

The  
Pragmatic  
Programmers

现代数据库和  
NoSQL运动指南

# 七周七数据库

## Seven Databases in Seven Weeks

A Guide to Modern Databases  
and the NoSQL Movement

[美] Eric Redmond Jim R. Wilson 著

王海鹏 田思源 王晨 译



 人民邮电出版社  
POSTS & TELECOM PRESS

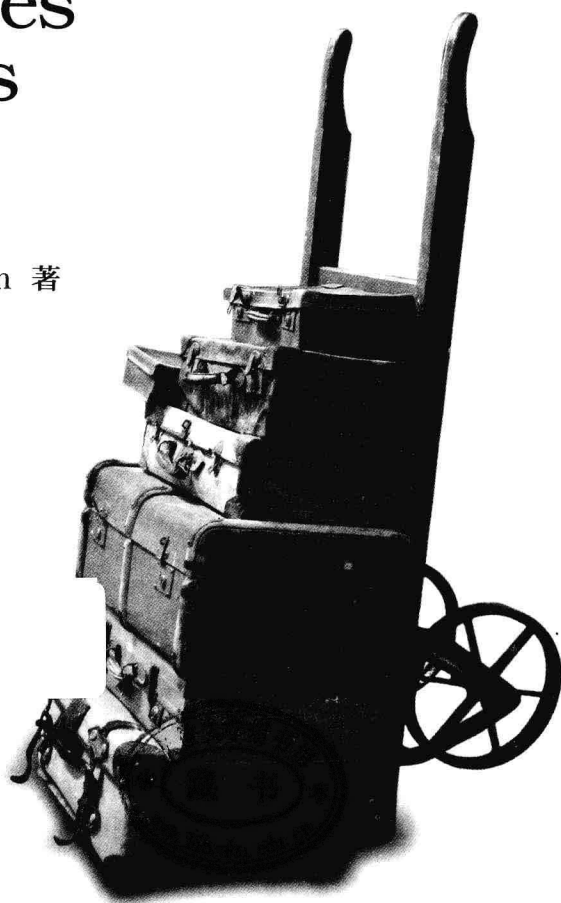
# 七周七数据库

## Seven Databases in Seven Weeks

A Guide to Modern Databases  
and the NoSQL Movement

[美] Eric Redmond Jim R. Wilson 著

王海鹏 田思源 王晨 译



人民邮电出版社  
北京

## 图书在版编目 (C I P) 数据

七周七数据库 / (美) 雷德蒙 (Redmond, E.), (美) 威尔逊 (Wilson, J. R.) 著; 王海鹏, 田思源, 王晨译  
— 北京: 人民邮电出版社, 2013. 7  
ISBN 978-7-115-31224-2

I. ①七… II. ①雷… ②威… ③王… ④田… ⑤王…  
… III. ①数据库系统 IV. ①TP311.13

中国版本图书馆CIP数据核字(2013)第042507号

## 版权声明

Simplified Chinese-language edition Copyright © 2013 by Posts & Telecom Press. All rights reserved.

Copyright © 2012 The Pragmatic Programmers, LLC. Original English language edition, entitled Seven Databases in Seven Weeks.

本书中文简体字版由 The Pragmatic Programmers, LLC 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] Eric Redmond  
Jim R. Wilson  
译 王海鹏 田思源 王 晨  
责任编辑 陈冀康  
责任印制 程彦红 杨林杰
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号  
邮编 100061 电子邮件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京鑫正大印刷有限公司印刷
- ◆ 开本: 800×1000 1/16  
印张: 20.5  
字数: 461 千字 2013年7月第1版  
印数: 1-3 500 册 2013年7月北京第1次印刷

---

著作权合同登记号 图字: 01-2012-4612 号

定价: 59.00 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223  
反盗版热线: (010)67171154

# 内容提要

如今，我们要面对和使用的数据正在变得越来越庞大和复杂。如果说数据是新的石油。那么数据库就是油田、炼油厂、钻井和油泵。作为一名现代的软件开发者，我们需要了解数据管理的新领域，既包括 RDBMS，也包括 NoSQL。

本书遵循《七周七语言》的写作风格和体例，带领你学习和了解当令最热门的开源数据库。在简单的介绍之后，本书分章介绍了 7 种数据库。这些数据库分别属于 5 种不同的数据库风格，但每种数据库都有自己保存数据和看待世界的方式。它们依次是 PostgreSQL、Riak、Apache HBase、MongoDB、Apache CouchDB、Neo4J 和 Redis。本书将深入每一种数据库，介绍它们的优势和不足，以及如何选取一种最符合你的应用需求的数据库。

