HZ BOOKS

PEARSON Addison Wesley

数据库重构

Refactoring Databases
Evolutionary Database Design

(加) Scott W. Ambler
(美) Pramod J. Sadalage 著

王海鹏 等译

机械工业出版社
China Machine Press

数据库重构

Refactoring Databases
Evolutionary Database Design

(加) Scott W. Ambler
(美) Pramod J. Sadalage 著
王海鹏 等译

机械工业出版社
China Machine Press

本书首次专门讨论数据库重构，向数据专业人员展示了如何运用重构、测试驱动及其他敏捷技术进行演进式数据库开发。书中通过许多实际例子，详细说明了数据库重构的过程、策略以及部署。

本书前5章介绍了演进式数据库开发的基本思想和技术，后6章详细描述了每一类重构，包括结构、数据质量、参照完整性、架构、方法的重构；另外还描述了不属于重构范畴的转换技术。

书中的示例代码是用Java、Hibernate和Oracle代码编写的，代码都很简单，读者可以毫无困难地将它们转换成C#、C++或Visual Basic代码。

本书版权登记号：图字：01-2006-2204

图书在版编目（CIP）数据

数据库重构/（加）安布勒（Ambler, S.W.），（美）塞得拉吉（Sadalage, P.J.）著；王海鹏译. -北京：机械工业出版社，2007.1

书名原文：Refactoring Databases: Evolutionary Database Design

ISBN 7-111-20209-0

Ⅰ.数…　Ⅱ.①安…　②塞…　③王…　Ⅲ.数据库系统　Ⅳ.TP311.13

中国版本图书馆CIP数据核字（2006）第126302号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：刘立卿

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2007年1月第1版第1次印刷

186mm×240mm·14.5印张

定价：32.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010)68326294

对本书的赞誉

“这本突破性的图书向我们揭示了为什么数据库 schema（模式）不一定难以改变，为什么数据不一定难以迁移，以及为什么数据库专业人员不一定要被来自顾客和开发者的变更请求压得喘不过气来。演进式设计是敏捷的核心。Ambler 和 Sadalage 向大家展示了如何演进敏捷的数据库。漂亮！”

——Joshua Kerievsky，Industrial Logic 公司的创始人，
《Refactoring to Patterns》一书的作者

“本书不仅提供了演进式数据库开发的基础知识，还提供了许多实际的、详细的数据库重构的例子。对于对敏捷开发感兴趣的数据库专业人员来说，这是一本必读的好书。”

——Doug Barry，Barry & Associates 公司的总裁，
《Web Services and Service-Oriented Architectures: The Savvy Manager's Guide》一书的作者

“Ambler 和 Sadalage 朝着别的作者觉得棘手的问题迈出了勇敢的一步。他们不仅关注数据库重构背后的理论，还解释了一步一步的过程，以一种受控的、深思熟虑的方式来进行数据库重构。但真正征服我的是超过 200 页的代码示例和深入的技术细节，展示了如何解决具体的数据库重构问题。这不仅是一本介绍一种好思想的书——更是一本教程和技术参考书，开发人员和 DBA 都会将它放在计算机旁。荣誉归于两位作者，他们成功地完成了其他人甚至都不敢尝试的事情。”

——Kevin Aguanno，IBM 加拿大公司高级项目经理

“任何承担真实项目的人都会认识到 Scott 和 Pramod 通过本书给软件开发生命周期带来的价值。与原有数据库打交道是一件相当麻烦的事情。尽管许多挑战可能是文化上的，改进可能被强势的 DBA 策略所忽略，但本书展示了重构和演进式数据库开发在技术上如何能够真正地以敏捷的方式进行。我希望遇到下一位坏心情的 DBA 时，将这本书放在他的桌上。”

——Jon Kern

“这本书很出色。它很适合数据专业人士，这些人需要在敏捷开发和对象技术的世界中完成他们的工作。通过精心组织，本书展示了什么是数据库重构，为什么要进行数据库重构以及如何进行数据库重构，同时也展示了相关的代码。就像一本最好的菜谱，我会在开发和改进数据库时经常用到它。”

