O'REILLY®

2nd Edition

Designing Data-Intensive Applications

The Big Ideas Behind Reliable, Scalable, and Maintainable Systems

Martin Kleppmann
& Chris Riccomini

"For the last decade, this has been the best book for understanding distributed systems, and the second edition makes it even better. It bridges the huge gap between distributed systems theory and practical engineering. I wish it had existed earlier so I could have saved myself a lot of mistakes."

Jay Kreps, creator of Apache Kafka and cofounder of Confluent

"This book should be required reading for software engineers. *Designing Data-Intensive Applications* is a rare resource that connects theory and practice to help developers make smart decisions as they design and implement data infrastructure and systems."

Kevin Scott, Chief Technology Officer at Microsoft

Designing Data-Intensive Applications

Data is at the center of many challenges in system design today. Difficult issues such as scalability, consistency, reliability, efficiency, and maintainability need to be resolved. In addition, there's an overwhelming variety of systems, including relational databases, NoSQL datastores, data warehouses, and data lakes. There are cloud services, on-premises services, and embedded databases. What are the right choices for your application? How do you make sense of all these buzzwords?

In this second edition, authors Martin Kleppmann and Chris Riccomini build on the foundation laid in the acclaimed first edition, integrating new technologies and emerging trends. You'll be guided through the maze of decisions and trade-offs involved in building a modern data system, learn how to choose the right tools for your needs, and understand the fundamentals of distributed systems.

- **Peer under the hood of the systems you already use, and learn to use them more effectively**
- **Make informed decisions by identifying the strengths and weaknesses of different tools**
- **Learn how major cloud services are designed for scalability, fault tolerance, and consistency**
- **Understand the core principles upon which modern databases are built**

Martin Kleppmann is an associate professor of distributed systems at the University of Cambridge. Previously he was a startup founder and a software engineer at LinkedIn, working on large-scale data systems.

Chris Riccomini is a software engineer, startup investor, and author. He cocreated Apache Samza and SlateDB, coauthored *The Missing README*, and runs Materialized View Capital.

DATA

US $69.99 CAN $87.99
ISBN: 978-1-098-11906-5

O'REILLY®

Praise for *Designing Data-Intensive Applications*

Designing Data-Intensive Applications has become the Bible of distributed systems—almost every serious practitioner I know owns a copy.

—Will Wilson, CEO at Antithesis

Mastering trade-offs is essential to solving real-world problems with distributed data systems. *Designing Data-Intensive Applications* explores them like none other, providing an unbiased view of how different systems have made these choices over time.

—Veena Basavaraj, head of application platform engineering at Benchling

An essential guide for distributed systems engineers—thorough and insightful for both beginners and experts.

—Zhengliang "Zane" Zhu, distributed systems engineer, Netflix

This book is a gateway to distributed systems. It's the best-in-class book for getting up to speed on concepts every systems engineer should know.

—Alex Petrov, author of Database Internals*, Apache Cassandra committer, and PMC member*

For the last decade, this has been the best book for understanding distributed systems, and the second edition makes it even better. It bridges the huge gap between distributed systems theory and practical engineering. I wish it had existed when I started working on distributed systems so I could have read it then and saved myself a lot of mistakes.

—Jay Kreps, creator of Apache Kafka and cofounder of Confluent

This book should be required reading for software engineers. *Designing Data-Intensive Applications* is a rare resource that connects theory and practice to help developers make smart decisions as they design and implement data infrastructure and systems.

—Kevin Scott, Chief Technology Officer at Microsoft

SECOND EDITION

Designing Data-Intensive Applications

The Big Ideas Behind Reliable, Scalable, and Maintainable Systems

Martin Kleppmann and Chris Riccomini

O'REILLY®

Designing Data-Intensive Applications
by Martin Kleppmann and Chris Riccomini

Published by O'Reilly Media, Inc., 141 Stony Circle, Suite 195, Santa Rosa, CA 95401.

