MySQL 8 查询性能优化

[澳] 杰斯帕 • 威斯堡 • 克罗(Jesper Wisborg Krogh) 著
史跃东 杨 欣 殷海英 译

清华大学出版社
北 京

北京市版权局著作权合同登记号　图字：01-2020-5561

图书在版编目(CIP)数据

MySQL 8查询性能优化 / (澳) 杰斯帕·威斯堡·克罗 (Jesper Wisborg Krogh) 著；史跃东，杨欣，殷海英译. —北京：清华大学出版社，2021.6

书名原文：MySQL 8 Query Performance Tuning：A Systematic Method for Improving Execution Speeds
ISBN 978-7-302-58391-2

I. ①M…　II. ①杰… ②史… ③杨… ④殷…　III. ①SQL语言－程序设计　IV. ①TP311.132.3

中国版本图书馆CIP数据核字(2021)第117382号

责任编辑：王　军
装帧设计：孔祥峰
责任校对：成凤进
责任印制：杨　艳

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：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　**印　　张**：36.25　　**字　　数**：999 千字
版　　次：2021 年 8 月第 1 版　　**印　　次**：2021 年 8 月第 1 次印刷
定　　价：158.00 元

产品编号：088758-01

译 者 序

历经数月，与好友兼同事 cindy、henry 通力合作，终于将《MySQL 8 查询性能优化》一书翻译完成。三人熟识已久，此次合作是愉悦的。

MySQL 在开源数据库领域的重要性毋庸置疑，在互联网行业，更是无处不在。作为一个从毕业至今一直与 Oracle 打交道的 IT 从业者，MySQL 的开放与灵活，以及在国内开源社区内的活跃度，令我颇感兴趣；我也曾阅读过多本 MySQL 相关书籍。此番有机会与好友们一起翻译这本基于 MySQL 8.0 的查询优化书籍，与业内诸位同仁一道，共同推动 MySQL 技术的应用与普及，也算是与有荣焉。

MySQL 从 5.7 到 8.0 版本，是一次极为重要的版本更新，尤其是在性能优化方面。本书从查询性能优化的入门知识谈起，层层深入，全面铺开，为广大读者详尽展示了 MySQL 8.0 中与 SQL 查询性能优化相关的方方面面。在注重实践的同时，兼顾内容的广度与深度，使得无论是初学者还是从业多年的老手，均可在查阅本书的过程中有所收益。

非常感谢清华出版社给予的这次机会，使得我们三人能参与到 MySQL 技术领域，为国内读者提供一本质量上乘的技术书籍。市面上的 MySQL 相关书籍已然众多，但能如此全面地阐述 SQL 查询优化相关知识的书籍，却少之又少。因此我们深信，本书具有独特价值，值得翻译出版。

也感谢 cindy 和 henry 二位的奉献和付出；正是二位的辛勤翻译，笔耕不辍，才使得本书能快速交稿并付梓出版。本书共 27 章，英文原版近千页，称得上是一项大翻译工程了。翻译期间，我们不断沟通，不断校对；cindy 和 henry 在百忙的工作之余，还要抽出晚间和周末的休息时间赶稿，也是不易。故在此一并谢过。他日再聚，定当把酒言欢，不醉不归。

史跃东

2021 年 6 月 21 日作于北京

作者简介

自 2006 年以来，Jesper Wisborg Krogh 先后以 SQL 开发人员和数据库管理员的身份参与到 MySQL 数据库工作中，并且作为 MySQL 技术支持团队的一员，工作了 8 年之久。他曾在 MySQL Connect 和 Oracle OpenWorld 上多次发表演讲。除了出版相关书籍外，他也会定期撰写一些以 MySQL 为主题的博客文章，并为 Oracle 知识库撰写了约 800 份文档。此外，Jesper Wisborg Krogh 也为 MySQL 中的 sys 库，以及 MySQL 5.6 等相关的 OCP 认证考试做出了许多贡献。

在 2006 年转向 MySQL 及软件开发之前，Jesper Wisborg Krogh 获得计算化学的博士学位。他现在居住在澳大利亚的悉尼，平时喜欢在户外散步、旅行以及阅读等。其研究领域涉及 MySQL 集群、MySQL Enterprise Backup(MEB)、性能优化，以及 performance 库和 sys 库等。

技术编辑简介

Charles Bell 喜欢专研新兴技术。他是 MySQL 开发团队的一员，也是 MySQL Enterprise Backup 团队的高级软件开发人员。目前，他携爱妻居住于弗吉尼亚州的一个小镇上。Charles Bell 于 2005 年从弗吉尼亚联邦大学获得了工程学博士学位。

Charles 是数据库领域的专家，在软件开发和系统工程方面有着丰富的知识和经验。同时他的研究兴趣还涉及 3D 打印、微型控制器、数据库系统、软件工程、高可用系统、云以及传感器网络等。在其有限的业余时间中，他经常专注于微型控制器项目，以及 3D 打印机的制作和改进。

致　　谢

我要感谢所有让本书的出版成为可能的人们。Apress 团队再次为我提供了极大帮助，尤其需要感谢与我合作的三位编辑：Jonathan Gennick、Jill Balzano 和 Laura Berendson。正是他们让本书的出版变成了现实。

