

需要整本电子书，联系我QQ: [2667271557](#);
此处是样章，取的完整版的前面几页，和最后
面几页；完整版是带书签的，样章没带书签；
另外需要其他书，也可以找我。

Making Sense of NoSQL

解读NoSQL

[美] Dan McCreary Ann Kelly 著

范东来 滕雨瞳 译



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS

Making Sense of NoSQL

解读NoSQL

[美] Dan McCreary Ann Kelly 著

范东来 滕雨瞳 译

人民邮电出版社

北京

图书在版编目 (CIP) 数据

解读NoSQL / (美) 麦克雷 (McCreary, D.), (美) 凯利 (Kelly, A.) 著; 范东来, 滕雨瞳译. — 北京: 人民邮电出版社, 2016. 2

书名原文: Making Sense of NoSQL

ISBN 978-7-115-41110-5

I. ①解… II. ①麦… ②凯… ③范… ④滕… III. ①数据库系统 IV. ①TP311.138

中国版本图书馆CIP数据核字(2015)第302980号

版权声明

Original English language edition, entitled *Making Sense of NoSQL* by Dan McCreary and Ann Kelly published by Manning Publications Co., 209 Bruce Park Avenue, Greenwich, CT 06830. Copyright ©2014 by Manning Publications Co.

Simplified Chinese-language edition copyright ©2016 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由 **Manning Publications Co.** 授权人民邮电出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

版权所有, 侵权必究。

-
- ◆ 著 [美] Dan McCreary Ann Kelly
 - 译 范东来 滕雨瞳
 - 责任编辑 杨海玲
 - 责任印制 张佳莹 焦志炜
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
 - 邮编 100164 电子邮件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 北京艺辉印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 16.25
 - 字数: 346 千字 2016 年 2 月第 1 版
 - 印数: 1—3 000 册 2016 年 2 月北京第 1 次印刷
 - 著作权合同登记号 图字: 01-2013-8988 号
-

定价: 49.00 元

读者服务热线: (010) 81055410 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

内容提要

本书从 NoSQL 的相关理论开始，深入浅出地探讨了 NoSQL 最核心的架构模式、解决方案和一些高级主题，内容循序渐进，从理论回归于实践。

全书分为 4 个部分。第一部分介绍 NoSQL 的相关理论，如 CAP 理论、BASE 理论、一致性散列算法等；第二部分介绍 NoSQL 最核心的架构模式——键值存储、图存储、列族存储、文档存储；第三部分展现一些常用的 NoSQL 解决方案，如 HA、全文搜索等；第四部分讨论 NoSQL 的一些高级主题，如函数式编程。

全书理论与实践并重，每章后面还有通俗的案例。对于 NoSQL 的初学者来说，不失为一本了解 NoSQL 技术全貌的优秀读物。

译者序

在大数据处理领域，NoSQL 无疑是一颗璀璨的明珠。近年来，NoSQL 数据库凭借其易扩展、高性能、高可用、数据模型灵活等特色获得了大批互联网公司的青睐，百度、阿里巴巴、京东等都分别在自己的业务系统中尝试了 NoSQL 解决方案。随着传统的 RMDBS 在某些业务场景下越来越乏力，可以预见的是，越来越多的组织将会乐意尝试使用 NoSQL 解决方案来解决问题。

得益于工业界和学术界的高度关注和大力支持，许多 NoSQL 产品在短时间内大量涌现，方兴未艾，如 HBase、Cassandra、CouchDB、MongoDB、Redis、Neo4j 等。如此多的产品为架构师提供了大量的选择，但却给初学者带来一些学习的困扰。

我认为，学习一门技术，起点很重要。如果初学者将自己的起点定位在某一种 NoSQL 产品上，那么学习完这种产品，得到的知识只是 NoSQL 技术中很小的一部分。但如果将起点定位在 NoSQL 的架构、模式上，那么初学者很快就会对 NoSQL 的全貌有个清晰的认识，这样在学习具体产品时，就会事半功倍。

本书对于 NoSQL 的学习者来说是一本很好的读物，它能让初学者快速地领略到 NoSQL 技术的全貌，为后面的学习打下坚实的基础。本书的两位作者 Dan McCreary 和 Ann Kelly 是两位经验丰富的架构师，具有高屋建瓴的视角，更为难得的是，他们这本书还具有深入浅出的特点。我相信，本书作为技术人员的第一本 NoSQL 书籍来说是很合适的。

在本书翻译过程中，得到了 BBD 小伙伴们的大力支持，在此对曾途、尹康、何宏靖、刘世林、赵飞、颜如宾、王维、杨舟、陈丽竹 Ju、利晓蕾、李倩、黄艳、宋开发、丁梦希、夏阳、唐婷婷、吴松珏等表示感谢。

就像本书第 2 章提到的 CAP 理论，无论是 RMDBS 还是 NoSQL 都不可能同时满足 CAP 理论的三个方面，要学会接受缺憾。技术如此，人生亦如此。

范东来

2015 年 10 月于成都

译者简介

范东来

北京航空航天大学硕士，BBD（数联铭品）大数据技术部负责人，大数据平台架构师，极客学院布道师，著有《Hadoop 海量数据处理：技术详解与项目实战》。研究方向：图挖掘、模式分类。

滕雨瞳

清华大学苏州汽车研究院大数据处理中心高级研发工程师。常年从事基于 HBase、Redis、Cassandra 等 NoSQL 数据库的应用开发，具有丰富的 NoSQL 系统架构和性能调优经验。研究方向：大数据处理、数学机械化。

序

如果一个主题被定义为它不是什么而不是它是什么，我们该如何向别人解释它？相信我，按照一位过去 3 年在这个领域传道授业的人的说法，这是非常困难的，这也是许多专家、学者、技术厂商的共识。尽管很少人认为 NoSQL 这个名字是最好的，但似乎每个人都同意，它比其他术语更加贴切地定义了这一类产品和技术。所以我强烈建议不要由于 NoSQL 的定义方式，而放弃学习这些新的技术。要我来说，这些是值得你花费时间的。

接下来，是一些个人背景的简要介绍：作为一个信息管理方面的出版商，我听说过 NoSQL 这个词，但当时并不知道它的含义，直到 3 年前我在多伦多参加一次会议时，在走廊上碰到了 Dan McCreary。他告诉了我一些情况，使我了解了他目前的项目、使用的令人兴奋的技术和优秀的工作伙伴。他使我相信 NoSQL 技术将呈星火燎原之势蓬勃发展，像我这样的人应该尽可能地学习这方面的知识。这无疑是一个极佳的建议，于是从那时起，我们就建立了良好的伙伴关系，如共同参加学术会议、发起在线研讨会、共同撰写论文。同所有 NoSQL 社区的参与者一样，Dan 是一位才华横溢的技术高手。

就像从事一些复杂而深奥的工作的人一样，为了使那些不具备和我同等学习热情和背景的人理解，我经常试图用简单的词汇来解释复杂的事物。但就算你理解了一次完美的电梯演讲的价值，或者拼命地想要向你的母亲解释你将要做的事情，正确的解释仍然是模糊的。有时向更有学识的人解释新事物反而更加困难。这尤其体现在 NoSQL 方面，由于 RDBMS 拥有庞大的社区和无数的拥簇者，他们只需要一个查询工具就高效而愉快地工作了 30 年。

这就是《解读 NoSQL》出现的原因。如果你参与企业信息系统的建设并且正试着理解 NoSQL 的价值，那么你会庆幸这本书的出现，因为它对你来说非常直观。诚然，新技术的使用能够使技术先驱者尝到甜头，但是对于企业 IT 从业人员来说，仍然存在一些令人望而生畏的障碍。其中的一个原因是由于这些年来成熟技术的沉淀和积累，周围的人会质疑你到底是为什么想要将数据转换成一个“四不像”而不是用优美的、有序的表来进行存储。

本书的作者也意识到了这一点，并且针对你将会面对到的技术架构之间的取舍问题做了大量的研究分析。除此之外，我还欣赏他们付出了大量努力所提供的贯穿全书的案例研究。案例通常

是说服的关键，当你向你的组织介绍这些新技术时，这些源自真实线上应用的案例对于你想要表达的主题来说，其价值是难以估量的。

虽然 Dan McCreary 和 Ann Kelly 给出了第一个 NoSQL 全面的定义和为什么在生产环境中使用 NoSQL 的原因，但本书并不仅仅是一本单纯的技术书籍，它背后包含了作者与架构师、开发人员一起付出的大量努力以确保不同 NoSQL 产品的细节和特点都在本书得到了完整呈现。

本书是一本易于理解的 NoSQL 指南，对于技术经理、架构师和开发人员来说有很高的参考价值。它适合需要全方位了解数据管理解决方案以应对世界上日益增长的、复杂的、海量的、快速流动的数据所带来的严峻挑战的任何人。第 1 章的标题是“NoSQL：明智的选择”，当你选择这本书的时候，你已经做出了一个明智的选择。