——David R. Haertzen，First Place Software 公司数据管理中心的编辑

“这本优秀的图书带来了在数据世界的重构敏捷实践。它提供了实际的指导，包括在机构中重构数据库的方法学，以及如何实现单次重构的细节。这本书也阐述了开发者与 DBA 互相协作的重要性；是开发者和 DBA 必备的一本参考图书。”

——Per Kroll，IBM RUP 开发经理，
Eclipse Process Framework 项目负责人

“Scott 和 Pramod 对数据库重构所做的事相当于 Martin Fowler 对代码重构所做的事。他们提供了一组一致的过程，你可以用这些过程来改进数据库的质量。如果你要与数据打交道，这本书就是为你写的。”

——Ken Pugh，《Prefactoring》一书的作者

“现在是数据人员加入敏捷行列的时候了，Ambler 和 Sadalage 正是合适的领头人。数据建模者、管理员以及软件团队都应该读这本书。长期以来我们在不同的技术世界里各行其事，本书将有助于消除我们之间的隔阂。”

——Gary K. Evans，敏捷过程传道者，Evanetics 公司

“演进式设计和重构已经令人很兴奋了，有了重构数据库后就更妙了。在这本书中，作者与我们分享了在数据库层面进行重构的技术与策略。通过这些重构，即使是数据库已经部署到生产环境中，数据库 schema 也可以安全地演进。通过本书，数据库就能被所有开发者访问。”

——Sven Gorts

“数据库重构是一个重要的新课题，这本书率先对社区作出了贡献。”

——Floyd Marinescu，InfoQ.com and TheServerSide.com 的创建者，
《EJB Design Pattern》一书的作者

㊀ 参加本书翻译工作的有王海鹏、王海燕、李国安、黄丽红、蔡黄辉、谢伟奇、漆巧林、周建鸣、贾立群和范俊。

序

10年前，重构（refactoring）还是一个只有少数人知道的词，主要用在Smalltalk社区。很高兴看到越来越多的人开始学习通过重构以一种训练有素和有效的方式来修改工作代码。许多人现在将代码重构视为软件开发的一个基本组成部分。

我的工作领域是企业应用开发，而企业应用开发的一个很大的部分就是与数据库打交道。在我最初关于重构的书籍中，我就已经指出了数据库是重构的一个主要问题领域，因为重构数据库会引入一些新的问题。在企业软件开发领域中，数据库专业人员与软件开发人员之间隔了一堵因互不理解和相互竞争所形成的墙，这个可悲的隔阂使得这些问题变得恶化。

我喜欢Scott和Pramod的一个原因是，他们以不同的方式努力工作，试图打破这种隔阂。Scott在其关于数据库的著作中一贯地尝试跨越这一鸿沟，他在对象关系映射方面的工作极大地影响了我关于企业应用架构的写作。Pramod可能名气小一些，但他对我的影响同样很大。当我们在ThoughtWorks一起做一个项目时，我们被告知重构数据库是不可能的。Pramod不同意这种看法，他产生了一些粗略的想法，并把这些想法变成了一种训练有素的计划，让数据库schema（模式）保持在一种稳定但受控的变化中。这使得应用开发人员能够在代码中自由地进行演进式设计。自那时起，Pramod将这些技术带给了我们的许多客户，并在ThoughtWorks的同事中传播这些技术，至少对我们来说，使数据库不再成为持续设计的拦路石。

本书汇集了两个人的经验，他们工作于应用与数据之间的无人地带，提供了对数据库重构的技术指导。如果你熟悉重构，你会注意到主要变化是你必须管理持续的数据迁移，而不只是改变程序和数据的结构。本书告诉了你如何进行数据迁移，其背后是两位作者积累的项目经验（和教训）。

尽管我很高兴看到本书出版，但我希望这只是第一步。在我的关于重构的书出版之后，我很高兴地看到出现了一些成熟的工具，利用这些工具能自动完成许多重构任务。我希望同样的情况也出现在数据库上，我们也看到一些供应商开始提供这方面的工具，使持续的schema和数据迁移变得更加容易。在这一切发生之前，本书将帮助你构建自己的过程和工具；以后，本书作为成功使用这类工具的基础，仍将体现其价值。