在本书的撰写过程中，有几个人是我最宝贵的伙伴。感谢 Charles Bell 提供的相关探讨，他的意见一向都是那么有价值；Jakub Lopuszanski 关于 InnoDB 锁的反馈也让我受益良多。同时，我与 MySQL 技术支持团队的合作，团队内部无数次的讨论，以及同事们的那些伟大的工作，都为本书提供了极有价值的启发和思路来源。另外，我还要感谢 Edwin Desouza 所提供的大力支持。

最后，当然也很重要的一点是，我要特别感谢我的妻子 Ann-Margrete，在我撰写本书期间，她表现出了极大的耐心和支持。

前　言

MySQL 性能优化是一个非常大的主题，人们通常需要花费数年时间才能掌握。本书的篇幅就证明了这一点，即使只专注与查询相关的优化主题，篇幅显然就不小了。一般而言，没有什么简单方法可以轻松地提升性能，恰恰相反，要找到相关的解决方法，你不仅需要了解 MySQL 内部各部分之间的关系，还需要了解相关技术栈其他部分的内容。如果你觉得单单在性能优化方面就很难入门，那么第一步你就跨不过去。但是，请不要对性能优化感到失望，与其他技巧一样，也可以通过实践逐步成为性能优化高手。

撰写本书的目的在于，使你能在 MySQL 性能优化方面登堂入室，从而熟练掌握如何提升在 MySQL 实例上运行的那些查询的性能。如前所述，这没有什么简单秘方，最佳办法就是学习并了解性能优化过程中涉及的各个组件。这也是本书的主要内容，当然，我们也提供了如何找到相关信息，以及如何执行一些常见任务的示例。另外，本书的内容仅限于对 MySQL 本身的探讨，因此关于操作系统、文件系统以及硬件级别的内容，就相对有限了。

众所周知，MySQL 以对各种存储引擎的支持而闻名。但除了对内部临时表相关的探讨外，本书只介绍 InnoDB 存储引擎。而对于 MySQL 的版本，则只考虑 MySQL 8。也就是说，本书中的大部分讨论内容虽然也适用于旧版本的 MySQL，但通常也只是为了说明 MySQL 8 中的新特性与旧版本的不同之处罢了。

本书面向的读者

本书是为那些具有丰富的 MySQL 数据库使用经验，并希望将知识扩展到查询性能优化领域的开发人员和数据库管理员而编写的。当然，在阅读本书之前，你不需要具备性能优化的相关经验。

在撰写本书的过程中，作者尝试添加了尽可能多的示例代码及其输出结果。当然，有些示例很短，有些则很长。但无论哪种情况，作者都希望读者能够跟上并重现这些示例的结果。同时请记住，由于实际环境的差异(当然，这种差异和索引统计信息一样明确)，示例结果可能会取决于在示例之前，相关的表和数据的获取方式。换句话说，即使读者完成了所有工作，得到的结果仍然可能与本书中的结果不同。尤其是涉及索引统计信息以及与计时等相关的数字时。

读者可扫封底的二维码，下载本书的示例代码。

本书结构

本书分为 6 部分，共计 27 章。在撰写本书时，作者试图让每章的内容都保持相对独立，以便读者将本书用作参考书。当然，这样做的缺点之一是有时会重复出现某些内容。例如第 18 章介绍了锁的理论方面的一些知识，以及如何对锁进行监控；而第 22 章则提供锁争用的一些示例。因此，第 22 章很自然会借鉴第 18 章中的部分信息，因此出现了一些内容上的重复。这是一个有意识的行为，作者希望各位读者在阅读本书的过程中可以减少翻页的次数，尽快找到所需内容。

在阅读过程中，本书的 6 部分将引导你逐步完成性能优化主题的相关探讨。我们先从一些基本的背景知识开始，然后给出面向问题的解决方案。第 I 部分将探讨相关的方法论、基准以及测试数据。第 II 部分重点介绍各种信息来源，如 performance 库等。第III部分介绍本书将用到的各种工具，如 MySQL shell。第IV部分则提供后面两部分将用到的理论知识。第 V 部分侧重分析查询、事务以及锁。第VI部分则探讨如何通过配置、查询优化、复制以及缓存等技术来提升性能。某些情况下，有些内容的编排可能较特殊，例如，所有关于复制的内容都包含在单独一章中(即第 26 章)。

第 I 部分　入门

第 I 部分介绍 MySQL 查询性能优化的相关概念，包括一些高级注意事项等。其中一些并非 MySQL 所独有(不过也是在 MySQL 上下文中进行探讨的)。第 I 部分包含 4 章。

第 1 章“MySQL 性能优化”—— 该章涵盖 MySQL 性能优化的一些高级概念，例如考虑整个堆栈和查询生命周期的重要性等。

第 2 章“查询优化方法论”—— 以有效方式解决性能问题至关重要。该章介绍有效工作的方法论，并强调积极工作的重要性。

第 3 章“使用 Sysbench 进行基准测试”—— 通常，我们需要使用基准测试来检验更改效果。该章将简要介绍基准测试，并专门探讨 Sysbench 工具，也包括如何创建自定义基准测试等内容。

第 4 章“测试数据”—— 列 出 本书主要使用的一些标准测试数据库。

第 II 部分　信息来源

MySQL 会通过一些信息来源提供有关性能的信息。在该部分，将介绍 performance 库、sys

库、information 库以及 SHOW 语句。虽然在该部分中使用这些信息来源的例子相对较少，但在本书其他部分，则广泛使用了这四个信息来源。如果你对这些信息来源还不太熟悉，我们强烈建议你详细阅读该部分中的各个章节。此外，该部分还包含慢查询日志的相关内容。第Ⅱ部分共包含 5 章。

