

Time to Relax



CouchDB

The Definitive Guide

O'REILLY®

*J. Chris Anderson,
Jan Lehnardt & Noah Slater*

CouchDB: The Definitive Guide

Three of CouchDB's creators show you how to use this document-oriented database as a standalone application framework or with high-volume, distributed applications. With its simple model for storing, processing, and accessing data, CouchDB is ideal for web applications that handle huge amounts of loosely structured data. That alone would stretch the limits of a relational database, yet CouchDB offers an open source solution that's reliable, scales easily, and responds quickly.

CouchDB works with self-contained data that has loose or ad-hoc connections. It's a model that fits many real-world items, such as contacts, invoices, and receipts, but you'll discover that this database can easily handle data of any kind. With this book, you'll learn how to work with CouchDB through its RESTful web interface, and you'll become familiar with key features such as simple document CRUD (create, read, update, delete), advanced MapReduce, deployment tuning, and more.

- Understand the basics of document-oriented storage and manipulation
- Interact with CouchDB entirely through HTTP, using its RESTful interface
- Model data as self-contained JSON documents
- Handle evolving data schemas naturally
- Query and aggregate data in CouchDB, using MapReduce views
- Replicate data between nodes
- Tune CouchDB for increased performance and reliability

*“This is a great book.
Clean, clear writing
with helpful examples.”*

—**Shelley Powers**
Author, *Learning JavaScript*

J. Chris Anderson is an Apache CouchDB committer and cofounder of Relaxed, Inc. Chris is obsessed with JavaScript CouchApps and bending the physics of the Web to give control back to users.

Jan Lehnardt is an Apache CouchDB committer and cofounder of Relaxed, Inc. Jan hacks on all parts of the web technology stack and focuses on making developers' lives easier.

Noah Slater is an Apache CouchDB committer and release manager. He works with the community to get CouchDB running in as many places as possible.

Experience with Internet programming is recommended.

O'REILLY[®]
oreilly.com

US \$39.99

CAN \$49.99

ISBN: 978-0-596-15589-6



Safari[®]
Books Online

Free online edition

for 45 days with purchase of this book. Details on last page.

CouchDB: The Definitive Guide

CouchDB: The Definitive Guide

J. Chris Anderson, Jan Lehnardt, and Noah Slater

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

CouchDB: The Definitive Guide

by J. Chris Anderson, Jan Lehnardt, and Noah Slater

Copyright © 2010 J. Chris Anderson, Jan Lehnardt, and Noah Slater. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editor: Mike Loukides
Production Editor: Sarah Schneider
Production Services: Appingo, Inc.

Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:

January 2010: First Edition.

O'Reilly and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *CouchDB: The Definitive Guide*, the image of a Pomeranian dog, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. This work has been released under the Creative Commons Attribution License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/2.0/legalcode> or send a letter to Creative Commons, 171 2nd Street, Suite 300, San Francisco, California, 94105, USA.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 978-0-596-15589-6

[M]

1263584573

*For the Web, and all the people who helped me
along the way. Thank you.*

—J. Chris

Für Marita und Kalle.

—Jan

For my parents, God and Damien Katz.

—Noah

Table of Contents

Foreword **xv**

Preface **xvii**

Part I. Introduction

1. Why CouchDB? **3**

- Relax 3
- A Different Way to Model Your Data 4
- A Better Fit for Common Applications 5
 - Self-Contained Data 5
 - Syntax and Semantics 6
- Building Blocks for Larger Systems 6
 - CouchDB Replication 8
- Local Data Is King 8
- Wrapping Up 9

2. Eventual Consistency **11**

- Working with the Grain 11
- The CAP Theorem 12
- Local Consistency 13
 - The Key to Your Data 13
 - No Locking 14
 - Validation 15
- Distributed Consistency 16
 - Incremental Replication 16
 - Case Study 17
- Wrapping Up 20

3. Getting Started	21
All Systems Are Go!	21
Welcome to Futon	23
Your First Database and Document	24
Running a Query Using MapReduce	27
Triggering Replication	31
Wrapping Up	32
4. The Core API	33
Server	33
Databases	34
Documents	38
Revisions	39
Documents in Detail	40
Replication	42
Wrapping Up	44