本书适合数据库架构师、数据库管理员，以及想要了解和学习各种 NoSQL 数据库技术的程序员阅读。本书将帮助读者了解、选择和应用这些数据库，从而更好地发挥日益增长的大数据的能力。

# 译者简介

王海鹏，1994年毕业于华东师范大学。软件开发者，独立的咨询顾问、培训讲师、译者。拥有20年编程经验，已翻译二十余本软件开发书籍。目前主要感兴趣的领域是软件架构和方法学，致力于提高软件开发的品质和效率。新浪微博：@王海鹏 Seal。

田思源，资深IT人士，从事软件、互联网行业多年，有多部译作面世。现居北京。新浪微博：@胡试之。

王晨，IBM中国系统与科技研发中心的软件工程师，从事硬件集成管理系统的开发测试工作。新浪微博：@wwwwwwch。

特别感谢李元佳（第2章）、谢磊（第4章）、程显峰（第5章）、李子骅（第8章）参与审阅译稿。

# 序

在科罗拉多州布雷肯里奇（Breckenridge）的滑雪季里，登上 Beaver 雪道运行超级缆车，一眼望去，滑雪道的斜坡被精心地修整过，而山上的植被和覆盖的雪层却依然如旧。我们在心里发问，新雪在哪里呢？没有新雪，滑雪的体验就不那么美妙了。

1994 年，作为 IBM 在奥斯汀的数据库开发实验室工作的雇员，我的感觉非常类似。当时我刚在奥斯汀的德克萨斯大学学习了面向对象数据库，因为在关系数据库主宰了 10 年之后，我想面向对象数据库真的有机会深入人心。但是，接下来的 10 年和之前一样，出现了更多同样的关系模型。我沮丧地关注着 Oracle、IBM 和其他以 MySQL 为首的开源解决方案，它们强劲伸展着的枝叶，完全挡住了阳光，妨碍了肥沃的土壤上正在萌芽的其他解决方案。

随着时间的推移，用户界面从绿屏幕变成了客户端-服务器的方式，又变成了基于互联网的应用，但关系层的代码同样张开无情的铁幕，几十年如一日地称职而单调。所以，我们期待着一场新雪。

然后新雪终于降临了。起初，雪花甚至不足以掩盖早行者的足迹，但暴雪随后到来，覆盖大地，带来了完美的滑雪体验，这正是我们渴望的不同和品质。在过去的一年里，我醒过来时发现，数据库的世界也覆盖了一层新雪。当然，关系数据库还在，你可以从开源 RDBMS 软件中获得令人吃惊的丰富体验。你可以创建集群，进行全文搜索，甚至进行模糊搜索。但你不再受限于某种方式。一年里我没有创建过一个完整的关系型解决方案。在这段时间里，我使用了一个文档数据库和几种键-值数据库。

真实情况是，关系数据库在灵活性和可伸缩性方面不再处于垄断地位。对于我们要构建的各类应用，还有更多合适的模型，更简单、更快速、更可靠。作为在 IBM 的奥斯汀实验室待了 10 年，与同事和客户从事数据库工作的人，我被这种进步惊呆了。在本书中，你会看到一些例子，完美地覆盖了数据库领域最重要的进展，正是这些数据库支撑了互联网的发展。在键-值存储库中，你会看到伸缩性极好、极为可靠的 Riak，还会看到 Redis 中漂亮的查询机制。在列型数据库社区，你将体验到 HBase 的威力，它是关系数据库模型的近亲。在文档数据库中，你会看到伸缩性极好的、优雅的方案，处理深层嵌套的文档。你还会看到 Neo4j 在图形数据库上的应用，支持快速地在关系上导航。

要成为更好的程序员或数据库管理员，你不必使用所有这些数据库。随着 Eric Redmond

和 Jim Wilson 引导你走过的这段神奇的旅程，每一步都会让你更聪明，所获得的深刻见解对于软件职业来说是无价的。你会知道每种平台的闪光之处和最大局限。你会看到行业的发展方向，理解行业的驱动力。