第 5 章“performance 库”—— 顾名思义，MySQL 中与性能相关的信息的主要来源是 performance 库。该章介绍相关的术语、基本概念、组织方式以及配置信息。

第 6 章“sys 库”——通过存储过程和函数中的预定义视图和工具，sys 库提供了各种报告信息。该章对 sys 库各种可用的特性进行了概述。

第 7 章“information 库”—— 如果想获得关于 MySQL 和数据库的元数据信息，就需要查看 information 库。该库还包含用于性能优化的重要信息，例如关于索引、索引统计以及直方图等信息。该章将概述 sys 库中可用的视图。

第 8 章“SHOW 语句”—— 这是获取信息的最古老方法，可通过它获得从执行查询到库的各级别信息。该章将 SHOW 语句与 information 库和 performance 库相关联，并在某种程度上更详细地介绍 SHOW 语句。而 SHOW 语句的某些内容，在 information 库和 performance 库中是没有对应内容的。

第 9 章“慢查询日志”—— 查找慢查询的传统方法，就是将其记录到慢查询日志中。该章将介绍如何配置慢查询日志，如何读取日志事件，以及如何使用 mysqldump 这一实用工具对事件进行聚合等。

第Ⅲ部分　工具

MySQL提供一些在执行日常任务以及特定任务时非常有用的工具。该部分涵盖与监控和简单查询执行相关的三种工具。本书将 Oracle 专用的 MySQL 监控解决方案作为监控示例。即使你当前正在使用其他监控解决方案，也建议你研究一下这个示例。因为这些不同的解决方案之间往往有一些重叠之处。此外，本书其余部分也广泛使用了这三种工具。该部分包含 3 章。

第 10 章“MySQL Enterprise Monitor”—— 监控是保证数据库稳定运行且性能良好的最重要内容之一。该章将介绍 MySQL Enterprise Monitor(MEM)，并说明如何安装试用版本，以及如何进行导航和使用图形化用户界面等。

第 11 章“MySQL Workbench”—— MySQL 通过 MySQL Workbench 为用户提供图形化界面。该章将介绍如何安装和使用这一工具。在本书中，MySQL Workbench 对创建查询执行计划的可视化图形(称为 Visual Explain，可视化解释)至关重要。

第 12 章“MySQL shell”—— Oracle 为 MySQL 推出的最新工具之一就是 MySQL shell。它是第二代的命令行客户端，支持在 SQL、Python 以及 JavaScript 模式下执行代码。该章将使你快速了解这一工具，并介绍该工具对外部代码模块的支持，分析该工具的报告基础结构，以及如何创建自定义模块、报告和插件等。

第Ⅳ部分　方案考量与查询优化器

在该部分中，介绍内容的节奏稍有改变。重点从与方案、查询优化器和锁相关的主题，逐步转移到性能优化的主题上。该部分包含 6 章。

第 13 章“数据类型”—— 在关系数据库中，每列都有各自的数据类型。数据类型定义了各列能够存储的值，两个值进行比较时所遵循的规则，以及数据的存储方式等。该章介绍 MySQL 中可用的各种数据类型，并提供应该使用何种数据类型的指导信息。

第 14 章“索引”—— 索引用于查找数据。而良好的索引可以极大地提升查询的性能。该章介绍索引的概念、关于索引的注意事项、索引类型以及索引的特性等。此外，介绍了 InnoDB 如何使用索引，以及应该使用何种索引策略等。

第 15 章“索引统计信息”—— 当优化器需要确定索引的有用程度以及与索引值上的条件相匹配的行数时，就需要关于索引中的数据的相关信息，即索引统计信息。该章介绍索引统计信息在 MySQL 中的工作方式，以及如何对其进行配置、监控和更新。

第 16 章“直方图”—— 如果希望优化器知道给定列中值出现的频率，则需要创建直方图。这是 MySQL 8 中新添加的特性。该章将介绍如何使用直方图，其内部结构如何，以及如何查询直方图的元数据和统计信息等。

第 17 章“查询优化器”——在执行查询时，优化器会决定查询的执行方式。该章将介绍优化器要完成的任务、使用的联接算法、联接优化、优化器相关配置以及资源组等内容。

第 18 章“锁原理与监控”—— 最容易给人带来挫败感的问题之一就是锁争用。该章首先说明数据库为何需要锁、锁的访问级别以及锁的类型(粒度)；然后介绍在无法获得锁时，如何减少锁争用，以及可在何处找到锁相关的信息。

第Ⅴ部分　查询分析

有了第Ⅳ部分的信息，现在就可对查询进行分析了。这包括如何查找查询，从而进行下一步的分析，然后使用 EXPLAIN 或 performance 库分析等。当你有两个或者两个以上的查询来争用相同的锁时，还需要考虑事务是如何工作的，并对锁争用问题进行调查。该部分包含 4 章。

第 19 章“查找待优化的查询”—— 无论是将其作为日常运维的一部分，还是在紧急状况下，你都需要查找那些需要分析和优化的查询。该章将介绍如何使用 performance 库、sys 库、MySQL Workbench、监控解决方案以及慢查询日志等工具来查找那些值得研究的查询。