——Martin Fowler，丛书编辑，ThoughtWorks首席科学家

自我开始软件开发生涯以来，行业和技术的许多方面都已发生了巨大的变化。但没有改变的是软件开发的基础本质。创造软件从来都不难——只需要一台计算机并开始大量地产出代码。但创造良好的软件却很难，而创建优秀的软件更是难上加难。这种情形直至今天也没有改变。今天，我们可以更容易地创建更大、更复杂的软件系统，只需将不同来源的一些部件拼凑在一起。软件开发工具得到了极大的发展，我们对创建软件的过程中哪些有效、哪些无效也有

了更多的认识。然而大多数软件仍然是脆弱的，并努力想达到可接受的品质。或许是因为我们正在创建更大、更复杂的系统，或许是因为我们使用的技术仍然存在某些基本的缺失。由于前面两个因素，我相信今天的软件开发会像以前一样富有挑战性。幸运的是，新的技术不断出现，为我们提供了帮助。在这些进展中，能够极大地影响我们在项目开始时认识其潜在能力的却极少。重构所涉及的技术，以及相关的敏捷方法学，就是这些少有的进展之一。本书所包含的工作在一个很重要的方向上扩展了这一基础。

重构是一种受控的技术，指安全地改进代码的设计而不改变它的行为语义。任合人都可能有机会改进代码，但重构带来一种训练有素的方法，能安全地进行改动（通过测试），并利用软件开发社区所积累的知识（通过重构）。自从 Fowler 推出了这个领域的开创性书籍以来，重构已经被广泛地应用，帮助检测重构的可能性和对代码进行重构的工具也已被广泛采用。然而在应用的数据层，重构却被证实要困难得多。部分问题无疑是文化上的原因，正如这本书所展示的；但是同时也缺少适用于数据层的清晰的过程和一组重构技术。这相当不幸，因为数据层的糟糕设计几乎总是会带来更高层的问题，通常会引起一连串的坏设计，而这只是为了徒劳地维护这个不可靠基础的稳定性。而且，不能演进数据层，不论是因为忽视了这一点或者是害怕改变，都将妨碍所有依赖于它的工作，最终导致不能交付最好的软件。正是这些问题使得这项工作变得很重要：我们现在有了一个过程和一组重构技术，使我们能在这个重要的领域进行迭代式设计改进。

我很高兴看到这本书的出版，希望它能推动创造一些工具来支持书中描述的技术。软件行业目前处于一个有趣的阶段，我们看到了开放源代码软件的兴起以及它所带来的协作工具。在像 Eclipse Data Tools Platform 这样的项目中，人们自然希望在工具中实现数据库重构。我希望开源社区努力工作来实现这一愿景，因为潜在的回报是巨大的。如果数据库重构像一般的重构那样得到广泛采用，软件开发将变得更加成熟。

——John Graham，Eclipse Data Tools Platform 项目管理委员会主席，
Sybase 公司高级工程师

数据社区在许多方面错过了整个敏捷软件开发革命。应用开发者已广泛应用重构、测试驱动开发以及其他技术，使迭代成为有效的高级软件开发方式；而数据专业人士却在很大程度上忽略了这些潮流，甚至把自己与潮流隔离开来。

当我还在一家大型金融服务机构做应用开发时，就清楚地看到了这一点。那时我的办公室就在开发团队和数据库团队之间。我很快发现，尽管他们之间只有几步之遥，但两个团队的文化、实践和过程是极为不同的。顾客的要求对开发团队来说就意味着一些重构，代码检入（check in），以及迅速的验收测试。对数据库团队而言，类似的请求就意味着正式的变更请求，要通过层层批准，最后才能开始修改 schema。这种繁杂的过程开销常常使开发人员和顾客变得沮丧，但事情只能这样，因为数据库团队不知道能有什么其他办法。

