

O'REILLY®



Vector Databases

A Practical Introduction

Nitin Borwankar

"Vector Databases by Nitin Borwankar demystifies the technology powering modern AI—from embeddings to semantic search and RAG systems. If you're building AI applications today, this is a book you need to understand and apply these concepts in practice."

Tom Taulli

Author of *AI-Assisted Programming*

Vector Databases

The AI revolution is here, and at its core lies a game-changing technology that most developers haven't fully explored: vector databases. From powering semantic search to enabling large language models (LLMs) and generative AI, vector databases are reshaping how we build applications with unstructured data like text, images, and audio. But how do you go from curious to capable with this vital technology? That's where this book comes in.

In this hands-on guide, author Nitin Borwankar takes you through the why, what, and how of vector databases, starting with the basic theory behind vector embeddings and progressing to building applications with real-world tools. You'll learn about Word2vec, how to convert open source SQL databases like SQLite3 and PostgreSQL into vector databases, and how to integrate them into retrieval-augmented generation (RAG) applications. Whether you're a Python developer, data engineer, or ML practitioner, this book gives you the foundation to leverage vector databases confidently in your AI projects.

- Understand the connection between vector databases, embeddings, and LLMs
- Learn practical approaches for transforming SQL databases into vector databases
- Build RAG applications for both personal and enterprise use
- Apply vector databases to solve real-world AI challenges
- Learn how to use vector databases with LLMs to build applications

Nitin Borwankar is a seasoned data scientist and database professional, known for his contributions to data education and open source machine learning tools over a career spanning three decades. He is a frequent conference speaker, offering a pragmatic approach to AI and large language models.

DATA

US \$69.99 CAN \$87.99

ISBN: 978-1-098-17759-1



9

O'REILLY®

Vector Databases

A Practical Introduction

Nitin Borwankar

O'REILLY®

Vector Databases

by Nitin Borwankar

Copyright © 2026 Nitin Borwankar. All rights reserved.

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 (<http://oreilly.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Acquisitions Editor: Andy Kwan

Development Editor: Sara Hunter

Production Editor: Beth Kelly

Copyeditor: Audrey Doyle

Proofreader: Krsta Technology Solutions

Indexer: WordCo Indexing Services, Inc.

Cover Designer: Susan Thompson

Cover Illustrator: José Marzan Jr.

Interior Designer: David Futato

Interior Illustrator: Kate Dullea

April 2026: First Edition

Revision History for the First Edition

2026-04-06: First Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781098177591> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Vector Databases*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the author and do not represent the publisher's views. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-098-17759-1

[LSI]

Table of Contents

Preface.....	xi
1. Introduction to Vector Databases.....	1
Why Do You Need Vector Databases?	1
A New Data Type: Vector	2
Similarity Search	3
What's Different About the Vector Type?	6
Where Do You Use Vector Databases?	8
SQL Versus Vector Databases	9
The Foundation of Business Math: Accounting Arithmetic	9
Vector Representation in a Relational Database Management System	10
The Need for Vector-Specific Capabilities	11
NoSQL Versus Vector Databases	11
NoSQL Databases and Vector Storage	11
Limitations of Vector Extensions in NoSQL Databases	12
When to Choose NoSQL with Vector Extensions	12
Hybrid Approaches: Combining Structured and Vector Data	13
The Need for Both Vector Data and Metadata	13
Limitations of Pure Vector Storage	13
Hybrid Database Architecture	14
Example of a Hybrid Query	14
Benefits of the Hybrid Approach	15
Conclusion	15
2. Embeddings.....	17
Understanding Vector Embeddings: Why We Need Them	17
Word2Vec: The Breakthrough That Changed Everything	19
Doc2Vec: From Words to Documents	20

