

# *Making Sense of NoSQL*

*A GUIDE FOR MANAGERS  
AND THE REST OF US*

DAN McCREARY  
ANN KELLY



MANNING  
SHELTER ISLAND

# *brief contents*

---

<b>PART 1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1	■ NoSQL: It's about making intelligent choices	3
2	■ NoSQL concepts	15
<b>PART 2</b>	<b>DATABASE PATTERNS .....</b>	<b>35</b>
3	■ Foundational data architecture patterns	37
4	■ NoSQL data architecture patterns	62
5	■ Native XML databases	96
<b>PART 3</b>	<b>NOSQL SOLUTIONS .....</b>	<b>125</b>
6	■ Using NoSQL to manage big data	127
7	■ Finding information with NoSQL search	154
8	■ Building high-availability solutions with NoSQL	172
9	■ Increasing agility with NoSQL	192
<b>PART 4</b>	<b>ADVANCED TOPICS .....</b>	<b>207</b>
10	■ NoSQL and functional programming	209
11	■ Security: protecting data in your NoSQL systems	232
12	■ Selecting the right NoSQL solution	254



# contents

---

*foreword xvii*  
*preface xix*  
*acknowledgments xxi*  
*about this book xxii*

## **PART 1 INTRODUCTION.....1**

### **1 NoSQL: It's about making intelligent choices 3**

- 1.1 What is NoSQL? 4
- 1.2 NoSQL business drivers 6
  - Volume 7* ▪ *Velocity 7* ▪ *Variability 7* ▪ *Agility 8*
- 1.3 NoSQL case studies 8
  - Case study: LiveJournal's Memcache 9* ▪ *Case study: Google's MapReduce—use commodity hardware to create search indexes 10*
  - Case study: Google's Bigtable—a table with a billion rows and a million columns 11* ▪ *Case study: Amazon's Dynamo—accept an order 24 hours a day, 7 days a week 11* ▪ *Case study: MarkLogic 12*
  - Applying your knowledge 12*
- 1.4 Summary 13

## 2 *NoSQL concepts* 15

- 2.1 Keeping components simple to promote reuse 15
- 2.2 Using application tiers to simplify design 17
- 2.3 Speeding performance by strategic use of RAM, SSD, and disk 21
- 2.4 Using consistent hashing to keep your cache current 22
- 2.5 Comparing ACID and BASE—two methods of reliable database transactions 24
  - RDBMS transaction control using ACID* 25
  - *Non-RDBMS transaction control using BASE* 27
- 2.6 Achieving horizontal scalability with database sharding 28
- 2.7 Understanding trade-offs with Brewer’s CAP theorem 30
- 2.8 Apply your knowledge 32
- 2.9 Summary 33
- 2.10 Further reading 33

## PART 2 DATABASE PATTERNS ..... 35

### 3 *Foundational data architecture patterns* 37

- 3.1 What is a data architecture pattern? 38
- 3.2 Understanding the row-store design pattern used in RDBMSs 39
  - How row stores work* 39
  - *Row stores evolve* 41
  - Analyzing the strengths and weaknesses of the row-store pattern* 42
- 3.3 Example: Using joins in a sales order 43
- 3.4 Reviewing RDBMS implementation features 45
  - RDBMS transactions* 45
  - *Fixed data definition language and typed columns* 47
  - *Using RDBMS views for security and access control* 48
  - *RDBMS replication and synchronization* 49
- 3.5 Analyzing historical data with OLAP, data warehouse, and business intelligence systems 51
  - How data flows from operational systems to analytical systems* 52
  - Getting familiar with OLAP concepts* 54
  - *Ad hoc reporting using aggregates* 55

- 3.6 Incorporating high availability and read-mostly systems 57
- 3.7 Using hash trees in revision control systems and database synchronization 58
- 3.8 Apply your knowledge 60
- 3.9 Summary 60
- 3.10 Further reading 61