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

Acquisitions Editor: Aaron Black
Development Editor: Melissa Potter
Production Editor: Katherine Tozer
Copyeditor: Rachel Wheeler
Proofreader: Sharon Wilkey
Indexer: Potomac Indexing, LLC
Cover Designer: Susan Brown
Cover Illustrator: Monica Kamsvaag
Interior Designer: David Futato
Interior Illustrator: Kate Dullea

March 2017: First Edition
February 2026: Second Edition

Revision History for the Second Edition
2026-02-18: First Release

See *https://oreilly.com/catalog/errata.csp?isbn=9781098119065* for release details.

978-1-098-11906-5

[LSI]

To everyone using technology and data to address the world's biggest problems.

Computing is pop culture. […] Pop culture holds a disdain for history. Pop culture is all about identity and feeling like you're participating. It has nothing to do with cooperation, the past or the future—it's living in the present. I think the same is true of most people who write code for money. They have no idea where [their culture came from].

—Alan Kay, in interview with *Dr. Dobb's Journal* (2012)

Table of Contents

Preface xvii

1. Trade-Offs in Data Systems Architecture 1
Operational Versus Analytical Systems 3
Characterizing Transaction Processing and Analytics 5
Data Warehousing 7
Systems of Record and Derived Data 10
Cloud Versus Self-Hosting 12
Pros and Cons of Cloud Services 13
Cloud Native System Architecture 14
Operations in the Cloud Era 17
Distributed Versus Single-Node Systems 19
Problems with Distributed Systems 20
Microservices and Serverless 21
Cloud Computing Versus Supercomputing 23
Data Systems, Law, and Society 24
Summary 25

2. Defining Nonfunctional Requirements 33
Case Study: Social Network Home Timelines 34
Representing Users, Posts, and Follows 34
Materializing and Updating Timelines 35
Describing Performance 37
Latency and Response Time 38
Average, Median, and Percentiles 40
Use of Response Time Metrics 41
Reliability and Fault Tolerance 43

Fault Tolerance 43
Hardware and Software Faults 44
Humans and Reliability 47
Scalability 49
Understanding Load 50
Shared-Memory, Shared-Disk, and Shared-Nothing Architectures 51
Principles for Scalability 52
Maintainability 52
Operability: Making Life Easy for Operations 53
Simplicity: Managing Complexity 54
Evolvability: Making Change Easy 55
Summary 56

3. Data Models and Query Languages. 65
Relational Versus Document Models 67
The Object-Relational Mismatch 68
Normalization, Denormalization, and Joins 72
Many-to-One and Many-to-Many Relationships 75
Stars and Snowflakes: Schemas for Analytics 77
When to Use Which Model 80
Graph-Like Data Models 84
Property Graphs 86
The Cypher Query Language 88
Graph Queries in SQL 90
Triple Stores and SPARQL 92
Datalog: Recursive Relational Queries 96
GraphQL 98
Event Sourcing and CQRS 101
DataFrames, Matrices, and Arrays 105
Summary 107

4. Storage and Retrieval. 115
Storage and Indexing for OLTP 116
Log-Structured Storage 118
B-Trees 125
Comparing B-Trees and LSM-Trees 129
Multicolumn and Secondary Indexes 132
Storing Values Within the Index 133
Keeping Everything in Memory 133
Data Storage for Analytics 134
Cloud Data Warehouses 135

Column-Oriented Storage 136
Query Execution: Compilation and Vectorization 142
Materialized Views and Data Cubes 143
Multidimensional and Full-Text Indexes 145
Full-Text Search 146
Vector Embeddings 147
Summary 150

5. Encoding and Evolution. 161
Formats for Encoding Data 163
Language-Specific Formats 164
JSON, XML, and Binary Variants 165
Protocol Buffers 169
Avro 172
The Merits of Schemas 177
Modes of Dataflow 178
Dataflow Through Databases 178
Dataflow Through Services: REST and RPC 180
Durable Execution and Workflows 187
Event-Driven Architectures 189
Summary 191