然而，如果他们的业务想在今天不断发生变化的竞争环境中得到发展，那么，他们就必须学习其他的办法。数据社区必须像开发者那样采用一些敏捷技术。

本书是一本无价的资源，它向数据专业人士展示了怎样实现飞越，自信而安全地拥抱变化。Scott 和 Pramod 展示了小的、迭代式重构的设计改进如何使敏捷 DBA 避免大的前期设计错误，以及如何随着对顾客需求理解的逐步加深与应用一起演进数据库 schema。

不要弄错，重构数据库是很难的。即使是像列改名这样的简单变化，都会引起 schema、相关对象、持久层框架和应用层的相应改变，这令 DBA 觉得重构似乎是一种不可用的技术。

本书列出了一组实践说明，向专业 DBA 展示了如何将敏捷方法用于数据库的设计和开发。Scott 和 Pramod 对实现每类数据库重构的详细描述证明了它是可行的，为数据库广泛采用重构铺平了道路。

因此，我建议数据专业人士行动起来。阅读这本书，拥抱变化，广为宣传。数据库重构是改进数据社区敏捷性的关键。

——Sachin Rekhi，微软公司项目经理

在系统开发的世界里，存在两种不同的文化：一种是由面向对象（OO）开发者主宰的，他们在 Java 和敏捷软件开发中自由呼吸；另一种由关系数据库人员主宰，他们喜欢仔细的工程方法和稳定的关系数据库设计。这两个阵营说着不同的语言，参加不同的会议，相互之间似乎很少交谈。这种分裂在许多组织机构的 IT 部门中得到了反映。OO 开发人员抱怨 DBA 们古板而保守，不能跟上快速变化的节奏。数据库专业人士哀叹 Java 开发人员的愚蠢行为，因为他们不知道怎样和数据库打交道。

Scott Ambler 和 Pramod Sadalage 属于能够跨越两种文化的很少的一部分人。本书是从 OO 架构师的角度来谈数据库设计的；因此，本书对于 OO 开发人员和关系数据库专业人员都有价值。它帮助 OO 开发人员在数据库领域应用敏捷代码重构技术，并让关系数据库专业人员了解 OO 架构师的想法。

这本书包含了许多提示和技巧，可用于改进数据库设计的质量。它明确关注如何解决真实世界的问题，即数据库已经存在，但设计有问题，或者原来的数据库设计不能产生一种好的模型。

这本书在不同的层面上获得了成功。首先，它可以作为实战开发人员的战术指导。另外，它也可以作为一本激发人思想的著作，让我们思考怎样融合 OO 和关系型思考方式。我希望更多的系统架构师能响应 Ambler 和 Sadalage 的观点，认识到数据库不仅仅是存放类的持久拷贝的地方。

——Paul Dorsey 博士，Dulcian 公司总裁，
New York Oracle User Group 总裁，J2EE SIG 主席

前　言

演进式开发以及敏捷软件开发方法学，如极限编程（XP）、Scrum、Rational 统一过程（RUP）、敏捷统一过程（AUP）和特征驱动开发（FDD），已经在过去几年中为信息技术（IT）产业带来了一场风暴。为了明确定义，演进式方法在本质上是迭代式和增量式的，敏捷方法在本质上是演进式和高度协作的。而且，像重构、结对编程、测试驱动开发（TDD）和敏捷模型驱动开发（AMDD）等敏捷技术也正在进入 IT 组织机构中。这些方法和技术已经形成，并在这些年里以草根的方式演进着，在实战中得到检验，而不是在象牙塔中形式化。简而言之，这些演进式开发和敏捷方法学似乎在实践中非常有效。

在 Martin Fowler 的开创性著作《Refactoring》中，他将重构描述为对源代码的一些小改动，这些改动会改进代码的设计，但不会改变其语义。换言之，你改进了工作的品质，但不会破坏或增加任何东西。在那本书中 Martin 提到，正如可以重构应用源代码一样，你也可能重构数据库 schema（模式）。但是，他指出数据库重构相当困难，因为数据库中有相当程度的耦合，因此他选择不在他的书中讨论这部分内容。