享受这段旅程吧。

Bruce Tate, 《七周七语言》的作者

2012 年 2 月

德克萨斯州奥斯汀

# 作者访谈

**Q: 你们怎么选择这七种数据库的?**

Eric: 我们确实有一些选择标准,但没有明确列出来。这些数据库必须是开源的,因为我们不想介绍让读者绑定某公司的数据库。对于 5 种数据库类型(关系型、键-值对型、列型、文档型、图型),每种至少需要一个实现。然后我们选择一些数据库,它们能够用实例展示我们想介绍的一些一般概念,如 CAP 原理或 MapReduce。最后,我们选择一些彼此是很好的竞争对手的数据库。所以我们选择了 MongoDB 和 CouchDB(二者实现文档数据库的不同方式)。我们选择 Riak 是因为它是 Dynamo(亚马逊的数据库)的一种实现,可以与 HBase 进行比较,而后者是 BigTable(谷歌的数据库)的一种实现。

Jim: 我们这本书的主要目标是向读者介绍现有的选择。我们的选择基本上服务于这个目标。即便如此,这也是一个相当长的迭代过程。我们知道,不论我们选了哪些数据库,都会有人问,为什么我们选择或没选择他们喜欢的产品。这首先取决于我们想讨论的数据库类型,然后选择的数据库既要代表某个类型,又要相对比较受欢迎。

例如,我们选择了 PostgreSQL,因为它几乎严格地遵守 SQL 标准,同时又不如 MySQL 这样的开源竞争产品那样知名。类似地,虽然 HBase 和 Cassandra 都是面向列的数据库,但我们选择了 HBase,因为 Cassandra 是混合类型,它同时包含了来自 BigTable 论文和 Dynamo 论文的思想。

**Q: 数据库正在快速变化。现在你们希望当初选了哪些?**

Eric: 有几百种数据库可以选择,但我们很高兴地看到,一年之后,我们的选择仍然在变得更强。但是,如果再来一次,我会加入 Triplestore(如 Mulgara),因为语义网正让这种数据存储方法逐渐变得热门。我也会在 Neo4j 的 Cypher 语言上花更多的时间,或更详细地介绍 Hadoop,因为分析是数据存储的一大部分工作。

Jim: 是的,数据库在快速地变化,这体现在两个方面。首先,可用数据存储技术的领域在近年来有了爆炸式增长。越来越多的不同数据库正在涌现,填补不同的小众需求。在另一方面,各种数据库本身也在快速发展。即使在小版本之间,现代的 NoSQL 数据库也在加入越来越多的特征,目的是占领更多的市场,保持竞争力。因此,也产生了一些趋同效应,这使得选择更为困难,因为更多的产品可以满足你的需求。

我还是认为，我们选择的 5 种类型和 7 种数据库满足了我们设定的条件。但我还想写其他的数据库。包括一些长期受欢迎的数据库，如 SQLite，以及一些你可能想不到的数据库，如 OpenLDAP 或 SOLR（一个倒排索引/查找引擎）。

**Q：为什么你们决定要写这本书？**

Eric: Jim 和我很早就在讨论写这样一本书了。大约一年半以前，他发了一封只有标题的电子邮件：“七周七数据库？”这个标题打动了我。我们都喜欢 Bruce 的《七周七语言》，这似乎是探讨这个新兴领域的完美方式。

Jim: 早在 2010 年 3 月，Eric 和我就对写本 NoSQL 的书进行了探讨。那时候，关于这个术语有许多讨论，但也有许多迷惑。我们认为我们可以为讨论带来某种结构，并将所有最新进展告诉不那么了解最新情况的人。

在读了 Bruce A. Tate 的《七周七语言》之后，我想，“七种数据库怎么样？”Eric 提交了一份建议，几周后我们就开始动手写了。

**Q：你对目前和将来的数据库怎么看？**

Eric: 我是 Neo4j 的粉丝。我们在本书中介绍了它，但老实说，我们选它是因为想探讨一个开源的图数据库。但在过去的一年里，它确实非常成功。我相信，今年会有更多的人采用图数据库。

在我们没有介绍的数据库中，我认为 ElasticSearch 很清楚获得了支持。OrientDB 也很有趣，因为它可以作为关系型、键值对型、文档型或图数据库。我认为将来你会看到更多这样的多类型表现。就像我们前面提到的，Triplestores 也得到一些支持，虽然他们的问题集与一般的图型数据库有很大的交集。

Jim: 当然，有很多数据库，但至少有两种是我个人希望仔细研究的：ElasticSearch 和 doozer。

ElasticSearch 是分布式的、对等的、支持 REST/JSON 的文档搜索引擎。ElasticSearch 采用分布式的 Lucene 索引作为核心，允许 REST 客户端根据模糊的条件查找文档。每个人都需要一个搜索引擎，ElasticSearch 让这一点变得容易了。

