Oracle PL/SQL Performance Tuning Tips & Techniques

Oracle PL/SQL 性能调优诀窍与方法

提升Oracle数据库整体速度、可靠性和安全性的最佳实践

[美] Michael Rosenblum 著
Paul Dorsey
张骏温 译

清华大学出版社

Oracle PL/SQL 性能调优决窍与方法

[美] Michael Rosenblum
Paul Dorsey 著
张骏温 译

清华大学出版社
北 京

Michael Rosenblum, Paul Dorsey
Oracle PL/SQL Performance Tuning Tips & Techniques
EISBN 978-0-07-182482-8

图书在版编目(CIP)数据

Oracle PL/SQL 性能调优诀窍与方法 / (美)罗森布拉姆(Rosenblum, M.)，(美)多西(Dorsey, P.) 著；张骏温　译. —北京：清华大学出版社，2015

书名原文：Oracle PL/SQL Performance Tuning Tips & Techniques

ISBN 978-7-302-41956-3

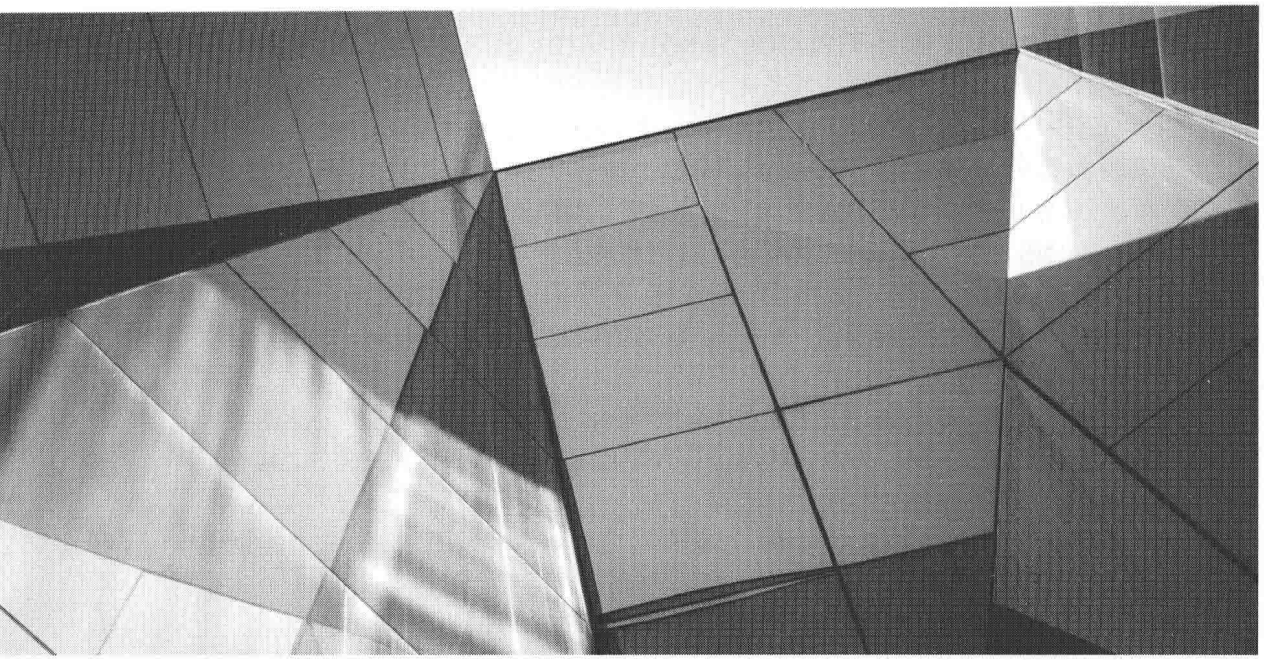
I. ①O…　II. ①罗…　②多…　③张…　III. ①关系数据库系统　IV. ①TP311.138

中国版本图书馆 CIP 数据核字(2015)第 257219 号

责任编辑：王　军　李维杰
封面设计：牛艳敏
责任校对：曹　阳
责任印制：何　芊

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 刷 者：北京鑫丰华彩印有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**16.25　　**字　　数：**426 千字
版　　次：2015 年 11 月第 1 版　　**印　　次：**2015 年 11 月第 1 次印刷
印　　数：1～3500
定　　价：49.80 元

产品编号：062382-01



译 者 序

众所周知，在关系型数据库领域，Oracle 是当之无愧的领导者。在国内各大行业、企业和政府部门的关键应用中，Oracle 数据库产品都被广泛使用，这已是不争的事实。既然如此，在实际开发和使用过程中，大型应用系统的管理人员和开发人员都会面临如何高效地用好 Oracle 数据库的问题。这其实涉及两个方面的问题：怎么使用和如何用好。从本人的经验来看，会用不难，用好却不易，而本书则详细地描述了如何用好 Oracle 数据库。

本书的两位作者——Michael Rosenblum 和 Paul Dorsey 是业内知名的数据库专家，他们不但有高深的理论造诣，而且还有极其丰富的实践经验，他们不仅是数据库设计的高手，还是数据库管理的专家。通过多年的亲身实践，他们发现数据库调优是一个非常值得深入研究的领域，并将自己多年积累的经验汇集成册，编写完成了本书。

按照本书作者的定义，所谓的性能调优就是“使代码运行得更快的过程”。在本书中，两位作者归纳阐述了大型 Web 应用系统的 9 大步骤，并对每个步骤可能消耗的时间进行了分析，说明了影响代码运行速度的原因有多种，并在本书中详细解释了影响数据库系统运行速度的根本原因。

