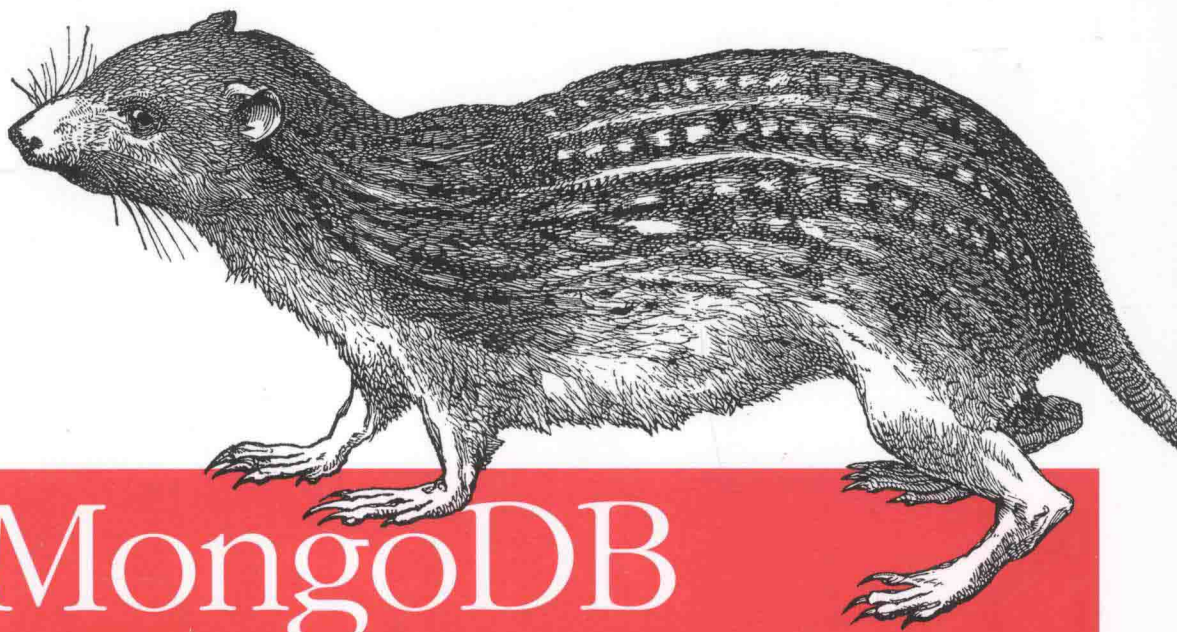


MongoDB设计模式 (影印版)



# MongoDB Applied Design Patterns

O'REILLY®

東南大學出版社

*Rick Copeland* 著

---

# MongoDB设计模式 (影印版)

## MongoDB Applied Design Patterns

*Rick Copeland* 著

**O'REILLY®**

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

## 图书在版编目 (CIP) 数据

MongoDB 设计模式: 英文 / (美) 科普兰 (Copeland, R.)

著. —影印本. —南京: 东南大学出版社, 2013.10

书名原文: MongoDB Applied Design Patterns

ISBN 978-7-5641-4458-6

I. ① M… II. ① 科… III. ① 关系数据库系统—英文

IV. ① TP311.138

中国版本图书馆 CIP 数据核字 (2013) 第 203656 号

江苏省版权局著作权合同登记

图字: 10-2013-137 号

©2013 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2013. Authorized reprint of the original English edition, 2013 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2013

英文影印版由东南大学出版社出版 2013。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式复制。

## MongoDB 设计模式 (影印版)

出版发行: 东南大学出版社

地 址: 南京四牌楼 2 号 邮编: 210096

出 版 人: 江建中

网 址: <http://www.seupress.com>

电子邮件: [press@seupress.com](mailto:press@seupress.com)

印 刷: 扬中市印刷有限公司

开 本: 787 毫米 × 980 毫米 16 开本

印 张: 11

字 数: 215 千字

版 次: 2013 年 10 月第 1 版

印 次: 2013 年 10 月第 1 次印刷

书 号: ISBN 978-7-5641-4458-6

定 价: 36.00 元 (册)

本社图书若有印装质量问题, 请直接与营销部联系。电话 (传真): 025-83791830

---

# Preface

Whether you're building the newest and hottest social media website or developing an internal-use-only enterprise business intelligence application, scaling your data model has never been more important. Traditional relational databases, while familiar, present significant challenges and complications when trying to scale up to such "big data" needs. Into this world steps MongoDB, a leading NoSQL database, to address these scaling challenges while also simplifying the process of development.