Doozer 是一个快速的、无头一致（headless consensus）的引擎。它是 Heroku 的一群聪明人用 Go 语言写的。Doozer 对于存放小块的重要信息是很好的，这些信息绝对需要一致（如集群的配置元数据），但没有单点失效。

# 前言

如果说数据是新的石油，那么数据库就是油田、炼油厂、钻井和油泵。数据存放在数据库中，如果你有兴趣利用它，那么掌握相应的现代化的工具就是好的开始。

数据库是工具，它们是到达终点的手段。每种数据库都有自己保存数据和看待世界的方式。你对它们的理解越多，就越能随心所欲，在日益增长的大数据上，就能更好地利用它们潜在的能力。

## 为什么是 7 种数据库

早在 2010 年 3 月，我们就想写本关于 NoSQL 的书。NoSQL 这个术语已经聚集了人气，虽然许多人都在谈论，但关于它似乎也存在相当多的困惑。NoSQL 到底意味着什么？包含哪些类型的系统？对于开发优秀的软件，它将产生怎样的影响？这些就是我们想要回答的问题，既是为我们自己，也是为别人。

在读了 Bruce Tate 的典范性的著作《七周七语言》后，我们知道他做得很对。他循序渐进地介绍语言的方式引起了我们的共鸣。我们觉得用同样的方式讲授数据库，将会提供一个很好的环境，回答这些棘手的 NoSQL 问题。

## 本书内容

本书针对的是有经验的开发者，他们希望全面地理解现代数据库的整体情况。本书不要求读者以前在数据库方面有经验，但有数据库经验会有助于学习本书。

在简单的介绍之后，本书分章介绍了 7 个数据库。这些数据库分属 5 种不同的数据库类型或风格，这在第 1 章中有介绍。它们依次是 PostgreSQL、Riak、Apache HBase、MongoDB、Apache CouchDB、Neo4j 和 Redis。

每章都设计成一个长周末的学习量，分为三天。每天结束时都有一些练习，扩展刚刚介绍的主题和概念。每章最后都有一段总结性的讨论，总结了这种数据库的优点和缺点。你既可以学得快一点，也可以学得慢一点，但重要的是先掌握每天的概念，再继续后面的学习。我们试着设计了一些例子来探索每种数据库的独特之处。要真正理解这些数据库提

供的能力，必须花些时间来使用它们，这意味着要动手实践。

虽然你可能想跳过某些章，但我们设想你是按章节顺序阅读这本书的。某些概念，如映射-归约（mapreduce），在前面的章节中深入地进行了介绍，所以在后面的章节中就略过了。本书旨在实现对现代数据库的一致理解，所以建议你完整地阅读本书。

## 不包含的内容

在阅读本书之前，你应该知道它不包含哪些内容。

## 本书不是安装指南

安装本书中提到的数据库有时候容易，有时候有些挑战，有时候非常棘手。对于某些数据库，可以使用提供的安装包；而对于另一些数据库，需要编译源代码。我们会不时提供一些有用的提示，但主要还是靠你自己。省略安装步骤让我们能安排更多有用的例子和概念讨论，这才是你真正想要的，对吗？

## 本书也不是管理手册

出于对安装同样的考虑，本书也不会介绍管理手册里的所有内容。每种数据库都有大量的选项、设置、开关和配置细节，绝大部分都能在 Web 上找到详尽的文档。我们更关心介绍有用的概念，完全深入进去，而不是仅关注日常操作。虽然数据库的一些特点会根据操作设置而改变（我们可能会讨论这些特点），但由于篇幅有限，我们不可能介绍所有可能配置的全部具体细节。

## 对 Windows 用户的说明

本书本身就讨论选择，主要是针对\*nix 平台上的开源软件。微软的环境作为集成环境有点困难，它限制了许多选择，只留下一个较小的、预定义的子集。因此，我们介绍的数据库是开源的，由\*nix 系统的用户开发（也主要为他们服务）。这不是我们的偏见，只是当前真实情况的反映。所以，我们假定教程式的例子运行在\*nix 的 shell 下。如果你运行 Windows 并希望给它一个尝试的机会，我们推荐安装 Cygwin<sup>1</sup>，这样更容易成功。你也可以考虑运行一个 Linux 虚拟机。