Part II. Developing with CouchDB

5. Design Documents	47
Document Modeling	47
The Query Server	48
Applications Are Documents	48
A Basic Design Document	51
Looking to the Future	52
6. Finding Your Data with Views	53
What Is a View?	53
Efficient Lookups	56
Find One	56
Find Many	57
Reversed Results	58
The View to Get Comments for Posts	59
Reduce/Rereduce	61
Lessons Learned	64
Wrapping Up	64
7. Validation Functions	67
Document Validation Functions	67
Validation's Context	69
Writing One	69
Type	69

Required Fields	71
Timestamps	72
Authorship	73
Wrapping Up	73
8. Show Functions	75
The Show Function API	76
Side Effect-Free	77
Design Documents	78
Querying Show Functions	78
Design Document Resources	79
Query Parameters	79
Accept Headers	80
Etags	81
Functions and Templates	81
The !json Macro	82
The !code Macro	82
Learning Shows	83
Using Templates	83
Writing Templates	85
9. Transforming Views with List Functions	87
Arguments to the List Function	87
An Example List Function	89
List Theory	91
Querying Lists	92
Lists, Etags, and Caching	93

Part III. Example Application

10. Standalone Applications	97
Use the Correct Version	97
Portable JavaScript	98
Applications Are Documents	99
Standalone	100
In the Wild	101
Wrapping Up	108
11. Managing Design Documents	109
Working with the Example Application	109
Installing CouchApp	110
Using CouchApp	110

Download the Sofa Source Code	111
CouchApp Clone	111
ZIP and TAR Files	111
Join the Sofa Development Community on GitHub	112
The Sofa Source Tree	112
Deploying Sofa	115
Pushing Sofa to Your CouchDB	115
Visit the Application	115
Set Up Your Admin Account	116
Deploying to a Secure CouchDB	117
Configuring CouchApp with .couchapprc	117
12. Storing Documents	119
JSON Document Format	120
Beyond _id and _rev: Your Document Data	122
The Edit Page	123
The HTML Scaffold	124
Saving a Document	125
Validation	128
Save Your First Post	130
Wrapping Up	130
13. Showing Documents in Custom Formats	131
Rendering Documents with Show Functions	132
The Post Page Template	133
Dynamic Dates	134
14. Viewing Lists of Blog Posts	135
Map of Recent Blog Posts	135
Rendering the View as HTML Using a List Function	137
Sofa's List Function	137
The Final Result	141

Part IV. Deploying CouchDB

15. Scaling Basics	145
Scaling Read Requests	146
Scaling Write Requests	146
Scaling Data	147
Basics First	147

16. Replication	149
The Magic	150
Simple Replication with the Admin Interface	150
Replication in Detail	151
Continuous Replication	152
That's It?	152
17. Conflict Management	153
The Split Brain	154
Conflict Resolution by Example	155
Working with Conflicts	158
Deterministic Revision IDs	161
Wrapping Up	161
18. Load Balancing	163
Having a Backup	163
19. Clustering	165
Introducing CouchDB Lounge	165
Consistent Hashing	166
Redundant Storage	167
Redundant Proxies	167
View Merging	167
Growing the Cluster	168
Moving Partitions	169
Splitting Partitions	170

Part V. Reference

20. Change Notifications	173
Polling for Changes	174
Long Polling	175
Continuous Changes	176
Filters	177
Wrapping Up	178
21. View Cookbook for SQL Jockeys	179
Using Views	179
Defining a View	179
Querying a View	180
MapReduce Functions	180
Look Up by Key	181