Tony Shaw

Dataversity 创始人和首席执行官

前言

有时候，现实迫使我们重新审视我们认为已经了解的事物。在花费了大量的工作时间专注于以行式数据结构存储数据的数据建模任务之后，我们发现，其实建模环节并不是非做不可的。但是这些信息并不意味着我们现有的知识体系是无效的，它迫使我们去审视应该如何解决企业的技术难题。有了新的知识、技术和解决问题方式的武装之后，我们的思路才能得以扩展。

2006 年，在一个涉及房地产交易的项目中，我们花了好几个月的时间设计 XML 的语言模式和形式以存储层次结构复杂的数据。根据我的一个朋友 Kurt Cagle 的建议，我们发现，用原生 XML 数据库对数据进行存储为我们的项目节省了数月的对象建模、设计关系型数据库以及对象关系映射时间，并最终形成一个可以由非专业人员进行维护的异常简单的架构。

对进入 NoSQL 领域的人来说，能意识到企业数据可以用 RDBMS 以外的架构进行存储是重要的转折点。最初，我们可能对这些消息持怀疑态度，会带着恐惧和自我怀疑的复杂心情来看待这些信息。我们会质疑自己的技能和为我们提供培训的教育机构以及那些强调 RDBMS 和对象是解决问题唯一途径的组织。但是，我们如果要公平地对待客户和用户，就必须进行一种全方位的尝试来寻找解决企业问题的最佳方案并评估其他数据库产品架构。

2010 年，一直缺乏关注的 NoSQL 数据库技术得以在大型企业数据会议中露面，当时我们和来自 Dataversity 的 Tony Shaw 一起讨论了关于启动一个新的会议的议题。该会议旨在为所有对 NoSQL 技术感兴趣的人提供一个平台，供他们学习和展示如何使用 NoSQL 数据库技术。2011 年 8 月第一届 NoSQL Now! 会议在加州圣何塞成功举行，吸引了大约 500 名感兴趣和好奇的参会者。

在会议上，我们发现没有任何一份资料覆盖了 NoSQL 架构，或者客观地介绍了一个针对业务来选择 NoSQL 数据库产品的流程。人们想要的不仅仅是开源项目中的“Hello World”示例，而是一本能够帮助他们根据具体业务选择技术架构并像评估商业数据库系统那样评估开源技术的指南。

找到一家能够认同我们现有技术文档的出版机构是第一步，幸运的是，我们发现 Manning 出版社理解这种出版价值。

致谢

我们要感谢 Manning 出版社的每一个人，是你们帮助我们最初的思想转化为一本书。Michael Stephens 带领我们的工作步入正轨；Elizabeth Lexleigh 是我们的开发编辑，她非常耐心地阅读每一章的每一版；Nick Chase 完成了所有的技术工作；还有市场营销和生产团队以及每一个幕后工作者，我们衷心感谢你们的努力、指导和鼓励。

感谢那些回顾了案例研究和为我们提供真实 NoSQL 案例的人们，感谢你们宝贵的时间和专业知识：George Bina、Ben Brumfield、Dipti Borkar、Kurt Cagle、Richard Carlsson、Amy Friedman、Randolph Kahle、Shannon Kempe、Amir Halfon、John Hudzina、Martin Logan、Michaline Todd、Eric Merritt、Pete Palmer、Amar Shan、Christine Schwartz、Tony Shaw、Joe Wicentowski、Melinda Wilken 和 Frank Weige。

感谢那些在书稿修改过程中提出了宝贵意见和反馈的评阅者们，有了你们的参与，我们的书才会更好：Aldrich Wright、Brandon Wilhite、Craig Smith、Gabriela Jack、Ian Stirk、Ignacio Lopez Vellon、Jason Kolter、Jeff Lehn、John Guthrie、Kamesh Sampah、Michael Piscatello、Mikkel Eide Eriksen、Philipp K. Janert、Ray Lugo, Jr.、Rodrigo Abreu 和 Roland Civet。

在此，特别感谢我们的朋友 Alex Bleasdale 为我们提供 NoSQL 安全机制对应章中有关基于用户的访问控制的案例研究的代码。特别感谢 Tony Shaw 为本书撰写的序和 Leo Polovets 在出版前对最终书稿的技术审校。

关于本书

编写本书的过程中，我们抱着两个目标：第一个目标是介绍 NoSQL 数据库；第二个目标则是为读者展示将 NoSQL 系统作为独立解决方案的方法，以及扩展当前 SQL 系统解决业务问题的途径。我们欢迎任何对 NoSQL 感兴趣的读者将本书作为参考。你会发现本书包含的信息、例子以及案例的目标受众是那些有兴趣学习 NoSQL 的技术经理、解决方案架构师和数据架构师。

本书将会帮助你客观地评估 SQL 和 NoSQL 数据库系统以明白它们各自解决了哪类业务问题。如果你想要的是一本关于某个产品的编程指导，那么本书可能就不是你想要的。在本书中，你将了解到 NoSQL 兴起背后的动机以及相关的术语和概念。对于书中有些章节介绍的内容，你可能已经了解，可以根据实际情况略看或直接跳过该部分，而把精力集中在你仍然不了解的部分。

最后，我们非常重视和关注行业标准。SQL 系统相关的行业标准允许应用在使用统一语言的情况下自由切换底层数据库。但遗憾的是，NoSQL 系统还不能作出这个保证。在不久的将来，NoSQL 应用提供商会敦促 NoSQL 数据库供应商采纳一系列标准以使其能像 SQL 一样具备兼容性。

路线图

本书分为 4 部分。第一部分由 NoSQL 定义和 NoSQL 运动背后的基本概念回顾组成。

在第 1 章“**NoSQL：明智的选择**”中，我们定义了术语 NoSQL，讨论了触发 NoSQL 运动的关键性事件，并阐述了 NoSQL 系统带来的大局上的商业收益。熟悉 NoSQL 运动及其商业收益的读者可以选择略过本章。

在第 2 章“**NoSQL 概念**”中，我们引入了 NoSQL 运动中的核心概念。读者在初次通读的过程中，可以选择略过本章，但这一章对于帮助理解之后的章节非常重要。我们建议将本章作为一个参考指导，在遇到相关概念时再返回本章仔细阅读相关内容。

在第二部分“**数据库模式**”中，我们深入回顾了 SQL 和 NoSQL 数据库的架构模式。我们探讨了不同的数据库结构、如何访问它们，并通过用例展示了每种架构最适用的场景。

第 3 章介绍了“**基础数据架构模式**”。本章以回顾关系型数据库背后的驱动力以及 ERP 系统

需求如何塑造出现今关系型数据库和商业智能/数据仓库 (BI/DW) 系统开始, 还简略地讨论了其他数据库系统, 如对象数据库、修改控制系统。如果已经熟悉了这些系统, 可以略过本章。

在第 4 章 “NoSQL 数据架构模式” 中, 我们介绍了 NoSQL 相关的数据库模式。我们探讨了键值存储、图存储、列族存储 (BigTable) 以及文档存储。本章将用定义、示例和案例研究等内容辅助读者理解。

第 5 章覆盖了 “原生 XML 数据库” 的内容。因其低廉的成本和对标准的支持, 该数据库常被用于政府和出版应用。我们为读者展示了两个来自金融和政府出版领域的案例。

第三部分探讨了将 NoSQL 系统应用到大数据、搜索、高可用性以及敏捷网站部署等问题上的方法。

在第 6 章 “用 NoSQL 管理大数据” 中, 你将了解到配置 NoSQL 系统在商用硬件上高效处理大体量数据的方法。我们还有一个关于分布式计算和横向可扩展性的讨论。同时, 我们还为读者展示了一个不能通过商用硬件扩展完成巨型图分析的案例。

在第 7 章 “用 NoSQL 搜索获取信息” 中, 你将学习到如何通过实现文档模型和保留文档内容的方式提升搜索质量。我们讨论了采用 MapReduce 转化技术创建倒排索引并最终获得快速搜索性能的方法, 还回顾了用于文档和数据库的搜索系统, 并展示了采用结构化搜索解决方案获得准确的搜索结果排序的方法。

第 8 章的内容覆盖了 “用 NoSQL 构建高可用的解决方案”。我们阐述了如何使用 NoSQL 系统自带的复制和分布式特性提升系统可用性。你将了解到在采用了数据同步技术的情况下, 如何使用大量低廉 CPU 来提供更长的在线时间。我们的案例研究揭示了纯点对点架构能比其他分布式模型提供更高可用性的原因。

在第 9 章中, 我们讨论了 “用 NoSQL 提升敏捷性”。因为移除了对象关系映射层, NoSQL 软件开发变得更简单且能快速适应业务需求的变更。你将了解到这些 NoSQL 系统是如何将富有经验的开发人员和非编程人员整合到软件开发生命周期过程中的。

