Real-World SQL and PL/SQL: Advice from the Experts

SQL和PL/SQL深度编程

数据建模 高级编程 高级分析 安全与管理

[美] 阿勒普·纳达(Arup Nanda)
[爱] 布伦丹·蒂尔尼(Brendan Tierney) 等著
[芬] 海利·希尔塔赫(Heli Helskyaho)
唐波 侯圣文 译

Oracle Press

清华大学出版社

SQL和PL/SQL深度编程

数据建模 高级编程 高级分析 安全与管理

[美] 阿勒普·纳达(Arup Nanda)
[爱] 布伦丹·蒂尔尼(Brendan Tierney) 等著
[芬] 海利·希尔塔赫(Heli Helskyaho)

唐 波 侯圣文 译

清华大学出版社

北 京

Arup Nanda, Brendan Tierney, Heli Helskyaho et al
Real-World SQL and PL/SQL: Advice from the Experts
EISBN：978-1-25-964097-1

图书在版编目(CIP)数据

SQL 和 PL/SQL 深度编程：数据建模　高级编程　高级分析　安全与管理 / (美)阿勒普·纳达(Arup Nanda) 等著；唐波等 译. —北京：清华大学出版社，2019

书名原文：Real-World SQL and PL/SQL: Advice from the Experts

ISBN 978-7-302-51926-3

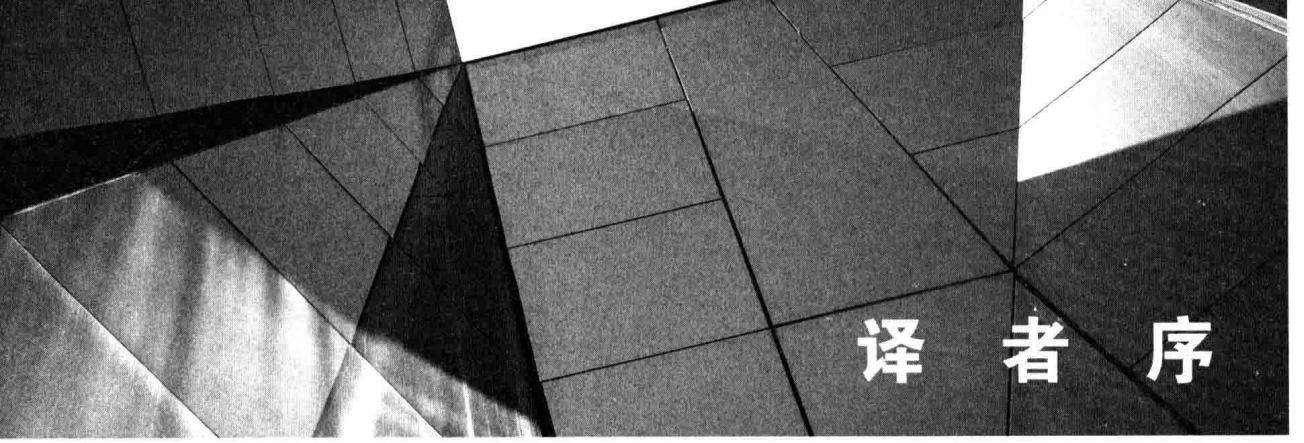
Ⅰ. ①S…　Ⅱ. ①阿…　②唐…　Ⅲ. ①关系数据库系统　Ⅳ. ①TP311

中国版本图书馆 CIP 数据核字(2018)第 288799 号

责任编辑：王　军
装帧设计：孔祥峰
责任校对：牛艳敏
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　印　　张：35　　字　　数：1152 千字
版　　次：2019 年 3 月第 1 版　　印　　次：2019 年 3 月第 1 次印刷
定　　价：128.00 元

