站点 http://webstore.ansi.org/ansidocstore/，该站点有一个搜索表单域。通过使用这个搜索域，输入“ANSI X3.135”，就可以找到 SQL 标准。可以得到该文档 PDF 格式的副本。在我编写本书时，下载这种副本的费用是 18 美元，该文档包含 600 多页。

SQL 标准与大部分标准一样，都是用 Backus Naur Form 语法(稍后会说明)定义的。以下是从 ANSI-X3.135-1999，“Database Language SQL”摘录出来的内容，用于说明该标准如何精确地定义一条 CREATE TABLE 语句的语法。

```
<table definition> ::=
CREATE [ { GLOBAL ) LOCAL } TEMPORARY ] TABLE <table name>
<table element list>
[ ON COMMIT { DELETE I PRESERVE } ROWS ]
<table element list> ::=
<left paren> <table element>
[ { <comma> <table element> }... ] <right paren>
<table element> ::=
<column definition>
| <table constraint definition>
```

因此，如果您是一位数据库供应商，要销售一种 SQL 数据库产品，那么您就会告诉客户，您的产品能够解析和执行 ANSI SQL 语句，例如前面提到的 CREATE TABLE 语句。

使用标准总是有好处的。也就是说，知识是可以转化的。如果理解了 SQL，那么在关系数据库技术方面的知识就可以在 Oracle 和 SQL Server 之类的产品之间转化。即使是在一个标准内也会有不同之处。SQL 也不例外。

SQL 随供应商的不同而不同，这有两点原因。首先，在标准中存在着“实现定义的(implementation-defined)”功能。也就是说，对于某些 SQL 功能，关于一种功能是如何实现的这样的特定性质是“实现定义的”——该标准通过语言“实现定义的”显式声明这一点。这意味着供应商可以随意选择该功能的实现方式。一个例子就是，供应商如何实现编目或数据字典。该标准不能强制供应商以某种方式存储元数据。在本章的后面我们将看到，Oracle 的数据字典是一组关系表，从这些表中我们可以选择大量的信息，例如描述我们所创建的表的信息。

不同数据库之间存在不同点的第二个原因是，供应商会用附加的功能去增强各自的特定 SQL 引擎，他们认为这些附加功能将有利于应用程序开发人员。在 Oracle 中可以找到的