自 1999 年《Refactoring》出版以来，我们俩发现了一些重构数据库 schema 的方法。开始，我们是独自工作的，在 Software Development（www.sdexpo.com）等一些会议上见面，并通过邮件列表（www.agiledata.org/feedback.html）交流。我们讨论一些思想，参加对方的会议报告，很快发现彼此的思想和技术有着共同之处，并且相互吻合。所以我们合作编写了本书，分享我们在通过重构演进数据库 schema 方面的经验和技术。

本书中的示例代码是用 Java、Hibernate 和 Oracle 编写的。对于每一类数据库重构，描述时基本上都包含了修改数据库 schema 的代码；对于一些更有趣的重构，我们展示了它们对Java 应用代码所产生的影响。因为各种数据库并不一样，所以我们也讨论了当数据库产品之间存在重要差别时一些替代的实现策略。有时候，我们讨论一类重构的替代实现，这类重构使用了 Oracle 专有的特征，如 SET UNUSED 或 RENAME TO 等命令；我们的许多代码示例也用到了 Oracle 的 COMMENT ON 特征。有些数据库包含另外一些使数据库重构变得容易的特征，好的 DBA 会知道如何利用这些特征。更好的是，将来数据库重构工具会为我们完成这些事情。而且，我们尽可能使 Java 代码保持简单，这样读者应该能够毫无困难地将这些代码转换成 C#、C++ 或 Visual Basic 代码。

为何需要演进式数据库开发

演进式数据库开发是一个适时出现的概念。不是在项目的前期试图设计数据库 schema，而是在整个项目生命周期中逐步地形成它，以反映项目涉众确定的不断变化的需求。不论你是否喜欢，需求会随着项目的推进而变化。传统的方式是忽略这个基本事实并试图以各种方式来

“管理变更”，这实际上是对阻止变更的一种委婉的说法。现代开发技术的实践者们选择拥抱变化，并使用一些技术，从而能够随着需求的变化演进他们的工作。程序员们采用了 TDD、重构和 AMDD 等技术，创建了新的开发工具使这些技术变得更容易。我们实现了这些之后，意识到还需要一些技术和工具来支持演进式数据库开发。

以演进的方式进行数据库开发有以下好处：

1. *将浪费减至最少。*演进的、即时（JIT）的生产方式能够避免一些浪费，在串行式开发方式中，如果需求发生变化，这些浪费是不可避免的。如果后来发现某项需求不再需要，所有对详细需求、架构和设计工件方面的早期投资都会损失掉。如果有能力事先完成这部分工作，那肯定也有能力以 JIT 的方式来完成同样的工作。

2. *避免了很多返工。*在第 1 章将会看到，仍需要进行一些初始的建模工作，将主要问题前期想清楚，如果在项目后期才确定这些问题，可能会导致大量返工。你只是不需要很早涉及其中的细节。

3. *总是知道系统可以工作。*通过演进的方式，定期产生能够工作的软件，即使只是部署到一个演示环境中，它也能工作。如果每一两周就得到一个系统的可工作版本，就会大大地降低项目的风险。

4. *总是知道数据库设计具有最高的品质。*这就是数据库重构所关注的：每次改进一点 schema 设计。

5. *与开发人员的工作方式一致。*开发人员以演进的方式工作，如果数据专业人员希望成为现代开发团队中的有效成员，他们也需要以演进的方式工作。

6. *减少了总工作量。*以演进的方式工作，只需要完成今天真正需要完成的工作，没有其他工作。

演进式数据库开发也有一些不足之处：

1. *存在文化上的阻碍。*许多数据专业人员喜欢按串行式的方式进行软件开发，他们常持这样的观点：在编程开始之前，必须创建某种形式的详细逻辑和物理数据模型。现代方法学已经放弃了这种方式，因为它效率不高，风险较大，因此许多数据专业人员受到了冷遇。更糟糕的是，数据社区中的许多“思想领袖”是在 20 世纪 70 年代和 80 年代获得他们的初步经验的，但却错过了 90 年代的对象革命，因此没有取得演进式开发方面的经验。世界已经改变，而他们似乎没有跟着改变。读者会在本书中看到，数据专业人员不仅可能以演进的方式工作，实际上这也是比较可取的工作方式。