作者从 PL/SQL 入手，根据其特性，将全书组织为 4 大部分共 12 章。

第 I 部分(第 1～3 章)从宏观的角度综合概述了系统调优涉及的关键领域，并结合 Web 应用系统的 9 大步骤探讨了性能调优问题的定位和根源。

第Ⅱ部分(第 4～6 章)阐述了 PL/SQL 与 SQL 的集成问题，这是 PL/SQL 使用中最关键的问题。用作者的话讲，最优化的核心就是在 SQL 和 PL/SQL 之间正确地分配负载，这涉及 SQL 中的 PL/SQL(用户定义的函数功能)、PL/SQL 中的 SQL(游标和批操作)、编写高效的触发器。

第Ⅲ部分(第 7～9 章)描述了 Dynamic SQL、缓冲机制及高级数据类型三类技术。

第Ⅳ部分(第 10～12 章)结合大量的案例，介绍了性能调优的日常工作内容、版本控制的意义及作用，并给出了性能调优的一些秘诀、技巧和理念。

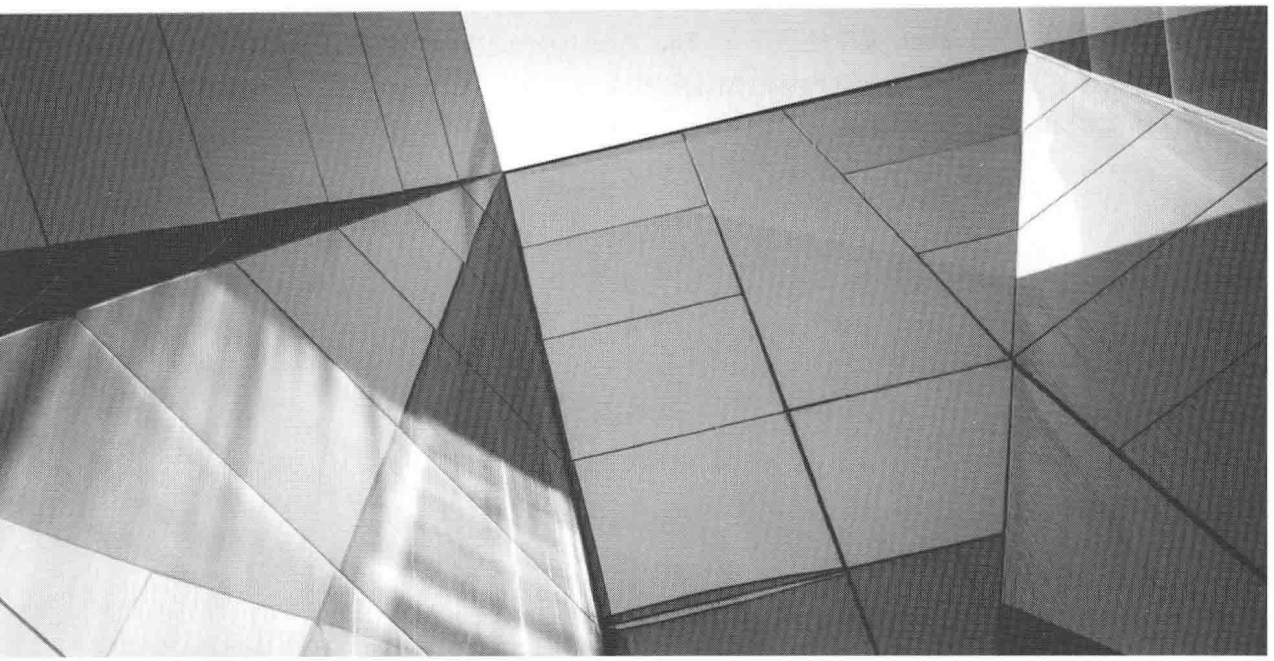
在本人看来，这是一本深入浅出的专业书籍，对于应用系统的开发人员及 DBA 都有帮助。本书语言精练简洁，叙述清晰到位，从宏观理念到微观技巧，面面俱到，而且给出大量案例，实用性很强。

很荣幸能够将这本好书翻译成中文版奉献给国内的广大读者。在这里要感谢清华大学出版社的王军和李维杰编辑，他们为本书的翻译投入了巨大的热情并付出了很多心血，没有他们的帮助和鼓励，本书不可能顺利付梓。

还要感谢甲骨文(中国)公司的技术产品售前总监许向东和她的团队，在百忙之中审阅了译稿的初稿，并提出很多修改意见。

本书全部章节由张骏温翻译，参与本次翻译的还有白宇宇、吴天爽、郭书华、贺风、温豆豆、赵爱华、薛群群、王欢，在此也一并谢过！

由于水平有限，翻译工作中可能会有不准确的内容，如果读者在阅读过程中发现有失误和遗漏之处，欢迎批评指正。敬请广大读者提供反馈意见，读者可以将意见发到 zjw@bjtu.edu.cn，我会仔细阅读读者发来的每一封邮件，这是一个很好的自我提高的过程。