在第四部分中, 我们先介绍了函数式编程、安全等 “高级主题”, 然后, 回顾了一个正式的 NoSQL 系统选型过程。

在第 10 章中, 我们介绍了关于 “NoSQL 与函数式编程” 和类似 MapReduce 的分布式转化架构的要求这两方面内容。我们探讨了函数式编程对 NoSQL 解决方案能使用大量低廉 CPU 能力的影响, 以及许多 NoSQL 数据库系统采用类似 Erlang 的基于角色的框架的原因。我们还通过 NetKernel 系统的案例介绍了结合函数式编程和面向资源编程构建性能可扩展的分布式系统的方法。

第 11 章覆盖了 “安全: 保护 NoSQL 系统中的数据” 相关的内容。我们回顾了 NoSQL 解决方案中安全性方面的历史以及关键的安全性问题。我们还通过例子为读者展示了键值存储、列族存储和文档存储实现一个健壮的安全模型的能力。

在第 12 章 “选择合适的 NoSQL 解决方案” 中, 我们遍历了企业为业务问题选型合适数据库的正式流程。最后, 我们将以一些关于这些技术会如何影响业务系统选型的想法和信息结束本章。

代码约定和下载

清单和文本中的源代码会以等宽字体出现，以便和普通文本区分开。你可以从 Manning 出版社的网站（www.manning.com/MakingSenseofNoSQL）上下载清单中的源代码。

作者在线论坛

购买本书后，读者享有由 Manning 出版社运营的私有网页论坛的免费访问权。在该论坛上，读者可以评论本书、询问技术问题以及从作者和其他用户那获得帮助。如果想访问和订阅这个论坛，读者可以访问 www.manning.com/MakingSenseofNoSQL。这个页面提供了如何在注册后登录论坛、可以获得哪些帮助以及论坛行为规范等信息。

Manning 出版社对读者承诺，要为读者与读者、读者与作者之间有意义的交流提供一个空间。这不是一个针对任何特定数量的作者的承诺。任何在论坛中做出过贡献的作者都是志愿且不计酬劳的。因此，我们建议读者试着询问作者一些更具挑战性的问题，以免他们对问题毫无兴趣。

只要书籍仍在印刷，这个作者在线论坛和历史讨论存档就一直可以通过出版社的网站访问。

关于作者

Dan McCrearay 是一个对行业标准具有强烈兴趣的数据架构咨询师。他曾经在贝尔实验室（负责集成电路设计）、超级计算行业（负责移植 UNIX）以及史蒂夫·乔布斯（Steve Jobs）创办的 NeXT 计算机公司（软件布道师）工作，也创办了自己的咨询公司。Dan 于 2002 年开始参与制定美国联邦数据标准，该标准也在国家信息交换模型（NIEM）中获得了启用。在 2006 年面对用原生 XML 数据库存储数据时，Dan 开始了他的 NoSQL 开发历程。他还是万维网 XForms 标准制定小组的客座专家以及 NoSQL Now! 会议的联合创始人。

Ann Kelly 是 Kelly McCrearay 咨询公司的软件咨询师。在从事多年保险行业软件开发和管理项目之后，她于 2011 年开始转投 NoSQL 领域。从那时起，她就开始为客户构建能快速高效地解决业务问题的 NoSQL 解决方案，同时还提供管理应用的培训。

目录

第一部分 了解 NoSQL

第 1 章 NoSQL:明智的选择 2

- 1.1 什么是 NoSQL 3
- 1.2 NoSQL 的商业驱动 4
 - 1.2.1 容量 5
 - 1.2.2 速度 5
 - 1.2.3 敏捷性 6
- 1.3 NoSQL 案例研究 6
 - 1.3.1 案例研究: LiveJournal 的 Memcache 技术 7
 - 1.3.2 案例研究: Google 的 MapReduce——利用商用硬件生成搜索索引 7
 - 1.3.3 案例研究: Google 的 Bigtable——一个有着数十亿行和百万列的表 8
 - 1.3.4 案例研究: 亚马逊的 Dynamo——每天 24 小时接收订单 9
 - 1.3.5 案例研究: MarkLogic 9
 - 1.3.6 实践 10
- 1.4 小结 10

第 2 章 NoSQL 概念 12

- 2.1 保持组件简单以促进重用 12
- 2.2 将应用分层以简化设计 14
- 2.3 策略地使用 RAM、SSD

- 和磁盘提升性能 17
- 2.4 使用一致性散列算法维护当前的缓存 18
- 2.5 比较 ACID 和 BASE——两种可靠的数据库事务方法 19
 - 2.5.1 RDBMS 的事务控制——ACID 21
 - 2.5.2 非 RDBMS 的事务控制——BASE 22
- 2.6 通过数据库分片获得水平扩展能力 23
- 2.7 基于 Brewer 的 CAP 定理进行权衡 25
- 2.8 实践 26
- 2.9 小结 27
- 2.10 延伸阅读 27

第二部分 数据库模式

第 3 章 基础数据架构模式 30

- 3.1 什么是数据架构模式 31
- 3.2 理解应用于 RDBMS 的行存储设计模式 31
 - 3.2.1 行存储如何工作 32
 - 3.2.2 行存储的演变 33
 - 3.2.3 分析行存储模式的优点和缺点 34
- 3.3 示例: 对销售订单进行

- 连接操作 35
- 3.4 回顾 RDBMS 实现的特性 36
 - 3.4.1 RDBMS 事务 37
 - 3.4.2 固定的数据定义语言和强类型的列 38
 - 3.4.3 通过 RDBMS 视图保证安全并进行访问控制 39
 - 3.4.4 RDBMS 的复制和同步 39
- 3.5 通过 OLAP、数据仓库和商业智能系统对历史数据进行分析 41
 - 3.5.1 数据如何从操作型系统流入分析型系统 42
 - 3.5.2 熟悉 OLAP 的概念 44
 - 3.5.3 通过汇总生成专项报表 45
- 3.6 将高可用性和以读为主的系统一体化 46
- 3.7 在修改控制系统和数据同步中使用散列树 47
- 3.8 实践 49
- 3.9 小结 49
- 3.10 延伸阅读 50

第 4 章 NoSQL 数据架构模式 51

- 4.1 键值存储 52
 - 4.1.1 什么是键值存储 52
 - 4.1.2 使用键值存储的好处 54
 - 4.1.3 使用键值存储 56
 - 4.1.4 使用案例: 用键值存储保存网页 59
 - 4.1.5 使用案例: 亚马逊简单存储服务 (S3) 59
- 4.2 图存储 60
 - 4.2.1 图存储概述 60
 - 4.2.2 用 RDF 标准来连接外部数据 62
 - 4.2.3 图存储的使用案例 63
- 4.3 列族 (Bigtable) 存储 68
 - 4.3.1 列族存储基础 69
 - 4.3.2 理解列族存储的键 69
 - 4.3.3 列族存储的优点 70

- 4.3.4 案例研究: 在 Bigtable 中存储分析信息 72
- 4.3.5 案例研究: Google 地图用 Bigtable 存储地理信息 72
- 4.3.6 案例研究: 使用列族存储用户偏好信息 73
- 4.4 文档存储 73
 - 4.4.1 文档存储基础 74
 - 4.4.2 文档集合 74
 - 4.4.3 应用集合 75
 - 4.4.4 文档存储的 API 75
 - 4.4.5 文档存储的实现 76
 - 4.4.6 案例研究: MongoDB 和广告服务器 76
 - 4.4.7 案例研究: 大型对象数据库 CouchDB 77
- 4.5 NoSQL 架构模式的变体 78
 - 4.5.1 定制 RAM 和 SSD 存储 78
 - 4.5.2 分布式存储 78
 - 4.5.3 分组的对象 79
- 4.6 小结 80
- 4.7 延伸阅读 81

第 5 章 原生 XML 数据库 82

- 5.1 什么是原生 XML 数据库 83
- 5.2 用原生 XML 数据库构建应用 85
 - 5.2.1 加载数据可以像拖曳那样简单 86
 - 5.2.2 使用集合来组织 XML 文档 87
 - 5.2.3 使用 XPath 运用简单的查询转换复杂的数据 89
 - 5.2.4 用 XQuery 转换数据 91
 - 5.2.5 用 XQuery 更新文档 93
 - 5.2.6 XQuery 全文搜索标准 94
- 5.3 在原生 XML 数据库中应用 XML 标准 94
- 5.4 用 XML Schema 和 Schematron 设计和验证数据 96
 - 5.4.1 XML Schema 96
 - 5.4.2 使用 Schematron 检查文档规则 97

- 5.5 用自定义模块扩展 XQuery 98
- 5.6 案例研究：在美国国务院历史学家办公室使用 NoSQL 98
- 5.7 案例研究：使用 MarkLogic 管理金融衍生品 102
 - 5.7.1 为什么 RDBMS 难以存储金融衍生品 102
 - 5.7.2 一个投资银行从 20 个 RDBMS 转换到 1 个原生 XML 数据库 102
 - 5.7.3 迁移至原生 XML 文档存储的商业好处 104
 - 5.7.4 项目成果 104
- 5.8 小结 105
- 5.9 延伸阅读 105

