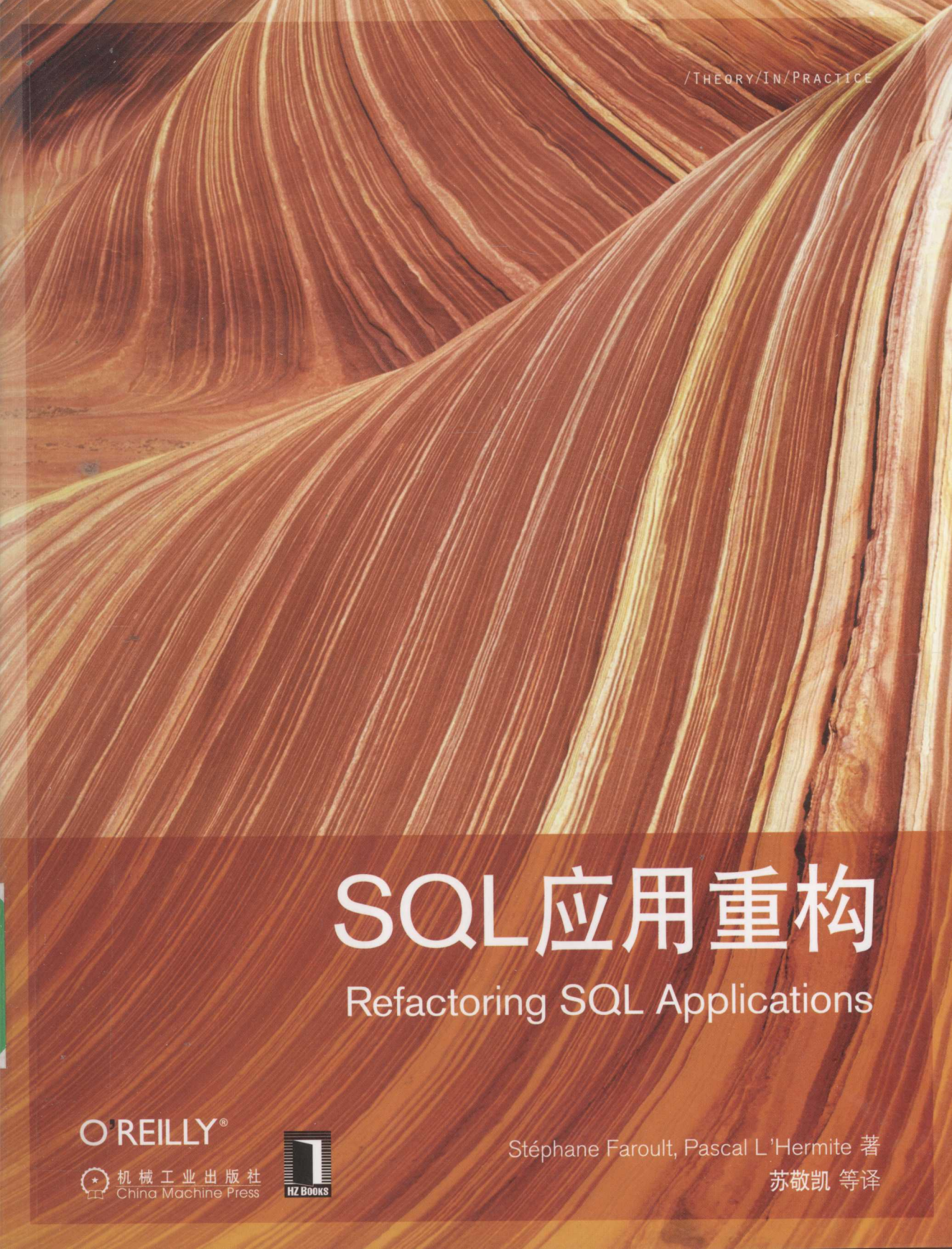

SQL应用重构

Refactoring SQL Applications

O'REILLY®
机械工业出版社
China Machine Press
HZ BOOKS

Stéphane Faroult, Pascal L'Hermite 著
苏敬凯 等译

SQL应用重构

Stéphane Faroult & Pascal L 'Hermite 著

苏敬凯 等译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权机械工业出版社出版

机械工业出版社

图书在版编目（CIP）数据

SQL 应用重构 /（美）法拉特（Faroult, S.）等著；苏敬凯等译．－北京：机械工业出版社，2009.6

（O'Reilly 精品图书系列）

书名原文：Refactoring SQL Applications

ISBN 978-7-111-26358-6

I. S… II. ①法… ②苏… III. 关系数据库－数据库管理系统 IV. TP311.138

中国版本图书馆 CIP 数据核字（2009）第 021349 号

北京市版权局著作权合同登记

图字：01-2009-1617 号

书　　名 / SQL 应用重构

书　　号 / ISBN 978-7-111-26358-6

责任编辑 / 李俊竹

封面设计 / Mark Paglietti，张健

出版发行 / 机械工业出版社

地　　址 / 北京市西城区百万庄大街 22 号（邮编 100037）

印　　刷 / 北京京师印务有限公司印刷

开　　本 / 178 毫米 × 233 毫米　16 开本　18.5 印张

版　　次 / 2010 年 1 月第 1 版　第 1 次印刷

定　　价 / 49.00 元（册）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com

O'Reilly Media, Inc. 介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc. 授权机械工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc. 是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc. 一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc. 是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc. 具有深厚的计算机专业背景，这使得 O'Reilly Media, Inc. 形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc. 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc. 还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc. 依靠他们及时地推出图书。因为 O'Reilly Media, Inc. 紧密地与计算机业界联系着，所以 O'Reilly Media, Inc. 知道市场上真正需要什么图书。