第 20 章“分析查询”—— 有了候选查询后，就需要分析为何执行速度慢或给系统造成如此大的影响。这里用到的主要工具是 EXPLAIN，该命令可提供有关优化器选择的查询计划的相关信息。如何使用 EXPLAIN 生成和理解查询(包括示例)计划是该章的重点。也可使用优化器跟踪程序来获取有关优化器如何选择某一查询的更多信息。分析查询的另一种方法是使用 performance 库和 sys 库将查询分解为较小的部分。

第 21 章“事务 ”—— InnoDB 将所有执行操作视作事务，并且事务是一个极为重要的概念。正确使用事务可以确保原子性、一致性和隔离性。但是，事务也可能导致严重的性能和锁争用问题。该章探讨事务为何成为问题，以及如何分析事务。

第 22 章“诊断锁争用”—— 该章介绍了四种锁争用的场景(刷新锁、元数据锁、记录锁以及死锁)，并探讨这四种锁的症状、形成原因、重建场景的方式、调查、解决方案以及预防方式等。

第Ⅵ部分　提升查询性能

现在，你已经找到有问题的查询，并对其进行分析，包括事务情况，以了解表现不佳的原因。但是，你应该如何改善查询呢？该部分介绍其他章节未涵盖的重要配置选项，以及如何改变查询计划、改变方案等，同时包含批量加载、复制以及缓存等相关知识。该部分包含 5 章。

第 23 章“配置”——MySQL 在执行查询时需要系统资源。该章将介绍配置这些资源的最佳实践，以及其他讨论中未曾介绍的一些重要配置选项。同时对 InnoDB 中的数据生命周期做了概述，将其作为探讨 InnoDB 配置的背景知识。

第 24 章“改变查询计划”——尽管优化器通常在找到最佳查询执行计划方面表现不错，但你

依然需要不时为其提供帮助。例如，可能是因为没有索引，或现有索引无法使用等，导致全表扫描的出现。你可能还想提高索引的使用率，或者可能需要重写某个复杂条件甚至整个查询等。该章将仔细介绍这些内容，同时说明了如何使用 SKIP LOCKED 子句来实现队列系统。

第 25 章“DDL 与批量数据加载”——在执行方案更改，或将大数据集加载到系统中时，MySQL 需要完成大量工作。该章将探讨如何提升这类任务的性能，包括使用 MySQL shell 的并行数据加载特性等。该章也包含关于数据加载注意事项的内容，这通常也适用于数据修改操作。此外，我们也显示了顺序插入和随机插入之间的不同之处。在完成此探讨后，我们就知道选择主键对性能来说意味着什么。

第 26 章“复制”——在实例之间进行数据复制是 MySQL 的一项颇为流行的特性。从性能的角度看，复制包含两方面的内容：首先，你需要确保复制操作性能良好，此外，可通过复制来提升系统性能。该章探讨这两方面的内容，介绍用于监控复制的 performance 库表。

第 27 章“缓存”——提升查询性能的终极方法是根本不执行查询，或至少避免执行某部分查询。该章将探讨如何使用缓存表来降低查询的复杂性，以及如何使用 Memcached、MySQL InnoDB Memcached 插件和 ProxySQL 来避免查询的完全执行。

下载示例代码和彩色图片

本书正文中，有些地方会提到图中标记、箭头或区域的颜色。本书是黑白印刷，无法显示彩色。读者可在实际界面中查看确切的颜色。另外，可扫描封底二维码，下载示例代码和彩色图片。

目　录

第Ⅲ部分 工 具

第Ⅳ部分 方案考量与查询优化器

第 I 部分

入　门

第 1 章

MySQL 性能优化

欢迎来到 MySQL 性能优化的世界！这个世界，有时看起来似乎是被黑魔法或运气所支配。但本书期望能帮助你以结构化方式进行工作，而且最终你能有条理地完成任务，从而获得更好的性能表现。

本章通过探讨与 MySQL 相关的方方面面，以及基于数据执行某些操作的重要性，来介绍 MySQL 性能优化。由于本书的内容主要是关于查询的，因此应先回顾一下查询的生命周期。

提示 如果你需要一个测试实例，以便在阅读本书或解决工作中遇到的问题时使用，那么云端环境可能是一个很好的选择。例如，如果你只需要一个很小的实例来处理本书中提到的示例，你甚至可使用网上的免费实例，例如，通过 Oracle Cloud 申请一个(当然，你仍然需要注册并使用信用卡信息，只是它不会真正扣钱罢了): https://mysql.wisborg.dk/oracle_cloude_ free_tier。

1.1 通盘考虑

在处理性能问题时，重要的是需要考虑系统涉及的所有部分，例如从最终用户到应用，再到 MySQL。当有人向你报告说应用运行缓慢，而你又知道 MySQL 是应用的核心部分时，就很容易

得到“MySQL 运行缓慢”这样的结论。但是，这有可能将那些真正造成性能不佳的潜在因素排除在外。

当应用程序需要查询结果，或需要在 MySQL 中存储数据时，将通过网络把请求发送给 MySQL；为完成这一请求，MySQL 将与操作系统进行交互，并使用主机资源(如内存和磁盘)。待请求的结果准备就绪后，再通过网络将其传输给应用程序，如图 1-1 所示。