第三部分 NoSQL 解决方案

第 6 章 用 NoSQL 管理大数据 108

- 6.1 什么才是大数据解决方案 109
- 6.2 线性扩展数据中心 112
- 6.3 理解线性可扩展性和表现力 113
- 6.4 了解大数据问题的类型 114
- 6.5 使用无共享架构分析大数据 116
- 6.6 选择分布式模型：主从模型与对等模型 117
- 6.7 在分布式系统上使用 MapReduce 处理数据 118
 - 6.7.1 MapReduce 和分布式文件系统 120
 - 6.7.2 MapReduce 怎样做到高效处理大数据问题 121
- 6.8 NoSQL 系统处理大数据问题的 4 种方式 122

- 6.8.1 分发查询到数据，而非数据到查询 122
- 6.8.2 使用散列环在集群中均匀分发数据 122
- 6.8.3 使用复制扩展读取性能 123
- 6.8.4 使数据库将查询均衡地分发到数据节点 124
- 6.9 案例研究：使用 Apache Flume 处理事件日志 125
 - 6.9.1 事件日志数据分析的挑战 125
 - 6.9.2 Apache Flume 搜集分布式事件日志的方法 126
 - 6.9.3 延伸思考 127
- 6.10 案例研究：计算机辅助发现医疗保险欺诈 128
 - 6.10.1 什么是医疗保险欺诈检测 128
 - 6.10.2 使用图和定制的内存共享硬件检测医疗保险欺诈 129
- 6.11 小结 130
- 6.12 延伸阅读 131

第 7 章 用 NoSQL 搜索获取信息 132

- 7.1 什么是 NoSQL 搜索 132
- 7.2 搜索分类 133
 - 7.2.1 布尔搜索、全文关键字搜索和结构化搜索的比较 133
 - 7.2.2 测试常见搜索类型 134
- 7.3 提高 NoSQL 搜索效率的策略和方法 135
- 7.4 使用文档结构提升搜索质量 137
- 7.5 搜索质量量化 139
- 7.6 本地索引与远程搜索服务 139
- 7.7 案例研究：使用 MapReduce 建立倒排索引 141
- 7.8 案例研究：搜索技术文档 142

- 7.8.1 什么是技术文档
搜索 142
- 7.8.2 在 NoSQL 文档存储中
保留文档结构 143
- 7.9 案例研究: 搜索领域
语言——可检索性和
重用性 144
- 7.10 实践 145
- 7.11 小结 146
- 7.12 延伸阅读 146

第 8 章 用 NoSQL 构建高可用 的解决方案 148

- 8.1 高可用 NoSQL 数据库的
定义 148
- 8.2 量化 NoSQL 数据库的
可用性 149
 - 8.2.1 案例研究: 亚马逊 S3 的
服务级别协议 151
 - 8.2.2 预测系统可用性 151
 - 8.2.3 实践 152
- 8.3 NoSQL 系统的高可用性
策略 153
 - 8.3.1 使用负载均衡器将流量转
向到最空闲的节点 153
 - 8.3.2 结合高可用分布式文件系
统和 NoSQL 数据库 154
 - 8.3.3 案例研究: 将 HDFS 作为
一个高可用的文件系统存
储主数据 155
 - 8.3.4 使用托管的 NoSQL
服务 156
 - 8.3.5 案例研究: 使用亚马逊的
DynamoDB 作为高可用数
据存储 157
- 8.4 案例研究: 使用 Apache
Cassandra 作为高可用的
列族存储 158
- 8.5 案例研究: 使用 Couchbase
作为高可用文档
数据库 161
- 8.6 小结 163
- 8.7 延伸阅读 163

第 9 章 用 NoSQL 提升敏 捷性 165

- 9.1 软件敏捷性的定义 165
- 9.2 量化敏捷性 169
- 9.3 使用文档存储来避免对象
关系映射 171
- 9.4 案例研究: 使用 XRX 管理
复杂表单 172
 - 9.4.1 什么是复杂业务
表单 173
 - 9.4.2 用 XRX 替换客户端
JavaScript 和对象关系
映射 173
 - 9.4.3 理解 XRX 对敏捷性的
影响 176
- 9.5 小结 177
- 9.6 延伸阅读 177

第四部分 高级主题

第 10 章 NoSQL 与函数式 编程 180

- 10.1 什么是函数式编程 181
 - 10.1.1 指令式编程就是管理
程序状态 181
 - 10.1.2 函数式编程是没有副作
用的并行转化 183
 - 10.1.3 比较指令式编程和函数
式编程的扩展性 186
 - 10.1.4 使用引用透明避免重复
计算 187
- 10.2 案例研究: 用 NetKernel
优化网页内容组装 188
 - 10.2.1 组装嵌套内容, 追踪组件
依赖 188
 - 10.2.2 用 NetKernel 优化组件
再生成 189
- 10.3 函数式编程语言
示例 191
- 10.4 完成指令式编程到函数
式的编程转变 192
 - 10.4.1 使用函数作为函数的

- 参数 192
- 10.4.2 使用递归处理非结构化文档数据 192
- 10.4.3 使用不可变变量而非可变变量 192
- 10.4.4 去除循环和条件语句 193
- 10.4.5 新的思维方式: 从状态记录到转化隔离 193
- 10.4.6 质量、校验和一致性单元测试 194
- 10.4.7 函数式编程的并发 194
- 10.5 案例研究: 用 Erlang 构建 NoSQL 系统 194
- 10.6 实践 197
- 10.7 小结 198
- 10.8 延伸阅读 198

第 11 章 安全: 保护 NoSQL 系统中的数据 200

- 11.1 NoSQL 数据库的一种安全模型 201
 - 11.1.1 使用服务减少数据库内部的安全性需求 202
 - 11.1.2 使用数据仓库和 OLAP 减少数据库内部的安全性需求 203
 - 11.1.3 应用级安全措施和数据库级安全措施的收益总结 203
- 11.2 收集安全需求 204
 - 11.2.1 认证 205
 - 11.2.2 授权 207
 - 11.2.3 审查和日志记录 210
 - 11.2.4 加密和数字签名 211
 - 11.2.5 保护公开网站免受拒绝服务攻击和注入攻击 212
- 11.3 案例研究: 键值存储的访问控制——亚马逊的 S3 213
 - 11.3.1 身份和访问管理 (IAM) 214
 - 11.3.2 访问控制列表 (ACL) 214

- 11.3.3 桶策略 214
- 11.4 案例研究: 在 Apache Accumulo 中使用键可见性技术 215
- 11.5 案例研究: 在安全发布流程中使用 MarkLogic 的 RBAC 模型 216
 - 11.5.1 使用 MarkLogic 的 RBAC 安全模型保护文档 217
 - 11.5.2 在安全出版中使用 MarkLogic 218
 - 11.5.3 MarkLogic 的安全模型的优势 218
- 11.6 小结 219
- 11.7 延伸阅读 219

第 12 章 选择合适的 NoSQL 解决方案 221

- 12.1 什么是架构利弊分析 222
- 12.2 数据库架构选型团队的组成变化 223
 - 12.2.1 选择合适的团队 224
 - 12.2.2 考虑经验偏好 224
 - 12.2.3 雇用外部咨询师 225
- 12.3 架构权衡分析步骤 225
- 12.4 解构分析: 质量树 228
 - 12.4.1 质量属性样例 229
 - 12.4.2 评估混合架构和云架构 231
- 12.5 与利益系相关者沟通结果 231
 - 12.5.1 用质量树作为导航图 232
 - 12.5.2 实践 233
 - 12.5.3 使用质量树进行项目风险交流 234
- 12.6 找到合适的验证架构的试点项目 235
- 12.7 小结 236
- 12.8 延伸阅读 237

第一部分

了解 NoSQL

在第一部分中，我们将介绍 NoSQL。我们将定义什么是 NoSQL，讨论为什么 NoSQL 运动开始流行，了解 NoSQL 的核心主题，讨论基于 NoSQL 的解决方案能给企业带来什么样的好处。

第 1 章从定义 NoSQL 开始，然后谈到 NoSQL 流行背后的商业驱动和动机。第 2 章会在第 1 章的基础上进行扩展，并对 NoSQL 相关的核心概念进行探讨。

如果你对 NoSQL 运动已经足够熟悉，你可能会希望略过第 1 章和第 2 章中介绍 NoSQL 相关概念的部分，但是我们仍然鼓励所有人都阅读第 2 章中的相关概念，因为这些概念将从始至终贯穿全书。

第 1 章 NoSQL: 明智的选择

本章主要内容

- 什么是 NoSQL
- NoSQL 的商业驱动
- NoSQL 的案例研究

最小部件所耗费的复杂度约每两年增长一倍。当然，如果这个增长速度不再增长的话，短期内这样的增长速度仍然会继续。

——Gordon Moore (戈登·摩尔, Intel 创始人之一), 1965

……你最好开始游泳，否则你将沉入水底……因为时代在变革。

——Bob Dylan

我们编写本书有两个初衷：第一，介绍 NoSQL 数据库；第二，展示如何使用 NoSQL 系统作为独立的解决方案或对现有 SQL 系统进行补充以解决实际业务问题。尽管我们希望所有对 NoSQL 感兴趣的人都能够将本书作为指南，但是本书中的内容、样例、案例研究面向的是希望学习 NoSQL 的技术经理、解决方案架构师和数据架构师。