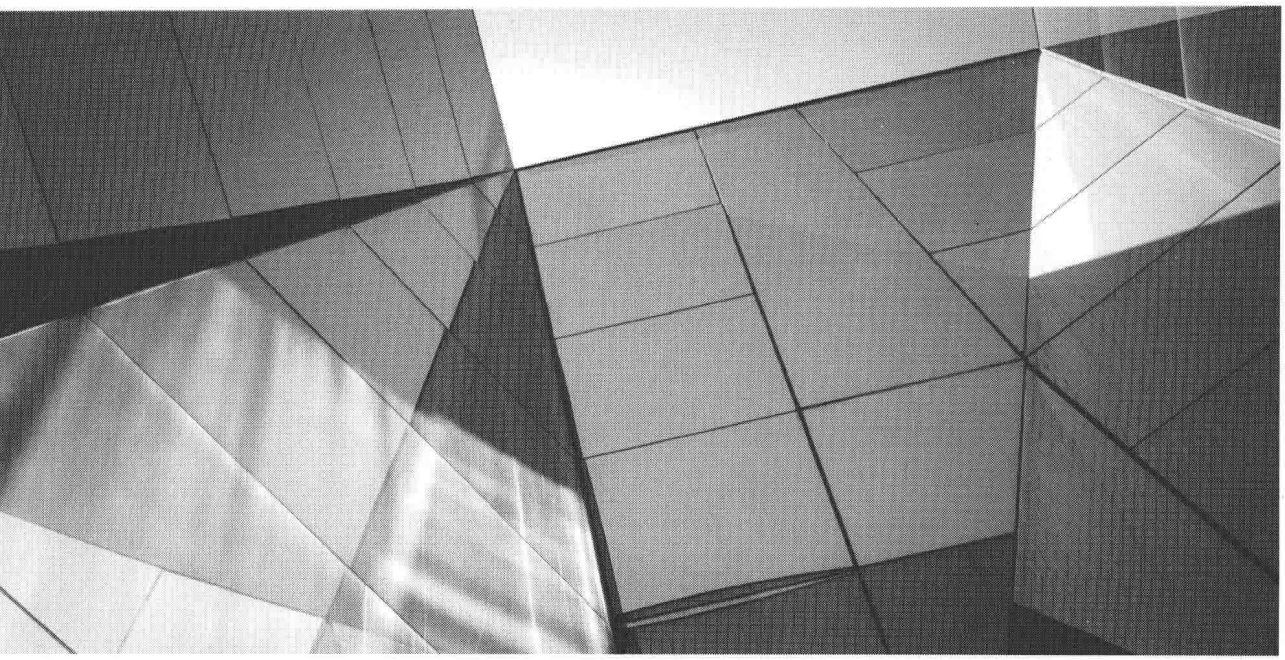


作者简介

Michael Rosenblum 是 Dulcian,Inc.的一位软件架构师/高级 DBA，负责系统调优和设计应用程序体系架构。Michael 通过编写复杂的 PL/SQL 例程和研究新特性来为 Dulcian 的开发者提供支持。他是 *PL/SQL for Dummies*(Wiley Press, 2006)一书的合著者，是 *Expert PL/SQL Practices* (Apress, 2011)一书的贡献作者，也是大量与数据库相关的期刊文章和会议论文的作者。Michael 是一位 Oracle ACE，是众多 Oracle 用户组会议的活跃主持人，包括 Oracle OpenWorld、ODTUG、IOUG Collaborate、RMOUG、NYOUG 等，并获得了 ODTUG Kaleidoscope 的 2009 年度最佳演讲人奖。他的祖国是乌克兰，他以优等生的身份毕业于基辅国家经济大学，获得信息系统专业的科学硕士学位。

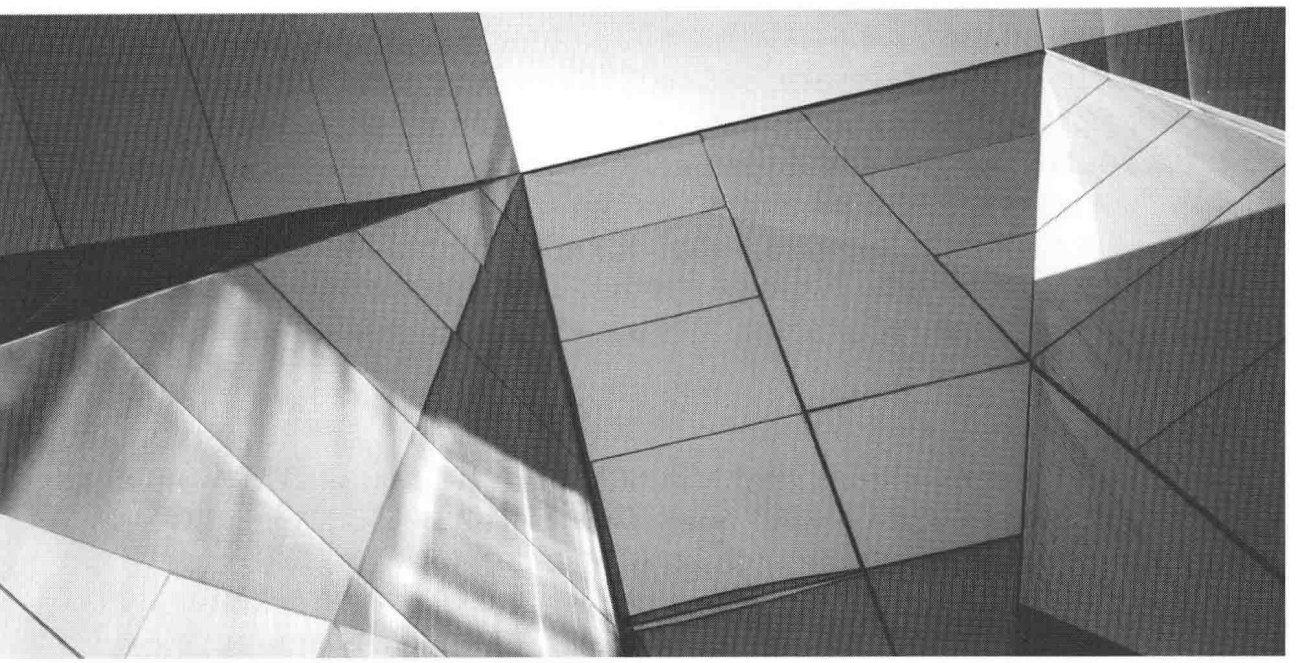
Paul Dorsey 博士是 Dulcian,Inc.的创始人和总裁，这是一家 Oracle 咨询公司，专门从事业务规则和基于 Web 的应用程序开发。他是 Dulcian,Inc.的 Business Rules Information Manager (BRIM)产品工具的总架构师，还是 Oracle 出版社出版的 7 本书的合著者，这些书涵盖了 Designer、数据库设计、Developer 和 JDeveloper 等主题，已被翻译为 9 种语言，他还是 Wiley 出版社出版的 *PL/SQL for Dummies* 图书的合著者。Paul 是一位 Oracle ACE，并且是第一个进入 IOUG SELECT 名人堂的人。他是 NYOUG 的名誉主席。Paul 于 2003 年被 ODTUG 授予年度志愿者称号，2001 年被 IOUG 授予年度志愿者称号，是最早被 Oracle 授予 Oracle 9*i* 认证大