产品编号：074924-01



译者序

刚刚拿到本书的英文原版时，我就被内容吸引。我也经常在Oracle公司定期发给我的ACE Newsletter邮件中看到本书这五位作者的事迹介绍。根据多年从事Oracle开发/管理/培训工作的经验，我认定本书是最适合我国一线Oracle技术人员，特别是一线Oracle开发者阅读的技术书籍之一。因其举例丰富、循序渐进，所以即使是初学者，阅读本书也不困难。于是邀约侯圣文老师共同翻译。本书第1、6、13、14、15、16章由我本人翻译，第2章由我和侯老师共同翻译，第3、4、5、7、8、9、10、11、12章由侯老师翻译，其他内容由我翻译。全书由我整理统稿。感谢侯老师所做的大量工作，没有他的分担，我想本书是不可能翻译完成的。

当我查看我所翻译的382页译稿时，翻译的酸甜苦辣历历在目。面对原著中复杂的技术表述、技术智慧、技术幽默以及母语人士的英文使用技巧，只有不断地努力琢磨，才能尽量把这几位ACE-D的技术思想完整展现给读者。

感谢中国Oracle用户组大家庭在我和出版社之间牵线搭桥，感谢本书的所有编辑，感谢在翻译过程中我妻子给我的帮助和儿子对我没有时间辅导他的理解。最后，我特别想把这本书献给我的母亲。

——唐波

序　言

或许你使用 Oracle 数据库，要决定选择哪种编程语言来开发应用程序。现在有以下 3 种选项可供选择：

- SQL
- PL/SQL
- 一些库外编程语言(Java、C#、JavaScript 等)

你要选择哪一个？让我们探讨一下。

使用 SQL 的情景

SQL 是关系型数据库的核心。数据库依靠它进行内部自我管理。向数据库服务器发送一条 SQL 语句时，在整个处理过程的每一步都会使用递归 SQL，以确保语句有效，检查是否有权限访问那些对象，并确定执行计划。执行计划决定如何检索数据。在关系型数据库内部很少有地方不涉及 SQL。

想象一下：如果你是关系型数据库(Oracle、MySQL 或 SQL Server)引擎的开发人员并致力于提高 SQL 的性能，那么你不仅提高了他人的应用的性能，实际也提高了数据库引擎自身的内部性能。很容易理解关系型数据库引擎的开发人员为什么会着迷于 SQL 的性能(SQL 都做了什么)。

SQL 不仅仅是基本的插入、更新、删除和多表连接查询。现代关系型数据库引擎包含海量的 SQL 功能，从而可以极大减少应用中的代码行数。分析函数、模式匹配和 model 子句等功能使我们能够在不离开 SQL 环境的情况下执行数据分析。在 SQL 中生成和处理 XML 和 JSON 数据是家常便饭，这样就可以在没有任何额外编码的情况下提供网页服务。

从性能角度来看，SQL 是王者。只在极少数的情况下，非 SQL 解决方案比 SQL 解决方案处理关系型数据库数据性能会更好。大多数其他语言都鼓励一行一行地逐行处理数据，而这完全不是关系型数据库希望你去做的。通过使用 SQL 并聚焦于数据集的处理，通常会看见批处理和分析操作的性能呈数量级提升。

考虑到这一点，处理关系型数据库中数据的最好语言显然是 SQL。任何时候使用非 SQL 语言时，都是把数据库拖离最佳状态。最终应用的性能和可扩展性都会显著下降。

使用 PL/SQL 的情景

PL/SQL 是 SQL 的过程化扩展，允许在不离开数据库引擎的前提下向数据处理过程添加过程逻辑。PL/SQL 引擎所在的位置就是数据库引擎的位置。这一点便是使 PL/SQL 如此强大的重要原因。

设想一个场景：有两种势均力敌的编程语言。其中一种在数据库内部运行，而另一种在位于独立计算机上的应用服务器中运行。当运行于数据库内部的语言需要数据时，立即就能拿到。而运行于应用服务器上的语言需要数据时，必须通过网络向数据库服务器提出请求，所有要返回的数据都必须通过网络传回。现在设想一个“聊兴