本书将帮你评估 SQL 数据库系统和 NoSQL 数据库系统能够胜任的业务场景。如果你在寻找一本详细的、具体的编程指南，那么本书不适合你。在这里你会发现 NoSQL 背后的发展动机、相关技术和概念。或许你已经理解本书某些章节所提到的主题，那么可以略过它们并专注于那些新的知识。

最后，我们对标准表示强烈的关注。SQL 系统的相关标准使得应用可以使用一种通用的语言在不同的数据库产品之间进行迁移。遗憾的是，NoSQL 系统目前并没有相关的标准。在将来，NoSQL 应用的开发者将会向 NoSQL 数据库厂商施压，迫使它们采用一系列标准，以使 NoSQL 应用的移植性能够媲美 SQL。

在本章，我们将给出 NoSQL 的定义并探讨使 NoSQL 产品如此有趣并在业界流行的原因。最后，我们将看到 5 个通过实施 NoSQL 产品解决具体业务问题的成功的行业案例。

1.1 什么是 NoSQL

准确定义 NoSQL 本身就具有挑战性。NoSQL 这个术语其实是有待商榷的，因为它并没有真正意义上揭示 NoSQL 运动的核心主题。这个术语来源于一群定期在湾区开会并讨论一些共同关注的可扩展的开源数据库的人们，它就这样出现了。不管它形不形象，它似乎出现在所有地方：行业期刊、产品说明和各种会议。在本书，我们将用 NoSQL 区别于传统的关系型数据库管理系统（RDBMS）。

按照我们的目标，我们将从以下几个方面定义 NoSQL。

NoSQL 是关于快速而高效地处理数据，专注于性能、可靠性和敏捷性的一组概念。

这听起来有些宽泛，是吧？它没有排除 SQL 或者 RDBMS，对吗？这其实并没有错。重要的是，我们需要搞清楚 NoSQL 背后的核心主题：它是什么，最重要的是，它不是什么。那么 NoSQL 究竟是什么？

- NoSQL 不仅仅是普通意义上的表——NoSQL 系统可以从许多格式中存储和检索数据：键值存储、图数据库、列族存储、文档存储甚至是普通的表。
- NoSQL 避免连接操作——NoSQL 系统能够通过简单的接口提取数据从而避免连接操作。
- NoSQL 是模式无关的——NoSQL 系统允许将数据拖曳到一个文件夹并进行查询，而不需要创建对象-关系模型。
- NoSQL 工作在多核处理器之上——NoSQL 系统允许将数据库部署在多核处理器之上从而保持良好的性能。
- NoSQL 运行在无共享的商用计算机——大多数（并不是所有的）NoSQL 系统利用廉价的商用处理器、独立的硬盘和内存进行搭建。
- NoSQL 支持线性扩展——当你增加更多的处理器时，你的单位性能增量始终是一致的。
- NoSQL 是创新的——NoSQL 对于存储、检索、操作数据提供了更多的选择。NoSQL 的支持者（也被称为 NoSQLers）对于 NoSQL 和 SQL 解决方案持一种兼收并蓄的态度。对于 NoSQL 社区来说，NoSQL 的意思是“不只是 SQL”。

同样重要的是，NoSQL 不是什么。

- NoSQL 不是一种 SQL 语言——NoSQL 并不是采用非 SQL 查询语言的应用。SQL 和其他查询语言也可以被用于 NoSQL 数据库。
- NoSQL 不仅是开源的——尽管许多 NoSQL 系统都有一个开源模式，但是借鉴 NoSQL 思想的商业产品同样也不排斥开源。你仍然可以通过商业产品创新地解决问题。
- NoSQL 不仅仅代表海量数据——大部分但不是所有的 NoSQL 应用都是来源于为应对海量数据而提升当前应用运行规模的需求。虽然数据的容量和数据处理速度很重要，但 NoSQL 也专注于数据的种类和敏捷性。
- NoSQL 和云计算没有关系——虽然很多 NoSQL 系统为了能在负载变化时利用云端动态扩展的优势而部署在云端，但是 NoSQL 系统也能像在云端运行那样运行在公司的数据中心。

- 这不是关于如何用好 RAM 和 SSD——NoSQL 专注于高效地使用 RAM 和固态硬盘获得性能的提升，尽管这很重要，但是 NoSQL 系统可以运行在普通硬件之上。
- NoSQL 并不是精英团体的专属产品——NoSQL 不是一个排他的、只有少数产品的俱乐部，也并没有为加入设置门槛。想成为一个 NoSQLer，你只需说服别人，对于他们的业务难题你有创新的解决思路。

NoSQL 应用采用很多数据存储类型（不同的数据库）。有简单的表现键值关系的键值存储、表现关联关系的图存储、用以存储可变数据的文档存储，每一种 NoSQL 数据存储类型都有其独特的属性和使用场景，如表 1-1 所示。

表 1-1 NoSQL 数据存储类型——4 种主要的 NoSQL 系统及其代表产品

类型	典型的应用	案例
键值对——一种简单的数据存储形式，可以通过键来访问值	<ul style="list-style-type: none"> • 图像存储 • 基于键的文件系统 • 对象缓存 • 设计为可扩展的系统 	<ul style="list-style-type: none"> • Berkeley DB • Memcache • Redis • Riak • DynamoDB
列族——稀疏矩阵的形式，可以用行和列作为键	<ul style="list-style-type: none"> • 网络爬虫的结果 • 大数据的问题 • 软一致性 	<ul style="list-style-type: none"> • Apache HBase • Apache Cassandra • Hypertable • Apache Accumulo
图存储——适用于对关联性要求高的问题	<ul style="list-style-type: none"> • 社交网络 • 欺诈侦测 • 强关联的数据 	<ul style="list-style-type: none"> • Neo4j • AllegroGraph • Bigdata (RDF 数据存储) • InfiniteGraph (Objectivity 公司)
文档存储——将层次化的数据结构直接存储到数据库中	<ul style="list-style-type: none"> • 高度变化的数据 • 文档搜索 • 集成中心 • 互联网内容管理 • 出版物 	<ul style="list-style-type: none"> • MongoDB (10Gen 公司) • CouchDB • Couchbase • MarkLogic • eXist-db • Berkeley DB XML

NoSQL 系统拥有独特的特性能够使其单独使用或者与已有系统配合使用。许多机构认为 NoSQL 系统这样做的原因是为了克服一些常见的问题，如数据的容量、流动的速度、数据的种类和数据的敏捷性以及 NoSQL 运动背后的商业驱动所使然。

1.2 NoSQL 的商业驱动

哲学家、科学家 Thomas Kuhn 提出了“模式转变”（paradigm shift）的概念来描述在科学试

验中反复出现的过程，也是这样才有很多创新的思想爆发出来，并以非线性方式影响了世界。我们将采用 Kuhn 对于模式转变的定义去思考和解释当今 NoSQL 运动以及思维模式、架构、涌现的方法的改变。

许多使用基于单 CPU 的关系型系统的机构都面临着技术的十字路口：机构的需求正在发生变化。企业通过不断采集和分析海量的可变数据获得价值，并基于获得的信息对业务作出快速调整。

图 1-1 展示了来自于数据容量、流动速度、多样性和敏捷性的需求是如何在 NoSQL 解决方案的涌现中起关键作用的。正由于这些商业驱动都对单处理器的关系型模型产生压力，它的基础正变得不那么稳定，再也不能满足机构的需求。

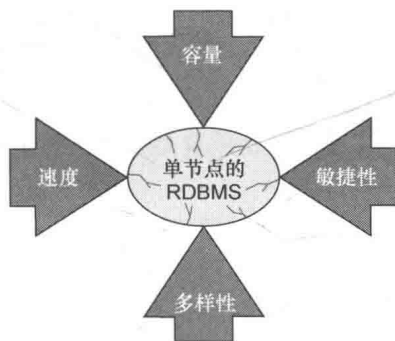


图 1-1 我们看见了诸如数据容量、处理速度、多样性和敏捷性的商业驱动是如何对单 CPU 系统产生压力，甚至导致崩溃。数据容量和流动速度涉及处理飞速到达的大型数据集的能力，而多样性则涉及如何将各种不同类型的数据转化为结构化的表，敏捷性则涉及系统如何快速地对业务改变进行响应