From Embeddings to Modern Language Models:	
The Transformer Connection	23
Encoder-Only Transformers (BERT and Its Variants)	24
Decoder-Only Transformers (GPT Family)	24
Encoder-Decoder Transformers (T5, BART)	25
Embedding Models: The Specialized Vector Generators	26
Distinction from Traditional Models	26
Role in Modern LLM Applications	27
Practical Applications and Use Cases	28
Simple RAG Pipeline	28
The sentence-transformers Library: The Swiss Army Knife of Text Embeddings	31
Best Practices for Using SentenceTransformers: A Detailed Guide	35
The Embedding Layer: The Gateway to Zero-Shot Learning	40
Anatomy of Transformer Embeddings	40
Connection to Zero-Shot Learning	42
Key Characteristics That Enable Zero-Shot Learning	43
Limitations and Considerations	45
Latest Developments and Trends	46
Vector Arithmetic with Word2Vec: A Hands-On Guide	46
Step 1: Setup and Installation	46
Step 2: Load Pretrained Word2Vec Model	46
Step 3: Implement Vector Arithmetic Functions	47
Step 4: Classic King–Queen Analogy	48
Step 5: More Interesting Analogies	49
Step 6: Interactive Exploration Tool	49
Final Words on Vector Arithmetic	50
Conclusion	51
3. Similarity Search with FAISS.....	53
Foundations	53
Vector Representations	55
Distance Metrics	56
Selection Heuristics	58
FAISS Indexes	58
Flat Indexes (Brute Force)	58
IVF-Based Indexes	59
LSH-Based Indexes	60
HNSW-Based Indexes	61
Other Specialized Indexes	61
Composite and Transformative Indexes	62
Choosing the Right Index	62

Quantization	65
SQ	65
PQ	67
The ANN Problem	71
The Problem	72
Avoid Computational Cost	72
Key ANN Techniques in FAISS	73
Choosing an Index in FAISS	75
Code Example	75
Understanding HNSW Indexes	76
What Is HNSW?	77
How HNSW Works	78
Key Parameters Explained	79
Practical Example: Building a Similarity Search System	80
Performance Characteristics	81
Best Practices	82
FAISS Architecture and Components	83
Foundation	83
Core Concepts	85
Key Components	85
Common Workflow	87
Illustrative Example	87
Key Takeaways	88
Further Exploration	88
Conclusion	89
4. Semantic Search with SQLite3.....	91
Understanding the SQLite Vector Similarity Search Extension	91
Core Capabilities	92
Architecture Overview	93
Limitations	93
Setting Up the Development Environment	94
Installing Dependencies	94
Verifying the Installation	95
Operational Pragmas	96
Designing the Database Schema	96
Schema Requirements	96
Table Definitions	96
Schema Design Decisions	98
Connecting to Reddit with the Python Reddit API Wrapper	99
Creating Reddit API Credentials	99
PRAW Client Implementation	99

Usage Example	102
Content Extraction and Preprocessing	102
Text Cleaning Pipeline	102
Quality Filtering	105
Generating and Storing Embeddings	105
Embedding Generator	106
Database Storage	108
Batch Processing Pipeline	112
Building the Vector Index	113
Understanding VSS Indexing	113
Index Management	114
Implementing Semantic Search	117
Search Result Container	117
Search Engine	117
Putting It All Together	123
Workflow Example	124
Example Output	126
Extension: Incremental Indexing	127
Conclusion	129
5. Building an ArXiv Paper Search System with PostgreSQL pgvector.....	131
The Challenge of Searching Scientific Literature	131
Why ArXiv Makes an Ideal Data Source	131
Real-World Use Cases	132
Technology Stack Rationale	132
Architecture Overview	133
System Components	133
Data Flow	134
Design Philosophy	135
Environment Setup and Dependencies	136
PostgreSQL and pgvector Installation	136
Python Environment Configuration	137
Directory Structure and Configuration	137
Verification and Testing	138
Database Design for Scientific Papers	139
Schema Design Principles	139
Core Tables Structure	140
Vector Storage Strategy	144
Indexing Strategy	144
ArXiv Integration and PDF Management	145
ArXiv API Client Implementation	146
PDF Download Pipeline	147