盎然”的应用，它重复呼叫数据库来请求数据完成任务。这就可以看出为何应用服务器上的代码无法扩展了。

可扩展性问题的一种解决方案是在数据库外添加硬件和缓存层。另一种解决方案就是直接将应用以 PL/SQL 代码重写，就把要实现的功能移到数据了。

历年来，我做过许多项目。这些项目常常只关注层，而把数据库仅仅作为数据的基本容器。这些项目最终都会遭遇性能问题，尤其是遇到批处理时。解决方案几乎都是用 PL/SQL 重写数据密集型的过程，以靠近数据。如果把以数据为中心的过程放在数据库内部来处理，会得到更好的可扩展性和性能，也更容易在不同客户端、表示层和界面之间共享这些功能。

PL/SQL 语言与 SQL 紧密集成在一起，使 SQL 可以和过程化逻辑之间无缝地进行数据传递。与 PL/SQL 比起来，我过去 20 多年用过的其他语言在处理数据库时总是显得笨拙得不可思议。

那么就是说 PL/SQL 仅仅适用于批处理作业了？根本不是这么一回事！Pl/SQL 不仅在考虑以数据为中心的代码时大放异彩，而且近些年已经发展成为一个功能丰富的开发平台。下面来看看 Oracle Application Express (APEX) 等开发工具提供的许多功能吧：它们都是用 PL/SQL 写的。你会意识到 PL/SQL 有多么能干。

使用其他语言的情景

总会有需要采用特殊语言应对特殊需求的情况，Oracle R Enterprise 允许在 Oracle 数据库内部运行 R 程序并与 SQL 和 PL/SQL 集成在一起。那么其他库外语言呢？

我在工作中曾使用过许多编程语言，包括 Perl、Python、PHP、JavaScript 和多种操作系统的脚本语言。我还曾用 Java、C#和 C 等语言编写过程序。虽然每一种语言都有各自的优点，但是一旦涉及与数据库交互，都不是 PL/SQL 的对手。

在开发表示层甚至开发一些业务逻辑时，会不可避免地用到其他编程语言和框架。但是编程方式离数据库越远，工作就会越艰难。

结论

大多数开发人员和 DBA 都得出一个相似的结论：如果可以，就用 SQL 编程。不能用 SQL 的场景就用 PL/SQL。实在连 PL/SQL 都不能用时，才用其他编程语言。

Tim Hall
DBA/数据库开发人员
oracle-base.com
Oracle ACE Director

前　言

Brendan Tierney 和 Heli Helskyaho 在 2015 年 3 月找到我，让我和 Arup Nanda 和 Alex Nuijten 编写本书。不久之后，我们也把 Martin Widlake 拉入伙。如果说被他们邀请对我来说是极大的荣耀，那仍是过于轻描淡写。很快，我意识到我没有足够的精力投入此项目，但我也不想让其他作者因为我陷入出版风险之中。但我仍然非常愿意为此书出一分力，遂自荐要成为此书的技术编辑。他们欣然接受了我的这个新角色。

这是我第一次正式成为技术编辑。但是我曾经做过多年类似的工作：检查自己的工作，检查他人的工作。我体会到追求完美去做这项工作会帮助良多。

本书所有的测试工作都是在 OTN/Oracle 提供的预建 Database App Development VM 上进行的。这使测试变得非常容易。根据相应文档配置测试环境也很容易。

在审核此书的过程中，我本人面临的最大挑战之一是 Oracle 12*c* 的多租户体系结构。我已经多年没有从事 DBA 类型的工作了，因此要弄清楚哪些事应该在根容器(CDB)上做、哪些应该在插件容器(PDB)上做真是新鲜有趣。此外，作者们给出的其他提示都很容易搞懂。

设计(数据建模、基于版本的重定义、VPD)、安全性(数据编写/掩蔽、加密/哈希)、编码(正则表达式，PL/SQL、SQL)、性能测量和剖析或把原始数据转换成可执行信息(数据挖掘，Oracle R，预测查询)——以上所有内容都会在本书中详细介绍。这些就是一名开发人员从头设计一个完整应用所需要做的全部。