师称号的 6 泰斗之一。Paul 提交的用于为 The Preeclampsia Foundation 收集数据的 Survey Generator 赢得了 2007 年度 Oracle Fusion Middleware Developer Challenge 奖，Oracle 选择他作为 2007 年度的 PL/SQL Developer。



技术编辑简介

Arup Nanda 从事 Oracle Database 和 PL/SQL 方面的工作已有 20 年(甚至在 PL/SQL 只能用于 SQL*Forms 而不是数据库中的时候就已如此)。他已撰写了 500 篇论文，参加了大约 300 个会议，是 5 本书的合著者，在 22 个国家讲授培训课程。Arup 是一名 Oracle ACE Director，是 Oak Table Network 的成员和 Exadata SIG 的 Board of Directors 的成员，是 *SELECT Journal* 的编辑。由于其专业水平和贡献，Oracle 授予 Arup 2003 年的年度 DBA 称号和 2012 年的年度 Enterprise Architect 称号。他居住在 Connecticut 州的 Danbury 市，闲暇时，他喜欢画水彩画、摄影和读书。



致　谢

Michael Rosenblum：

我想按时间顺序来表达我的谢意，从我的学生时代开始。实话说，我是一个很好的学生，但就是有点懒。如果类似的问题能用 5 行解释清楚的话，我绝不想用 10 行。非常幸运，我的初中、高中数学老师 Bronislava Olshevskaya 和 Tamara Mozdolevskaya (Lyceum #4，Kremenchuk, Ukraine)总是允许我可以不受课程内容的约束，去寻找其他方法，我非常感谢他们。当然，我得知道按书本要求怎么做，但找出自己的理论依据却更令我着迷。这是我专门从事性能调优的开始。随后，在移民美国后，我有关优化的概念的形成极大地受到我这位令人尊敬的合著者——Paul Dorsey 博士的影响。他费了好几年的时间才使我明白了还有“适度优化”这回事。具体来说，如果我能使耗在一个模块上的时间从 10 秒降到 0.05 秒，或许我就没必要再花一个星期的努力来把时间从 0.05 秒降到 0.04 秒了。另外，Paul 的体系结构观总能为我提供有关整个系统开发过程的宏图。我要感谢他的全部见解、工作和支持，这些见解、工作和支持并不局限于本书的写作，而是贯穿于我们相识以来的所有这些年。

若没有支持团队，也不可能有这本书。我们的项目经理 Caryl Lee Fisher，尽管她的计划有些疯狂，但她却能把 Misha's English 翻译为 American English(比我想象的还好！)并能够使整个项目按计划推进。我们的技术编辑 Arup Nanda，要求我们一定要诚实，我们宣称自己知道的任何事情，一定要真的知道。他深厚的背景，包括开发生命周期的所有方面(从 DBA 到系统架构

师)，都有助于确保本书可为多种不同类型的读者所用。不尽的感激献给 Alex Nuijten，他是 Oracle 的 ACE Director 和著名的 PL/SQL 专家。他自告奋勇地担当了本书的第一审稿人，并利用其宝贵的空余时间为确保本书在技术上的准确可靠做出了主要贡献。我还要感谢两位 XML 专家：Marco Gralike，他是 Oracle 的 ACE Director；以及 Mark Drake，他是 Oracle 的 Senior Product Manager。感谢他们的支持、理解和耐心(太多了！)。

最后，但并非最不重要，我要感谢我的家庭。为了使本书的计划得以实现，Elina 和女儿们做出了许多牺牲。许多个周末，当我撰写不同章节时，她们要保持安静，我知道这对于三个孩子来讲有多难。另一方面，我无法想象向一个 8 岁的孩子解释出版过程的细节是一件多么令人着迷的事情。

Paul Dorsey：

首先且最重要的，我要感谢我的合著者 Michael(“Misha”) Rosenblum。我认识 Misha 已有 15 年了。我是通过一位前雇员、老朋友 Sergey Guberman 认识 Misha 的。Sergey 当时在教 Oracle 的课程，Misha 是他的学生。Sergey 给我打电话并说：“你最好雇用这个家伙。作为一个学生，他写的 SQL 比我写的都好。”确实，Sergey 是对的。我从未遇到过更有技术创造性和更聪明的头脑。现在有一个流行的段子，说 Misha 将会发布一个不可能解决的问题，但几个小时后，Misha 就能给出答案。

在签本书的出版合同时，我被列为第一作者。但作为合著者，我的第一个行动就是把 Misha 写为第一作者。我知道他的技术经验是本书成书的动力。我确信，讲这个故事是正确的，但我想要承认，这个故事是 Misha 的。在我们共同度过的这些年中，眼看着他成长为业内最好的技术专家之一，真的非常令人高兴。

Caryl Lee Fisher(除了其他职业职责外)还帮助我们使本书的写作能够按计划推进，并且本书编辑方面的所有重大提升都是她做的。她协调各方，使 Misha 和我能够集中精力于写作。在写作一本书的过程中，会有大量与写作无关的事情，Caryl Lee 都为我们把这些事情处理得妥妥帖帖。我们感谢你所做的一切。在过去的 20 年中，Caryl Lee 为我此前出版的 9 部书的每一部都提供了帮助。我确信，没有她的帮助，就绝不会有这第 10 部书。