However, in all the hype surrounding big data, many sites have launched their business on NoSQL databases without an understanding of the techniques necessary to effectively use the features of their chosen database. This book provides the much-needed connection between the features of MongoDB and the business problems that it is suited to solve. The book's focus on the practical aspects of the MongoDB implementation makes it an ideal purchase for developers charged with bringing MongoDB's scalability to bear on the particular problem you've been tasked to solve.

## Audience

This book is intended for those who are interested in learning practical patterns for solving problems and designing applications using MongoDB. Although most of the features of MongoDB highlighted in this book have a basic description here, this is not a beginning MongoDB book. For such an introduction, the reader would be well-served to start with *MongoDB: The Definitive Guide* by Kristina Chodorow and Michael Dirolf (O'Reilly) or, for a Python-specific introduction, *MongoDB and Python* by Niall O'Higgins (O'Reilly).

## Assumptions This Book Makes

Most of the code examples used in this book are implemented using either the Python or JavaScript programming languages, so a basic familiarity with their syntax is essential to getting the most out of this book. Additionally, many of the examples and patterns

are contrasted with approaches to solving the same problems using relational databases, so basic familiarity with SQL and relational modeling is also helpful.

## Contents of This Book

This book is divided into two parts, with Part I focusing on general MongoDB design patterns and Part II applying those patterns to particular problem domains.

### Part I: Design Patterns

Part I introduces the reader to some generally applicable design patterns in MongoDB. These chapters include more introductory material than Part II, and tend to focus more on MongoDB techniques and less on domain-specific problems. The techniques described here tend to make use of MongoDB distinctives, or generate a sense of “hey, MongoDB can’t do that” as you learn that yes, indeed, it can.

#### *Chapter 1: To Embed or Reference*

This chapter describes what kinds of documents can be stored in MongoDB, and illustrates the trade-offs between schemas that embed related documents within related documents and schemas where documents simply reference one another by ID. It will focus on the performance benefits of embedding, and when the complexity added by embedding outweighs the performance gains.

#### *Chapter 2: Polymorphic Schemas*

This chapter begins by illustrating that MongoDB collections are schemaless, with the schema actually being stored in individual documents. It then goes on to show how this feature, combined with document embedding, enables a flexible and efficient polymorphism in MongoDB.

#### *Chapter 3: Mimicking Transactional Behavior*

This chapter is a kind of apologia for MongoDB’s lack of complex, multidocument transactions. It illustrates how MongoDB’s modifiers, combined with document embedding, can often accomplish in a single atomic document update what SQL would require several distinct updates to achieve. It also explores a pattern for implementing an application-level, two-phase commit protocol to provide transactional guarantees in MongoDB when they are absolutely required.

### Part II: Use Cases

In Part II, we turn to the “applied” part of *Applied Design Patterns*, showing several use cases and the application of MongoDB patterns to solving domain-specific problems. Each chapter here covers a particular problem domain and the techniques and patterns used to address the problem.

#### *Chapter 4: Operational Intelligence*

This chapter describes how MongoDB can be used for operational intelligence, or “real-time analytics” of business data. It describes a simple event logging system, extending that system through the use of periodic and incremental hierarchical aggregation. It then concludes with a description of a true real-time incremental aggregation system, the Mongo Monitoring Service (MMS), and the techniques and trade-offs made there to achieve high performance on huge amounts of data over hundreds of customers with a (relatively) small amount of hardware.

#### *Chapter 5: Ecommerce*

This chapter begins by describing how MongoDB can be used as a product catalog master, focusing on the polymorphic schema techniques and methods of storing hierarchy in MongoDB. It then describes an inventory management system that uses optimistic updating and compensation to achieve eventual consistency even without two-phase commit.

#### *Chapter 6: Content Management Systems*

This chapter describes how MongoDB can be used as a backend for a content management system. In particular, it focuses on the use of polymorphic schemas for storing content nodes, the use of GridFS and Binary fields to store binary assets, and various approaches to storing discussions.

#### *Chapter 7: Online Advertising Networks*

This chapter describes the design of an online advertising network. The focus here is on embedded documents and complex atomic updates, as well as making sure that the storage engine (MongoDB) never becomes the bottleneck in the ad-serving decision. It will cover techniques for frequency capping ad impressions, keyword targeting, and keyword bidding.

#### *Chapter 8: Social Networking*

This chapter describes how MongoDB can be used to store a relatively complex social graph, modeled after the Google+ product, with users in various circles, allowing fine-grained control over what is shared with whom. The focus here is on maintaining the graph, as well as categorizing content into various timelines and news feeds.

#### *Chapter 9: Online Gaming*

This chapter describes how MongoDB can be used to store data necessary for an online, multiplayer role-playing game. We show how character and world data can be stored in MongoDB, allowing for concurrent access to the same data structures from multiple players.

