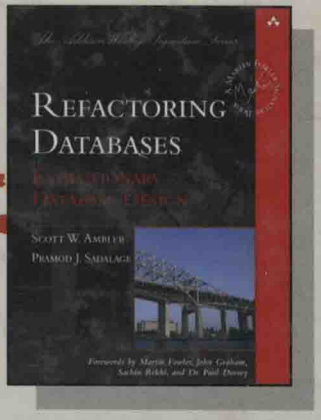
PEARSON
Addison Wesley

典藏原版书苑

Refactoring Databases
Evolutionary Database Design

[美] Scott W. Ambler Pramod J. Sadalage 著

数据库重构(英文版)



荣获2007年第17届Jolt生产力大奖

人民邮电出版社
POSTS & TELECOM PRESS

重构技术已经在众多领域的开发项目中证明了其自身的价值——帮助软件专业人士改善系统的设计、可维护性、可扩展性和性能。如今，世界一流的敏捷方法理论家Scott W. Ambler和著名咨询顾问Pramod J. Sadalage首次披露了为数据库系统专门设计的强大重构技术。

Ambler和Sadalage展示了如何在不改变语义的情况下，对表结构、数据、存储过程及触发器等略作改动，就可以给几乎任何数据库设计带来实质上的飞跃。读者将学会如何让数据库模式与源代码同步渐进——从而依靠迭代的敏捷方法在项目中发挥更多的作用。

Scott W. Ambler，世界一流的软件过程改进咨询顾问，开创了敏捷建模、敏捷数据、企业统一过程（Enterprise UP）以及敏捷统一过程方法论。曾在软件开发、JavaOne、OOPSLA、DAMA等会议上作主题发言和演讲。

Pramod J. Sadalage，ThoughtWorks公司咨询顾问，1999年在一大型J2EE应用程序上使用极限编程方法的同时，率先实践、处理过渐进式数据库设计和数据库重构。目前正在就渐进项目上的数据库管理、渐进过程在数据库设计和管理上的应用等问题著书、演讲。

运用本书的技术和实例，读者可以减少浪费和重复工作，降低风险和成本，建立能够顺利发展以适应未来需求的数据库系统。

这是一本内容全面的参考、指南书，全面介绍了数据库重构涉及的每个基本观念，可以帮助读者克服重构数据库所遇到的实际困难。本书作者运用完整的实例，带领读者学习从重构简单的孤立数据库应用程序到重构复杂的多应用程序环境的全过程。读者将掌握与重构数据库模式有关的全部任务，找到最佳的方法，将重构的结果布置到哪怕是最复杂的产品环境中。

本书系统讲述了数据库重构的五大主要类别。读者将学会如何运用重构改善数据库结构、数据质量和参照完整性，如何同时对结构和方法进行重构。本书提供了用Oracle和Java建立的多种实例，可以方便地转换成C#、C++、VB.NET等其它语言或DB2、SQL Server、MySQL、Sybase等其他数据库。

• 装帧设计：胡平利

ISBN 978-7-115-15570-2

9 787115 155702 >

ISBN 978-7-115-15570-2/TP

定价：65.00 元

分类建议：计算机/计算机科学/数据库理论

人民邮电出版社网址：www.ptpress.com.cn

PEARSON
Addison Wesley
典藏原版书苑

数据库重构

（英文版）

人民

数据库重构

（英文版）

Refactoring Databases
Evolutionary Database Design

[美] Scott W. Ambler
Pramod J. Sadalage 著

人民邮电出版社
北 京

图书在版编目（CIP）数据

数据库重构/（美）安布勒（Ambler, S. W.），（美）萨达拉戈（Sadalage, P. J.）著.
—北京：人民邮电出版社，2007.6
（典藏原版书苑）

ISBN 978-7-115-15570-2

Ⅰ. 重… Ⅱ. ①安… ②萨… Ⅲ. 数据库系统—程序设计—英文 Ⅳ. TP311.13

中国版本图书馆 CIP 数据核字（2006）第 147794 号

典藏原版书苑
数据库重构（英文版）

◆ 著 ［美］Scott W. Ambler Pramod J. Sadalage
责任编辑 付 飞

◆ 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 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京铭成印刷有限公司印刷
新华书店总店北京发行所经销