---

<sup>1</sup> <http://www.cygwin.com/>

## 代码示例和惯例

本书包含各种语言的代码。部分原因是因为我们介绍的这些数据库本身使用语言不同。我们曾试着将语言局限在 Ruby/JRuby 和 JavaScript。我们更喜欢命令行工具，而不是脚本语言，但我们会引入其他一些语言来完成工作，如 PL/pgSQL (Postgres) 和 Gremlin/Groovy (Neo4j)。我们也会尝试使用 Node.js，编写一些服务器端的 JavaScript 应用。

除非特别注明，代码清单都是完整的，通常可以直接执行。根据所涉及的语言规则，突出了示例和代码片段中的语法。shell 命令以\$开始。

## 在线资源

本书的 Pragmatic Bookshelf 页面<sup>1</sup>是很好的资源。从上面可以下载本书中的所有源代码。你也会找到一些反馈工具，如社区论坛和勘误提交，你可以通过它们对本书未来的版本提出改进建议。

感谢你陪伴我们完成现代数据库的观光之旅。

Eric Redmond 和 Jim R. Wilson

---

<sup>1</sup> <http://pragprog.com/book/rwdata/seven-databases-in-seven-weeks>

# 致 谢

本书内容涉及范围都比较宽泛，只靠两位作者是无法完成的。它需要许多非常聪明的人的努力，他们有超人般的眼睛，能尽可能多地发现错误，针对这些技术的细节提供有价值的见解。

我们要感谢所有贡献出时间和专业知识的人（不分先后）：Ian Dees、Mark Phillips、Jan Lenhardt、Robert Stam、Oleg Bartunov、Dave Purrington、Daniel Bretoi、Matt Adams、Sean Copenhaver、Loren Sands-Ramshaw、Emil Eifrem 和 Andreas Kollegger。最后，还要感谢 Bruce Tate 的经验和指导。

我们还要真诚地感谢 Pragmatic Bookshelf 的整个团队。感谢他们提出这个大胆的项目，并看着我们完成。我们特别要感谢编辑 Jackie Carter。是你的耐心反馈，才有了今天这本书。感谢整个团队辛勤的工作，找出了我们所有的错误。

最后但同样重要的是，感谢 Frederic Dumont、Matthew Flower、Rebecca Skinner 和所有严格的读者，如果没有你们的学习热情，我们就不会有机会提供服务了。

对于这里遗漏的人，我们希望你接受道歉。我们肯定不是有意的。

Eric 想说：亲爱的 Neolle，你不是特别的，你是唯一的，这要好得多。谢谢你又忍受我完成了一本书。感谢数据库的创建者和贡献者，为我们提供了写书的内容和谋生的工具。

Jim 想说：首先，我要感谢我的家庭。Ruthy，你无限的耐心和鼓励温暖了我的心。Emma 和 Jimmy，你们是两个聪明鬼，爸爸永远爱你们。还要感谢所有无名英雄，他们盯着 IRC、消息板、邮件列表和 bug 管理系统，时刻准备帮助需要的人。你们对开源的贡献让这些项目一直激动人心。

# 目 录

|                        |    |                          |    |
|------------------------|----|--------------------------|----|
| 第 1 章 概述               | 1  | 2.3.7 视图                 | 27 |
| 1.1 从一个问题开始            | 1  | 2.3.8 规则是什么              | 28 |
| 1.2 风格                 | 2  | 2.3.9 联表分析               | 30 |
| 1.2.1 关系数据库            | 3  | 2.3.10 第 2 天总结           | 32 |
| 1.2.2 键-值数据库           | 3  | 2.4 第 3 天: 全文检索和多维查询     | 32 |
| 1.2.3 列型数据库            | 4  | 2.4.1 模糊搜索               | 34 |
| 1.2.4 文档型数据库           | 5  | 2.4.2 SQL 标准的字符串匹配       | 34 |
| 1.2.5 图数据库             | 5  | 2.4.3 字符串相似比较算法          |    |
| 1.2.6 混合使用多种数据库        | 6  | levenshtein              | 35 |
| 1.3 前进和提升              | 6  | 2.4.4 三连词                | 36 |
| 第 2 章 PostgreSQL       | 7  | 2.4.5 全文检索               | 37 |
| 2.1 这就是 Post-greS-Q-L  | 7  | 2.4.6 组合使用字符串匹配方法        | 42 |
| 2.2 第 1 天: 关系、CRUD 和联接 | 8  | 2.4.7 把电影风格表示成多维超立方体     | 42 |
| 2.2.1 从 SQL 开始         | 10 | 2.4.8 第 3 天总结            | 45 |
| 2.2.2 使用表              | 10 | 2.5 总结                   | 46 |
| 2.2.3 使用联接的查询          | 13 | 2.5.1 PostgreSQL 的优点     | 46 |
| 2.2.4 外联接              | 15 | 2.5.2 PostgreSQL 的缺点     | 47 |
| 2.2.5 使用索引快速查找         | 16 | 2.5.3 结束语                | 47 |
| 2.2.6 第 1 天总结          | 18 | 第 3 章 Riak               | 48 |
| 2.2.7 第 1 天作业          | 18 | 3.1 Riak 喜欢 Web          | 48 |
| 2.3 第 2 天: 高级查询、代码和规则  | 19 | 3.2 第 1 天: CRUD、链接和 MIME | 49 |
| 2.3.1 聚合函数             | 19 | 3.2.1 REST 是最棒的          |    |
| 2.3.2 分组               | 20 | (或用 cURL 时)              | 51 |
| 2.3.3 窗口函数             | 21 | 3.2.2 将值放于桶中             | 52 |
| 2.3.4 事务               | 22 | 3.2.3 链接                 | 54 |
| 2.3.5 存储过程             | 24 | 3.2.4 Riak 的 MIME 类型     | 58 |
| 2.3.6 触发器              | 26 | 3.2.5 第 1 天总结            | 58 |
|                        |    | 3.3 第 2 天: Mapreduce 和服务 |    |