1.2.1 容量

毫无疑问，迫使机构关注他们目前的 RDBMS 的替代品的关键因素是需要通过商用处理器集群查询海量数据。直到 2005 年左右，对性能的提升还停留在购买更快的处理器。但现在，提高处理器的处理速度并不是一个合适的选择。这是因为随着芯片集成度的提高，当芯片过热时，热量将不再能够及时地消散。这种现象，被称之为“功率墙”（power wall），也正是如此，迫使了系统的设计者将注意力从提高单个芯片的处理速度转移到让更多的芯片协同工作。规模向外扩展（也叫横向扩展）而不是规模向上扩展（更快的处理器）的需求，使机构将数据问题切分成独立的路径并交给独立的处理器去分而治之地工作，也就是从串行执行到并行执行。

1.2.2 速度

尽管海量数据已经成为了一个使用户放弃 RDBMS 的原因，但是单处理器系统的快速读写能力的瓶颈亦是关键。许多基于单处理器的 RDBMS 已经不能满足由一些面向公众的网站所发出的

实时写入和在线查询的需求。每当新加入一行，RDBMS 都会频繁地对该行的许多列新建索引，这个过程会影响系统性能。当 RDBMS 被作为网上商店的后台，网络拥塞所引发的随机突发事件会使系统对所有人的响应变慢，而且对系统进行调优以满足必须的高速读写吞吐量的代价是非常高的。

1.2.3 敏捷性

基于 RDBMS 构建应用最复杂的部分莫过于数据读取和写入的过程。如果你的数据具有嵌套和重复子组的数据结构，你就需要一个对象关系映射层（ORM）。该层负责根据对数据库的增删改查的操作在关系型数据库持久化层对对象数据进行导出或导入。这个过程并不简单，并且当开发新应用或者修改现有的应用需要快速改变时，该层并不能很好地作出变化。

通常，对象关系映射的工作需要对对象关系映射框架（如 Java 的 Hibernate 或者 .Net 系统的 NHibernate）非常熟悉和有经验的工程师。但就算是有经验的工程师，小小的改动也将拖慢开发速度和测试流程。

从上面可以看到，速度、容量、多样性和敏捷性是 与 NoSQL 运动关系最紧密的驱动力。现在你已经熟悉了这些驱动力，你也可以审视一下自己的机构，看看 NoSQL 的解决方案是否能够对这些驱动力产生积极的影响，从而帮助业务面对当今竞争激烈的市场的需求变化。

1.3 NoSQL 案例研究

我们的经济正在发生变革，企业想要保持竞争力就必须找到吸引并留住客户的新方法。要做到这一点，就必须得到技术和相关技术人员及时有效的支持。在这个技术前沿时代，解决方案需要运用新的思考方式，即如何实现从传统的思维方式向流程化、技术化的思维方式转变。

以下的案例研究展示了如何用打破陈规的思维方式更快、更经济、更有效地解决问题。表 1-2 总结了 NoSQL 解决方案用于解决特定业务问题的 5 个案例研究。表中展示了问题、业务驱动因素和最终结果。当你查看后面详细案例研究部分的内容时，你会发现这些案例都有一个共同的主题：很多业务问题需要新思路和新技术来提供最佳的解决方案。

表 1-2 与 NoSQL 相关的关键案例研究——所选方案的案例名称/标准、业务驱动、结果（发现）

案例名称/标准	业务驱动	结果
LiveJournal 的 Memcache 技术	需要提高数据库查询性能	通过使用散列和高速缓存来共享 RAM 中的数据，这减少了发送到数据库的读取请求并提高了性能
Google 的 MapReduce	需要基于低成本的硬件为搜索服务生成数十亿的网页索引	通过使用并行处理，数十亿的网页索引可以通过大量的商用处理器迅速完成
Google 的 BigTable	需要在分布式系统中灵活地存储表格数据	通过使用这种稀疏矩阵模式，用户可以认为不需要提前定义数据模型，数据是存储在一个由数十亿行、数百万列组成的表格中

续表

案例名称/标准	业务驱动	结果
亚马逊的 Dynamo	需要每天 24 小时不间断地接收网络订单	键值存储，有着简单的查询接口，即使是海量数据也能完成复制操作
MarkLogic	需要用标准的查询语言来查询存储在商用硬盘中的大量的 XML 文档	通过将包含 XML 文档索引的查询分发至商用服务器，每个服务器通过在本地磁盘处理数据，并将结果返回至查询服务器的方式进行响应

1.3.1 案例研究：LiveJournal 的 Memcache 技术

LiveJournal 博客系统的工程师们正致力于研究如何运用他们最宝贵的资源——每个 Web 服务器中的 RAM——来运行系统。LiveJournal 网站存在一个问题：这个网站太受欢迎了，浏览网站的用户数量也在不断增长。要满足这种不断增长的需求就需要不断增加 Web 服务器，并且每个服务器都要有自己单独的 RAM。

为了提高性能，LiveJournal 工程师发现了一些将最常被数据库查询的数据保存在 RAM 中的方法，避免了在数据库中进行相同 SQL 查询的昂贵成本。但是查询数据的副本是保存在各自 Web 服务器的 RAM 中，所以即使是同一个机架上的服务器也不会知道旁边的服务器已经在 RAM 中保存了查询数据的副本。

因此，LiveJournal 的工程师发明了一个简单的方法来区分每一个 SQL 查询，那就是为每一个 SQL 查询设计一个“签名”。每个签名或散列值就代表一个 SQL SELECT 语句。Web 服务器之间只需要发送一个请求，就可以知道其他服务器中是否已经有执行的 SQL 结果的副本。如果其中一个服务器已有副本，它会将查询结果回传给发出请求的服务器，这样就避免了已经不堪重负的 SQL 数据库数据往返的昂贵成本。他们将这个新系统称作 Memcache，这是因为它管理了 RAM 中的内存高速缓存。

许多其他软件工程师在以前也遇到过这个问题。大型的共享内存的服务器资源池这个概念其实并不新，不同的是这次 LiveJournal 的工程师们在这个概念上领先了一步。他们不仅让这些系统运行（并且运行良好）并通过开放资源许可共享了他们的软件，还标准化了 Web 前端的通信协议（被称为 memcached 协议）。现在，如果有人想缓解因用户反复查询而导致的数据库超负荷运行的话，前端工具是一个不错的选择。

1.3.2 案例研究：Google 的 MapReduce——利用商用硬件生成搜索索引

关于 NoSQL 运动最有影响力的一个案例研究就是 Google 的 MapReduce 系统。在本节，Google 将和我们分享如何使用廉价的商用 CPU 将大量的 Web 数据转换为内容搜索索引。

尽管它们分享的内容是标志性的，但其实映射函数和化简函数的概念并不新。映射函数和化简函数仅仅是数据转换两个阶段的名称而已，如图 1-2 所示。

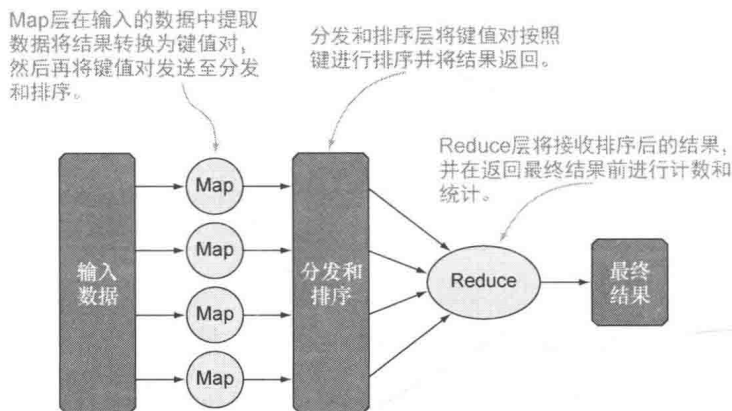


图 1-2 Map 和 Reduce 函数可以在独立的转换系统中将大数据集划分成小块。关键是要将每个函数隔离，这样就可以将其扩展到多个服务器

转换的初始阶段被称为映射操作（map operation），它们负责数据的提取、转换和过滤。然后将结果送到下一层，即化简函数（reduce function）。化简函数将结果进行排序、整合和汇总，从而得到最终结果。

映射和化简函数背后的核心概念基于坚实的计算机科学工作，那还是在 20 世纪 50 年代，麻省理工学院的程序员通过当时比较有影响力的 LISP 系统实现了这些函数。与其他编程语言不同，LISP 重视转化独立列表数据的函数，这是许多在分布式系统下表现出色的函数式编程语言的基础。

Google 扩展了映射和化简函数使其能够处理数十亿网页并能在数以千计的低成本商用 CPU 上可靠运行。利用这两个函数，Google 在海量数据上实现了让映射任务和化简任务经济高效地运行。Google 对 MapReduce 的运用使其他人对函数式编程的威力以及函数式编程系统通过数以千计的低成本 CPU 进行扩展的能力刮目相看。一些软件包，如 Hadoop，很大程度上就是模仿这些函数。

MapReduce 的应用启发了来自雅虎和其他组织的工程师们对 Google 的 MapReduce 创建了开源版本。它让我们逐渐意识到传统过程编程的局限性并鼓励我们使用函数式编程系统。