## 4 *NoSQL data architecture patterns* 62

- 4.1 Key-value stores 63
  - What is a key-value store?* 64
  - *Benefits of using a key-value store* 65
  - *Using a key-value store* 68
  - *Use case: storing web pages in a key-value store* 70
  - *Use case: Amazon simple storage service (S3)* 71
- 4.2 Graph stores 72
  - Overview of a graph store* 72
  - *Linking external data with the RDF standard* 74
  - *Use cases for graph stores* 75
- 4.3 Column family (Bigtable) stores 81
  - Column family basics* 82
  - *Understanding column family keys* 82
  - *Benefits of column family systems* 83
  - Case study: storing analytical information in Bigtable* 85
  - Case study: Google Maps stores geographic information in Bigtable* 85
  - *Case study: using a column family to store user preferences* 86
- 4.4 Document stores 86
  - Document store basics* 87
  - *Document collections* 88
  - Application collections* 88
  - *Document store APIs* 89
  - Document store implementations* 89
  - *Case study: ad server with MongoDB* 90
  - *Case study: CouchDB, a large-scale object database* 91
- 4.5 Variations of NoSQL architectural patterns 91
  - Customization for RAM or SSD stores* 92
  - *Distributed stores* 92
  - Grouping items* 93
- 4.6 Summary 95
- 4.7 Further reading 95

- 5 Native XML databases 96**
- 5.1 What is a native XML database? 97
  - 5.2 Building applications with a native XML database 100
    - Loading data can be as simple as drag-and-drop 101* ▪ *Using collections to group your XML documents 102* ▪ *Applying simple queries to transform complex data with XPath 105*
    - Transforming your data with XQuery 106* ▪ *Updating documents with XQuery updates 109* ▪ *XQuery full-text search standards 110*
  - 5.3 Using XML standards within native XML databases 110
  - 5.4 Designing and validating your data with XML Schema and Schematron 112
    - XML Schema 112* ▪ *Using Schematron to check document rules 113*
  - 5.5 Extending XQuery with custom modules 115
  - 5.6 Case study: using NoSQL at the Office of the Historian at the Department of State 115
  - 5.7 Case study: managing financial derivatives with MarkLogic 119
    - Why financial derivatives are difficult to store in RDBMSs 119*
    - An investment bank switches from 20 RDBMSs to one native XML system 119* ▪ *Business benefits of moving to a native XML document store 121* ▪ *Project results 122*
  - 5.8 Summary 122
  - 5.9 Further reading 123

## **PART 3 NOSQL SOLUTIONS ..... 125**

- 6 Using NoSQL to manage big data 127**
- 6.1 What is a big data NoSQL solution? 128
  - 6.2 Getting linear scaling in your data center 132
  - 6.3 Understanding linear scalability and expressivity 133
  - 6.4 Understanding the types of big data problems 135
  - 6.5 Analyzing big data with a shared-nothing architecture 136
  - 6.6 Choosing distribution models: master-slave versus peer-to-peer 137

- 6.7 Using MapReduce to transform your data over distributed systems 139
  - MapReduce and distributed filesystems* 140
  - *How MapReduce allows efficient transformation of big data problems* 142
  
- 6.8 Four ways that NoSQL systems handle big data problems 143
  - Moving queries to the data, not data to the queries* 143
  - *Using hash rings to evenly distribute data on a cluster* 144
  - *Using replication to scale reads* 145
  - *Letting the database distribute queries evenly to data nodes* 146
  
- 6.9 Case study: event log processing with Apache Flume 146
  - Challenges of event log data analysis* 147
  - *How Apache Flume works to gather distributed event data* 148
  - *Further thoughts* 149
  
- 6.10 Case study: computer-aided discovery of health care fraud 150
  - What is health care fraud detection?* 150
  - *Using graphs and custom shared-memory hardware to detect health care fraud* 151
  
- 6.11 Summary 152
- 6.12 Further reading 153
  
- 7 Finding information with NoSQL search 154**
  - 7.1 What is NoSQL search? 155
  - 7.2 Types of search 155
    - Comparing Boolean, full-text keyword, and structured search models* 155
    - *Examining the most common types of search* 156
  - 7.3 Strategies and methods that make NoSQL search effective 158
  - 7.4 Using document structure to improve search quality 161
  - 7.5 Measuring search quality 162
  - 7.6 In-node indexes versus remote search services 163
  - 7.7 Case study: using MapReduce to create reverse indexes 164
  - 7.8 Case study: searching technical documentation 166
    - What is technical document search?* 166
    - *Retaining document structure in a NoSQL document store* 167



- 7.9 Case study: searching domain-specific languages—  
findability and reuse 168
- 7.10 Apply your knowledge 170
- 7.11 Summary 170
- 7.12 Further reading 171