|                               |     |                                       |     |
|-------------------------------|-----|---------------------------------------|-----|
| 集群 .....                      | 59  | 4.3.10 运行脚本 .....                     | 114 |
| 3.3.1 填充脚本 .....              | 59  | 4.3.11 检查输出 .....                     | 114 |
| 3.3.2 mapreduce 介绍 .....      | 60  | 4.3.12 第 2 天总结 .....                  | 116 |
| 3.3.3 Riak 中的 mapreduce ..... | 63  | 4.4 第 3 天: 放入云端 .....                 | 117 |
| 3.3.4 关于一致性和持久性 .....         | 69  | 4.4.1 开发 Thrift 协议的 HBase<br>应用 ..... | 117 |
| 3.3.5 第 2 天总结 .....           | 75  | 4.4.2 Whirr 简介 .....                  | 121 |
| 3.4 第 3 天: 解决冲突和扩展 Riak ..... | 76  | 4.4.3 设置 EC2 .....                    | 121 |
| 3.4.1 以向量时钟解决冲突 .....         | 76  | 4.4.4 准备 Whirr .....                  | 122 |
| 3.4.2 扩展 Riak .....           | 83  | 4.4.5 配置集群 .....                      | 122 |
| 3.4.3 第 3 天总结 .....           | 87  | 4.4.6 启动集群 .....                      | 123 |
| 3.5 总结 .....                  | 87  | 4.4.7 连接集群 .....                      | 124 |
| 3.5.1 Riak 的优点 .....          | 88  | 4.4.8 销毁集群 .....                      | 125 |
| 3.5.2 Riak 的缺点 .....          | 88  | 4.4.9 第 3 天总结 .....                   | 125 |
| 3.5.3 Riak 之于 CAP .....       | 88  | 4.5 总结 .....                          | 126 |
| 3.5.4 结束语 .....               | 89  | 4.5.1 HBase 的优点 .....                 | 126 |
| 第 4 章 HBase .....             | 90  | 4.5.2 HBase 的缺点 .....                 | 127 |
| 4.1 介绍 HBase .....            | 91  | 4.5.3 HBase on CAP .....              | 127 |
| 4.2 第 1 天: CRUD 和表管理 .....    | 91  | 4.5.4 结束语 .....                       | 128 |
| 4.2.1 配置 HBase .....          | 92  | 第 5 章 MongoDB .....                   | 129 |
| 4.2.2 HBase 的 shell .....     | 93  | 5.1 其大无比 .....                        | 129 |
| 4.2.3 创建表 .....               | 93  | 5.2 第 1 天: CRUD 和嵌套 .....             | 130 |
| 4.2.4 插入、更新和读取数据 .....        | 95  | 5.2.1 命令行的乐趣 .....                    | 131 |
| 4.2.5 修改表 .....               | 96  | 5.2.2 Mongo 的更多有趣内容 .....             | 134 |
| 4.2.6 通过编程方式添加数据 .....        | 99  | 5.2.3 深入挖掘 .....                      | 136 |
| 4.2.7 第 1 天总结 .....           | 100 | 5.2.4 更新 .....                        | 140 |
| 4.3 第 2 天: 处理大数据 .....        | 101 | 5.2.5 引用 .....                        | 142 |
| 4.3.1 导入数据, 调用脚本 .....        | 101 | 5.2.6 删除 .....                        | 143 |
| 4.3.2 流式 XML .....            | 102 | 5.2.7 用代码来读取 .....                    | 144 |
| 4.3.3 流式维基百科 .....            | 103 | 5.2.8 第 1 天总结 .....                   | 145 |
| 4.3.4 压缩和 Bloom 过滤器 .....     | 105 | 5.3 第 2 天: 索引、分组和<br>mapreduce .....  | 146 |
| 4.3.5 开始 .....                | 106 | 5.3.1 索引: 如果还不够快 .....                | 146 |
| 4.3.6 区域和监控磁盘使用简介 .....       | 107 | 5.3.2 聚合查询 .....                      | 150 |
| 4.3.7 区域的问讯 .....             | 108 | 5.3.3 服务器端命令 .....                    | 152 |
| 4.3.8 扫描一个表来建立另一个表 .....      | 111 |                                       |     |
| 4.3.9 构建扫描程序 .....            | 112 |                                       |     |