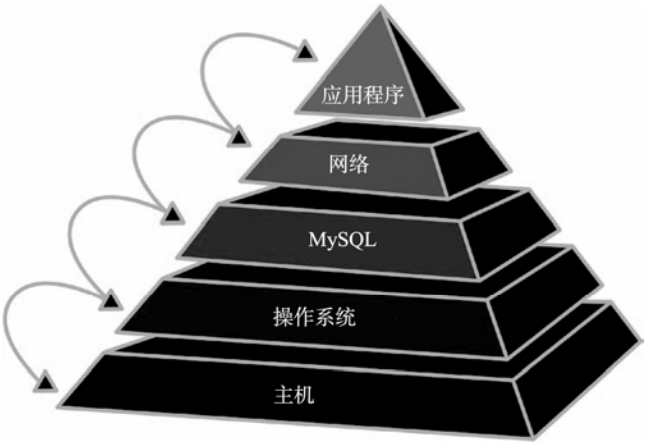

图 1-1　围绕 MySQL 的技术栈

当然，这里的金字塔图片其实已经非常简化了，它将应用程序之上的各种内容都省略了。应用程序将与用户通信，并使用自己的资源。而在通过网络进行通信时，则又涉及主机和操作系统。

为说明上述各层之间是如何相互作用的，请考虑一个实例。一位 MySQL 用户报告，MySQL 遇到了临时停顿的问题。在 Linux 上使用 perf 工具进行检查发现，发生停顿的原因是内存变得非常零散，这主要是由 I/O 缓存引起的。当你通过网络提交数据时，Linux 会请求一块连续内存(使用 kmalloc)。但由于内存碎片化过重，Linux 需要先对内存进行碎片整理(压缩)。虽然进行了压缩处理，但此时包括 MySQL 在内的所有程序都停滞了。在最糟糕的情况下，系统可能需要花费 1 分钟(服务器具有大量可用于 I/O 缓存的内存)的时间完成压缩，因此造成严重影响。这种情况下，可更改 MySQL 的配置，从而使用直接 I/O 来解决问题。虽然这是一个比较极端的情况，但值得注意的是，各层之间的交互可能带来令人意外的阻塞和瓶颈问题。

另一个更直接的案例，则是使用框架来生成查询语句的应用程序。该框架存在一个 bug，即对大表进行查询时，会自动省略 WHERE 子句。这很容易导致一系列问题，例如应用程序的查询重试。系统在数秒钟内就完成 50 次查询操作(由于数据最终已被读入缓冲池，因此最后一次查询的速度要比第一次快得多)，从而最终导致问题出现：MySQL 将大量数据发回给应用程序，导致网络超载，并且应用程序的内存也出现不足。

本书侧重于 MySQL，以及影响查询的各个方面，但依然不要忘记系统的其他部分。当然，这其中就包括对系统的监控。

1.2　监控

监控对于保持系统的正常运行至关重要。你所做的一切事情，都应该围绕监控进行。在某些情况下，通过专门的监控解决方案，可为你提供解决问题需要的全部数据。而在其他情况下，则

可能还需要一些临时观察。

你的监控应该使用多个信息来源。包括但不限于：

- **performance 库。**包含从低级的互斥量(mutex)到查询和事务度量的各种信息。这是用于查询性能优化的最重要的信息来源。sys 库则提供一个方便的界面，尤其是用于即席查询时。
- **information 库。**包含 schema 信息、InnoDB 存储引擎统计信息等。
- **SHOW 语句。**包含诸如 InnoDB 详细统计信息的内容。
- **慢查询日志。**用于记录满足某些条件(如执行时间超过预定义阈值)的查询。
- **EXPLAIN 语句。**返回查询的执行计划。这是一个无价之宝，可用于调查为什么由于缺少索引，查询只能以次优方式执行；或者用于确认 MySQL 选择次优方式是导致查询性能不佳的原因。EXPLAIN 语句在处理特定查询时，通常以临时方式使用。
- **操作系统指标。**如磁盘使用率、内存使用率以及网络使用率等。不要忘记那些简单指标(如可用存储空间等)，因为存储空间不足也会导致中断。

上述信息来源都将在本书中进行探讨及使用。

在整个性能优化的过程中，使用监控，可以验证问题所在，找到原因，或证明你已解决问题。当然，在研究解决方案时，了解一下查询的生命周期也很有价值。

1.3 查询的生命周期

当执行查询时，往往经过几个处理步骤，然后查询结果才能返回给应用程序或客户端。其中的每个步骤都需要时间进行处理，并且有些步骤本身就可能包含一个或多个复杂的子处理过程。

查询的生命周期的简化图如图 1-2 所示。当然在实际中，则涉及更多步骤。如果你安装了诸如查询重写器(query rewriter)的插件，则可能添加自己的处理步骤。不过，本图确实已经包含基本处理步骤，稍后将更详细地介绍其中的几个步骤。

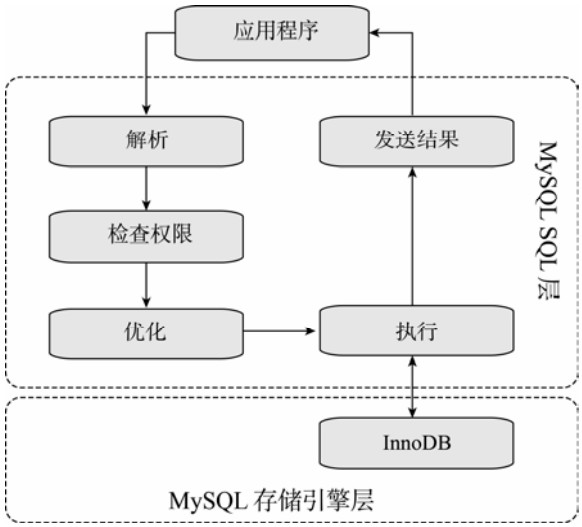

图 1-2 基本的查询生命周期

MySQL Server 可以分为两层。上面是 SQL 层，用于处理诸如用户连接，以及准备要执行的语句等操作。实际数据则由下面的存储引擎层负责存储。存储引擎以插件形式予以实现，这就使