Look Up by Prefix	182
Aggregate Functions	183
Get Unique Values	185
Enforcing Uniqueness	187
22. Security	189
The Admin Party	189
Creating New Admin Users	190
Hashing Passwords	191
Basic Authentication	191
Update Validations Again	192
Cookie Authentication	193
Network Server Security	194
23. High Performance	195
Good Benchmarks Are Non-Trivial	195
High Performance CouchDB	197
Hardware	197
An Implementation Note	197
Bulk Inserts and Mostly Monotonic DocIDs	198
Optimized Examples: Views and Replication	198
Bulk Document Inserts	198
Batch Mode	199
Single Document Inserts	200
Hovercraft	201
Trade-Offs	201
But...My Boss Wants Numbers!	202
A Call to Arms	202
24. Recipes	205
Banking	205
Accountants Don't Use Erasers	205
Wrapping Up	208
Ordering Lists	208
A List of Integers	208
A List of Floats	210
Pagination	211
Example Data	211
A View	212
Setup	213
Slow Paging (Do Not Use)	213
Fast Paging (Do Use)	215
Jump to Page	216

Part VI. Appendixes

A. Installing on Unix-like Systems	219
B. Installing on Mac OS X	221
C. Installing on Windows	223
D. Installing from Source	225
E. JSON Primer	231
F. The Power of B-trees	233
Index	237

Foreword

As the creator of CouchDB, it gives me great pleasure to write this Foreword. This book has been a long time coming. I've worked on CouchDB since 2005, when it was only a vision in my head and only my wife Laura believed I could make it happen.

Now the project has taken on a life of its own, and code is literally running on millions of machines. I couldn't stop it now if I tried.

A great analogy J. Chris uses is that CouchDB has felt like a boulder we've been pushing up a hill. Over time, it's been moving faster and getting easier to push, and now it's moving so fast it's starting to feel like it could get loose and crush some unlucky villagers. Or something. Hey, remember "Tales of the Runaway Boulder" with Robert Wagner on *Saturday Night Live*? Good times.

Well, now we are trying to safely guide that boulder. Because of the villagers. You know what? This boulder analogy just isn't working. Let's move on.

The reason for this book is that CouchDB is a very different way of approaching data storage. A way that isn't inherently better or worse than the ways before—it's just another tool, another way of thinking about things. It's missing some features you might be used to, but it's gained some abilities you've maybe never seen. Sometimes it's an excellent fit for your problems; sometimes it's terrible.

And sometimes you may be thinking about your problems all wrong. You just need to approach them from a different angle.

Hopefully this book will help you understand CouchDB and the approach that it takes, and also understand how and when it can be used for the problems you face.

Otherwise, someday it could become a runaway boulder, being misused and causing disasters that could have been avoided.

And I'll be doing my best Charlton Heston imitation, on the ground, pounding the dirt, yelling, "You maniacs! You blew it up! Ah, damn you! God damn you all to hell!" Or something like that.

—Damien Katz
Creator of CouchDB

Preface

Thanks for purchasing this book! If it was a gift, then congratulations. If, on the other hand, you downloaded it without paying, well, actually, we're pretty happy about that too! This book is available under a free license, and that's important because we want it to serve the community as documentation—and documentation should be free.

So, why pay for a free book? Well, you might like the warm fuzzy feeling you get from holding a book in your hands, as you cosy up on the couch with a cup of coffee. On the couch...get it? Bad jokes aside, whatever your reasons, buying the book helps support us, so we have more time to work on improvements for both the book and CouchDB. So thank you!

We set out to compile the best and most comprehensive collection of CouchDB information there is, and yet we know we failed. CouchDB is a fast-moving target and grew significantly during the time we were writing the book. We were able to adapt quickly and keep things up-to-date, but we also had to draw the line somewhere if we ever hoped to publish it.

At the time of this writing, CouchDB 0.10.1 is the latest release, but you might already be seeing 0.10.2 or even 0.11.0 released or being prepared—maybe even 1.0. Although we have some ideas about how future releases will look, we don't know for certain and didn't want to make any wild guesses. CouchDB is a community project, so ultimately it's up to you, our readers, to help shape the project.

On the plus side, many people successfully run CouchDB 0.10 in production, and you will have more than enough on your hands to run a solid project. Future releases of CouchDB will make things easier in places, but the core features should remain the same. Besides, learning the core features helps you understand and appreciate the shortcuts and allows you to roll your own hand-tailored solutions.