2. *学习曲线。*需要花时间来学习这些新技术，甚至需要花更多的时间将串行式的思维方式转变成演进式的思维方式。

3. *工具支持还在发展之中。*当《Refactoring》在 1999 年出版时，没有支持这一技术的工具。短短几年之后，每种集成开发环境（IDE）都内建支持代码重构的特征。在本书写作时，还没有数据库重构的工具，尽管我们在书中包含了手工实现重构所需的全部代码。幸运的是，Eclipse Data Tools Project（DTP）已经在他们的项目计划书中说明需要在 Eclipse 中开发数据库重构功能，所以工具厂商赶上来只是时间问题。

敏捷简介

虽然这不是一本专门讲述敏捷软件开发的图书，但实际上数据库重构是为敏捷开发者准备的主要技术。如果一个过程满足敏捷联盟（www.agilealliance.org）的4种价值观，它就被认为是敏捷的。这些价值观定义了一些偏好，而不是以某种方式取代另一种方式；鼓励关注特定领域，但并不排除其他领域。例如，过程和工具是重要的，但个人和他们之间的互动更重要。这4种敏捷价值观如下：

1. *个人和他们之间的互动胜过过程和工具*。需要考虑的最重要的因素是人以及他们如何一起工作；如果没有找对人，最好的工具和过程都会毫无用处。

2. *工作的软件胜过面面俱到的文档*。软件开发的主要目标是创建能工作的软件，满足项目涉众的要求。当然文档也有它的位置，编写得当的话，它能描述系统如何构建，为什么构建，以及如何利用系统工作。

3. *顾客协作胜过合同谈判*。只有顾客能告诉你他们想要什么。不幸的是，他们不擅长此道：他们可能缺少准确描述系统所需的技能，也可能一开始就没弄对，更糟的是，他们可能随着时间的推移而改变想法。与顾客签订合同是重要的，但合同不能代替有效的沟通。成功的IT专业人员会与他们的顾客密切协作，他们会投入一定的精力来发现顾客想要的东西，并一直教育他们的顾客。

4. *响应变化胜过遵循计划*。随着系统工作的推进，项目涉众对他们想要的东西的理解发生了变化，业务环境发生了变化，底层技术也发生了变化。变化是软件开发的现实，因此，你的项目计划和整体方式必须反映变化的环境，才能是有效的。

如何阅读本书

本书的大部分内容（从第6章到第11章）包含了详细描述每一类重构的参考材料。前面5章描述了演进式数据库开发的基本思想和技术，特别介绍了数据库重构。读者应该依次阅读下面的章节：

- 第1章“演进式数据库开发”，概述了演进式开发的基本思想和支持它的技术。综述了重构、数据库重构、数据库回归测试、通过AMDD方法进行演进式数据建模、数据库资产的配置管理，以及对独立的开发者沙盒的需求。
- 第2章“数据库重构”，详细地探讨了数据库重构背后的概念，以及为什么它在实践中如此困难。展示了在“简单的”单应用环境下和复杂的多应用环境下数据库重构的例子。
- 第3章“数据库重构过程”，描述了在简单和复杂的环境下，重构数据库schema所需的详细步骤。对于单应用数据库来说，你对环境拥有更大的控制力，因此重构schema所需的工作也要少得多。在多应用环境中，你需要支持一个转换期，在这段时间内数据库同时支持原来的schema和新的schema，让应用团队能更新他们的代码并将代码部署到生产环境中去。

- 第 4 章“部署到生产环境”，描述了将数据库重构部署到生产环境中去的过程。这在多应用环境下特别富有挑战性，因为多个团队所做的改动必须合并和测试。
- 第 5 章“数据库重构策略”，汇集了我们在这些年来发现的一些关于重构数据库 schema 的“最佳实践”。我们也提出了一些打算尝试但还没有机会尝试的想法。