得在选择不同的存储引擎或数据处理方式时相对容易一些。InnoDB 是主要的存储引擎(当然也是本书唯一考虑的存储引擎)，它是完全事务性的，对高并发的工作负载也具有很好的支持效果。其他存储引擎还有 NDB Cluster，也是事务性的，可作为 MySQL NDB Cluster 的一部分。

当应用程序需要执行查询时，第一件事是创建一个连接(图 1-2 中并未包括连接部分，因为连接可能会被重用，以执行多次查询)。查询到达时，MySQL 将对其进行解析。其中包括将查询拆分为令牌(token)，因此查询类型就是已知的，并且存在查询所需的表和列的列表信息。在下一步中，将需要此列表信息；这里将检查用户是否具有执行查询所需的权限。

此时，查询已经到达确定如何执行查询的重要步骤。这就是优化器的工作了，其工作内容涉及查询重写、确定表的访问顺序以及要使用哪些索引等。

实际执行过程还涉及从存储引擎层请求数据。存储引擎本身可能会很复杂。对于 InnoDB 来说，它包括缓冲池(用于缓存数据和索引)、重做日志及回滚日志，还包括其他缓冲区以及表空间文件。如果查询返回行，则这些行将从存储引擎通过 SQL 层发给应用程序。

在查询优化过程中，最重要的就是优化器、执行步骤以及存储引擎。本书大部分内容都将直接或间接地涉及这三部分。

1.4　本章小结

本章从头开始介绍性能优化相关内容，从而为你完成本书的其余内容做好准备。关键是你要通盘考虑从最终用户到主机和操作系统等低级层面的各个因素。而监控又是性能优化中的绝对必要条件。执行查询包含多个步骤，其中的优化器、执行步骤是你最需要关注的内容。

第 2 章将详细介绍对解决性能问题非常有用的一些方法。

第 2 章

查询优化方法论

解决问题的方法通常不止一种。在某些极端情况下，你可能会比较冒失地尝试一些更改，尽管这似乎能节约时间，但往往会带来挫败感。所做的更改似乎奏效了，但依然无法确定是真正解决了问题，还是情况只是暂时好了一点。

相反，我们推荐你进行分析，并使用监控来确认更改效果。也就是说，你应该更合理地开展工作。本章将介绍这样一种方法，该方法在解决 MySQL 问题时通常非常有用，尤其是在性能优化方面。首先，我们将介绍该方法中涉及的步骤，然后详细探讨每个步骤，分析为何要花费尽可能多的时间来主动工作(也是一件很重要的事情)。

注意 这里描述的其实是 Oracle 支持服务中用于解决客户报告的问题的方法。

2.1 综述

MySQL性能优化可被视为一个永无止境的过程。在此过程中，我们使用迭代方法来逐步改善性能。显然，有时可能出现一些特定问题，例如查询需要消耗半小时才能完成等。但请记住，性能并不是二进制状态，这一点很重要，因此你有必要知晓何为足够好的性能。否则，哪怕是一个

简单的优化任务，你也可能永远无法完成。

图 2-1 显示了一个示例，用于描述性能优化的生命周期。该循环从左上角开始，一共包含四个阶段。其中第一个是核实问题。

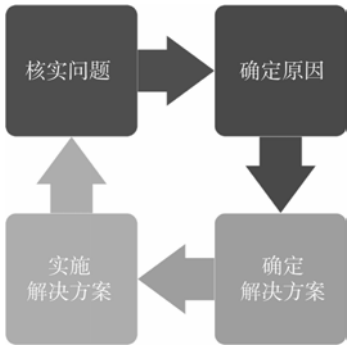

图 2-1　性能优化的生命周期

在遇到性能问题时，首先需要核实问题之所在，这包括收集问题相关的证据，并确定在考虑解决问题时有何需求。

第二阶段涉及确定造成性能问题的原因，第三阶段则确定解决方案。最后，第四阶段实施解决方案，其中应该包含检验更改操作的效果。

提示　*无论是在出现问题时解决问题，还是在主动工作期间，该循环均有效。*

然后，可以从头开始，或者进行第二次迭代，以进一步改善刚遇到的性能问题，或者你可能需要处理下一个性能问题。当然，两次循环之间也可能间隔很长时间。

2.2　核实问题

在尝试确定导致问题的原因和解决方案之前，重要的是，你要弄清楚应该解决的问题。只是说“MySQL慢”是远远不够的。这是什么意思？一个准确的问题描述可以是“该网页中第二部分使用的查询需要 5 秒才能完成”，或者“MySQL 每秒只能处理 5000 个事务”。问题描述越具体，解决问题的机会就越大。

最初看起来是什么问题，可能与真正的问题之间有所差别。核实问题可能很简单，例如执行查询，然后观察该查询是否真的花了那么长时间。也可能需要去查看监控。

本阶段的准备工作还应该包含从监控中收集基准数据，或收集能够说明问题的数据。没有基准数据，你就无法在故障排除之后，证明自己真正解决了问题。

最后，你需要确定性能优化的目标是什么。在这里，我们可以引用 Stephen R. Covey 所著的 *The 7 Habits of Highly Effective People* 中的一句话作为参考：