Writing an open book was great fun. We're happy O'Reilly supported our decision in every way possible. The best part—besides giving the CouchDB community early access to the material—was the commenting functionality we implemented on the [book's website](#). It allows anybody to comment on any paragraph in the book with a simple click. We used some simple JavaScript and Google Groups to allow painless commenting. The result was astounding. As of today, 866 people have sent more than 1,100

messages to our little group. Submissions have ranged from pointing out small typos to deep technical discussions. Feedback on our original first chapter led us to a complete rewrite in order to make sure the points we wanted to get across did, indeed, get across. This system allowed us to clearly formulate what we wanted to say in a way that worked for you, our readers.

Overall, the book has become so much better because of the help of hundreds of volunteers who took the time to send in their suggestions. We understand the immense value this model has, and we want to keep it up. New features in CouchDB should make it into the book without us necessarily having to do a reprint every three months. The publishing industry is not ready for that yet, but we want to continue to release new and revised content and listen closely to the feedback. The specifics of how we'll do this are still in flux, but we'll be posting the information to the book's website the first moment we know it. That's a promise! So make sure to visit the book's website at <http://books.couchdb.org/relax> to keep up-to-date.

Before we let you dive into the book, we want to make sure you're well prepared. CouchDB is written in Erlang, but you don't need to know anything about Erlang to use CouchDB. CouchDB also heavily relies on web technologies like HTTP and JavaScript, and some experience with those does help when following the examples throughout the book. If you have built a website before—simple or complex—you should be ready to go.

If you are an experienced developer or systems architect, the introduction to CouchDB should be comforting, as you already know everything involved—all you need to learn are the ways CouchDB puts them together. Toward the end of the book, we ramp up the experience level to help you get as comfortable building large-scale CouchDB systems as you are with personal projects.

If you are a beginning web developer, don't worry—by the time you get to the later parts of the book, you should be able to follow along with the harder stuff.

Now, sit back, relax, and enjoy the ride through the wonderful world of CouchDB.

Using Code Examples

This book is here to help you get your job done. In general, 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.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/2.0/legalcode> or send a letter to Creative Commons, 171 2nd Street, Suite 300, San Francisco, California, 94105, USA.

An attribution usually includes the title, author, publisher, and ISBN. For example: “*CouchDB: The Definitive Guide* by J. Chris Anderson, Jan Lehnardt, and Noah Slater. Copyright 2010 J. Chris Anderson, Jan Lehnardt, and Noah Slater, 978-0-596-15589-6.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

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

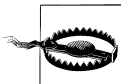
Shows commands or other text that should be typed literally by the user.

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.

Safari® Books Online



Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at <http://my.safaribooksonline.com>.

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://www.oreilly.com/catalog/9780596155896>

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

<http://www.oreilly.com>

Acknowledgments

J. Chris

I would like to acknowledge all the committers of CouchDB, the people sending patches, and the rest of the community. I couldn't have done it without my wife, Amy, who helps me think about the big picture; without the patience and support of my coauthors and O'Reilly; nor without the help of everyone who helped us hammer out book content details on the mailing lists. And a shout-out to the copyeditor, who was awesome!

Jan

I would like to thank the CouchDB community. Special thanks go out to a number of nice people all over the place who invited me to attend or talk at a conference, who let me sleep on their couches (pun most definitely intended), and who made sure I had a good time when I was abroad presenting CouchDB. There are too many to name, but all of you in Dublin, Portland, Lisbon, London, Zurich, San Francisco, Mountain View, Dortmund, Stockholm, Hamburg, Frankfurt, Salt Lake City, Blacksburg, San Diego, and Amsterdam: you know who you are—thanks!

To my family, friends, and coworkers: thanks you for your support and your patience with me over the last year. You won't hear, "I've got to leave early, I have a book to write" from me anytime soon, promise!

Anna, you believe in me; I couldn't have done this without you.

Noah

I would like to thank O'Reilly for their enthusiasm in CouchDB and for realizing the importance of free documentation. And of course, I'd like to thank Jan and J. Chris for being so great to work with. But a special thanks goes out to the whole CouchDB community, for making everything so fun and rewarding. Without you guys, none of this would be possible. And if you're reading this, that means you!

PART I

Introduction

Why CouchDB?