在此次技术编辑工作中我最喜欢的部分可能是：不仅要检查书中的内容是否可运行，而且必须做更多的工作。通常，当我阅读一本书或博文时，会快速抓住技术要点，而略过 why、when 和 where 的部分。我很容易会这样做。多年来，我每天都会读 AskTom 上的文章，这是使我避免陷入困局而采取的短时充电办法。首先在这些文章中能看到特定问题是如何解决的，这些解决方案偶尔也可以应用到我自己的问题中。这样做了一两年后，我便会进一步想到应该去搞清楚为什么要按照某种方式去处理特定问题，并且想要搞清楚那些 Tom 详尽解释的处理方式的后果。

审阅此书我受益颇丰。在这个过程中，我走进作者的思维世界：不仅能看到他们是如何解决技术难题的，而且能看到他们为什么以这种方式去处理。对开发人员和 DBA 来说，能看到这些都是极其有价值的。我们中的大部分人都能找到解决特定问题的方法，但是需要更上一层楼，我们需要理解 why、when 和 where。这本书提供的正是这些。

Chet Justice
Oracle ACE Director
技术编辑

引　言

要说明编写本书的初衷时，Rod Stewart 唱的《航行》歌词跳进我的脑海："我们在航行，我们在航行，穿越海洋，游子回乡"。这是因为写此书的想法诞生于一艘船上。有些人叫它轮船，有些人叫它邮轮。不管叫它什么，这本书诞生于 2015 年 3 月举行的 OUG 挪威大会期间。这一届 OUG 挪威大会十分特别是因为它是在一艘往返于挪威奥斯陆和德国基尔之间的邮轮上举行的。这意味着会议的演讲嘉宾和参会人员都会被"困"于这艘船上长达两天。这期间每个人都心无旁骛地沉浸在 Oracle 社区的技术演讲、研讨会、技术探讨和思想分享之中。

在这次会议期间，Heli 和 Brendan 开始讨论这本书。Heli 刚刚出版了 *Oracle SQL Developer Data Modeler* 一书。Brendan 前一年出版了 Oracle 数据挖掘方面的书。探讨写作经验、分享技术知识及写作乐趣时，他们都意识到有很多书可供人们在开始 Oracle 职业生涯时阅读，也有许多书籍针对专门的高深主题。但位于这两者中间的书却严重缺乏。一个始终需要解答的问题是：读完入门书籍后，有什么书可读，进而让人们读懂那些高深的书籍呢？对，这就是 SQL 和 PL/SQL 方面的书籍！

他们感觉很多书，特别是技术介绍类的书缺少基于经验处理问题的"why"和"how" 的总结部分。熟记命令的语法和选项固然很好，但是只有从运用该语言完成真实任务过程中获得经验才能进阶：从理解一门语言到能灵活自如地运用它。分享这方面的一些经验真是太好了。

然后，在 OUG 挪威会议最后一天的早餐过后，当邮轮穿行过哈当厄尔峡湾，通向环绕奥斯陆的小岛时，Heli 和 Brendan 最终决定本书应该面世。他们之后详细列出本书的内容类型，以及要由哪些知名专家(或技术明星)来阐述相关专题。专家列表很快就出来了：Oracle ACE 总监的作者团队形成了，包括 Arup Nanda、Martin Widlake、Alex Nuijten、Heli Helskyaho 和 Brendan Tierney。然后作者团队着手确定章节和内容。利用他们合在一起超过 120 年的 SQL 和 PL/SQL 经验，最终在 Oracle Open World 上定下了本书的范围和内容。

本书分为 4 个部分，每个部分都致力于帮助 Oracle 开发人员更好地理解他们每天都会用到的 SQL 和 PL/SQL 的技术核心。

第 I 部分关注所有数据库项目的绝对核心之一——数据模型的设计，并阐明什么时候该用 SQL，什么时候该用 PL/SQL 这一基本问题。第 I 部分介绍了 Oracle SQL Developer Data Modeler 这一数据库设计工具。