|       |                                         |     |               |                                         |
|-------|-----------------------------------------|-----|---------------|-----------------------------------------|
| 5.3.4 | mapreduce (以及 Finalize) .....           | 155 | CouchDB ..... | 187                                     |
| 5.3.5 | 第 2 天总结 .....                           | 159 | 6.3.8         | 第 2 天总结 .....                           |
| 5.4   | 第 3 天: 副本集、分片、地理<br>空间和 GridFS .....    | 159 | 6.4           | 第 3 天: 进阶视图、Changes API<br>以及复制数据 ..... |
| 5.4.1 | 副本集 .....                               | 159 | 6.4.1         | 用规约器创建进阶视图 .....                        |
| 5.4.2 | 偶数节点的问题 .....                           | 162 | 6.4.2         | 规约器调用详解 .....                           |
| 5.4.3 | 分片 .....                                | 163 | 6.4.3         | 监控 CouchDB 的变化 .....                    |
| 5.4.4 | 地理空间查询 .....                            | 165 | 6.4.4         | 连续监控变化 .....                            |
| 5.4.5 | GridFS .....                            | 167 | 6.4.5         | 在 CouchDB 中复制数据 .....                   |
| 5.4.6 | 第 3 天总结 .....                           | 168 | 6.4.6         | 第 3 天总结 .....                           |
| 5.5   | 总结 .....                                | 168 | 6.5           | 总结 .....                                |
| 5.5.1 | Mongo 的优点 .....                         | 168 | 6.5.1         | CouchDB 的优点 .....                       |
| 5.5.2 | Mongo 的缺点 .....                         | 169 | 6.5.2         | CouchDB 的缺点 .....                       |
| 5.5.3 | 结束语 .....                               | 169 | 6.5.3         | 结束语 .....                               |
| 第 6 章 | CouchDB .....                           | 170 | 第 7 章         | Neo4j .....                             |
| 6.1   | 在沙发上放松 .....                            | 170 | 7.1           | Neo4j, 白板友好的数据库 .....                   |
| 6.2   | 第 1 天: CRUD、Futon 与<br>cURL Redux ..... | 171 | 7.2           | 第 1 天: 图、Groovy 和<br>CRUD .....         |
| 6.2.1 | 享受 Futon .....                          | 171 | 7.2.1         | Neo4j 之 Web 接口 .....                    |
| 6.2.2 | 用 cURL 执行基于 REST 的<br>CRUD 操作 .....     | 175 | 7.2.2         | 通过 Gremlin 操作 Neo4j .....               |
| 6.2.3 | 用 GET 读取文档 .....                        | 175 | 7.2.3         | pipe 的威力 .....                          |
| 6.2.4 | 用 POST 新建文档 .....                       | 176 | 7.2.4         | Pipeline 与顶点 .....                      |
| 6.2.5 | 用 PUT 更新文档 .....                        | 177 | 7.2.5         | 无模式的社会性数据 .....                         |
| 6.2.6 | 用 DELETE 移除文档 .....                     | 178 | 7.2.6         | 垫脚石 .....                               |
| 6.2.7 | 第 1 天总结 .....                           | 178 | 7.2.7         | 引入 Groovy .....                         |
| 6.3   | 第 2 天: 创建/查询视图 .....                    | 179 | 7.2.8         | 特定领域的步骤 .....                           |
| 6.3.1 | 通过视图访问文档 .....                          | 179 | 7.2.9         | 更新、删除与完成 .....                          |
| 6.3.2 | 编写你的第一个视图 .....                         | 181 | 7.2.10        | 第 1 天总结 .....                           |
| 6.3.3 | 将视图另存为“设计文档” .....                      | 183 | 7.3           | 第 2 天: REST、索引与算法 .....                 |
| 6.3.4 | 由 Name 查找 Artists .....                 | 184 | 7.3.1         | 引入 REST .....                           |
| 6.3.5 | 由 name 查找 albums .....                  | 184 | 7.3.2         | 用 REST 新建节点与关系 .....                    |
| 6.3.6 | 查询自定义的 Artist 与<br>Album 视图 .....       | 185 | 7.3.3         | 查找路径 .....                              |
| 6.3.7 | 使用 Ruby 将数据导入                           |     | 7.3.4         | 索引 .....                                |
|       |                                         |     | 7.3.5         | REST 与 Gremlin .....                    |
|       |                                         |     | 7.3.6         | 大数据 .....                               |