关于封面

Martin Fowler 签名丛书中，每本书的封面都有一座桥梁的图片。这个惯例是因为 Martin 的妻子是一名土建工程师，在丛书开始时，她正在为桥梁和隧道类的项目设计水平投影图。这座桥是南安大略的 Burlington Bay James N. Allan Skyway，它穿越了 Hamilton 港。这里有 3 座桥：图中的两座以及没有显示出来的 Eastport Drive 吊桥。桥的重要性有两个原因。最重要的是它体现了增量式的交付方式。吊桥最初承担了这一地区的交通，原来承担交通的桥在 1952 年因轮船撞击而倒塌。Skyway 的第一跨——前部在道路上方有金属支撑的部分，在 1958 年开发，以取代倒塌的桥。因为 Skyway 是多伦多向北和尼亚加拉瀑布向南的主要交通要道，所以交通量很快就超过了设计能力。第二跨是没有金属支撑的部分，它在 1985 年开放，以支持新的负载。增量式交付不论是在土木工程领域还是在软件开发领域，在经济上都很有意义。我们使用这张图的第二个原因是，Scott 是在安大略的 Burlington 长大的，他出生于 Joseph Brant 医院，靠近 Skyway 的北部出口。Scott 用 Nikon D70S 拍下了封面的照片。

致　谢

我们想感谢这些人，他们为本书的编写提供了信息：Doug Barry、Gary Evans、Martin Fowler、Bernard Goodwin、Joshua Graham、Sven Gorts、David Hay、David Haertzen、Michelle Housely、Sriram Narayan、Paul Petralia、Sachin Rekhi、Andy Slocum、Brian Smith、Michael Thurston、Michael Vizdos 和 Greg Warren。

另外，Pramod 想感谢 Irfan Shah、Narayan Raman、Anishek Agarwal 和其他团队成员，是他们经常对我的观点提出挑战，教给我许多软件开发的知识。我也感谢 Martin 给予我写作的机会，让我去演讲，积极参加 ThoughtWorks 之外的活动；感谢 Kent Beck 的鼓励，感谢我在 ThoughtWorks 的同事，他们在许多方面给予我帮助，让工作变得有趣；感谢我的父母 Jinappa 和 Shobha，他们花了许多精力将我养育成人；感谢我的哥哥，从我小时候起，他就批评并帮助我改进写作方式。

目　录

第 1 章　演进式数据库开发

瀑布是美妙的旅游景点。但用来组织软件开发项目，瀑布式却是一种特别差劲的策略。

——Scott Ambler

现代软件过程，也称为方法学，在本质上都是演进式的，要求你以迭代和增量的方式工作。这些过程的例子包括 Rational 统一过程（RUP）、极限编程（XP）、Scrum、动态系统开发方法（DSDM）、水晶方法系列、团队软件过程（TSP）、敏捷统一过程（AUP）、企业统一过程（EUP）、特征驱动开发（FDD）和快速应用开发（RAD），等等。工作以迭代的方式进行，你在一段时间内，每项活动做一点，例如建模、测试、编码或部署，然后是下一个迭代，再下一个迭代。这种过程有别于串行的方式，在串行方式中，你首先确定所有打算实现的需求，然后创建详细的设计，实现该设计并进行测试，最后部署你的系统。在增量式的开发方式中，你会把系统组织成一系列的发布版本，而不是一个大的发布版本。

而且，许多现代过程是敏捷的，为简单起见，我们把这些过程的特点归结为本质上的演进式以及高度协作。当团队采取一种协作的方式时，他们会主动地寻求有效协同工作的方法；你甚至应该尝试让业务顾客这样的项目涉众（stakeholder）也成为积极的团队成员。Cockburn（2002）建议，你应该追求采用适合你情况的“最热”的沟通技巧：面对面围绕一块白板的对话要优于打电话，打电话要优于发送 E-mail，发送 E-mail 要优于发送一份详细的文档。软件开发团队内部的沟通和协作越好，你成功的机会就越大。

尽管演进式和敏捷的工作方式已经在开发社区中得到了采用，但在数据社区中，情况却还不是这样。大多数面向数据的技术在本质上是串行式的，要求先创建相当详细的模型，然后才“允许”开始实现。更糟糕的是，这些模型常常被基线化，并纳入变更管理控制之下，以期尽量减少变化（如果你考虑到最终效果，这应该被称为一个变更防止过程）。这其中出现了摩擦：一般的数据库开发技术没有反映出现代软件开发过程的实际情况。事情并非一定如此。