译者序

在翻译本书时，正流行着几本名为“啥啥之美”的书。本书也可以说讲的就是“如何将SQL写得更美”。

SQL之美就在于用清楚合理的方式告诉DBMS你要做什么，让DBMS总能以最优的方式高效地完成任务。SQL是一种描述式语言，在描述中要表达清楚能够影响优化器决策的要点，同时避免过程化思维的那些细节。而要做到这一点，理解优化器的工作方式和SQL的理念至关重要，正如某条广告语讲的那样“知其道，用其妙”。

本书作者从事数据库咨询工作多年，在本书中，不仅举出了很多具体生动的例子，还讲述了很多重构SQL应用的思想。无论你是DBA还是程序员，本书都能帮你写出或改写（也就是重构）出优美的SQL语句、清楚明了的处理过程，从而交付出高效的系统，赢得众人的赞许，就像书中的很多故事那样。

参加本书翻译的还有赵龙刚、金振林、周志强、杨宁等。本书翻译力求忠于原著，但由于时间仓促，译者水平有限，翻译错误和不妥之处在所难免，欢迎广大读者批评指正。

苏敬凯

2008年12月

目录

前言

我写书的目的是让它对于能理解它的人有用，因此，我更应当去探究问题的真相，而不是想当然。

——尼可罗·马基亚维利

《君主论》第15章

在本书的背后有一个故事。那时我刚刚写完《The Art of SQL》，该书还没有开始销售，该书的编辑Jonathan Gennick就产生了让我写一本关于SQL重构的书的想法。我知道SQL，但从未听过SQL重构。于是，我Google了一下这个词，在莫里哀的名剧《贵人迷》中，茹尔丹先生（Monsieur Jourdain），一位富有但没受过良好教育的男子，在他成年以后才开始上一些课程，他惊讶地发现原来他一辈子一直都在说“散文”。就像茹尔丹先生那样，我发现多年来我一直在重构SQL代码，却不知道这个词——对客户的系统进行性能分析会非常自然地引发“通过增量的小改动来改进代码同时又不改变程序”的行为。重构SQL应用就是：尽你所能设计出一个好数据库来，并对架构和程序进行合理布局以使对数据库的访问更加高效。另一个问题就是：你所面对的系统未必是一个刚开始设计的系统，它可能已经运行了若干年，一切都已超出了控制而你却还必须要使用它，那么如何能使这些系统的性能达到最佳呢。以这样一个在我的职业生涯中经常出现的视角来展现SQL语言，这种想法确实有些吸引力。

在写完一本书之后，最不想做的事情就是再写另一本书。但这个想法却一下子抓住了我的心。我曾和几个朋友讨论过这一想法，其中一个人是我所认识的最令人敬畏的SQL专家。这个朋友爆发了，这些时髦词汇使他义愤填膺。不过，这一次我不能同意他的意见