[Download at WoweBook.com](http://WoweBook.com)

Apache CouchDB is one of a new breed of database management systems. This chapter explains why there's a need for new systems as well as the motivations behind building CouchDB.

As CouchDB developers, we're naturally very excited to be using CouchDB. In this chapter we'll share with you the reasons for our enthusiasm. We'll show you how CouchDB's schema-free document model is a better fit for common applications, how the built-in query engine is a powerful way to use and process your data, and how CouchDB's design lends itself to modularization and scalability.

Relax

If there's one word to describe CouchDB, it is *relax*. It is in the title of this book, it is the byline to CouchDB's official logo, and when you start CouchDB, you see:

Apache CouchDB has started. Time to relax.

Why is relaxation important? Developer productivity roughly doubled in the last five years. The chief reason for the boost is more powerful tools that are easier to use. Take Ruby on Rails as an example. It is an infinitely complex framework, but it's easy to get started with. Rails is a success story because of the core design focus on ease of use. This is one reason why CouchDB is relaxing: learning CouchDB and understanding its core concepts should feel natural to most everybody who has been doing any work on the Web. And it is still pretty easy to explain to non-technical people.

Getting out of the way when creative people try to build specialized solutions is in itself a core feature and one thing that CouchDB aims to get right. We found existing tools too cumbersome to work with during development or in production, and decided to focus on making CouchDB easy, even a pleasure, to use. Chapters 3 and 4 will demonstrate the intuitive HTTP-based REST API.

Another area of relaxation for CouchDB users is the production setting. If you have a live running application, CouchDB again goes out of its way to avoid troubling you.

Its internal architecture is fault-tolerant, and failures occur in a controlled environment and are dealt with gracefully. Single problems do not cascade through an entire server system but stay isolated in single requests.

CouchDB’s core concepts are simple (yet powerful) and well understood. Operations teams (if you have a team; otherwise, that’s you) do not have to fear random behavior and untraceable errors. If anything should go wrong, you can easily find out what the problem is—but these situations are rare.

CouchDB is also designed to handle varying traffic gracefully. For instance, if a website is experiencing a sudden spike in traffic, CouchDB will generally absorb a lot of concurrent requests without falling over. It may take a little more time for each request, but they all get answered. When the spike is over, CouchDB will work with regular speed again.

The third area of relaxation is growing and shrinking the underlying hardware of your application. This is commonly referred to as *scaling*. CouchDB enforces a set of limits on the programmer. On first look, CouchDB might seem inflexible, but some features are left out by design for the simple reason that if CouchDB supported them, it would allow a programmer to create applications that couldn’t deal with scaling up or down. We’ll explore the whole matter of scaling CouchDB in [Part IV, *Deploying CouchDB*](#).

In a nutshell: CouchDB doesn’t let you do things that would get you in trouble later on. This sometimes means you’ll have to unlearn best practices you might have picked up in your current or past work. [Chapter 24](#) contains a list of common tasks and how to solve them in CouchDB.

A Different Way to Model Your Data

We believe that CouchDB will drastically change the way you build document-based applications. CouchDB combines an intuitive document storage model with a powerful query engine in a way that’s so simple you’ll probably be tempted to ask, “Why has no one built something like this before?”

Django may be built *for* the Web, but CouchDB is built *of* the Web. I’ve never seen software that so completely embraces the philosophies behind HTTP. CouchDB makes Django look old-school in the same way that Django makes ASP look outdated.

—Jacob Kaplan-Moss, Django developer

CouchDB’s design borrows heavily from web architecture and the concepts of resources, methods, and representations. It augments this with powerful ways to query, map, combine, and filter your data. Add fault tolerance, extreme scalability, and incremental replication, and CouchDB defines a sweet spot for document databases.

A Better Fit for Common Applications

We write software to improve our lives and the lives of others. Usually this involves taking some mundane information—such as contacts, invoices, or receipts—and manipulating it using a computer application. CouchDB is a great fit for common applications like this because it embraces the natural idea of evolving, self-contained documents as the very core of its data model.