以始为终。

例如，慢查询语句所能接受的最低目标是什么？系统所需的最小事务吞吐量是多少？这样，就可以确保在更改之后是否已完成优化目标。

在明确定义并核实问题后，就可以开始分析问题并确定原因了。

2.3 确定原因

第二阶段是确定性能低下的原因。在此，你需要进行通盘考虑。这样就不会因为只盯着某个毫无关系的因素，而忽视了其他重要因素。

当你认为自己已经知道原因后，依然需要探讨一下为何会是这个原因。你可能会在 EXPLAIN 语句的输出结果中清楚地看到，查询使用了全表扫描，因此这可能就是问题发生的原因。或者有一张图显示 InnoDB 重做日志使用率已经超过 75%，所以你推测可能进行了异步刷新，从而导致临时的性能问题。

查找问题原因往往是最难的部分。而一旦找到原因，就可以确定解决方案了。

2.4 确定解决方案

确定要处理的问题的解决方案，往往需要两个步骤。第一步是寻找可能的解决方案；第二步需要选择实施哪个方案。

在寻找可能的解决方案时，进行头脑风暴，然后写下能想到的所有想法，这可能会很有用。重要的是，不要将自己局限在与问题产生的根本原因相关的思考范围之内。实际上，你经常可以在其他很多地方找到解决方案。其中一个例子是上一章提到的内存碎片导致的系统停顿问题，解决方案是更改 MySQL 的配置，使用直接 I/O，进而减少操作系统对 I/O 缓存的使用。此外，还需要考虑临时解决方案和长期解决方案，例如可能需要重启 MySQL 实例，或者需要执行升级、更换硬件等操作，这可能就不是能够立即实施的解决方案了。

提示 有时，一些不会被重视的解决方案，可能是需要升级 MySQL 或操作系统，以便使用新特性来解决问题。不过在这种情况下，就需要进行非常仔细、全面的测试，从而确保应用程序可与这些新特性很好地兼容。尤其是要注意优化器是否进行了调整，这也可能导致查询性能出现下降。

确定解决方案的第二部分内容是选择效果最佳的候选解决方案。为做到这一点，你需要探讨每个解决方案为何能起作用，以及它们的优缺点各是什么。在这一步中，你需要对自己足够诚实，因为必须仔细考虑解决方案可能带来的副作用。

一旦你对所有可能的解决方案都有了很好的了解，就可选择其中一种来解决问题了。在使用更可靠的解决方案时，也可选择另一种作为临时缓解措施。当然，无论是哪一种情况，下一步都是要实施解决方案了。

2.5 实施解决方案

可通过一系列步骤来实施解决方案。这其中包含定义行动计划、测试行动计划以及完善行动计划等，直到最终将解决方案应用到生产系统为止。重要的不是急于执行，因为这里可能是发现解决方案是否存在问题的最后机会。某些情况下，测试可能表明你需要放弃已经选定的解决方案，然后返回上一阶段并选择其他解决方案。图 2-2 说明了实施解决方案的流程。

你需要使用选定的解决方案，然后为其创建一个行动计划。在这里，非常重要的一点是，该行动计划一定要非常具体。这样就能确保在测试系统上执行的行动计划，也就是在接下来在生产系统上要执行的行动计划。写下你将要使用的确切命令和语句将很有用，这样就可以直接复制粘

贴，或者将其整理到脚本中以便自动执行。

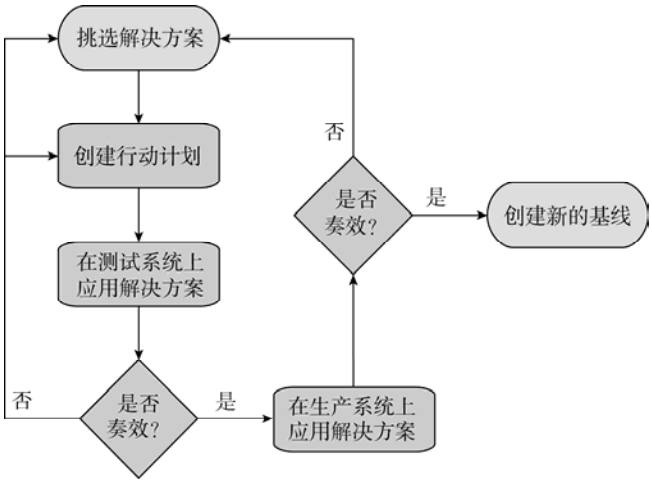

图 2-2　实施解决方案的工作流程

然后，你需要在测试系统上测试执行计划。重要的是，此时要尽可能反映在实际生产系统上执行的情况。因此，你在测试系统上使用的数据，必须能代表生产数据。可以复制生产数据，并在此过程中对敏感数据(如个人详细信息以及信用卡信息等)进行屏蔽处理。

提示　可以订阅(付费订阅)MySQL 企业版，其中包含数据屏蔽功能：www.mysql.com/products/enterprise/masking.html。