了。确实，这一首先由Martin Fowler（注1）普及开来的“通过小得几乎微不足道的局部修改来改进代码”的想法看起来就像一种时尚——完全是由那些刚刚从大学毕业的企业顾问们用报告堆积出来的时尚。但对于我来说，重构的真正意义不仅在于“我们不再把那些已投入生产的代码视作是神圣不可侵犯的”，还在于我们认识到“只要经过一点努力，很多表现平庸的系统就能够表现得好很多”。重构还是一种认可，承认“性能不令人满意的错误在于我们自己，而不在于我们的运气”，这也确实是业界的一个启示。

在很多地方我都看到，IT经理们对于性能有着一种几乎悲观的态度，人们感觉是受着命运的折磨，他们把最后的一点希望放在“调优”上。如果数据库和系统管理员的努力失败了，在他们看来，唯一的选择就只能是签发订单采购性能更高的硬件了。我也读到过很多自称是数据库专家的人所写的审计报告，在将系统工具的输出重新编排了一下格式之后，他们就得出了“需要提高某几个参数”以及“应该增加更多内存”的结论。公平地说，有些报告中也提到了应该对一些糟糕的查询进行“调优”，但除了把SQL语句的执行计划贴在附录里之外，没有更明确地说明如何调优。

我已经很多年没有碰过数据库的参数了（客户的技术团队通常很称职）。不过，我改进了很多程序，毫无畏惧地研究它们，我尽可能和开发人员一起工作，而不是呆在我的象牙塔里从很远的地方发号施令。我几乎总是遇到渴望学习和理解的人、几乎不需要鼓励就能走上正确道路的人、喜欢学习SQL技能的人以及能很快开始为自己设定性能目标的人。

当流逝的时光将写书的痛苦从我的记忆中抹去之后，我做出了果断的决定，又一次开始了写作，因为我打算扩展这些在我与开发人员一起工作时想要传达给他们的思想。数据库访问可能是能从代码改进中获益最多的区域之一。我写本书的目的不是给出一些具体的做法，而是给出一个框架，以此来改进我们周围那些不够理想的SQL应用，而不用从头开始重写它们（尽管有时重写会有非常强的诱惑力）。

为什么重构

大多数的应用迟早都会遇到性能问题。最好的情况是，一些辉煌的应用系统曾经是成功的，随着时间的推移，它们要处理的数据量已达到了设计时根本想不到的程度。因此，这些旧程序需要重生，直到在生产环境中投入了一个新的应用系统来替代它为止。最坏的情况是，在产品上线之前所进行的性能测试中，就发现性能不能满足服务水平要求。

注1：Fowler, M. 等所著的《Refactoring: Improving the Design of Existing Code》, Boston: Addison-Wesley Professional 出版。

而介乎于最好与最坏之间的情况是，数据量增长、增加新功能、软件升级或配置变更都时常能揭示出一些一直隐藏着的瑕疵，而又不是永远都可以选择回溯。所有这些情况都有极其苛刻的性能改进截止期限和很高的压力。

首先到达的救援队通常都是系统工程师和数据库管理员，他们被请来表演神奇的参数舞蹈。除非之前没有注意到某些非常大的错误（这确实发生过），对于系统性能的提升来说，数据库和系统调优的帮助通常很有限。

这时，下一步的传统做法一直就是：为应用系统中投入更多的硬件。这是一个成本非常高的选择，因为硬件的价格中可能还包含了更高的软件许可成本。这样做还会中断业务的运行，并且必须为此制订计划。令人担忧的是，无法真正保证这一投资能有所回报。很多大型硬件升级后都没有达到预期效果，这种情况不止一次地发生过。似乎与我们的直觉相反，很多大型硬件升级的恐怖故事实际上是：硬件升级反而导致了性能的退化。在很多情况下，在一台机器里多加几个处理器也只不过是进一步加剧了进程之间的争夺而已。

重构的概念引入了一个在调优和大型硬件投入之间的急需的中间状态。在Martin Fowler的那部关于重构的有创意的著作中，他所关注的是面向对象技术。但是数据库所处的情景与用对象语言或过程语言编写的应用程序有着明显的差异，而这些差异也带来了一些对于重构工作的特殊的曲解。例如：