|                                    |     |                             |     |
|------------------------------------|-----|-----------------------------|-----|
| 7.3.7 功能全面的算法 .....                | 237 | 8.3.6 数据转储 .....            | 277 |
| 7.3.8 第 2 天总结 .....                | 242 | 8.3.7 Redis 集群 .....        | 279 |
| 7.4 第 3 天: 分布式高可用性 .....           | 243 | 8.3.8 Bloom 过滤器 .....       | 280 |
| 7.4.1 事务 .....                     | 243 | 8.3.9 SETBIT 和 GETBIT ..... | 282 |
| 7.4.2 高可用性 .....                   | 244 | 8.3.10 第 2 天总结 .....        | 283 |
| 7.4.3 HA 集群 .....                  | 245 | 8.4 第 3 天: 与其他数据库合作 .....   | 284 |
| 7.4.4 备份 .....                     | 250 | 8.4.1 多持久并存服务 .....         | 284 |
| 7.4.5 第 3 天总结 .....                | 251 | 8.4.2 数据填充 .....            | 285 |
| 7.5 总结 .....                       | 251 | 8.4.3 关系存储 .....            | 291 |
| 7.5.1 Neo4j 的优点 .....              | 251 | 8.4.4 服务 .....              | 293 |
| 7.5.2 Neo4j 的缺点 .....              | 252 | 8.4.5 第 3 天总结 .....         | 296 |
| 7.5.3 Neo4j 之于 CAP .....           | 252 | 8.5 总结 .....                | 297 |
| 7.5.4 结束语 .....                    | 252 | 8.5.1 Redis 的优点 .....       | 297 |
| 第 8 章 Redis .....                  | 254 | 8.5.2 Redis 的缺点 .....       | 297 |
| 8.1 数据结构服务器存储库 .....               | 254 | 8.5.3 结束语 .....             | 298 |
| 8.2 第 1 天: CRUD 与数据类型 .....        | 255 | 第 9 章 结束语 .....             | 299 |
| 8.2.1 入门指南 .....                   | 255 | 9.1 类型终极版 .....             | 299 |
| 8.2.2 事务 .....                     | 257 | 9.1.1 关系型 .....             | 299 |
| 8.2.3 复杂数据类型 .....                 | 258 | 9.1.2 键-值存储库 .....          | 300 |
| 8.2.4 到期 .....                     | 265 | 9.1.3 列型 .....              | 301 |
| 8.2.5 数据库命名空间 .....                | 266 | 9.1.4 文档型 .....             | 301 |
| 8.2.6 更多命令 .....                   | 267 | 9.1.5 图 .....               | 302 |
| 8.3 第 2 天: 高级用法、分布 .....           | 268 | 9.2 选择 .....                | 303 |
| 8.3.1 一个简单的接口 .....                | 268 | 9.3 我们将走向哪里 .....           | 303 |
| 8.3.2 服务器信息 .....                  | 271 | 附录 A 数据库概述表 .....           | 305 |
| 8.3.3 Redis 配置 .....               | 272 | 附录 B CAP 定理 .....           | 307 |
| 8.3.4 AOF (append only file) ..... | 274 |                             |     |
| 8.3.5 主从复制 .....                   | 276 |                             |     |