1.3.3 案例研究：Google 的 Bigtable——一个有着数十亿行和百万列的表

当 Google 发布 Bigtable 系统白皮书《A Distributed Storage System for Structured Data》的时候也影响了许多软件开发人员。Bigtable 背后的动力是存储用网页爬虫在互联网上收集到的 HTML 页面、图片、声音、视频和其他媒体的需求。这些数据太过庞大，以至于不太适合用单个关系型数据库进行存储，所以 Google 建立自己的存储系统。该系统建立的基本目标是可以轻易扩容以应对不断增长的数据存储需求并且不需要在硬件上投入过多。它既不是一个完整的数据库也不是一个文档系统，Google 称它为与结构化数据一起工作的“分布式存储系统”。

据说, Bigtable 项目非常成功。它通过创建一个表存储 Google 开发人员所需的数据从而为他们提供了数据的单一表格视图。此外, 这个系统还允许物理硬件位于任何数据中心甚至世界上的任何角落, 在这样的环境中, 开发人员不需要关心自己所操控的数据所在的物理位置。

1.3.4 案例研究: 亚马逊的 Dynamo——每天 24 小时接收订单

Google 的工作主要关注如何使分布式的批处理和报表生成更简单, 而没有考虑对高度可伸缩的 Web 店面每天 24 小时运行需求的支持。在这方面, 亚马逊考虑到了。亚马逊发表了另一篇的标志性论文《Amazon's 2007 Dynamo: A Highly Available Key-Value Store》。Dynamo 背后的业务驱动是亚马逊需要创建一个高可用的 Web 店面来支持来自世界各地每天 24 小时不间断的交易。

在几个不同地点经营的传统实体零售商的优势在于他们的收银机和销售设备只需要在营业时间运行, 而在非营业时间可以做日常报告、备份和软件升级。亚马逊的运营模式与之不同, 亚马逊的客户来自世界各地, 每时每刻都有人在网站上购物。在购买周期内, 任何的宕机都可能带来数百万美元的损失, 所以亚马逊的系统在服务过程中需要用铁一般的可靠性和可伸缩性来确保零损失。

在 Dynamo 刚开始应用时, 亚马逊使用关系型数据库来支持它的购物车系统和结账系统。他们拥有 RDBMS 软件的无限许可和充足的咨询预算, 允许他们为项目雇用最好的和最精明的顾问。尽管有着充足的预算和权力, 但他们最终还是意识到仅靠关系模型无法满足未来的业务需求。

NoSQL 社区里面的很多人都把亚马逊的 Dynamo 白皮书视为 NoSQL 运动的重要转折点。在关系模型还广泛使用的年代, NoSQL 挑战了当时的现状和当时的最佳应用。亚马逊发现, 键值存储接口简单, 更易于复制数据且更加可靠。最终, 亚马逊用键值存储的形式构建了一个可靠的、可扩展的, 并且可以支持每天 24 小时运行的商业模式的一站式服务系统, 这使得亚马逊成为世界上最成功的网络零售商之一。

1.3.5 案例研究: MarkLogic

2001 年, 一群住在旧金山湾区附近富有文档搜索经验的工程师们成立了一家专注于处理海量 XML 文档的公司。由于 XML 文档包含标记 (markup), 所以他们将公司命名为 MarkLogic。

MarkLogic 定义了两类类型的集群节点: 查询和文档节点。查询节点接收查询请求并协调与执行查询相关的所有活动。文档节点包含 XML 文档, 并负责在本地文件系统中执行查询。

查询请求被发送到一个查询节点后, 即被分发到每个包含 XML 文档索引的远端服务器。所有符合要求的文档会被返回至查询节点。当所有文档节点完成响应后, 查询结果即被返回。

MarkLogic 的架构是将查询任务移动至文档而不是将文档移动至查询服务器, 这样的架构对千兆字节的文档具有线性可扩展性。

MarkLogic 发现他们在美国联邦政府系统中的产品有一个需求, 即 TB 级的智库信息以及大型出版物需要存储和搜索它们的 XML 文档。自 2001 年以来, MarkLogic 已经发展成为成熟的、

通用的、高度可扩展的文档存储并对 ACID 事务和细粒度的、基于角色的访问控制提供支持。最初，MarkLogic 开发者使用的主要语言是 XQuery 与 REST 的组合，新版本支持 Java 和其他语言的编程接口。

MarkLogic 是一个商业产品，对于任何超过 40 GB 的数据集都需要软件许可。NoSQL 与商用产品和开源产品联系紧密，为业务问题提供了创新的解决方案。

1.3.6 实践

为了展示这些概念在这本书中如何应用，我们将为您介绍 Sally 的解决方案。Sally 在一个有许多业务部门的大型组织中担任解决方案架构师。解决方案架构师帮助存在信息管理问题的业务部门选择最好的解决方案来应对这些信息挑战。Sally 致力于解决需要定制开发的应用程序项目，她对于 SQL 和 NoSQL 技术有着深入的了解。Sally 的工作就是为业务问题寻求最适合的解决方案。

现在让我们通过两个案例来看看 Sally 是如何应用她的知识的。在第一个案例中，一个需要跟踪硬件采购的设备授权信息的团队来寻求 Sally 的建议。由于在 RDBMS 中已经保存了硬件的信息，并且这个团队在 SQL 方面很有经验，Sally 推荐他们扩展 RDBMS 以包含授权信息并使用连接操作创建报表。在这种情况下，SQL 就很合适了。

在第二个示例中，一个负责在关系型数据库中存储数字图片信息的团队找到 Sally。他们遇到的问题是数据库性能正对他们的 Web 应用页面产生负面影响。在这种情况下，Sally 推荐他们将所有图片移至键值对形式的存储系统，用一个 URL 代表一个图片。键值存储对读密集型应用程序做了优化并能与内容分发网络相协作。当我们把图片管理负载从 RDBMS 迁出之后，Web 应用程序以及其他应用程序的性能都得到了改善。

请注意，Sally 没有将她的工作简单地按照非此即彼的方式，即不是 RDBMS 就是 NoSQL 的方式进行选择。有时最好的解决方案是兼收并蓄的。

14 小结

本章首先介绍了 NoSQL 的概念，回顾了 NoSQL 运动背后的核心业务驱动力。然后展示了性能瓶颈如何迫使系统设计师使用高度并行处理设计，用创新的思维来管理数据。你也可以了解到，使用对象-中间层和 RDBMS 数据库的传统系统需要使用复杂的对象-关系映射系统来操作数据。这些层通常会阻碍组织对变化作出快速反应的能力（敏捷性）。

任何一项新的技术都是有风险的，至关重要的是要理解每个领域都有自己解决问题的模式，这些模式所使用的技术是明显不同的。从 SQL 过渡到 NoSQL 也不例外。NoSQL 是一种新的范型，需要一系列新的模式识别的能力、新的思维方式和新的解决方案。也就是说它需要我们具备一种新的认知风格。

选择使用 NoSQL 技术可以帮助企业在他们所处的市场中获得竞争优势，使他们更敏捷、更好地适应不断变化的商业环境。NoSQL 可以利用大量的商用处理器为公司节省时间和金钱，并提高服务的可靠性。

正如在案例研究中看到的，这些变化带来的影响比早期的技术用户带来的影响还要大：使世界各地的工程师认识到 RDBMS 并不是我们唯一的选择，它是可以被替代的。新的公司专注于新思维、新技术以及新架构的涌现不是由于一时兴起，而是缘于解决那些不适用于关系模型的真实业务问题的必要性。随着企业持续变化和进入经济全球化时代，这一趋势将继续扩大。

在下一章中，我们将开始讨论关于 NoSQL 的核心理念和技术。我们将讨论其简洁的设计，同时会构建一个模块化、可扩展以及低成本 of NoSQL 系统的基础。

第 2 章 NoSQL 概念

本章主要内容

- NoSQL 概念
- 对于可靠的数据库事务的 ACID 和 BASE
- 最小化由于数据库分区所造成的宕机时间
- Brewer 的 CAP 定理

少即是多。

——Ludwig Mies van der Rohe

在本章，我们将介绍一些 NoSQL 系统的核心概念。阅读完本章之后，你将有能力识别和定义 NoSQL 的概念和术语，了解 NoSQL 厂商的产品及其特性并且能够决定这些特性是否适合你的 NoSQL 系统。接下来，我们将讨论如何在应用开发过程中利用简单的组件来降低复杂性和促进重用，从而使你在系统设计和维护时节约时间、降低开销。

2.1 保持组件简单以促进重用

如果你和关系型数据库打交道，那么你应该知道它们是多么复杂。一开始，它们是一个简单的系统，当被请求时，从单一的平面文件返回一个被选择的行即可。随着时间的迁移，它们需要管理不止一张表、执行连接操作、对查询进行优化、复制事务、运行存储过程、设置触发器、保证安全性、维护索引等。对于这些复杂的问题，NoSQL 系统另辟蹊径，通过在网络中构建简单的分布式应用来满足不同维度的需求。保持架构级别的组件简单使得在不同的应用中可以重用这些组件，帮助开发人员理解和测试，而且应用的架构迁移也变得更加容易。