第 II 部分介绍在结构改写过程中能有效提升应用程序可靠性但未被充分使用却在 SQL 和 PL/SQL 中极为有用的工具，比如处理复杂数据集、正则表达式和基于版本的重定义特性。还介绍了使用 PL/SQL 的领域，虽然并不常用，但是其威力和灵活性需要进一步开发到极限。你会看到如何利用 PL/SQL 扩展 SQL，以避免一些 SQL 带来的常见却很少正确处理的问题；为了剖析和诊断性能问题，对代码如何进行性能测量，如何有效地使用动态 SQL(和它的扩展 PL/SQL)和 PL/SQL 的强大功能辅助数据库管理和执行自动化任务。

第III部分介绍如何利用 Oracle 数据库的特性来支持数据科学的方方面面。这包括如何运用 Oracle 数据挖掘算法里的 PL/SQL 和 SQL 函数，也包括如何使用 Predictive Queries(预测查询)，这是 Oracle 12*c* 中的新特性，允许在对算法一无所知的情况下进行自动数据挖掘。这部分还介绍了如何使用作为 Oracle 数据库引擎一部分的 R 语言。

到此为止，不仅能分析数据而且能合并使用 SQL、PL/SQL 和 Oracle 企业级 R 统计编程语言。Oracle 数据库的这些高级分析功能使你能够在不需要做任何数据移动的情况下快速有效地使用各种数据挖掘和机器学习技术来分析数据。

第Ⅳ部分介绍如何在 PL/SQL 编程中考虑安全性以保证包括数据库和应用程序在内的整个系统的安全。这部分详细讨论如何加固代码以抵御 SQL 注入攻击和内部劫持攻击，也讨论如何进行数据编写和加密敏感生产环境数据以免受到非法入侵。这部分不涉及数据库架构安全性，但涉及如何在编程时脑海中时刻考虑安全性，以及使用 Oracle 提供的工具创建安全的应用程序。

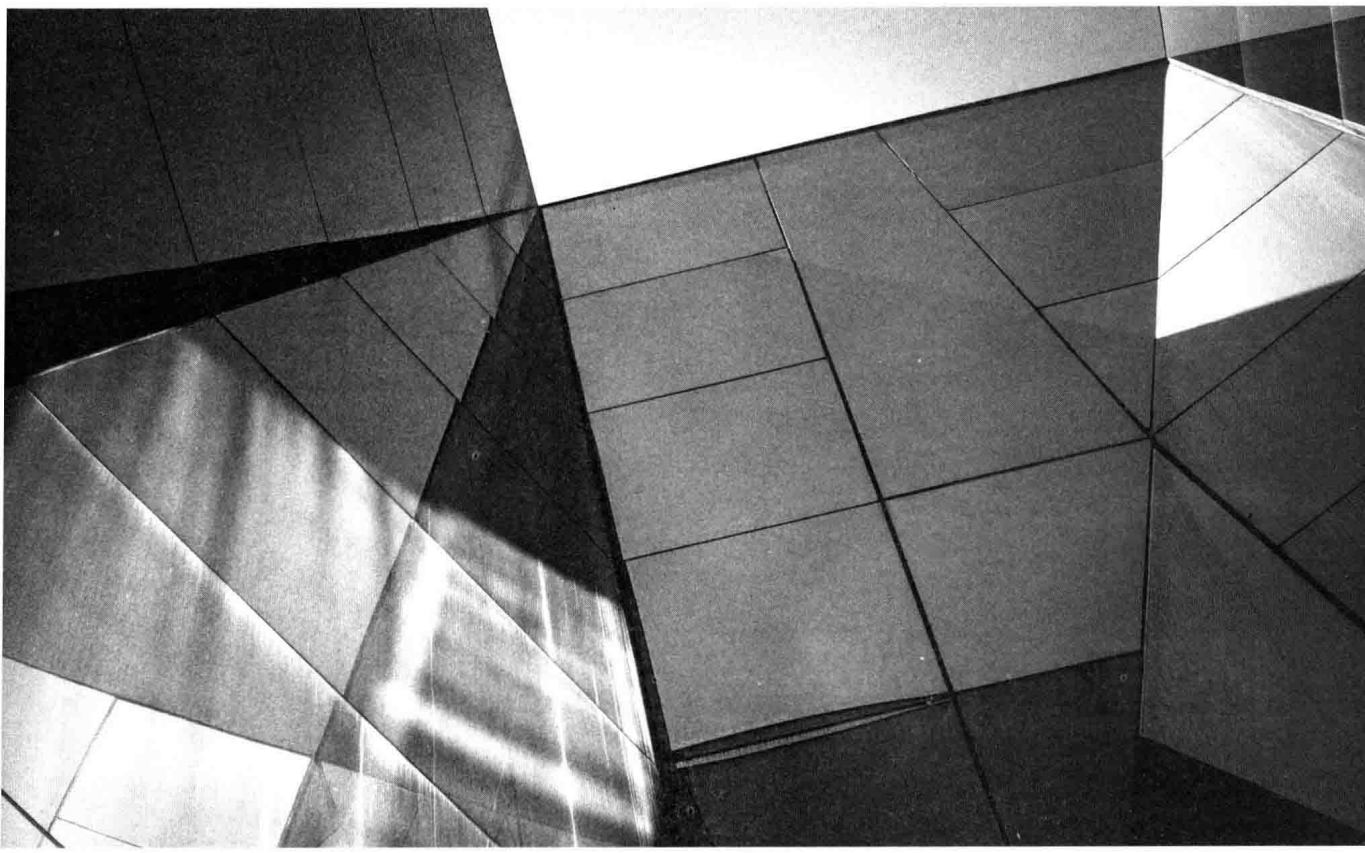
为了完成使命，本书致力于在人们从初学者转变成高级 SQL 和 PL/SQL 用户过程中架起一座桥梁。我们的目标是让内容覆盖 PL/SQL 和 SQL 架构所提供的强大而又平时并不常用的功能。无论是开发人员还是架构师甚至是 DBA，本书都能使你变成更有学识、更有效率并且更有安全意识的技术专业人士。每个和 Oracle 打交道的朋友都能从本书获益。