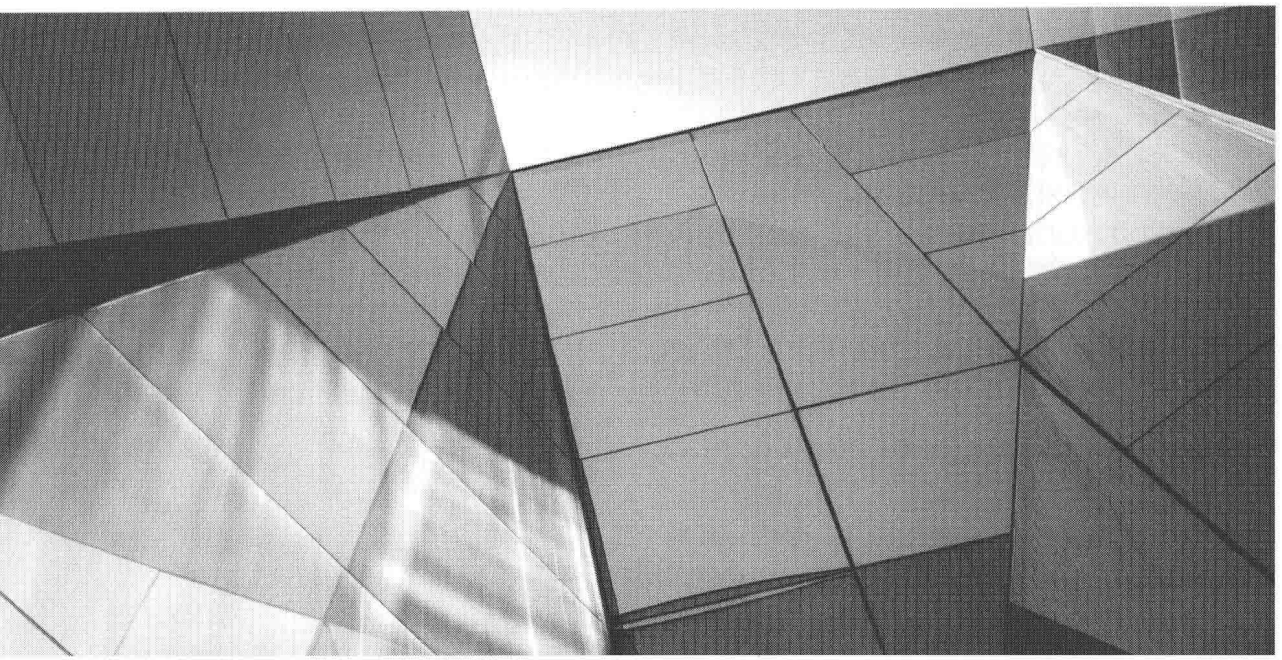
Arup Nanda(世界上最好的 DBA 之一)是我们的技术编辑。他能献出宝贵的时间来帮助我们出版这本书让我感到非常荣幸。他抓住了本书的许多问题，这些都有助于使本书比预期的要更好。他的贡献对于本书的准确性和有用性来说都是非常重要的。

Alex Nuijten 为我们做了完整的技术审核并产生了巨大的影响。Marco Gralike 在本书的 XML 部分为我们提供了帮助。很难说清，在只给 XML 很少篇幅的情况下，我们能介绍些什么。Marco 也帮助我们进行了部分技术审核，Mark Drake——Oracle 的 Senior Product Manager——也给予了我们大量的帮助。

没有上述这些人所献出的宝贵时间，本书就不会是现在的样子。

本书两位作者要感谢他们的同事 Grigoriy Novikov 和 John Ryzdy，感谢他们作为研究团队成员所付出的努力以及对那些最终落实到书中的想法所给出的参考意见。

本书两位作者还要感谢由 Paul Carlstroem、Amanda Russell、Janet Walden 和 Bill McManus 组成的 McGraw-Hill Professional 团队，以及 Cenveo Publisher Services 的 Yashmita Hota，感谢他们给予本书的帮助。



序

本书的书名有些保守，书中的实际内容比起书名所蕴含的内容要广。在第 1 章中，作者们实际上介绍了如何认识那些利用 Oracle Database 来实现数据库记录的应用程序的端到端的性能，以及在那样的上下文中，如何理解这种应用程序最佳的从上到下的体系结构。他们演示了使用 PL/SQL 而不是完全弃之不用会给性能带来巨大的好处。只有当使用 PL/SQL 的环境建立起来时，本书才会发挥其最大作用。在这一最优体系结构中，PL/SQL 的目的是要发布 SQL 语句并处理结果，在这个意义上，SQL 语句可被看作由若干子程序组成的闭包中的一种特殊的子程序，这些子程序实现 PL/SQL 中条目的网络影响。所以，在数据库中划分 PL/SQL 和 SQL 子系统各自的工作是整个 PL/SQL 性能的最关键的决定因素。PL/SQL 和 SQL 的背景及相互之间的影响在本书的前两部分介绍。

但是，试图提升一个不正确的应用程序的性能是没有意义的！首先要正确，然后才能考虑性能。所以，笔者要用本序来详细说明在本书所涉及的问题领域中，如何将软件工程的核心的、通用的原则运用于最大化应用程序的正确性机会。由于一个非常愉快的巧合，我们发现，我们不但有蛋糕，而且还能吃；用于最大化应用程序的正确性机会的体系结构方法与本书第 1 章中所描述的可以带来最优性能的方法是一样的。

没有人会否认，成功实现一个大型软件系统依赖于好的模块设计。模块是一个代码组织单元，它实现系统功能的一个相干子集，通过一个直接表达且只表达此功能的 API 来将这一功能展现出来，并把所有实现细节都隐藏到这个 API 之后。当然，这个模块的分解原则是可以递归

使用的：把系统分解为数量较少的主要模块，然后把每个主要模块再分解，如此下去。可以证明，这条原则在众多的软件工程最佳实践原则中是最重要的——并且在过去的 50 年中，这一观点是得到普遍认可的。