6. Replication. 197
Single-Leader Replication 198
Synchronous Versus Asynchronous Replication 200
Setting Up New Followers 201
Handling Node Outages 204
Implementation of Replication Logs 206
Problems with Replication Lag 209
Solutions for Replication Lag 214
Multi-Leader Replication 215
Geographically Distributed Operation 216
Sync Engines and Local-First Software 220
Dealing with Conflicting Writes 222
Leaderless Replication 229
Writing to the Database When a Node Is Down 229
Single-Leader Versus Leaderless Replication Performance 235
Multi-Region Operation 236
Detecting Concurrent Writes 237
Summary 243

7. Sharding. 251
Pros and Cons of Sharding 253
Sharding for Multitenancy 254
Sharding of Key-Value Data 255
Sharding by Key Range 256
Sharding by Hash of Key 258
Skewed Workloads and Relieving Hot Spots 263
Operations: Automatic Versus Manual Rebalancing 264
Request Routing 265
Sharding and Secondary Indexes 268
Local Secondary Indexes 268
Global Secondary Indexes 270
Summary 271

8. Transactions. 277
What Exactly Is a Transaction? 278
The Meaning of ACID 279
Single-Object and Multi-Object Operations 284
Weak Isolation Levels 288
Read Committed 290
Snapshot Isolation and Repeatable Read 293
Preventing Lost Updates 299
Write Skew and Phantoms 303
Serializability 308
Actual Serial Execution 309
Two-Phase Locking 313
Serializable Snapshot Isolation 317
Distributed Transactions 323
Two-Phase Commit 324
Distributed Transactions Across Different Systems 328
Database-Internal Distributed Transactions 333
Exactly-Once Message Processing Revisited 334
Summary 335

9. The Trouble with Distributed Systems. 345
Faults and Partial Failures 346
Unreliable Networks 347
The Limitations of TCP 348
Network Faults in Practice 350
Fault Detection 351
Timeouts and Unbounded Delays 352

Synchronous Versus Asynchronous Networks 355
Unreliable Clocks 358
Monotonic Versus Time-of-Day Clocks 359
Clock Synchronization and Accuracy 360
Relying on Synchronized Clocks 362
Process Pauses 366
Knowledge, Truth, and Lies 371
The Majority Rules 372
Distributed Locks and Leases 373
Byzantine Faults 377
System Model and Reality 380
Formal Methods and Randomized Testing 384
Summary 388

10. Consistency and Consensus. 401
Linearizability 402
What Makes a System Linearizable? 404
Relying on Linearizability 408
Implementing Linearizable Systems 411
The Cost of Linearizability 413
ID Generators and Logical Clocks 417
Logical Clocks 420
Linearizable ID Generators 423
Consensus 425
The Many Faces of Consensus 427
Consensus in Practice 433
Coordination Services 437
Summary 440

11. Batch Processing. 451
Batch Processing with Unix Tools 454
Simple Log Analysis 454
Chain of Commands Versus Custom Program 456
Sorting Versus In-Memory Aggregation 456
Batch Processing in Distributed Systems 457
Distributed Filesystems 458
Object Stores 460
Distributed Job Orchestration 461
Batch Processing Models 466
MapReduce 466
Dataflow Engines 468

Shuffling Data 469
Joins and Grouping 471
Query Languages 473
DataFrames 475
Batch Use Cases 476
Extract–Transform–Load 476
Analytics 477
Machine Learning 478
Serving Derived Data 479
Summary 481

12. Stream Processing. 487
Transmitting Event Streams 488
Messaging Systems 489
Log-Based Message Brokers 495
Databases and Streams 500
Keeping Systems in Sync 501
Change Data Capture 503
State, Streams, and Immutability 508
Processing Streams 513
Uses of Stream Processing 514
Reasoning About Time 518
Stream Joins 523
Fault Tolerance 526
Summary 529