书中代码的下载地址

本书的大部分章节都配有示例。为了节约重新录入这些代码的时间和键盘错误，作者提供了一组代码文件供下载。有代码的章节，它的代码独立放在单个文件里，这样你就可以快速找到每个章节中的示例。

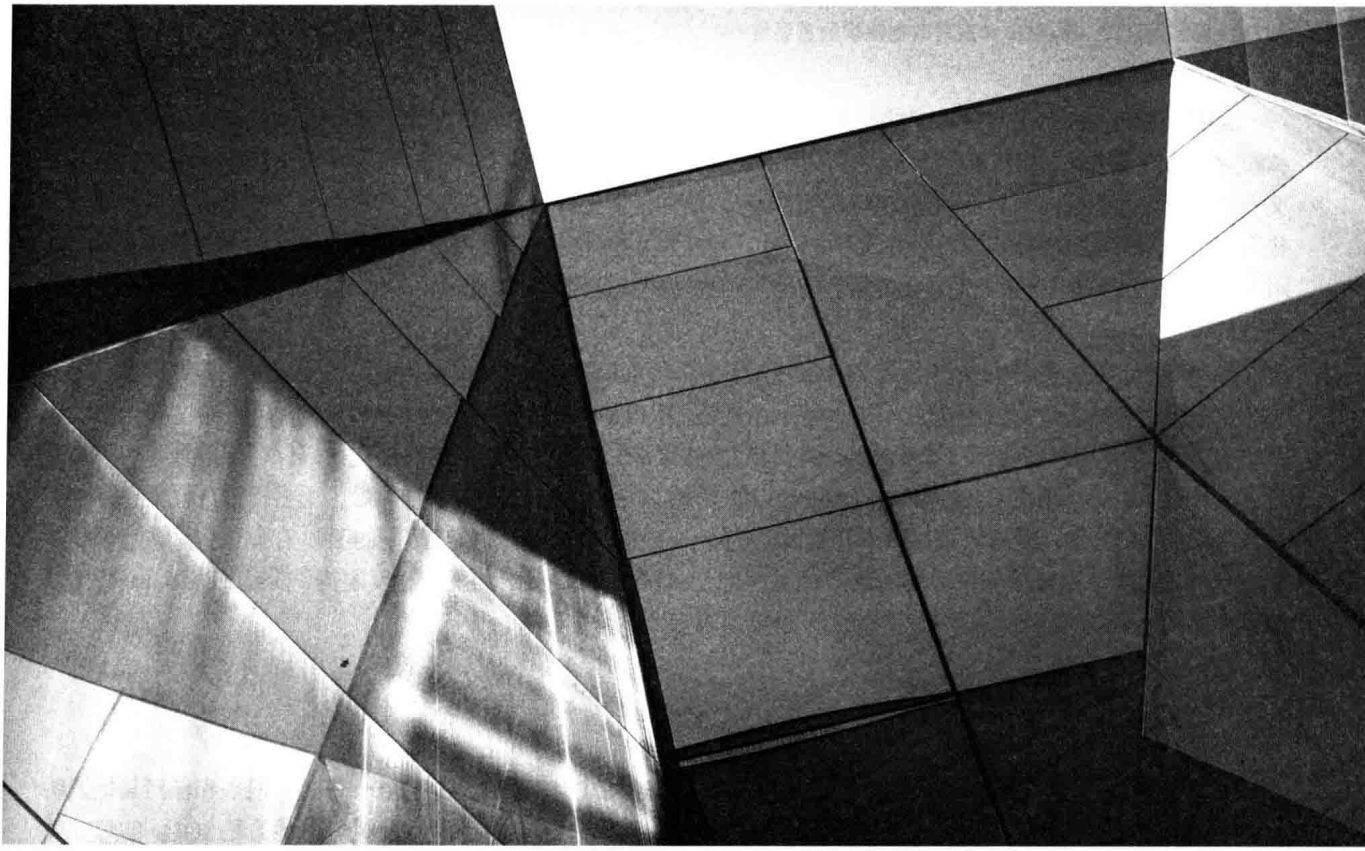
可以从 McGraw-Hill Professional 的网站 www.mhprofessional.com 下载这个 ZIP 文件包。在搜索框中直接输入本书的书名或 ISBN 号，然后单击本书主页上的 Downloads & Resources 链接。也可扫描封底二维码获取本书下载资源。

目　录



第 I 部分

SQL、PL/SQL 和良好数据模型的重要性



第1章

SQL 和 PL/SQL

业务活动通常基于知识和信息，而所有这些重要的信息通常被存储于数据库中。关系型数据库通常只能由结构化查询语言(Structured Query Language，SQL)来访问。它是管理关系型数据库系统(Relational Database Management System，RDBMS)中所保存的数据的标准语言。过程化语言/结构化查询语言(Procedural Language/Structured Query Language，PL/SQL)是 Oracle 公司对 SQL 语言的扩展，它使你能够为 Oracle RDBMS 开发存储过程代码(例如 IF...THEN...ELSE、循环等)。

1.1 SQL 和 PL/SQL 介绍

数据库是存放所有重要信息的地方。当然，用户需要一种从数据库获取信息的方式，也需要能使信息保持最新状态的方式。数据库的优良设计对于其信息处理的使命来说非常重要。好的设计保证了其应该达到的性能和数据质量以及其他指标。可以在第 2 章中读到更多关于数据库设计方面的内容。数据保持最新状态很重要，因此必须有一种有效的方式维护它们。

SQL是管理RDBMS中的信息的标准语言。它允许查询信息并管理这些信息。SQL支持向数据库插入、更新和删除数据，当然也支持从其中查询数据。使用SQL是处理数据的最有效方式。这意味着它在数据库内执行一个又一个的操作，但仅向外界返回最终需要的数据行。这样使得处理和移动数据变得更安全，性能也变得更好。一些人喜欢把所有数据都拖到应用服务器、中间件，并在那儿执行过滤和连接。但显然这样做并不安全和有效。使用SQL和数据库的组合是处理数据的最好方法，因为它们就是为此目的而诞生的，而且SQL是数据库唯一能够理解的语言。