目前，人们会把一个以 Oracle Database 为其持久机制的应用程序粗略地分解为数据库模块、应用服务器模块和客户端模块。

数据库模块的最终实现是 SQL 语句，这些语句从应用程序的表中查询内容或改变表中的内容。然而，最常见的是，对一个单独表的操作只能实现该应用程序的说明书中所描述的业务交易的一部分。典型的例子是在某特定的银行里某个应用程序所管理的所有账目中的现金转账业务功能。将这一功能参数化主要是通过识别源账目、目标账目、现金量等实现的，其他参数，如交易产生的日期、交易备忘录等有时也是需要的。由此立即就会引出如下 API：

```
FUNCTION Transfer_Funds (Source IN..., Target IN..., Amount IN..., ...)
  RETURN Outcome_t IS...
```

这个 API 就是一个函数，反映一个尝试被禁止的可能性，其返回值是非标量的，反映出如下事实：该尝试被禁止的原因可能来自多个值，例如，可能包括源账目中的短缺量。

我们立即就可以看出有许多不同的设计选项。例如，每种账目都可能有一个单独的表，这种情况反映的事实是不同种类的账目具有不同的账目细节；也可能是一个单独的表，其中具有一个账目种类列，以及一个保存每种账目细节的表族。类似地，也会有一个账目拥有者的代表，还可能会有不同的种类，诸如个人和公司，且都有各自的细节。毫无疑问，还会有一个表来保存未来肯定要进行的转账需求。

上述指向是明确的：一个能够精确反映功能说明的 API 设计可以通过不同的方法来实现。结论同样明确：

数据库模块应通过一个 PL/SQL 的 API 来展现。

而名称的细节和表的结构以及实现它们的 SQL，则应该安全地隐藏在应用程序服务器模块之后。

这个范式有时被称为“厚数据库”。它为有关何时使用 SQL、何时使用 PL/SQL 的讨论设定了上下文。应用程序服务器可能发布的唯一一种 SQL 语句是一个调用 API 的多个子程序中某个子程序的 PL/SQL 匿名块：

```
BEGIN :r := Transfer_Funds(:s, :t, :a, ...); END;
```

在第 5 章中，作者们主张使用 SQL 著名的声明式的基于集合的方法，这有利于在 PL/SQL 中程序性地编程实现相应的功能。当然，前提是从要实现对数据库顶层调用的 PL/SQL 中执行 SQL，而不是用执行 SQL 来代替 PL/SQL 的这种使用。

作为一种红利，PL/SQL 比其他编程语言更适用于那种执行 SQL 语句并处理其结果的任务。例如：

- 这种语言支持的嵌入式 SQL 可以成为语法定义及其语义的固有部分。
- PL/SQL 标识符可被用于嵌入式 SQL 中，而在普通 SQL 中，嵌入标识符的地方是要使用占位符的。这一事实不但把编程人员从编写代码以便编程连接占位符这样的杂务中解脱出来，而且能保证代码不受 SQL 注入攻击。
- 将 PL/SQL 变量和类型的声明固定于模式级表(%TYPE、%ROWTYPE 和 FOR LOOP 指针的迭代器)中列的数据类型中的能力意味着 PL/SQL 程序会随着它们所操作的表的定义的改变而自动地调整自己。只对其他 PL/SQL 程序、表、视图等产生静态引用的 PL/SQL 程序，能够保证它们只利用它们所依赖的最近的定义来执行。

- 具有如下能力：通过有选择地给 PL/SQL 子程序授予 EXECUTE 特权来控制哪些用户可以执行哪些业务功能，而不是通过授予 SELECT、INSERT、UPDATE 和 DELETE 特权来控制哪些用户可以看到和修改哪些特定表中的数据，这一能力是保护数据完整性的关键。
- PL/SQL 具有固有的异常处理机制，SQL 错误(诸如著名的 ORA-nnnnn 错误代码)会被映射为 PL/SQL 异常，另外 COMMIT 和 ROLLBACK 这两条 SQL 语句在 PL/SQL 中成为内嵌的 SQL 语句，这两个事实保证了改变多于一个表时业务功能的一致性。这是保护数据完整性的又一个关键因素。
- PL/SQL 的 SQL 处理是性能最优的，不仅因为 SQL 执行在与发布它的 PL/SQL 相同的服务器进程中，还因为各种各样的后台优化，诸如著名的软解析避免。