Batch Processing System	148
PDF Text Extraction and Processing	150
PDF Extraction Challenges	150
Intelligent Text Chunking	152
Embedding Generation and Storage	153
Embedding Model Strategy	154
Batch Processing Pipeline	155
Similarity Search Implementation	156
Interactive Application and UI	158
Docker Packaging for Local Deployment	160
Container Architecture	161
Docker Compose Configuration	161
Database Initialization Scripts	163
Development Workflow	164
Cloud-Ready Design	164
Basic Performance Tuning	165
Index Configuration	165
Query Performance	165
Resource Management	165
Next Steps	165
Current Limitations	165
Enhancement Ideas	166
What We Did	166
System Achievements	166
Technical Skills Gained	167
Practical Research Tool	167
Foundation for Advanced Systems	167
Future Potential	167
Conclusion	167
6. Building a Retrieval-Augmented Generation System with SQLite VSS and Ollama. . .	169
System Architecture Overview	170
Database Foundation with Vector Support	171
Setting Up the Vector-Enabled Database	171
Schema Design for RAG	172
Creating Search Indexes	173
Text Processing and Embedding Generation	174
Embedding Model Management	174
Intelligent Text Chunking	175
Storing Content with Embeddings	176
Hybrid Search Implementation	177
Hybrid Search Algorithm	177

Semantic Search Component	178
Keyword Search Component	179
Score Fusion and Ranking	180
LLM Integration with Ollama	181
Ollama API Client	181
Health Check Function	182
The RAG Pipeline	182
Context Formatting	182
Question-Answering Pipeline	183
Demonstration and Testing	185
Sample Data Loading	185
Main Demonstration Function	186
Interactive Q&A Interface	187
Quick Testing Utility	188
Next Steps: Extending the System	188
Missing Reddit Data Features	188
Performance Optimizations	190
Production Considerations	190
Advanced RAG Patterns	191
Conclusion	191
 7. Building a Scientific RAG System with PostgreSQL and pgvector.	193
System Goals and Capabilities	194
Architecture Overview	194
Database Foundation with pgvector	196
Database Configuration and Setup	197
Schema Design for Scientific Papers	197
High-Performance Vector Indexes	199
Embedding Generation Strategy	199
ArXiv Integration and PDF Processing	200
Paper Discovery with ArXiv API	200
Intelligent PDF Text Extraction	201
Advanced Text Chunking	203
Storage Pipeline with Embeddings	204
Multilevel Semantic Search	206
Abstract-Level Search	206
Section-Level Search	207
The RAG Pipeline: Deep Dive	208
Local LLM Integration with Ollama	209
Health Check and Model Discovery	210
Intelligent Context Retrieval	210
Scientific Prompt Engineering	211

Complete RAG Execution Pipeline	212
Demonstration and Interactive Interface	213
Main Demonstration Flow	213
Search Demonstrations	214
RAG Demonstration	215
Interactive Search Interface	216
Entry Point with Mode Selection	217
Technical Note on HNSW	217
How to Evaluate Your Results	219
Next Steps: Extending the Scientific RAG System	220
Conclusion	223
8. Building a Complete Conversation Search and RAG System.	225
System Goals and Capabilities	226
System Architecture Overview	227
What We'll Build Together	229
Database Foundation for Conversation Storage	230
Designing the Conversation Schema	230
Three-Table Architecture for Optimal Performance	231
High-Performance Vector Indexing	232
Conversation Import and Data Processing Pipeline	233
Robust JSON Import with Error Handling	233
Atomic Transaction Processing	234
Timestamp Handling and Data Validation	234
Error Recovery and Logging	235
Efficient Embedding Generation and Batch Processing	236
Singleton Pattern for Model Management	236
Incremental Processing Strategy	237
Batch Processing for Optimal Performance	237
Database Insertion with Conflict Handling	238
Contextual Search with Conversational Understanding	239
Semantic Similarity Search	239
Multitable Joins for Rich Context	240
Result Formatting and Structure	240
Conversation Context Retrieval	241
Context Window Calculation	241
RAG Integration for Conversation History	242
Structured Context Management	242
Local LLM Integration with Ollama	243
Health Monitoring and Model Discovery	243
Context Retrieval and Assembly	244
Conversational Prompt Engineering	244