NoSQL 的观点认为，简单就是好的。构建一个应用时，并不需要在一个软件中包含所有的功能，应用的功能可以被分发到多个 NoSQL（或者 SQL）数据库上完成，这些系统由许多简单工具组成并具有简单的接口和清晰定义的功能。NoSQL 产品遵循的原则是做好几件事。为了说明这一点，我们来

- 4.3.4 案例研究：在Bigtable中存储分析信息
 - 4.3.5 案例研究：Google地图用Bigtable存储地理信息
 - 4.3.6 案例研究：使用列族存储用户偏好信息
 - 4.4 文档存储
 - 4.4.1 文档存储基础
 - 4.4.2 文档集合
 - 4.4.3 应用集合
 - 4.4.4 文档存储的API
 - 4.4.5 文档存储的实现
 - 4.4.6 案例研究：MongoDB和广告服务器
 - 4.4.7 案例研究：大型对象数据库CouchDB
 - 4.5 NoSQL架构模式的变体
 - 4.5.1 定制RAM和SSD存储
 - 4.5.2 分布式存储
 - 4.5.3 分组的对象
 - 4.6 小结
 - 4.7 延伸阅读
- 第5章 原生XML数据库
- 5.1 什么是原生XML数据库
 - 5.2 用原生XML数据库构建应用
 - 5.2.1 加载数据可以像拖曳那样简单
 - 5.2.2 使用集合来组织XML文档
 - 5.2.3 使用XPath运用简单的查询转换复杂的数据
 - 5.2.4 用XQuery转换数据
 - 5.2.5 用XQuery更新文档
 - 5.2.6 XQuery全文搜索标准
 - 5.3 在原生XML数据库中应用XML标准
 - 5.4 用XMLSchema和Schematron设计和验证数据
 - 5.4.1 XML Schema
 - 5.4.2 使用Schematron检查文档规则
 - 5.5 用自定义模块扩展XQuery
 - 5.6 案例研究：在美国国务院历史学家办公室使用NoSQL
 - 5.7 案例研究：使用MarkLogic管理金融衍生品
 - 5.7.1 为什么RDBMS难以存储金融衍生品
 - 5.7.2 一个投资银行从20个RDBMS转换到1个原生XML数据库
 - 5.7.3 迁移至原生XML文档存储的商业好处

5.7.4 项目成果

5.8 小结

5.9 延伸阅读

第三部 NoSQL解决方案

第6章 用NoSQL管理大数据

6.1 什么才是大数据解决方案

6.2 线性扩展数据中心

6.3 理解线性可扩展性和表现力

6.4 了解大数据问题的类型

6.5 使用无共享架构分析大数据

6.6 选择分布式模型：主从模型与对等模型

6.7 在分布式系统上使用MapReduce处理数据

6.7.1 MapReduce和分布式文件系统

6.7.2 MapReduce怎样做到高效处理大数据问题

6.8 NoSQL系统处理大数据问题的4种方式

6.8.1 分发查询到数据，而非数据到查询

6.8.2 使用散列环在集群中均匀分发数据

6.8.3 使用复制扩展读取性能

6.8.4 使数据库将查询均衡地分发到数据节点

6.9 案例研究：使用Apache Flume处理事件日志

6.9.1 事件日志数据分析的挑战

6.9.2 Apache Flume搜集分布式事件日志的方法

6.9.3 延伸思考

6.10 案例研究：计算机辅助发现医疗保险欺诈

6.10.1 什么是医疗保险欺诈检测

6.10.2 使用图和定制的内存共享硬件检测医疗保险欺诈

6.11 小结

6.12 延伸阅读

第7章 用NoSQL搜索获取信息

7.1 什么是NoSQL搜索

7.2 搜索分类

7.2.1 布尔搜索、全文关键字搜索和结构化搜索的比较

7.2.2 测试常见搜索类型

7.3 提高NoSQL搜索效率的策略和方法

7.4 使用文档结构提升搜索质量

7.5 搜索质量量化

- 7.6 本地索引与远程搜索服务
- 7.7 案例研究：使用MapReduce建立倒排索引
- 7.8 案例研究：搜索技术文档
 - 7.8.1 什么是技术文档搜索
 - 7.8.2 在NoSQL文档存储中保留文档结构
- 7.9 案例研究：搜索领域语言——可检索性和重用性
- 7.10 实践
- 7.11 小结
- 7.12 延伸阅读
- 第8章 用NoSQL构建高可用的解决方案
 - 8.1 高可用NoSQL数据库的定义
 - 8.2 量化NoSQL数据库的可用性
 - 8.2.1 案例研究：亚马逊S3的服务级别协议
 - 8.2.2 预测系统可用性
 - 8.2.3 实践
 - 8.3 NoSQL系统的高可用性策略
 - 8.3.1 使用负载均衡器将流量转向到最空闲的节点
 - 8.3.2 结合高可用分布式文件系统和NoSQL数据库
 - 8.3.3 案例研究：将HDFS作为一个高可用的文件系统存储主数据
 - 8.3.4 使用托管的NoSQL服务
 - 8.3.5 案例研究：使用亚马逊的DynamoDB作为高可用数据存储
 - 8.4 案例研究：使用Apache Cassandra作为高可用的列族存储
 - 8.5 案例研究：使用Couchbase作为高可用文档数据库
 - 8.6 小结
 - 8.7 延伸阅读
- 第9章 用NoSQL提升敏捷性
 - 9.1 软件敏捷性的定义
 - 9.2 量化敏捷性
 - 9.3 使用文档存储来避免对象关系映射
 - 9.4 案例研究：使用XRX管理复杂表单
 - 9.4.1 什么是复杂业务表单
 - 9.4.2 用XRX替换客户端JavaScript和对象关系映射
 - 9.4.3 理解XRX对敏捷性的影响
 - 9.5 小结
 - 9.6 延伸阅读
- 第10章 NoSQL与函数式编程

- 10.1 什么是函数式编程
 - 10.1.1 指令式编程就是管理程序状态
 - 10.1.2 函数式编程是没有副作用的并行转化
 - 10.1.3 比较指令式编程和函数式编程的扩展性
 - 10.1.4 使用引用透明避免重复计算
 - 10.2 案例研究：用NetKernel优化网页内容组装
 - 10.2.1 组装嵌套内容，追踪组件依赖
 - 10.2.2 用NetKernel优化组件再生成
 - 10.3 函数式编程语言示例
 - 10.4 完成指令式编程到函数式的编程转变
 - 10.4.1 使用函数作为函数的参数
 - 10.4.2 使用递归处理非结构化文档数据
 - 10.4.3 使用不可变变量而非可变变量
 - 10.4.4 去除循环和条件语句
 - 10.4.5 新的思维方式：从状态记录到转化隔离
 - 10.4.6 质量、校验和一致性单元测试
 - 10.4.7 函数式编程的并发
 - 10.5 案例研究：用Erlang构建NoSQL系统
 - 10.6 实践
 - 10.7 小结
 - 10.8 延伸阅读
- 第11章 安全：保护NoSQL系统中的数据
- 11.1 NoSQL数据库的一种安全模型
 - 11.1.1 使用服务减少数据库内部的安全性需求
 - 11.1.2 使用数据仓库和OLAP减少数据库内部的安全性需求
 - 11.1.3 应用级安全措施和数据库级安全措施的收益总结
 - 11.2 收集安全需求
 - 11.2.1 认证
 - 11.2.2 授权
 - 11.2.3 审查和日志记录
 - 11.2.4 加密和数字签名
 - 11.2.5 保护公开网站免受拒绝服务攻击和注入攻击
 - 11.3 案例研究：键值存储的访问控制——亚马逊的S3
 - 11.3.1 身份和访问管理（IAM）
 - 11.3.2 访问控制列表（ACL）
 - 11.3.3 桶策略

- 11.4 案例研究：在Apache Accumulo中使用键可见性技术
- 11.5 案例研究：在安全发布流程中使用MarkLogic的RBAC模型
 - 11.5.1 使用MarkLogic的RBAC安全模型保护文档
 - 11.5.2 在安全出版中使用 MarkLogic
 - 11.5.3 MarkLogic的安全模型的优势
- 11.6 小结
- 11.7 延伸阅读
- 第12章 选择合适的NoSQL解决方案
 - 12.1 什么是架构利弊分析
 - 12.2 数据库架构选型团队的组成变化
 - 12.2.1 选择合适的团队
 - 12.2.2 考虑经验偏好
 - 12.2.3 雇用外部咨询师
 - 12.3 架构权衡分析步骤
 - 12.4 解构分析：质量树
 - 12.4.1 质量属性样例
 - 12.4.2 评估混合架构和云架构
 - 12.5 与利益系相关者沟通结果
 - 12.5.1 用质量树作为导航图
 - 12.5.2 实践
 - 12.5.3 使用质量树进行项目风险交流
 - 12.6 找到合适的验证架构的试点项目
 - 12.7 小结
 - 12.8 延伸阅读

封底