当然，PL/SQL 独一无二地拥有这些特性并不是偶然的。在发明它们的时候，它们就已经被特殊地定义为这种语言应该满足的需求。

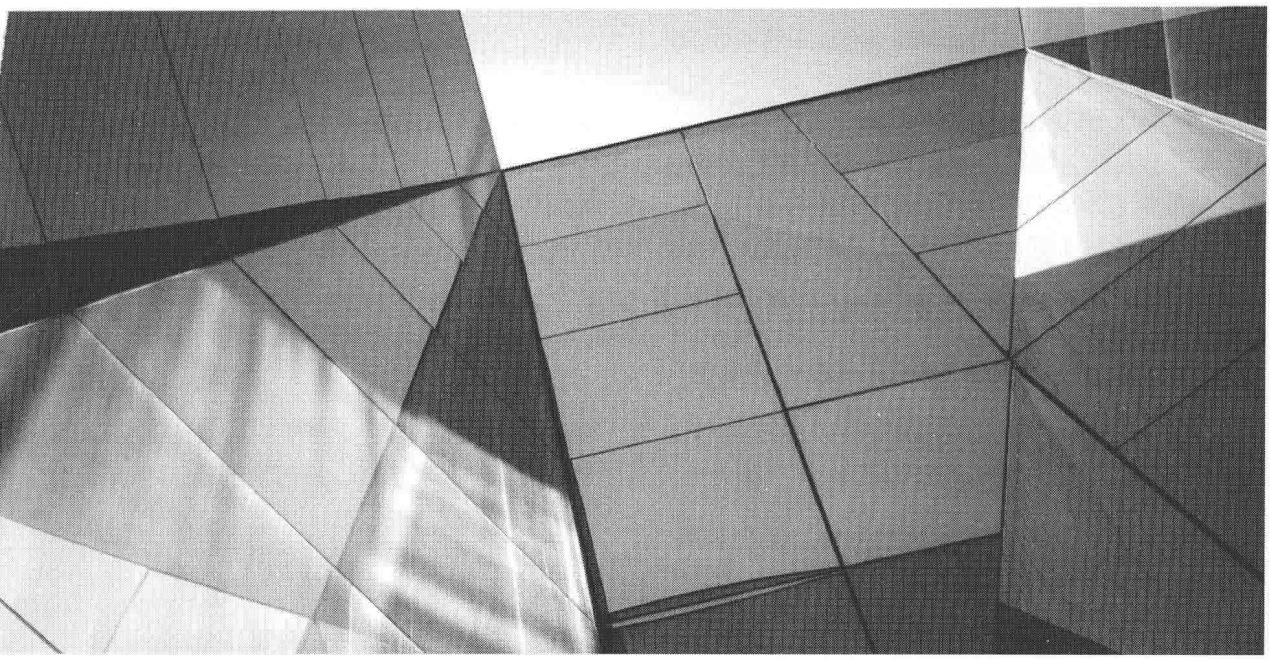
笔者认识许多严格遵循厚数据库范式的客户；也认识许多不遵循的人——他们都到了这样的程度，把所有对数据库的调用都实现为对应用程序的表进行精确操纵的 SQL 语句。这当然使数据库配不上“模块”这个词！第一组中的客户似乎通常对他们的应用程序的性能和可维护性很满意。具有讽刺意味的是，第二组中的客户总是抱怨性能问题，因为执行一个单独的业务交易通常都会涉及应用程序服务器模块和数据库模块之间的许多次往返。并且他们也抱怨维护问题，因为甚至给应用程序打个小补丁，都意味着要改变数据库模块的实现和应用程序服务器模块的实现。

一个使用 Oracle Database 作为其持久机制的应用程序，其设计在不可胜数的不同软件系统中独一无二，笔者相信，该应用程序没有哪个特性会与广为接受的人们的智慧相违背。

然而——现实生活中好像总是有“然而”——开发者使用系统体系结构无法控制的现有系统进行工作，有时还要使用非功能因素(例如生产应用程序服务器的开发框架的可用性)胜过功能因素的新系统进行工作。在这类系统中，表达数据库 API 的是正规的 SQL 语句——可以联合多个表的 SELECT 语句，以及对单个表进行操作的 INSERT、UPDATE 和 DELETE 语句——而不是 PL/SQL 匿名块。本书介绍了如何通过用单词“视图”替换单词“表”(这是一种对应用程序服务器代码的透明改变)来把 PL/SQL 隐藏在视图抽象之后使用，以便能给 API 带来与子程序抽象所能带来的相同的模块化益处。在 PL/SQL 中编写的 INSTEAD OF 触发器允许单独的 INSERT、UPDATE 和 DELETE 语句对某个视图操作以便改变数个底层表。并且对于复合查询需求，PL/SQL 的流水线表功能(pipelined table function)可用作视图的编程行源(programmatic rowsource)。在本书的第 II 部分会介绍本通用领域中的一些有趣技术。

作者们的注意力并不仅局限于高层的体系结构概念。书中还有丰富的、具有良好说明的性能增强技术和建议，可用来对全部的细致入微但又十分关键的细节进行处理。每一名专业的 PL/SQL 编程人员都应该学习本书并按照其中所教的来做。

Bryn Llewellyn
著名产品经理，Oracle Corporation HQ
Redwood Shores CA 94065 USA
2014 年 4 月



前　言

当你听说或读到有关 PL/SQL 性能调优时，一定要明白，通常人们说的某些东西和你想象的截然不同：不仅仅是 PL/SQL 代码本身！真正的主题要宽泛得多——人们想要的，是把 PL/SQL 用作粘合剂，把所有能得到的元素和技术(包括 SQL、XML、Java 等)片段粘合在一起，以此来优化 Oracle 数据库。

PL/SQL 的普适性决定了本书的结构。第 I 部分(第 1～3 章)始于“宏图”而不去管代码细节。系统调优的全局性方法把努力集中于更为关键的性能问题领域，而不是仅集中于最可见的那些问题上，这些方法是通过用户需求的 9 步表示法来说明的。然而，如果“归咎于数据库”的陈词滥调还存在的话，就需要有最好的、最高级的设备(要么来自 Oracle，要么来自本土)来解决这些问题。

第 II 部分(第 4～6 章)解决 PL/SQL 使用中最关键领域的问题，即它与 SQL 的集成。这一主题很容易分成三个逻辑块：SQL 中的 PL/SQL(用户定义的函数功能)、PL/SQL 中的 SQL(鼠标和批操作)、编写有效的触发器。贯穿本部分的主题是，最优化是由在 SQL 和 PL/SQL 之间适当地分配负载组成的。当然，编写有效的代码和使用最佳实践是必要的，但目前数据库的环境是复杂的，需要许多不同的资源。由于上述因素，就需要通过调整功能平衡来获取可用资源的最大价值。

PL/SQL 是一种丰富的语言环境，包含很多同等重要的特性。然而其中有些特性的同等重要性比其他特性更高。在第III部分(第 7～9 章)所涉及的主题中，作者们选取了最常用的性能优

化技术中的两个技术(Dynamic SQL 和缓存机制)，以及最被滥用和误解的技术之中的一个技术(高级数据类型)。

第Ⅳ部分(第 10～12 章)提醒我们，尽管发现新功能和再架构数据库环境具有很大的乐趣，但事实上，性能调优专家的日常工作却主要是进行细致的、耗时的研究，随时准备和进行修整(此处就需要版本控制了！)，并调整各处出现的看似微小的事物(但它们合并起来就不小了！)。