Complete RAG Pipeline with Performance Monitoring	245
Complete Web API with FastAPI	247
FastAPI Application Structure	247
Request Models with Validation	247
Search Endpoint Implementation	248
RAG Question-Answering Endpoint	248
System Statistics and Monitoring	248
Server Startup and Configuration	249
Demonstration and Sample Data	250
Realistic Sample Data Generation	250
Multitopic Sample Coverage	251
Sample Data Processing Pipeline	252
Comprehensive System Demonstration	252
Progressive Feature Demonstration	253
RAG Demonstration with Conditional Execution	254
Production Import Functionality	254
Application Entry Points	255
Conclusion: A Complete Personal Knowledge System	255
9. Vector Query Language.....	257
Core Concepts	258
Data Model	258
Basic Syntax Structure	259
Vector Operations	260
Similarity Search	260
Hybrid Search	261
Range Search	261
Batch Operations	261
Vector Functions and Aggregations	262
Vector Functions	262
Vector Aggregations	263
Index.....	265

Preface

To repeat a well-worn phrase, this is the book I wish I had when I started exploring vector databases. I was looking for a single source of basic theory and practical applications of vector databases, but I found none.

Existing vector database resources are fragmented across vendor documentation, blog posts, YouTube videos, and GitHub code. I wanted practical working examples rather than theoretical coverage. I also have a personal preference for proceeding systematically from basics to simple working systems to help organize data locally—especially private data.

Thus began my journey to pull together information from multiple sources and multiple directions into a coherent whole. And this book is what emerged. It is meant to be useful to the intermediate developer, from the theoretically minded to the “just show me the code” type. There’s something for everyone.

The book follows a clear pedagogical arc: concepts → tools → applications → future directions. Each chapter builds on previous ones while remaining practical with working code examples. The progression moves from understanding vector databases to building working applications that are meant for personal use, hands-on interaction, and extension by developers who want to build personal data management systems. It culminates in a forward-looking proposal for industry standardization of a Vector Query Language modeled after SQL.

The somewhat opinionated part of this book’s approach is a focus on simple hybrid vector relational databases—created by adding vector extension plug-ins to the well-known open source databases SQLite3 and PostgreSQL. The reason for this is that these databases are well known among database developers. Simply adding a data type of “vector” opens up a whole new world of AI data applications. This makes onboarding as low-effort as possible.



It can't be emphasized enough that these applications are for hands-on learning and not for web-scale production deployment. Modify them as you see fit; make them your own.

And now let's dive into the contents.

What's in This Book

Chapters 1 and 2 establish foundational concepts—vector databases, embeddings, and semantic search fundamentals:

Chapter 1, "Introduction to Vector Databases"

- Introduces vector databases as foundational technology for AI applications, explaining why unstructured data (text, images, audio, video) requires semantic search rather than keyword matching
- Covers the vector data type, similarity search, and key operations (cosine similarity, nearest-neighbor search, vector arithmetic)
- Contrasts vector databases with SQL and NoSQL databases, highlighting hybrid architectures that combine structured metadata with vector similarity search
- Establishes the conceptual foundation for understanding how embeddings preserve semantic meaning in vector space

Chapter 2, "Embeddings"

- Takes a deep dive into the history, evolution, and current usage of vector embeddings with code examples for language model applications
- Explains how embeddings bridge the gap between unstructured data and machine-processable representations, mapping raw data into vectors that capture semantic relationships
- Covers the mathematical foundations and how proximity in vector space reflects semantic similarity
- Provides the conceptual framework needed for practical applications later in the book