Self-Contained Data

An invoice contains all the pertinent information about a single transaction—the seller, the buyer, the date, and a list of the items or services sold. As shown in [Figure 1-1](#), there’s no abstract reference on this piece of paper that points to some other piece of paper with the seller’s name and address. Accountants appreciate the simplicity of having everything in one place. And given the choice, programmers appreciate that, too.

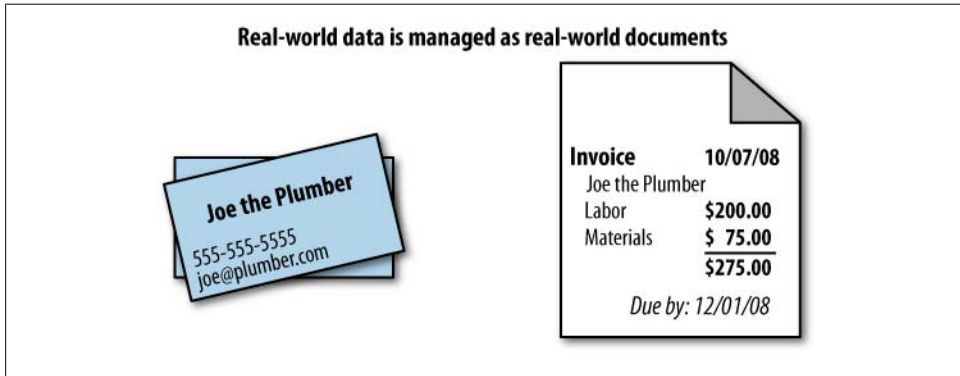


Figure 1-1. Self-contained documents

Yet using references is exactly how we model our data in a relational database! Each invoice is stored in a table as a row that refers to other rows in other tables—one row for seller information, one for the buyer, one row for each item billed, and more rows still to describe the item details, manufacturer details, and so on and so forth.

This isn’t meant as a detraction of the relational model, which is widely applicable and extremely useful for a number of reasons. Hopefully, though, it illustrates the point that sometimes your model may not “fit” your data in the way it occurs in the real world.

Let’s take a look at the humble contact database to illustrate a different way of modeling data, one that more closely “fits” its real-world counterpart—a pile of business cards. Much like our invoice example, a business card contains all the important information, right there on the cardstock. We call this “self-contained” data, and it’s an important concept in understanding document databases like CouchDB.

Syntax and Semantics

Most business cards contain roughly the same information—someone’s identity, an affiliation, and some contact information. While the exact form of this information can vary between business cards, the general information being conveyed remains the same, and we’re easily able to recognize it as a business card. In this sense, we can describe a business card as a *real-world document*.

Jan’s business card might contain a phone number but no fax number, whereas J. Chris’s business card contains both a phone and a fax number. Jan does not have to make his lack of a fax machine explicit by writing something as ridiculous as “Fax: None” on the business card. Instead, simply omitting a fax number implies that he doesn’t have one.

We can see that real-world documents of the same type, such as business cards, tend to be very similar in *semantics*—the sort of information they carry—but can vary hugely in *syntax*, or how that information is structured. As human beings, we’re naturally comfortable dealing with this kind of variation.

While a traditional relational database requires you to model your data *up front*, CouchDB’s schema-free design unburdens you with a powerful way to aggregate your data *after the fact*, just like we do with real-world documents. We’ll look in depth at how to design applications with this underlying storage paradigm.

Building Blocks for Larger Systems

CouchDB is a storage system useful on its own. You can build many applications with the tools CouchDB gives you. But CouchDB is designed with a bigger picture in mind. Its components can be used as building blocks that solve storage problems in slightly different ways for larger and more complex systems.

Whether you need a system that’s crazy fast but isn’t too concerned with reliability (think logging), or one that guarantees storage in two or more physically separated locations for reliability, but you’re willing to take a performance hit, CouchDB lets you build these systems.

There are a multitude of knobs you could turn to make a system work better in one area, but you’ll affect another area when doing so. One example would be the CAP theorem discussed in the next chapter. To give you an idea of other things that affect storage systems, see Figures 1-2 and 1-3.

By reducing latency for a given system (and that is true not only for storage systems), you affect concurrency and throughput capabilities.