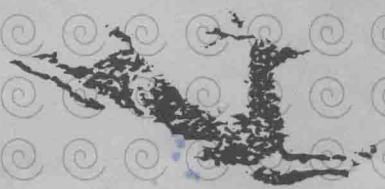
◆ 开本：800×1000 1/16
印张：23.5
字数：470 千字 2007 年 6 月第 1 版
印数：1－3 000 册 2007 年 6 月北京第 1 次印刷

著作权合同登记号 图字：01-2007-0860 号

ISBN 978-7-115-15570-2/TP

定价：65.00 元

读者服务热线：(010)67132705 印装质量热线：(010)67129223



智慧巅峰

Summit of Human Intelligence

内容提要

重构技术已经在领域广泛的开发项目中证明了自身的价值——帮助软件专业人士改善系统的设计、可维护性、可扩展性和性能。本书首次披露了为数据库系统专门设计的强大重构技术。

本书展示了如何在不改变语义的情况下，对表结构、数据、存储过程及触发器等略作改动，就可以给数据库设计带来实质上的飞跃。

这是一本内容全面的参考、指南书，全面介绍了数据库重构涉及的每个基本观念，运用完整的实例，带领读者学习从重构简单的孤立数据库应用程序到重构复杂的多应用程序环境的全过程，并讲述了数据库重整的五大主要类别。读者将学会如何运用重构改善数据库结构、数据质量和参照完整性，如何同时对结构和方法进行重整。本书提供了用 Oracle 和 Java 建立的多种实例，并可以方便地转换成 C#、C++、VB.NET 等其他语言或 DB2、SQL Server、MySQL、Sybase 等其他数据库。

运用本书的技术和实例，读者可以减少浪费和重复工作，降低风险和成本，建立能够顺利发展以适应未来需求的数据库系统。

Praise for *Refactoring Databases*

"This groundbreaking book finally reveals why database schemas need not be difficult to change, why data need not be difficult to migrate, and why database professionals need not be overburdened by change requests from customers and developers. Evolutionary design is at the heart of agility. Ambler and Sadalage have now shown the world how to evolve agile databases. Bravo!"
—Joshua Kerievsky, founder, Industrial Logic, Inc.; author, *Refactoring to Patterns*

"This book not only lays out the fundamentals for evolutionary database development, it provides many practical, detailed examples of database refactoring. It is a must read for database practitioners interested in agile development."
—Doug Barry, president, Barry & Associates, Inc.; author of *Web Services and Service-Oriented Architectures: The Savvy Manager's Guide*

"Ambler and Sadalage have taken the bold step of tackling an issue that other writers have found so daunting. Not only have they addressed the theory behind database refactoring, but they have also explained step-by-step processes for doing so in a controlled and thoughtful manner. But what really blew me away were the more than 200 pages of code samples and deep technical details illustrating how to overcome specific database refactoring hurdles. This is not just another introductory book convincing people that an idea is a good one—this is a tutorial and technical reference book that developers and DBAs alike will keep near their computers. Kudos to the brave duo for succeeding where others have failed to even try."
—Kevin Aguanno, senior project manager, IBM Canada Ltd.

"Anybody working on non-greenfield projects will recognize the value that Scott and Pramod bring to the software development life cycle with Refactoring Databases. *The realities of dealing with existing databases is one that is tough to crack. Though much of the challenge can be cultural and progress can be held in limbo by strong-armed DBA tactics, this book shows how technically the refactoring and evolutionary development of a database can indeed be handled in an agile manner. I look forward to dropping off a copy on the desk of the next ornery DBA I run into."*
—Jon Kern

"This book is excellent. It is perfect for the data professional who needs to produce results in the world of agile development and object technology. A well-organized book, it shows the what, why, and how of refactoring databases and associated code. Like the best cookbook, I will use it often when developing and improving databases."
—David R. Haertzen, editor, The Data Management Center, First Place Software, Inc.