# Conventions Used in This Book

The following typographical conventions are used in this book:

## *Italic*

Indicates new terms, URLs, email addresses, filenames, and file extensions.

## Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

## Constant width bold

Shows commands or other text that should be typed literally by the user.

## *Constant width italic*

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

# Using Code Examples

This book is here to help you get your job done. In general, if this book includes code examples, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*MongoDB Applied Design Patterns* by Rick Copeland (O'Reilly). Copyright 2013 Richard D. Copeland, Jr., 978-1-449-34004-9.”

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at [permissions@oreilly.com](mailto:permissions@oreilly.com).

## Safari® Books Online

**Safari**<sup>®</sup> Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

## How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.  
1005 Gravenstein Highway North  
Sebastopol, CA 95472  
800-998-9938 (in the United States or Canada)  
707-829-0515 (international or local)  
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at <http://oreil.ly/mongodb-applied-design-patterns>.

To comment or ask technical questions about this book, send email to [bookquestions@oreilly.com](mailto:bookquestions@oreilly.com).

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.



Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

## Acknowledgments

Many thanks go to O'Reilly's Meghan Blanchette, who endured the frustrations of trying to get a technical guy writing a book to come up with a workable schedule and stick to it. Sincere thanks also go to my technical reviewers, Jesse Davis and Mike Dirolf, who helped catch the errors in this book so the reader wouldn't have to suffer through them.

Much additional appreciation goes to 10gen, the makers of MongoDB, and the wonderful employees who not only provide a great technical product but have also become genuinely close friends over the past few years. In particular, my thanks go out to Jared Rosoff, whose ideas for use cases and design patterns helped inspire (and subsidize!) this book, and to Meghan Gill, for actually putting me back in touch with O'Reilly and getting the process off the ground, as well as providing a wealth of opportunities to attend and speak at various MongoDB conferences.

Thanks go to my children, Matthew and Anna, who've been exceedingly tolerant of a Daddy who loves to play with them in our den but can sometimes only send a hug over Skype.

Finally, and as always, my heartfelt gratitude goes out to my wonderful and beloved wife, Nancy, for her support and confidence in me throughout the years and for inspiring me to many greater things than I could have hoped to achieve alone. I couldn't possibly have done this without you.

---

# Table of Contents

<b>Preface.....</b>	<b>vii</b>
---------------------	------------

---

## **Part I. Design Patterns**

<b>1. To Embed or Reference.....</b>	<b>3</b>
Relational Data Modeling and Normalization	3
What Is a Normal Form, Anyway?	4
So What's the Problem?	6
Denormalizing for Performance	7
MongoDB: Who Needs Normalization, Anyway?	8
MongoDB Document Format	8
Embedding for Locality	9
Embedding for Atomicity and Isolation	9
Referencing for Flexibility	11
Referencing for Potentially High-Arity Relationships	12
Many-to-Many Relationships	13
Conclusion	14
<b>2. Polymorphic Schemas.....</b>	<b>17</b>
Polymorphic Schemas to Support Object-Oriented Programming	17
Polymorphic Schemas Enable Schema Evolution	20
Storage (In-)Efficiency of BSON	21
Polymorphic Schemas Support Semi-Structured Domain Data	22
Conclusion	23
<b>3. Mimicking Transactional Behavior.....</b>	<b>25</b>
The Relational Approach to Consistency	25
Compound Documents	26
Using Complex Updates	28

---

Optimistic Update with Compensation	29
Conclusion	33

---

## Part II. Use Cases

<b>4. Operational Intelligence.....</b>	<b>37</b>
Storing Log Data	37
Solution Overview	37
Schema Design	38
Operations	39
Sharding Concerns	48
Managing Event Data Growth	50
Pre-Aggregated Reports	52
Solution Overview	52
Schema Design	53
Operations	59
Sharding Concerns	63
Hierarchical Aggregation	63
Solution Overview	64
Schema Design	65
MapReduce	65
Operations	67
Sharding Concerns	72
<b>5. Ecommerce.....</b>	<b>75</b>
Product Catalog	75
Solution Overview	75
Operations	80
Sharding Concerns	83
Category Hierarchy	84
Solution Overview	84
Schema Design	85
Operations	86
Sharding Concerns	90
Inventory Management	91
Solution Overview	91
Schema	92
Operations	93
Sharding Concerns	100
<b>6. Content Management Systems.....</b>	<b>101</b>