我们的前提是，数据专家需要像开发人员一样，采用演进式技术。虽然你可以争辩说，在数据社区中开发人员应该回到“经检验正确”的传统开发方式，但是传统的方式不能很好地工作，这一点正变得越来越明显。Craig Larman 在他的著作（Larman，2004）的第 5 章 Agile & Iterative Development 中总结了支持演进式开发的研究证据，以及信息技术（IT）社区中的领导者对演进式开发的压倒性支持。简单地说，与在数据社区中流行的传统技术相比，在开发社区中流行的演进式和敏捷技术工作得更好。

数据专家也可以在他们工作的各个方面采用演进式技术，如果他们愿意的话。第一步是重新考虑你的 IT 组织中的“数据文化”，以反映现代 IT 项目团队的需要。敏捷数据（AD）方法（Ambler 2003）正是这样做的，它描述了现代面向数据的活动中的一些原则和角色。这些原则

反映出数据如何成为商业软件的诸多重要方面之一，暗示开发人员需要更擅长数据技术，数据专家需要学习现代的开发技术与技能。AD方法意识到每个项目团队都是独特的，需要遵守一个按照他们的情况进行剪裁的过程。超越你当前的项目而考虑整个企业的问题，这一点也得到了强调。因为像操作型数据库管理员和数据架构师这样的企业专家需要这样做，才能灵活地与采用敏捷方式的项目团队共同工作。

第二步是针对数据专家的，特别是数据管理人员。他们要采用新的技术，确保他们能以一种演进式的方式工作。在本章中，我们简单地介绍了这些关键的技术。在我们看来，最重要的技术就是数据库重构，这就是本书关注的重点。演进式数据库开发技术包括：

1. 数据库重构。让现有的数据库schema每次演进一点，在不改变语义的情况下改善其设计质量。

2. 演进式数据建模。以迭代和增量的方式对系统的数据进行建模，就像对系统的其他方面进行建模时一样，从而确保数据库schema与应用的代码一起演进。

3. 数据库回归测试。确保数据库schema确实能工作。

4. 数据库工件的配置管理。你的数据模型、数据库测试、测试数据等都是重要的项目工件（artifact），应该像其他工件那样管理起来。

5. 开发者沙盒。开发者需要属于他们自己的工作环境，他们可以在其中修改正在构建的系统的一部分，使这部分修改能工作，然后再将他们的工作与其他团队成员的工作集成起来。

让我们来仔细考虑一下每一种演进式数据库技术。

1.1 数据库重构

重构（Fowler 1999）是一种训练有素的方式，对源代码进行小的改动以改进其设计，使代码处理起来变得更容易。重构的一个关键方面是，它保持了代码的行为语义——你在重构时既不添加东西也不减少东西，你只是改进代码的质量。重构的一个例子可以是将getPersons()操作改名为getPeople()。为了实现这一重构，你必须改变操作定义，然后改变应用代码中所有对这个操作的调用。直到所有的代码重新运行之后，重构才算完成。

类似地，数据库重构是对数据库schema进行的简单改动，在保持其行为和信息语义的前提下改进其设计。你既可以重构数据库schema的结构部分，如表和视图的定义，也可以重构其功能部分，如存储过程和触发器。当你重构数据库schema时，不仅需要改变schema本身，也需要修改与你的schema耦合在一起的外部系统，如业务应用或数据extract。数据库重构明显地要比代码重构更难以实现，因此你需要小心。数据库重构将在第2章中详细描述，执行数据库重构的过程将在第3章中描述。

1.2 演进式数据库建模

不论你曾经听到过什么传言，演进式技术和敏捷技术绝不只是“编码加修复”的一个新名字。在开始构建之前，你仍然需要探索需求并完整地思考架构和设计；在编码之前，你需要一种好的方式进行建模。图1.1展示了敏捷模型驱动开发（AMDD）的生命周期（Ambler 2004；