"This excellent book brings the agile practice of refactoring into the world of data. It provides pragmatic guidance on both the methodology to refactoring databases within your organization and the details of how to implement individual refactorings. Refactoring Databases *also articulates the importance of developers and DBAs working side by side. It is a must have reference for developers and DBAs alike."*
—Per Kroll, development manager, RUP, IBM; project lead, Eclipse Process Framework

"Scott and Pramod have done for database refactoring what Martin Fowler did for code refactoring. They've put together a coherent set of procedures you can use to improve the quality of your database. If you deal with databases, this book is for you."
—Ken Pugh, author, *Prefactoring*

"It's past time for data people to join the agile ranks, and Ambler and Sadalage are the right persons to lead them. This book should be read by data modelers and administrators, as well as software teams. We have lived in different worlds for too long, and this book will help to remove the barriers dividing us."
—Gary K. Evans, Agile Process evangelist, Evanetics, Inc.

"Evolutionary design and refactoring are already exciting, and with Refactoring Databases *this gets even better. In this book, the authors share with us the techniques and strategies to refactor at the database level. Using these refactorings, database schemas can safely be evolved even after a database has been deployed into production. With this book, database is within reach of any developer."*
—Sven Gorts

"Database refactoring is an important new topic and this book is a pioneering contribution to the community."
—Floyd Marinescu, creator of InfoQ.com and TheServerSide.com; author of *EJB Design Patterns*

About the Authors

Scott W. Ambler is a software process improvement (SPI) consultant living just north of Toronto. He is founder and practice leader of the Agile Modeling (AM) (www.agilemodeling.com), Agile Data (AD) (www.agiledata.org), Enterprise Unified Process (EUP) (www.enterpriseunifiedprocess.com), and Agile Unified Process (AUP) (www.ambysoft.com/unifiedprocess) methodologies. Scott is the (co-)author of several books, including *Agile Modeling* (John Wiley & Sons, 2002), *Agile Database Techniques* (John Wiley & Sons, 2003), *The Object Primer, Third Edition* (Cambridge University Press, 2004), *The Enterprise Unified Process* (Prentice Hall, 2005), and *The Elements of UML 2.0 Style* (Cambridge University Press, 2005). Scott is a contributing editor with *Software Development* magazine (www.sdmagazine.com) and has spoken and keynoted at a wide variety of international conferences, including Software Development, UML World, Object Expo, Java Expo, and Application Development. Scott graduated from the University of Toronto with a Master of Information Science. In his spare time Scott studies the Goju Ryu and Kobudo styles of karate.

Pramod J. Sadalage is a consultant for ThoughtWorks, an enterprise application development and integration company. He first pioneered the practices and processes of evolutionary database design and database refactoring in 1999 while working on a large J2EE application using the Extreme Programming (XP) methodology. Since then, Pramod has applied the practices and processes to many projects. Pramod writes and speaks about database administration on evolutionary projects, the adoption of evolutionary processes with regard to databases, and evolutionary practices' impact upon database administration, in order to make it easy for everyone to use evolutionary design in regards to databases. When he is not working, you can find him spending time with his wife and daughter and trying to improve his running.

Scott:
For Beverley, my lovely new bride.

Pramod:
To the women I love most, Rupali and our daughter, Arula.

Acknowledgments

We want to thank the following people for their input into the development of this book: 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, and Greg Warren.

In addition, Pramod wants to thank Irfan Shah, Narayan Raman, Anishek Agarwal, and my other teammates who constantly challenged my opinions and taught me a lot about software development. I also want to thank Martin for getting me to write, talk, and generally be active outside of ThoughtWorks; Kent Beck for his encouragement; my colleagues at ThoughtWorks who have helped me in numerous ways and make working fun; my parents Jinappa and Shobha who put a lot of effort in raising me; and Praveen, my brother, who since my childhood days has critiqued and improved the way I write.

