DATA ENGINEERING AND ANALYTICS

Introduction to Vector Databases for AI

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Every serious AI application eventually needs to find things by meaning rather than by keyword: the support article that answers a question phrased three different ways, the document a chatbot should cite, the products that are similar rather than identical. Vector databases are the infrastructure that makes this possible, and they behave unlike any database most engineers have used: results are approximate by design, "correct" is a matter of degree, and quality depends as much on how you prepare the data as on the engine you choose.

This is a hands-on, practitioner course. It builds the subject in dependency order: first embeddings, because nothing about a vector database makes sense until you understand what a vector of meaning is; then similarity search and the indexes that make it fast; then choosing and operating a database; and finally building real semantic search and seeing how it becomes the retrieval half of RAG. In keeping with a less-but-deeper philosophy, we go deep on the concepts and skills that transfer across every vector database rather than surveying vendor feature lists. Every module ends with a lab, and each module builds on the one before.

Who Should Attend

- Developers and data engineers adding semantic search or AI retrieval to an application
- Database professionals extending relational or NoSQL experience into vector workloads
- Architects evaluating vector database options for an AI initiative

  Learners who want to go straight to building full retrieval-augmented applications should follow this course with *Retrieval-Augmented Generation (RAG) with Vector Databases*.

Prerequisites

- Basic Python: enough to read and modify short scripts
- Comfort with core database concepts (tables or collections, queries, indexes)
- No machine learning background required

What You Will Learn

- Explain what embeddings are and why they let software compare things by meaning
- Explain how similarity search works, including distance metrics and approximate nearest neighbor indexes
- Select a vector database sensibly, from pgvector to dedicated engines and managed services
- Load, index, and query vectors, including metadata filtering and hybrid search
- Build a working semantic search system over real documents
- Judge quality, cost, and scale tradeoffs, and explain how vector search powers RAG

Course Outline

Day one: embeddings and how vector search works

- Embeddings: Vectors of Meaning
  - From words to vectors: what an embedding model does
  - Why similar meanings land near each other, shown concretely
  - Choosing an embedding model and what its dimensions cost you
  - Lab: generate embeddings for real text and measure similarity between them
- How Similarity Search Works
  - Distance metrics: cosine, dot product, and Euclidean, and when the choice matters
  - Why exact search does not scale, and what approximate nearest neighbor buys you
  - HNSW and friends at an intuition level: recall versus speed versus memory
  - Lab: compare exact and approximate search on the same dataset and observe the tradeoff
- Choosing a Vector Database
  - The landscape: pgvector inside PostgreSQL, dedicated engines, and managed cloud services
  - Honest selection criteria: existing stack, scale, filtering needs, and operational appetite
  - When you do not need a vector database at all
  - Lab: stand up a vector database and load the embeddings from the first lab

Day two: building and operating semantic search

- Preparing Data for Retrieval
  - Chunking documents: sizes, overlap, and why chunking drives result quality
  - Metadata design: what to store alongside vectors and why
  - Lab: chunk, embed, and load a real document set with useful metadata
- Querying Well
  - Top-k search, metadata filtering, and combining the two
  - Hybrid search: adding keyword signals to vector similarity
  - Evaluating result quality with a small, honest test set
  - Lab: build and tune a semantic search over the loaded documents
- Operating and the Bridge to RAG
  - Updates, deletes, and re-embedding when models or documents change
  - Cost and scale: index memory, query volume, and what actually gets expensive
  - How this becomes RAG: retrieval feeding an LLM, and what that adds to the picture
  - Lab: wire the search system into a simple LLM prompt and see grounded answers

Extended Version

The three-day version keeps the same gradient and adds depth and a fuller build:

- Deeper evaluation: building a retrieval quality harness and tuning against it
- Reranking and query rewriting to improve relevance
- Operating at scale: sharding, index rebuilds, and monitoring in production
- A capstone that builds a complete semantic search service over a realistic corpus, evaluated and tuned end to end