另一方面，SQL每次仅支持一条SQL语句的处理。这在许多场合是不够的，还需要过程化的结构来获得所需的业务逻辑和结果。为了填补这一空白，Oracle提供了PL/SQL。这是为Oracle RDBMS准备的过程化扩展。PL/SQL允许你使用过程化程序设计结构，例如IF...THEN...ELSE和循环，并有可能把代码保存于数据库中以便今后能从数据库内外调用它们。从Oracle Database 12*c*开始，Oracle对PL/SQL的性能做了更多提升。虽然使用SQL被认为是最快的方法，但是从12.1版本开始可以使用pragma UDF来定义模式级别的函数/存储过程。这几乎和纯SQL一样快。在WITH子句中使用函数/存储过程也可以和pragma UDF的执行速度媲美。Oracle Database 12.1以前版本中的普通方案级别的函数/存储过程则要慢很多。

1.2 SQL

创建SQL是为了与RDBMS系统交互。SQL提供了与RDBMS系统交互的唯一方式。但是和任何其他发明一样，SQL也不可能尽善尽美。其中的一个问题是：关系模型必须基于关系型理论，但遗憾的是SQL，的每一处细节都不是关系型的。这肯定会导致问题。为了避免这些遗漏，必须理解关系模型、了解SQL如何背离关系型、了解如何以关系型的方式去编写SQL。这方面的相关内容可以参阅C. J. Date的*Database in Depth: Relational Theory for Practitioners*一书(O'Reilly, 2005)。在本章中，我们要审阅一些SQL非关系型方面的示例。但是要学习更多关系模型，并学会编写真正关系型的SQL代码。相关内容可参阅C. J. Date的*SQL and Relational Theory: How to Write Accurate SQL Code*一书(O'Reilly, 2009)。

最典型的非关系型的SQL实施案例是重复行和空值。让我们先看看重复行。由于表结构是一个结果集(一个元组)，数学理论上表不应该包含重复行。数学上集合的概念中不允许包含重复的元素。这样，当为了更好的性能执行查询重写时，优化器就会做出基于以上的表不包含重复行的假设。但在一些场景下这样认为最终会返回错误的结果。因为不包含重复行的假设是错误的。另一个示例是错误处理。在有重复行时这会变得很困难，因为你不知道这些重复行到底是应该有还是不应该有。

因此如何处理重复行呢？一种情形是：数据库设计者在表上没有定义主键(Primary Key，PK)，或者至少没有定义唯一键，这样便允许表上出现重复行。另外一种情景是：设计者定义了没有包含唯一约束的代理键，这样导致没有任何约束真正在维护数据的完整性和质量。如果我们能够假设数据库设计者完全了解如何在每一个用到的表上定义主键(或者完全了解如何在每一个用到的表上定义使用代理主键时的唯一约束)，那么我们有理由相信不会存在重复行的风险。但是我们错了，由于SQL的处理方式，这种风险依然存在——不在数据库的基表中但可能在SQL查询的结果集中。

设想数据库中有两个表：CUSTOMERS表和ORDERS表。在这个场景中，为登记客户信息而准备的CUSTOMERS表包含这些列：customer_no和customer_name，还有一个去范式的为计算每个客户所有订单总数而存在的tot_order_count列。在ORDERS表中，我们存有客户的所有订单信息。它包含名为order_no、customer_customer_no和order_amount(订单的金额)这些列。把那个列称作customer_customer_no是因为我们的命名规则要求这样命名："父表名_外键名"。为了使你的开发项目有更好的一致性和质量，确立一个命名规则非常重要。好的数据库设计工具都对命名规则具有良好的支持。

UNION、INTERSECT和EXCEPT操作符默认会进行DISTINCT操作。但剩余的其他操作符，例如SELECT ALL或UNION ALL，都不会执行DISTINCT操作。假设我们有一个历史客户表(CUSTOMERS_HISTORY)。由于