Forewords

A decade ago *refactoring* was a word only known to a few people, mostly in the Smalltalk community. It's been wonderful to watch more and more people learn how to use refactoring to modify working code in a disciplined and effective manner. As a result many people now see code refactoring as an essential part of software development.

I live in the world of enterprise applications, and a big part of enterprise application development is working with databases. In my original book on refactoring, I picked out databases as a major problem area in refactoring because refactoring databases introduces a new set of problems. These problems are exacerbated by the sad division that's developed in the enterprise software world where database professionals and software developers are separated by a wall of mutual incomprehension and contempt.

One of the things I like about Scott and Pramod is that, in different ways, they have both worked hard to try and cross this division. Scott's writings on databases have been a consistent attempt to bridge the gap, and his work on object-relational mapping has been a great influence on my own writings on enterprise application architecture. Pramod may be less known, but his impact has been just as great on me. When he started work on a project with me at ThoughtWorks we were told that refactoring of databases was impossible. Pramod rejected that notion, taking some sketchy ideas and turning them into a disciplined program that kept the database schema in constant, but controlled, motion. This freed up the application developers to use evolutionary design in the code, too. Pramod has since taken these techniques to many of our clients, spreading them around our ThoughtWorks colleagues and, at least for us, forever banishing databases from the list of roadblocks to continual design.

This book assembles the lessons of two people who have lived in the no-mans land between applications and data, and presents a guide on how to use refactoring techniques for databases. If you're familiar with refactoring, you'll notice that the major change is that you have to manage continual migration of the data itself, not just change the program and data structures. This book tells you how to do that, backed by the project experience (and scars) that these two have accumulated.

Much though I'm delighted by the appearance of this book, I also hope it's only a first step. After my refactoring book appeared I was delighted to find sophisticated tools appear that automated many refactoring tasks. I hope the

same thing happens with databases, and we begin to see vendors offer tools that make continual migrations of schema and data easier for everyone. Before that happens, this book will help you build your own processes and tools to help; afterward this book will have lasting value as a foundation for using such tools successfully.

—Martin Fowler, series editor; chief scientist, ThoughtWorks

In the years since I first began my career in software development, many aspects of the industry and technology have changed dramatically. What hasn't changed, however, is the fundamental nature of software development. It has never been hard to create software—just get a computer and start churning out code. But it was hard to create good software, and exponentially harder to create great software. This situation hasn't changed today. Today it is easier to create larger and more complex software systems by cobbling together parts from a variety of sources, software development tools have advanced in bounds, and we know a lot more about what works and doesn't work for the process of creating software. Yet most software is still brittle and struggling to achieve acceptable quality levels. Perhaps this is because we are creating larger and more complex systems, or perhaps it is because there are fundamental gaps in the techniques still used. I believe that software development today remains as challenging as ever because of a combination of these two factors. Fortunately, from time to time new technologies and techniques appear that can help. Among these advances, a rare few are have the power to improve greatly our ability to realize the potential envisioned at the start of most projects. The techniques involved in refactoring, along with their associate Agile methodologies, were one of these rare advances. The work contained in this book extends this base in a very important direction.

Refactoring is a controlled technique for safely improving the design of code without changing its behavioral semantics. Anyone can take a chance at improving code, but refactoring brings a discipline of safely making changes (with tests) and leveraging the knowledge accumulated by the software development community (through refactorings). Since Fowler's seminal book on the subject, refactoring has been widely applied, and tools assisting with detection of refactoring candidates and application of refactorings to code have driven widespread adoption. At the data tier of applications, however, refactoring has proven much more difficult to apply. Part of this problem is no doubt cultural, as this book shows, but also there has not been a clear process and set of refactorings applicable to the data tier. This is really unfortunate, since poor design at the data level almost always translates into problems at the higher tiers, typically causing a chain of bad designs in a futile effort to stabilize the shaky foundation. Further, the inability to evolve the data tier, whether due to denial or

fear of change, hampers the ability of all that rests on it to deliver the best software possible. These problems are exactly what make this work so important: we now have a process and catalog for enabling iterative design improvements on in this vital area.