小改动有时不那么小

由于SQL语言的描述式的本质，对于代码的小改动往往给在SQL引擎中的执行带来巨大的影响，这也就会导致性能上的巨大变化 —— 可能更好也可能更差。

对改动的有效性进行测试可能很困难

检查返回一个值的函数非常简单，只要检查在各种情况下代码修改前后的返回值是否相同就行了。而在改写了一个主要的update语句后，要检查一个大表的内容是否保持相同，这完全是另外一回事。

上下文环境往往是决定性的

在问题出现前的若干年里，数据库应用系统可能一直运行得非常令人满意；但在数据量或负载超过了某些阈值之后，或者在一次软件升级改变了优化器的行为之后，性能往往会突然变得无法让人接受。对于数据库应用系统的性能改进工作通常就发生在一场危机当中。

因此，数据库应用系统是重构的一个完全不同的战场，而同时这种努力也可能是（往往也就是）有高额回报的。

数据库访问的重构

长时间以来，数据库专家们都知道，提升性能最有效的方式是：先检查索引，再评审和细调访问数据库的模式。虽然表面上SQL语言有着描述式的本质，但这一语言还有这样的恶名：功能完全相同的语句可以有不同写法，而有时不同的写法在执行时间上有着令人吃惊的差别。

然而，数据库访问的重构不仅仅是改写有问题的查询语句，但大多数人却都止步于此了。例如，经过SQL语言若干年来缓慢但持续的改进，有时开发人员可以编写一些高效率的语句来替代以前的含有多条语句的复杂过程，也能完成同样的功能。很多新内置到数据库引擎中的机制能够让你以一种与过去不同的方式来更高效地做事。按照（数据库的）新特性来检查旧程序往往能产生大幅度的性能提升。

如果重构背后的原因只是期望利用新特性使旧应用程序返老还童，那么这真的是一个美丽新世界。对于那些被我策略地称为“不够优化”的代码来说，数据库应用系统的正确解决办法也能够创造奇迹。

修改应用系统的逻辑结构，似乎与前面所说的“保持较小变更”的目标相矛盾。实际上，你对于“小和增量意味着什么”的理解取决于你的阅历；当你第一次到一个没去过的地方时，路似乎总是很长，而当你再次回到这个地方，成为常客以致于熟悉这里时，路似乎就没那么长了。

我们可以期待从重构中得到什么

重要的是要理解，主要控制着重构效果的是下面这两个因素（实际上，它们是相互矛盾的因素）：

- 首先，重构的收益与最初的应用系统有着直接的关系：如果代码的质量很差，那么一次成功的改进就很有可能唾手可得。如果代码非常理想，那么可能就没有什么重构的机会 —— 引入新特性除外，这也可能就是故事结束的时候了。就像公司一样，只有管理糟糕的公司才能被神奇地扭转。
- 其次，如果数据库设计得真的很糟糕，那么重构也不会有太大效果。事情只能变得略微不那么糟糕，但决不会产生令人满意的结果。重构是一个渐进的过程。在数据库这一特定情形下，如果没有最初的明智设计，即便是明智的演变过程也无法使应用系统适于生存。这样的应用系统必将崩溃并绝迹。

像伟大的拉丁诗人贺拉斯那样，在把《aurea mediocritas》（即 golden mediocrity，中庸是金）写出来之前，先把它在脑海里重构一遍，是不可能的。但对于一些平凡的应用系统来说，我们确实可以对重构寄予最好的愿望。这样的应用系统有很多，因为正如本书的评论者Roy Owens说的那样“如果第一种每个人都同意的方式在功能上可以工作，那么它也就变成了设计”，这种情况太常见了。

本书的组织方式

本书尽量以一种现实和诚实的眼光来看待那些在SQL起重要作用的应用系统中的改进工作，并且详细描述了一个可用于战术演习的合理框架。往往是在疯狂追求“快速取胜”和“显著改进”的时候，才会进行重构，因为这样能防止预算被削减，能保住肩膀上的脑袋。严格地说，在普遍恐慌时，保持头脑冷静并采取系统化的方法才最有作用。让我们提前申明一下：奇迹，根据它的定义，是少数极有天赋的人的领地，它们通常适用于比你的应用系统更有价值的理由（无论你怎么理解这句话）。而理由充分地系统化应用一些合理原则反而更能取得引人注目的结果。本书将尽力帮你明确不同的策略，评估不同解决方案的可行性以及评估对于增量一词的不同解释所带来的风险。