Chapters 3 and 4 discuss practical implementations with Facebook AI Similarity Search (FAISS) and SQLite, showing how to build personal-scale search systems:

Chapter 3, "Similarity Search with FAISS"

- Explores FAISS as both a production similarity search engine and a flexible toolkit for building custom vector databases

- Is designed for developers interested in the internals of vector search and those wanting to extend capabilities of systems such as `sqlite-vss`
- Covers the mathematical foundations and implementation details of efficient similarity search at scale
- Focuses on CPU-based implementations for maximum accessibility

Chapter 4, “Semantic Search with SQLite3”

- Builds a personal knowledge management system using semantic search over Reddit content, demonstrating meaning-based search versus keyword matching
- Introduces `sqlite-vss`, a SQLite extension that wraps FAISS and integrates vector search into the relational database world
- Combines semantic retrieval with relational metadata filtering in a single SQL workflow
- Uses an “overfetch-then-filter” pattern to handle the fact that metadata filters aren’t pushed down into FAISS search

Chapters 5 through 8 progress through increasingly sophisticated applications—ArXiv paper search, local retrieval-augmented generation (RAG) systems, scientific RAG, and personal conversation search—moving from public to private data:

Chapter 5, “Building an ArXiv Paper Search System with PostgreSQL pgvector”

- Constructs a scientific literature search system addressing the challenge of discovering relevant research across millions of papers
- Demonstrates how vector embeddings capture semantic relationships that keyword search misses (e.g., connecting “neural network optimization” with “gradient descent improvements”)
- Leverages PostgreSQL with the `pgvector` extension for enterprise-scale applications
- Shows how to handle technical terminology, structured academic content, and metadata in a unified search system

Chapter 6, “Building a Retrieval-Augmented Generation System with SQLite VSS and Ollama”

- Assembles a complete RAG system combining vector search with local large language model (LLM) capabilities
- Builds an entirely local, private system running on a single desktop using SQLite Vector Similarity Search (SQLite VSS) and Ollama
- Addresses the fundamental LLM limitation: knowledge frozen at training time without access to private or recent information
- Creates a question-answering system that grounds LLM responses in actual retrieved data to reduce hallucinations

Chapter 7, “Building a Scientific RAG System with PostgreSQL and pgvector”

- Develops a RAG system specifically designed for scientific literature with unique challenges: technical terminology, structured content, citation networks, and evidence quality
- Enables semantic discovery, cross-paper synthesis, and contextual understanding across ArXiv’s 15,000+ monthly publications
- Uses PostgreSQL with pgvector for scientific knowledge retrieval
- Demonstrates handling domain-specific language and structured academic conventions

Chapter 8, “Building a Complete Conversation Search and RAG System”

- Moves from searching public data to building a “second brain” for personal chat history with AI assistants
- Addresses unique characteristics of conversational data: contextual dependencies, conversational flow across exchanges, personal language patterns, and privacy requirements
- Handles the messiness of real conversational data where ideas develop across multiple message exchanges
- Creates a system for retrieving valuable insights, solutions, and learned knowledge from thousands of personal AI conversations

Chapter 9 proposes an experimental Vector Query Language (VQL) to standardize the fragmented vector database landscape:

Chapter 9, “Vector Query Language”

- Presents a prototype SQL-inspired query language for vector databases, designed to start community discussion and consensus building
- Addresses the needs of three communities: application developers seeking database abstractions, database maintainers needing cross-vendor tooling, and AI/machine learning researchers requiring efficient data manipulation
- Defines a data model (collections, tables, vectors, embeddings, metadata) and query syntax combining SQL familiarity with vector operations
- Covers similarity search, hybrid search, range search, batch operations, vector functions, and aggregations
- Proposes a path toward standardization across the fragmented vector database landscape

Who This Book Is For

This book's target audience comprises:

- Enterprise software developers building their first AI applications
- Data scientists implementing semantic search
- Machine learning engineers working on RAG systems
- Backend developers adding vector capabilities to existing systems

Additionally, it is geared toward people who learn by doing, so there are also practical applications focused on individual use.