本书包括 4 部分，分为 12 章。

第 I 部分：PL/SQL 性能调优的核心理念和要素

第 1 章　“PL/SQL 在当前开发中的角色”

本章讨论性能调优对任何数据库系统的重要性。本章介绍 Web 应用程序所用的 9 步过程，并识别所有可能产生性能问题的地方。本章也提供处理此类问题领域中每种问题的策略。

第 2 章　“DBA/开发者的界线：工具和特性”

第 2 章讨论开发者和 DBA 间的协同，这是创建成功的、有效执行的数据库系统所需的。本章列出并解释性能调优的关键要素，包括数据字典视图、日志、跟踪、PL/SQL Hierarchical Profiler、PL/Scope 和 RUNSTATS。

第 3 章　“PL/SQL 中的代码插桩”

本章介绍多个用于定位性能问题的代码插入方法。本章讨论如何确定性能问题是否真的是由数据库问题引起的，还说明了诸如调用栈 API、差错栈 API 和计时器之类的信息源是如何被用来帮助定位和检修数据库的性能问题的。

第 II 部分：链接 SQL 和 PL/SQL

第 4 章　“扩展 SQL 的范围”

正确理解用户定义的函数在 SQL 环境中是怎样工作的是成功开发数据库的最关键因素之一。第 4 章涉及如何管理用户定义的函数的统计数量和如何影响调用总数。本章也包括 PL/SQL 延伸 SQL 的已有功能(诸如定制聚合功能和返回对象集合功能)等的场景。只有 Oracle Database 12*c* 拥有的以降低 SQL 和 PL/SQL(PRAGMA UDF 子句和 WITH 子句中的各种函数)之间上下文切换的成本，甚至以取消这种切换为目的的特性，被用来说明管理此类切换的重要性。

第 5 章　“用集合的概念来思考”

本章强调在集成 SQL 和 PL/SQL 时以集合的方式进行思考的重要性。数据量非常大时，需要内部优化，这可以用基于集合的方法来完成。FORALL 子句和 MULTISET 操作的适当使用是创建成功的和可伸缩的数据库解决方案的关键因素。

第 6 章　“使用触发器”

本章涉及触发器在系统性能中的关键角色，包含不适当使用触发器导致系统过载并引起性能下降的例子，以及对反规范化的讨论。有关几个方法的权衡的分析有助于决定何时使用、何时不使用特定的触发器类型。

第III部分：调优人员的工具包

第 7 章　“不仅限于标量数据类型”

高级数据类型是 Oracle 开发者和 DBA 需要理解和使用的重要工具。第 7 章讨论 LOB 和 XML 的用法以及它们可能对系统性能的影响。本章也包含 XMLType 和 XQuery 语言的例子，还有使用高级数据类型的最佳诀窍和技术。

第 8 章　“保持使用缓存”

Oracle 数据库有不同的与 PL/SQL 相关的缓存选项。对于有效的数据库系统运行来讲，了解这些机制是如何工作的以及会产生对性能有影响的什么样的副作用是很关键的。本章讨论各种不同的方法，诸如 DETERMINISTIC 子句、标量子查询、PL/SQL Function Result Cache 等，如何能用来提高数据库的性能。

第 9 章　“射击移动目标”

Dynamic SQL 是一个广泛涉及但又常被误解的概念。错用这一特性会给系统带来极大的毁坏。第 9 章介绍如何正确使用 Dynamic SQL 来减少所需的代码量和使得系统管理更容易、更有效。近来，在 Oracle Database 版本 11*g* 和 12*c* 所改进的内容中，已经扩展了对 Dynamic SQL 的支持，本章对此也有讨论。

第IV部分：日常生活中的 PL/SQL

第 10 章　“来自战壕的传奇”

在第 10 章中，你会看到是什么使性能调优专家不停地忙碌。通常，这种工作最耗时间的地方不是如何修复问题，而是如何确切定位问题区域。把“这个程序太慢”这一问题精确到“这个函数常出问题”需要大量的技巧和耐心。另外，确信整体系统架构是稳定的且能够支持需求也是需要大量时间的。总之，所有的事情都可归结为管理和预测资源利用率。书中使用大量真实的例子来说明这一点。

第 11 章　“真实系统中的代码管理”

尽管版本控制与性能调优没有直接的联系，但它在系统的生命周期中却扮演着重要的角色。本章介绍如何在生产环境中执行计划好的修改而不产生任何新的问题和副作用。本章也探讨了代码管理的不同方法(人工的和自动的)，包括 Oracle 自己的 Edition-Based Redefinition (EBR)。

第 12 章　“额外的秘诀、技巧和理念”

每个 Oracle 开发者和 DBA 都在长期的工作中积累了一系列用于成功创建数据库系统的秘诀和技巧。第 12 章提供作者从长期经验中领悟到的最佳实践，可帮助读者优化和提高代码的效率。所涉及的主题包括通用数据类型的空间分配、通过参考来传递变量和管道化函数。

本书适合于以下读者：

- 希望提高开发技能的初中级 Oracle PL/SQL 开发人员。
- 有其他编程语言(例如 Java)经验且正在 Oracle 环境中工作的开发人员。
- 运行 Oracle 应用程序的组织中的技术人员，包括 DBA 和数据库架构师。

示例获取

书中所有的 SQL 脚本、程序和其他文件都能从 Oracle Press 网站 www.OraclePressBooks.com 下载。文件都压缩在一个 zip 文件中。下载该 zip 文件后，需要将其内容解压缩。解压缩后，会产生一个名为 PLSQL_Tuning 的文件夹，其中从第 2 章开始，每一章都有一个专门的子文件夹(第 1 章没有任何脚本)。