## 8 *Building high-availability solutions with NoSQL* 172

- 8.1 What is a high-availability NoSQL database? 173
- 8.2 Measuring availability of NoSQL databases 174
  - Case study: the Amazon's S3 SLA* 176
  - *Predicting system availability* 176
  - *Apply your knowledge* 177
- 8.3 NoSQL strategies for high availability 178
  - Using a load balancer to direct traffic to the least busy node* 178
  - *Using high-availability distributed filesystems with NoSQL databases* 179
  - *Case study: using HDFS as a high-availability filesystem to store master data* 180
  - Using a managed NoSQL service* 182
  - Case study: using Amazon DynamoDB for a high-availability data store* 182
- 8.4 Case study: using Apache Cassandra as a high-availability column family store 184
  - Configuring data to node mappings with Cassandra* 185
- 8.5 Case study: using Couchbase as a high-availability document store 187
- 8.6 Summary 189
- 8.7 Further reading 190

## 9 *Increasing agility with NoSQL* 192

- 9.1 What is software agility? 193
  - Apply your knowledge: local or cloud-based deployment?* 195
- 9.2 Measuring agility 196
- 9.3 Using document stores to avoid object-relational mapping 199
- 9.4 Case study: using XRX to manage complex forms 201
  - What are complex business forms?* 201
  - *Using XRX to replace client JavaScript and object-relational mapping* 202
  - Understanding the impact of XRX on agility* 205

- 9.5 Summary 205
- 9.6 Further reading 206

## PART 4 ADVANCED TOPICS .....207

### 10 *NoSQL and functional programming* 209

- 10.1 What is functional programming? 210
  - Imperative programming is managing program state* 211
  - *Functional programming is parallel transformation without side effects* 213
  - *Comparing imperative and functional programming at scale* 216
  - *Using referential transparency to avoid recalculating transforms* 217
- 10.2 Case study: using NetKernel to optimize web page content assembly 219
  - Assembling nested content and tracking component dependencies* 219
  - *Using NetKernel to optimize component regeneration* 220
- 10.3 Examples of functional programming languages 222
- 10.4 Making the transition from imperative to functional programming 223
  - Using functions as a parameter of a function* 223
  - *Using recursion to process unstructured document data* 224
  - *Moving from mutable to immutable variables* 224
  - *Removing loops and conditionals* 224
  - *The new cognitive style: from capturing state to isolated transforms* 225
  - *Quality, validation, and consistent unit testing* 225
  - *Concurrency in functional programming* 226
- 10.5 Case study: building NoSQL systems with Erlang 226
- 10.6 Apply your knowledge 229
- 10.7 Summary 230
- 10.8 Further reading 231

### 11 *Security: protecting data in your NoSQL systems* 232

- 11.1 A security model for NoSQL databases 233
  - Using services to mitigate the need for in-database security* 235
  - Using data warehouses and OLAP to mitigate the need for in-database security* 235
  - *Summary of application versus database-layer security benefits* 236

- 11.2 Gathering your security requirements 237
  - Authentication* 237 ▪ *Authorization* 240 ▪ *Audit and logging* 242 ▪ *Encryption and digital signatures* 243
  - Protecting public websites from denial of service and injection attacks* 245
- 11.3 Case Study: access controls on key-value store—Amazon S3 246
  - Identity and Access Management (IAM)* 247 ▪ *Access-control lists (ACL)* 247 ▪ *Bucket policies* 248
- 11.4 Case study: using key visibility with Apache Accumulo 249
- 11.5 Case study: using MarkLogic’s RBAC model in secure publishing 250
  - Using the MarkLogic RBAC security model to protect documents* 250 ▪ *Using MarkLogic in secure publishing* 251
  - Benefits of the MarkLogic security model* 252
- 11.6 Summary 252
- 11.7 Further reading 253

## 12 *Selecting the right NoSQL solution* 254

- 12.1 What is architecture trade-off analysis? 255
- 12.2 Team dynamics of database architecture selection 257
  - Selecting the right team* 258 ▪ *Accounting for experience bias* 259 ▪ *Using outside consultants* 259
- 12.3 Steps in architectural trade-off analysis 260
- 12.4 Analysis through decomposition: quality trees 263
  - Sample quality attributes* 264 ▪ *Evaluating hybrid and cloud architectures* 266
- 12.5 Communicating the results to stakeholders 267
  - Using quality trees as navigational maps* 267 ▪ *Apply your knowledge* 269 ▪ *Using quality trees to communicate project risks* 270
- 12.6 Finding the right proof-of-architecture pilot project 271
- 12.7 Summary 273
- 12.8 Further reading 274
  - index* 275