Prerequisites include Python programming proficiency, a basic understanding of databases and SQL, and familiarity with running LLMs locally for the RAG chapters. Readers should also be comfortable with simple mathematical concepts.

The book is for those who want to learn how to build small- to medium-scale semantic search systems, implement RAG applications with local LLMs, optimize vector similarity search, choose the right indexes, and integrate vector search with relational databases.

The book focuses on Python implementations rather than other languages, covers CPU-based solutions rather than GPU optimization, and excludes cloud-managed vector database services in favor of self-hosted solutions. The scope of the applications and the theory is to cover enough to get started and do something useful with personal data as a first step. This is not a textbook on how to build web-scale production vector database applications. However, the lessons in the chapters can serve as a foundation and a springboard to take the next leap up in scope and complexity.

How to Use This Book

For the reader who is coming from a purely relational database background, reading the first two chapters is a must. **Chapter 3** is more detailed on the innards of FAISS, but you can just use the code examples to dive right in and use it.

Chapters **4** through **8** use SQLite3 and Postgres to build semantic search and RAG applications using local data and local LLMs. **Chapter 8** is especially topical, as it creates a semantic search over your LLM conversations—in this case, exported from Claude.

Finally, those interested in exploring similarities with SQL can dive into **Chapter 9**, which is a speculative design of a Vector Query Language to enable a unified, vendor-independent, client access metaphor for vector databases.

Software, Environment, and Resource Requirements

With the following resources, the example applications will run on standard developer laptops:

- Python version 3.11+
- Key dependencies: PostgreSQL, SQLite3, FAISS, Ollama
- OS: Linux, macOS, or WSL on Windows
- Minimum 16 GB RAM; preferably 24 GB

For a developer machine that can run PostgreSQL and an 8B LLM:

- Storage requirements: 512 GB to 1 TB

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 element signifies a tip or suggestion.



This element signifies a general note.



This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at <https://github.com/nborwankar/VectorDatabaseBook>.

If you have a technical question or a problem using the code examples, please send email to support@oreilly.com.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it 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 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 generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Vector Databases* by Nitin Borwankar (O'Reilly). Copyright 2026 Nitin Borwankar, 978-1-098-17759-1.”

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.

O'Reilly Online Learning



For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit <https://oreilly.com>.

How to Contact Us

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

O'Reilly Media, Inc.
141 Stony Circle, Suite 195
Santa Rosa, CA 95401
800-889-8969 (in the United States or Canada)
707-827-7019 (international or local)
707-829-0104 (fax)
support@oreilly.com
<https://oreilly.com/about/contact.html>

We have a web page for this book, where we list errata and any additional information. You can access this page at *<https://oreil.ly/vector-databases>*.

For news and information about our books and courses, visit *<https://oreilly.com>*.

Find us on LinkedIn: *<https://linkedin.com/company/oreilly-media>*

Watch us on YouTube: *<https://youtube.com/oreillymedia>*

Acknowledgments

I want to thank all the technical reviewers, especially Sunil Sawant. Because of their efforts, the book is a much better product.

Sara Hunter, my development editor, deserves praise for her patience during the yo-yo delivery cadence of the chapters. I would also like to thank the rest of the O'Reilly editorial team—both the early release and final production teams—for putting up with the inexperience of a first-time author.

My wife, Garima, and daughter, Vanita, had to tolerate my minimal participation in family events during all of last year. I would like to thank them deeply for their support and understanding during the writing of “the book.” It’s finally done, guys!

Of course, none of this would be possible without the open source developer communities of PostgreSQL, SQLite, and the vector extensions, whose work enabled the example code.