在很多情况下，SQL应用系统的重构工作都会按照一种与开发相反的顺序来进行：从容易的事情开始，逐渐深入地切入，直到你到了一种会有所损害的程度或你已达到了自愿设置的限制时为止。在本书中，我也尽量遵照同样的顺序来组织内容。

第1章“评估”

这一章是开场白，它讲述的是如何对形势进行评估。重构通常都与需要仔细分配的稀缺资源有关。因此没有犯错和对错误的目标进行改进的余地。该章将指引你评估的是：1）是否有希望进行重构；2）你有理由抱有哪种希望。

接下来的两章将讨论每个经理的梦想：快速取胜。在这两章中，我讨论的是主要发生在数据库一侧的修改，而不是应用程序一侧的修改。有时，你甚至还可以把其中一些修改用于你没有权限访问其源代码的“罐头式的应用系统”中。

第2章“健全检查”

讨论那些必须按照优先级进行控制的要点——特别是对索引的评审。

第3章“用户函数和视图”

解释为何大量使用用户编写的函数和视图有时会使应用系统的性能大大降低，以及怎样能使它们对性能的影响最小化。

在接下来的三章中，我将讲述可以对应用系统做哪些修改。

第4章“测试框架”

讲解如何组建一个正确的测试框架。在修改代码时，要确保我们仍然能够得到相同的结果，这一点很关键，因为任何一处修改，无论它多么小，都可能引入缺陷；没有任何一处修改是完全无风险的。该章将讨论比较一个程序的前后两种版本的策略。

第5章“语句重构”

深入讨论编写各种SQL语句的正确方法。优化器会改写次优（即未达最佳标准的）语句，这是优化器应当做的事情。但是，最精明的优化器也只能尽量利用现有的情况。我将向你展示如何分析并改写SQL语句，从而将优化器变为你的朋友，而不是你的对手。

第6章“任务重构”

将第5章的讨论进一步推向深入，解释操作模式的修改——特别是去除以记录为基础的处理方式——如何能把我们带入一个更高的层次。通常，改写单个语句所产生的结果仅仅是潜在改进成果中很小的一部分。应该更大胆地向前走，比如将几个语句联合起来，或者用气势磅礴的SQL语句来替代循环的过程化的语句，这些做法常常会取得令人惊喜的收获。这些收获要求有很好的SQL技能，而SQL的理念与传统的过程化理念及面向对象的理念非常不同。我将介绍一些这样的例子。

到了这个阶段，如果你仍然对性能不满意，那么你最后的希望就在下一章中。

第7章“重构流程和数据库”

回到数据库上来，讨论一些更底层的修改。首先，我将讨论如何通过改变流程和引入并行来提升性能，展示在将处理并行化以后需要注意的一些新问题，比如数据一致性、竞争和加锁。然后，我将讨论最后一招：有时可以对数据库结构进行物理上和逻辑上的修改，从而获得一些额外的性能提升。

接下来是对本书的总结。

第8章“实践中的重构”

这一章是对全书的总结，提供了一个扩展的检查列表。在该章中，我将引用前面的章节来描述：当我必须处理数据库应用系统的性能问题时，我的脑海里想的是什么，我做的是什么。这对于我是个难题，因为经验（以及通过经验获取的直觉）所建议的快捷方法有时没有真正经过清晰思考并且合乎逻辑的分析，但我还是希望它能成为一个有用的参考。

附录A“脚本及样例程序”和附录B“工具”。

描述了脚本、样例程序和工具，这些都可以从O'Reilly上本书的网页中下载：*http://www.oreilly.com/catalog/9780596514976*。