I am very excited to see the publication of this book, and hope that it drives the creation of tools to support the techniques it describes. The software industry is currently in an interesting stage, with the rise of open-source software and the collaborative vehicles it brings. Projects such as the Eclipse Data Tools Platform are natural collection areas for those interested in bringing database refactoring to life in tools. I hope the open-source community will work hard to realize this vision, because the potential payoff is great. Software development will move to the next level of maturity when database refactoring is as common and widely applied as general refactoring itself.

—John Graham, Eclipse Data Tools Platform, Project Management, committee chair, senior staff engineer, Sybase, Inc.

In many ways the data community has missed the entire agile software development revolution. While application developers have embraced refactoring, test-driven development, and other such techniques that encourage iteration as a productive and advantageous approach to software development, data professionals have largely ignored and even insulated themselves from these trends.

This became clear to me early in my career as an application developer at a large financial services institution. At that time I had a cubicle situated right between the development and database teams. What I quickly learned was that although they were only a few feet apart, the culture, practices, and processes of each group were significantly different. A customer request to the development team meant some refactoring, a code check-in, and aggressive acceptance testing. A similar request to the database team meant a formal change request processed through many levels of approval before even the modification of a schema could begin. The burden of the process constantly led to frustrations for both developers and customers but persisted because the database team knew no other way.

But they must learn another way if their businesses are to thrive in today's ever-evolving competitive landscape. The data community must somehow adopt the agile techniques of their developer counterparts.

Refactoring Databases is an invaluable resource that shows data professionals just how they can leap ahead and confidently, safely embrace change. Scott and Pramod show how the improvement in design that results from small, iterative refactorings allow the agile DBA to avoid the mistake of big upfront design and evolve the schema along with the application as they gradually gain a better understanding of customer requirements.

Make no mistake, refactoring databases is hard. Even a simple change like renaming a column cascades throughout a schema, to its objects, persistence frameworks, and application tier, making it seem to the DBA like a very inaccessible technique.

Refactoring Databases outlines a set of prescriptive practices that show the professional DBA exactly how to bring this agile method into the design and development of databases. Scott's and Pramod's attention to the minute details of what it takes to actually implement every database refactoring technique proves that it can be done and paves the way for its widespread adoption.

Thus, I propose a call to action for all data professionals. Read on, embrace change, and spread the word. Database refactoring is key to improving the data community's agility.

—Sachin Rekhi, program manager, Microsoft Corporation

In the world of system development, there are two distinct cultures: the world dominated by object-oriented (OO) developers who live and breathe Java and agile software development, and the relational database world populated by people who appreciate careful engineering and solid relational database design. These two groups speak different languages, attend different conferences, and rarely seem to be on speaking terms with each other. This schism is reflected within IT departments in many organizations. OO developers complain that DBAs are stodgy conservatives, unable to keep up with the rapid pace of change. Database professionals bemoan the idiocy of Java developers who do not have a clue what to do with a database.

Scott Ambler and Pramod Sadalage belong to that rare group of people who straddle both worlds. *Refactoring Databases: Evolutionary Database Design* is about database design written from the perspective of an OO architect. As a result, the book provides value to both OO developers and relational database professionals. It will help OO developers to apply agile code refactoring techniques to the database arena as well as give relational database professionals insight into how OO architects think.

This book includes numerous tips and techniques for improving the quality of database design. It explicitly focuses on how to handle real-world situations where the database already exists but is poorly designed, or when the initial database design failed to produce a good model.

The book succeeds on a number of different levels. First, it can be used as a tactical guide for developers in the trenches. It is also a thought-provoking treatise about how to merge OO and relational thinking. I wish more system architects echoed the sentiments of Ambler and Sadalage in recognizing that a database is more than just a place to put persistent copies of classes.

—Dr. Paul Dorsey, president, Dulcian, Inc.; president, New York Oracle Users Group; chairperson, J2EE SIG