应该进行测试，以确保解决方案已经解决问题，并且没有带来意外的副作用。要执行什么样的测试，取决于你要解决的问题以及建议的解决方案。如果查询速度慢，则需要在实施解决方案之后测试查询的性能。如果修改了一个或者多个表上的索引，则还需要验证是否影响其他查询。在实施解决方案后，可能还需要对系统进行基准测试。当然，在所有情况下，你都需要将其与核实问题期间收集的基准数据进行比较。

当然，有时第一次尝试可能无法按照预想的方式进行。一般情况下，你需要对行动计划进行一些微调。有时可能要放弃所选的解决方案，然后返回上一阶段并选择其他解决方案。如果提出的解决方案解决了部分问题，也可将其应用到生产系统上，然后重新进行评估，并继续改善性能。

如果你对测试结果感到满意，就可将该解决方案实施到登台(stage)系统上。如果一切还很不错，则可将其应用到生产系统上。完成此操作后，你需要再次验证其是否有效。当然，无论你是多么谨慎地设置测试系统，以便让其尽可能接近生产系统，但由于种种原因，解决方案也依然可能无法按照设想的方式运行。本书作者遇到的一种可能情形是索引的统计信息不同。因此在将解决方案应用到生产系统上时，需要使用 ANALYZE TABLE 语句来更新索引的统计信息。

如果该解决方案有效，则应该收集新的基线，以便用于将来的监控和优化。如果该解决方案不起作用，则需要通过回滚并寻找新的解决方案，或者是进行新一轮的故障排除，确定该解决方案为何不起作用，并应用其他解决方案来确定如何继续。

2.6 主动工作

我们知道，性能优化是一个永无止境的过程。如果你的系统从根本上来说是健康的，那么大部分工作其实是在预防紧急情况，是在紧急程度较低的地方开展的。当然，这不会为你的工作带来太多关注，但这会减轻你的日常工作压力，客户也会更开心。

注意 这里的讨论，某种程度上是基于 Stephen R. Covey 所著的 *The 7 Habits of Highly Effective People* 中的第三个习惯：要事第一。

图 2-3 显示了如何按照紧急程度和重要性对任务进行分类。紧急任务通常会引起他人的注意，而其他任务可能也很重要，但通常是在未及时完成时才会引起注意，因此会忽然变得很紧急。

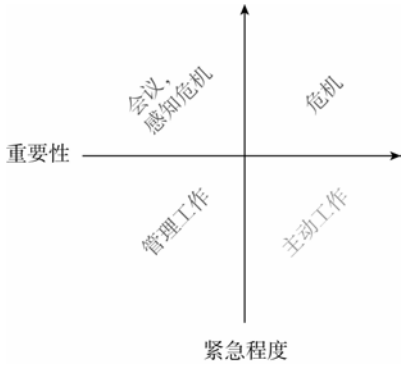

图 2-3 按照紧急程度和重要性对任务进行分类

最容易分类的任务通常是与危机相关的任务。例如生产系统故障或公司收入受损，因为客户无法使用产品或无法购买产品。这些任务既紧急又重要。在这些任务上花费大量时间可能会让你感到自己非常重要，但这也是一种紧张的工作方式。

解决性能问题的最有效方法是处理重要但不紧急的任务。这是预防危机发生的主动性工作，包括监控，以及在问题变得严重之前就进行改进等。此类别中的另一项重要任务就是“准备”。因此你也就会为危机做好准备。例如，这可能是需要建立一个备用系统，从而在发生危机时进行故障转移，或者快速启动替换实例等。这可帮助减少危机的持续时间，并使危机重新回到重要但不紧急的分类中。一般情况下，你在此类任务上耗费的时间越多，往往就越成功。

最后两类包括不太重要的任务。紧急但不重要的任务可能包括无法重新安排的会议、其他人推动的任务，以及可感知的(但不是真实的)危机。不紧急也不重要的任务则包括管理任务及检查电子邮件等。当然，其中的一些任务可能对于保持你的工作来说必不可少，但是对于保持 MySQL 的良好性能则并不重要。尽管你总是需要耗费时间来处理这些类别中的任务，但重要的是，尽量减少耗费在这些任务上的时间。

通过定义什么样的性能才是足够好的性能，就能避免对查询或者吞吐量进行过度优化。在实践中，如果不重要的任务引起了组织中其他人员的注意，就很难撤回这些任务了(因为这些任务通常是紧急任务)。但重要的是，你应该尽可能尝试将工作重心转移到处理那些重要而不紧急的任务上，以免日后处理那些危机。

2.7　本章小结

本章探讨了可用于解决 MySQL 性能问题的方法论，以及主动工作的重要性。

在报告问题后，就可以核实问题并确定已解决的问题。对于那些天生具有开放性的问题，重要的是要知道什么程度是足够好的，否则你将冒着永无休止地执行危机管理且无法回到主动工作的风险。

一旦有了清晰的问题描述，就可以确定问题发生的原因。待原因明确之后，就可以确定要解决的问题。最后一个阶段是实施解决方案，如果事实证明此前选择的解决方案不起作用，或会带来无法接受的副作用，就需要考虑替代方案。在这方面，重要的是，你需要尽可能紧贴实际来测试解决方案。

本章最后探讨要花费尽可能多的时间来主动工作的重要性，这些工作可以尽量防止危机的发生，并在危机真正发生时帮助你做好准备。这将帮助你减轻工作压力，并以更好的状态管理数据库。

如本章所述，在将解决方案部署到生产环境之前，测试其影响非常重要。因此第 3 章将介绍基准测试，尤其是 Sysbench 基准测试。