Metadata and Asset Management	101
Solution Overview	101
Schema Design	102
Operations	104
Sharding Concerns	110
Storing Comments	111
Solution Overview	111
Approach: One Document per Comment	111
Approach: Embedding All Comments	114
Approach: Hybrid Schema Design	117
Sharding Concerns	119
<b>7. Online Advertising Networks.....</b>	<b>121</b>
Solution Overview	121
Design 1: Basic Ad Serving	121
Schema Design	122
Operation: Choose an Ad to Serve	123
Operation: Make an Ad Campaign Inactive	123
Sharding Concerns	124
Design 2: Adding Frequency Capping	124
Schema Design	124
Operation: Choose an Ad to Serve	125
Sharding	126
Design 3: Keyword Targeting	126
Schema Design	127
Operation: Choose a Group of Ads to Serve	127
<b>8. Social Networking.....</b>	<b>129</b>
Solution Overview	129
Schema Design	130
Independent Collections	130
Dependent Collections	132
Operations	133
Viewing a News Feed or Wall Posts	134
Commenting on a Post	135
Creating a New Post	136
Maintaining the Social Graph	138
Sharding	139
<b>9. Online Gaming.....</b>	<b>141</b>
Solution Overview	141
Schema Design	142

Character Schema	142
Item Schema	143
Location Schema	144
Operations	144
Load Character Data from MongoDB	145
Extract Armor and Weapon Data for Display	145
Extract Character Attributes, Inventory, and Room Information for Display	147
Pick Up an Item from a Room	147
Remove an Item from a Container	148
Move the Character to a Different Room	149
Buy an Item	150
Sharding	151
<b>Afterword.....</b>	<b>153</b>
<b>Index.....</b>	<b>155</b>

**PART I**

---

# **Design Patterns**



---

# To Embed or Reference

When building a new application, often one of the first things you'll want to do is to design its data model. In relational databases such as MySQL, this step is formalized in the process of normalization, focused on removing redundancy from a set of tables. MongoDB, unlike relational databases, stores its data in structured *documents* rather than the fixed *tables* required in relational databases. For instance, relational tables typically require each row-column intersection to contain a single, scalar value. MongoDB BSON documents allow for more complex structure by supporting arrays of values (where each array itself may be composed of multiple subdocuments).

This chapter explores one of the options that MongoDB's rich document model leaves open to you: the question of whether you should *embed* related objects within one another or *reference* them by ID. Here, you'll learn how to weigh performance, flexibility, and complexity against one another as you make this decision.

## Relational Data Modeling and Normalization

Before jumping into MongoDB's approach to the question of embedding documents or linking documents, we'll take a little detour into how you model certain types of relationships in relational (SQL) databases. In relational databases, data modeling typically progresses by modeling your data as a series of *tables*, consisting of *rows* and *columns*, which collectively define the *schema* of your data. Relational database theory has defined a number of ways of putting application data into tables, referred to as *normal forms*. Although a detailed discussion of relational modeling goes beyond the scope of this text, there are two forms that are of particular interest to us here: first normal form and third normal form.



## What Is a Normal Form, Anyway?

Schema normalization typically begins by putting your application data into the first normal form (1NF). Although there are specific rules that define exactly what 1NF means, that's a little beyond what we want to cover here. For our purposes, we can consider 1NF data to be any data that's *tabular* (composed of rows and columns), with each row-column intersection ("cell") containing exactly one value. This requirement that each cell contains exactly one value is, as we'll see later, a requirement that MongoDB does not impose, with the potential for some nice performance gains. Back in our relational case, let's consider a phone book application. Your initial data might be of the following form, shown in Table 1-1.

Table 1-1. Phone book v1

id	name	phone_number	zip_code
1	Rick	555-111-1234	30062
2	Mike	555-222-2345	30062
3	Jenny	555-333-3456	01209

This data is actually already in first normal form. Suppose, however, that we wished to allow for multiple phone numbers for each contact, as in Table 1-2.

Table 1-2. Phone book v2

id	name	phone_numbers	zip_code
1	Rick	555-111-1234	30062
2	Mike	555-222-2345;555-212-2322	30062
3	Jenny	555-333-3456;555-334-3411	01209

Now we have a table that's no longer in first normal form. If we were to actually store data in this form in a relational database, we would have to decide whether to store `phone_numbers` as an unstructured BLOB of text or as separate columns (i.e., `phone_number0`, `phone_number1`). Suppose we decided to store `phone_numbers` as a text column, as shown in Table 1-2. If we needed to implement something like caller ID, finding the name for a given phone number, our SQL query would look something like the following:

```
SELECT name FROM contacts WHERE phone_numbers LIKE '%555-222-2345%';
```

Unfortunately, using a `LIKE` clause that's not a prefix means that this query requires a full table scan to be satisfied.

Alternatively, we can use multiple columns, one for each phone number, as shown in Table 1-3.