Vector Stores Vs Vector Databases

Section: Vector Stores And Vector Databases

Overview

In previous lessons we generated **embeddings** using models like OpenAI and Hugging Face and ran **semantic search** (cosine similarity) across small sets of sentences. The next step is deciding **where those vectors live** so we can search them efficiently. This lesson clarifies the difference between **vector stores** and **vector databases**, and when to use each.

Cleaned & Organized Transcript

What problem are we solving?

Once you convert document chunks into vectors, you must **store** them somewhere so you can **retrieve** similar items later. You'll hear two terms a lot: **vector store** and **vector database**. They sound similar but aren't the same.

Definitions

- **Vector Store**: A **lightweight library/tool** embedded in your application that specializes in **storing vectors and running similarity search** (e.g., k-NN with cosine similarity). Often **in-memory** or **local-file** based, quick to set up, minimal infrastructure.
- **Vector Database**: A **fully featured database system** purpose-built for **vector data at scale**. Provides **durability**, **replication**, **sharding**, **high availability**, **security**, and **rich metadata filtering** in addition to fast vector search.

Core Functionality

- **Vector Store**
  - Similarity search / **k-NN** over embeddings
  - Simple APIs, fast iteration
  - Typically no multi-node features
- **Vector Database**
  - Advanced search (ANN with **filters**, hybrid vector+keyword)

    - **CRUD** operations at scale (Create/Read/Update/Delete)
    - Operational features: **indexes**, **backups**, **RBAC**, **observability**

Architecture & Deployment

- **Vector Store**: Runs **inside your app process**, often **in-memory** or a single **local file**. Deployed on your laptop/server/container.
- **Vector Database**: **Client–server** or **managed cloud** service. Supports **replication**, **sharding**, **horizontal/vertical scaling**, and production SLAs.

When to Use Which

- **Choose a Vector Store** when:
    - You're building a **prototype/POC** or a small internal tool
    - Data volume is **small to moderate** (roughly up to ~1M vectors, hardware-dependent)
    - You want **fast setup**, low cost, and full control in code
- **Choose a Vector Database** when:
    - You're going to **production** and need reliability and scale
    - You expect **large datasets** (hundreds of millions/billions of vectors)
    - You require **metadata filters**, **concurrency**, **security**, **high availability**, and **observability**

Performance & Cost (Rules of Thumb)

- **Vector Store**: Setup in **minutes**; **µs–low-ms** latency in memory; near-zero infra cost beyond your machine. Persistence is your responsibility.
- **Vector Database**: Setup ranges from **minutes** (managed) to **hours/days** (self-hosted). Low-ms queries with proper indexes; **monthly costs** scale with storage, throughput, and features (backups, replicas).

Popular Tools (spellings corrected)

- **Vector Stores / Libraries**: **FAISS**, **hnswlib**, **NMSLIB**, **ScaNN**, **Annoy**, **Chroma** (commonly used as an embedded store for quick projects).

- **Vector Databases**: **Pinecone**, **Qdrant** (local & cloud), **Milvus**, **Weaviate**, **Vespa**. Also **DataStax Astra DB** (vector support) and **Postgres + pgvector** if you prefer a SQL database with vector search.

Note on names mentioned informally elsewhere: “fires/scan nims lib/quadrant” → **FAISS/ScaNN/NMSLIB/Qdrant**; and **DataStax** (not “data stacks”).

Comparison Table

Aspect	**Vector Store**	**Vector Database**
What it is	Embedded **library/tool**	**Database system** for vectors
Core ops	**Similarity search**, k-NN	ANN + **filters**, hybrid, **CRUD**
Deployment	**In-process**, local file	**Client–server**, managed cloud, clusters
Durability	App-managed (files/snapshots)	Built-in persistence, backups
Scale	**~≤1M** vectors (depends on HW)	**Hundreds of millions to billions**
HA/DR	Manual/none	**Replication**, **sharding**, failover
Security	App-level	**Auth/RBAC**, network controls
Observability	Minimal	Metrics, logs, tracing
Cost	Near-zero (your machine)	Opex by storage/throughput/features
Setup time	**Minutes**	**Minutes–days**
Latency	**µs–low ms** (in-memory)	**Low ms** with indexes/caches

These are heuristics; actual limits depend on vector dimension, index type, hardware, and query load.

End-to-End Flow (High Level)

1. **Prepare content** → split into chunks with sensible size/overlap.
2. **Embed** chunks into vectors.

3. **Store** vectors (choose vector store or vector database) with helpful **metadata** (document id, source, section, page, timestamps, language, tags).
4. At query time, **embed** the query, run **approximate nearest neighbor** search (optionally add metadata filters), and return top-k results.

Minimal Pseudocode

```
# Ingestion
chunks = chunk(documents)
emb = embedder.encode([c.text for c in chunks])
store.upsert([
 {
  "id": c.id,
  "vector": emb[i],
  "metadata": {
   "doc_id": c.doc_id,
   "source": c.source,
   "section": c.section,
   "page": c.page,
   "created_utc": c.created_utc,
   "tags": c.tags,
  }
 }
 for i, c in enumerate(chunks)
])

# Retrieval
q_vec = embedder.encode([user_query])[0]
```

results = store.search(q_vec, top_k=10, filter={"section": "Vector Stores And Vector Databases"})

Quick Examples

- **Prototype**: Chroma + FAISS index on < 1M chunks, fast cosine search.
- **Production**: Qdrant/Milvus managed cluster using HNSW; add filters like {tenant, language, doc_type} and enable replication.
- **Hybrid**: Combine BM25 keyword filtering (e.g., Elasticsearch/OpenSearch) with vector re-ranking.

Key Takeaways

- **Vector store** = embedded library for fast similarity search.
- **Vector database** = full database system for vector data at scale.
- Start small with a store; move to a database when you need **scale, reliability, and rich queries**.
- Add **metadata** from day one to enable filtering later.

FAQ

Is a vector store the same as a database?
No. A store focuses on similarity search; a database adds durability, scaling, security, and rich queries.

Can I stay with Postgres?
Yes—**pgvector** brings vector search to Postgres alongside standard SQL.

Do I need managed services?
Not for small projects. Managed options help when you need uptime, scaling, and team access.

Notes & Corrections

- Correct spellings: **FAISS**, **ScaNN**, **NMSLIB**, **Qdrant**, **DataStax**.

- The “~≤1M vectors” guideline is a heuristic; real limits vary with dimensions, hardware, and index choice.

What’s Next

Hands-on demos:

- In-process **Chroma + FAISS** quick start.
- Deploy **Qdrant** locally and then switch to managed cloud.
- Add **metadata filters** and experiment with **hybrid search**.