读者对象

本书为IT专业人士、开发人员、项目经理、维护团队、数据库管理员及调优专家而写，他们可能会参与到对一个数据库组件占很重要作用的应用系统进行的拯救活动中。

阅读本书的前提

本书假设读者有很好的SQL语言知识，当然，至少熟悉一种编程语言会更便于理解本书。

本书排版约定

本书使用如下的排版约定：

斜体（*Italic*）

表明是需要强调的部分、新术语、URL、文件名和文件扩展名。

等宽字体（`Constant width`）

表明是广义的计算机代码，包括命令、选项、变量、属性、键、请求、函数、方法、类型、类、模块、对象属性、参数、值、对象、事件、事件处理程序、XML和XHTML标签、宏及关键字。它也会用来表示表名、字段名之类的标识符，还用来表示代码样本和命令输出结果。

等宽粗体（**`Constant width bold`**）

表明是代码样本中需要强调的部分。

等宽斜体（*`Constant width italic`*）

表示这部分文本应当被替换为用户提供的具体值。

源代码例子的使用

本书的目的是为了帮助你完成工作。总的来说，你可以在自己的程序和文档中使用本书中的这些源代码。你不需要联系我们以获得许可，除非你要复制这些代码中的很大一部分。例如，在编写一个程序时使用本书中的几段代码，不需要许可。销售或分发O'Reilly图书所带的例子CD-ROM，不需要许可。在回答问题时引用本书或引用例子代码，不需要许可。在你的产品文档中采用本书中很大一部分示例代码，需要许可。

我们感谢注明出处，但并不要求必须这样做。在注明出处时，通常应包括标题、作者、出版商和ISBN号。例如“*Refactoring SQL Applications* by Stéphane Faroult with Pascal L'Hermite. Copyright 2008 Stéphane Faroult and Pascal L'Hermite, 978-0-596-51497-6”。

如果你感觉自己对于代码例子的使用已超出了这里给出的许可范围，就请联系我们，邮件地址为*permissions@oreilly.com*。

如何联系我们

关于本书的评论和疑问都可以发送给出版社：

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街2号成铭大厦C座807室（100035）
奥莱利技术咨询（北京）有限公司

我们为本书建立了一个网页，上面列出了勘误表、例子和一些额外的信息。你可以用如下地址来访问它：

http://www.oreilly.com/catalog/9780596514976（英文版）
http://www.oreilly.com.cn/book.php?bn=978-7-111-26358-6（中文版）

评论本书或者提一些关于本书的技术问题，请发电子邮件至：

bookquestions@oreilly.com

在我们的网站上有更多的关于我们的图书、会议、资源中心和O'Reilly网络的信息，欢迎访问：

http://www.oreilly.com
http://www.oreilly.com.cn

致谢

一本书的写作是很多人工作的成果，不止是名字印在封面上的那些人。首先，我要感谢Pascal L'Hermite，他的Oracle和SQL Server的知识在我写本书时极有价值。在一本技

术书籍中，文字只是一切努力中可见的那一部分。设置测试环境，设计样例程序，将它们移植到各种数据库平台上，有时还要尝试一些最终不会有什么结果的想法，所有这些任务都要花很长时间。在本书的背后有很多看不见的工作，很多的工作在最终成书时都只是表现为一些看似随意的引用和模糊的影子。没有Pascal的帮助，本书可能要花更长的时间才能写完。

每个项目都需要经理人，而我的编辑Mary Treseler就是O'Reilly出版社的项目经理。Mary选择了一个非常好的评审团队，其中的几个人也是作家。首先要感谢的是Brand Hunt，他是本书的技术编辑。我衷心地感谢Brand，他帮助我对本书进行了最终的定型。还要感谢Dwayne King，特别感谢他修改本书的行文及代码。David Noor、Roy Owens和Michael Blaha也非常有帮助。我还要感谢两位专家，同时也是我的老朋友：Philippe Bertolino和Cyril Thankappan，他们也仔细地评审了本书的草稿。

除了纠正了一些错误之外，这些评审者都贡献了自己对于最终产品的评论和说明，这使本书变得更好。

在作者和评审者的工作完成之后，很多O'Reilly人的工作才开始：在制作编辑的领导下，他们对本书进行编辑加工，书籍设计，封面设计，将我那些难看的图表改成更符合O'Reilly标准的东西，为本书建立索引，所有这些都有助于本书的最后面世。我由衷地感谢Rachel Monaghan、Audrey Doyle、Mark Paglietti、Karen Montgomery、Marcia Friedman、Rob Romano和Lucie Haskins。