13. A Philosophy of Streaming Systems. 539
Data Integration 539
Combining Specialized Tools by Deriving Data 540
Batch and Stream Processing 544
Unbundling Databases 546
Composing Data Storage Technologies 547
Designing Applications Around Dataflow 551
Observing Derived State 555
Aiming for Correctness 561
The End-to-End Argument for Databases 562
Enforcing Constraints 566
Timeliness and Integrity 571
Trust, but Verify 575
Summary 579

14. Doing the Right Thing. 585
Predictive Analytics 586
Bias and Discrimination 586
Responsibility and Accountability 587
Feedback Loops 588
Privacy and Tracking 589
Surveillance 590
Consent and Freedom of Choice 591
Privacy and Use of Data 592
Data as Assets and Power 594
Remembering the Industrial Revolution 595
Legislation and Self-Regulation 596
Summary 597

Glossary. 603

Index. 609

Preface

There are hundreds of databases to choose from. Which one should you use for your application? The short answer is, "It depends." The long answer is…this book.

Different technologies for storing and processing data make different trade-offs, and no one approach is best for all situations. The system that is a perfect fit for one application is badly suited to another. This book is a guide to the entire landscape of data systems, not just looking at one product, but comparing the strengths and weaknesses of many systems.

Although the landscape of technologies for processing and storing data is diverse and fast-changing, the underlying principles endure. If you understand those principles, you're in a position to see where each tool fits in, how to make good use of it, and how to avoid its pitfalls. This book focuses on those principles.

While this book is not a tutorial for using one particular tool, it isn't a textbook full of dry theory either. Instead, we will look at many examples of successful data systems: technologies that form the foundation of numerous popular applications and that have to meet scalability, performance, and reliability requirements in production every day. We will dig into the internals of those systems, tease apart their key algorithms, and discuss the trade-offs they have made. On this journey, we will try to find useful ways of *thinking about* data systems—not just *how* they work, but also *why* they work that way.

After reading this book, you will be in a great position to determine which kinds of technologies are appropriate for which purposes and to understand how tools can be combined to form the foundation of a solid application architecture. You will develop a strong intuition for what your systems are doing under the hood so that you can reason about their behavior, make good design decisions, and track down any problems that may arise.

The guiding philosophy of this book is to bring together a broad range of perspectives: both theoretical and practical, both recent and old. The computing industry tends to be attracted to things that are new and shiny and to look down on ideas that are considered old or academic. That is a mistake; many powerful, foundational ideas in computing arose from research, some recent, some decades ago. On the other hand, academia sometimes lacks a clear idea of what issues are important in practice. This book combines the best of both: academic precision and attention to detail with an industrial focus on practicality.

Who Should Read This Book?

If any of the following are true for you, you'll find this book valuable:

- You're a software engineer, software architect, or technical manager who needs to make decisions about the architecture of the systems you work on—for example, you need to choose tools for solving a given problem and figure out how best to apply them. This applies especially to backend systems.
- You're a data engineer who wants to understand the wider context of the systems you deal with, or a cloud engineer who wants insights into the underpinnings of the systems you're using. You will find that even though modern distributed systems hide a lot of complexity from you, understanding their underlying principles is extremely useful for performance optimization and debugging.
- You want to learn how to make data systems scalable (e.g., to support apps with millions of users), highly available (minimizing downtime), operationally robust, and easier to maintain in the long run (even as they grow and as requirements and technologies change).
- You are preparing for a "system design" job interview in which you will be asked to sketch an architecture for an application, and you need to learn the principles for good data architectures.
- You are curious to find out what goes on behind the scenes at major websites and online services, and inside various databases and data processing systems—especially if you like to dig deeper than buzzwords to gain a technically accurate and precise understanding of various technologies and their trade-offs.

This book assumes that you already have some experience building web-based applications and that you are familiar with relational databases and SQL. A high-level understanding of common network protocols like TCP and HTTP is helpful. Your choice of programming language or framework makes no difference for this book.