Vector Library versus Vector Database

December 1, 2022 · 7 min read

Erika Cardenas
Developer Advocate

In the world of Vector Search, we use **vector embeddings** – generated by **Machine Learning models** – to represent data objects (text, images, audio, etc.). The key idea here is that embeddings that are semantically similar to each other have a smaller distance between them.

We can use vector distance functions like euclidean distance or cosine distance to determine if these objects are similar to other objects in the collection. However, to do this we need to compare the distances between the query vector and **every** vector in the collection. This type of calculation can be difficult to scale to millions or billions of vectors.

This is why we have **Vector Databases** and **Vector Libraries**. They both use the Approximate Nearest Neighbor (ANN) algorithm to search through vectors in a tiny fraction of the time. You can learn more about this topic from "Why Vectors Search is so Fast."

The Big Question

So, if both vector databases and vector libraries allow you to efficiently search through your vectors. What are the key differences between them, and why/when should you choose one over the other?

Vector Libraries

Vector libraries store vector embeddings in in-memory indexes, in order to perform similarity search. Most vector libraries share the following characteristics:

1. they store vectors only,
2. index data is immutable,
3. query during import limitation

Store Vectors Only

Vector libraries only store vector embeddings and not the associated objects they were generated from.

When you run a query, a vector library will respond with the relevant vectors and object ids. This is limiting since the actual information is

stored in the object and not the id. To solve this problem, you would need to store the objects in a secondary storage. You can then use the returned ids from the query and match them to the objects to understand the results.

Immutable Data

Indexes produced by vector libraries are immutable. This means that once you have imported your data and built the index, you cannot make any modifications (no new inserts, deletes or changes). To make any changes to your index, you will need to rebuild it from scratch.

Query during Import Limitation

Most vector libraries cannot be queried while importing your data. It is required to import all of your data objects first. Then the index is built after the objects have been imported. This can be a concern for applications that require importing millions or even billions of objects.

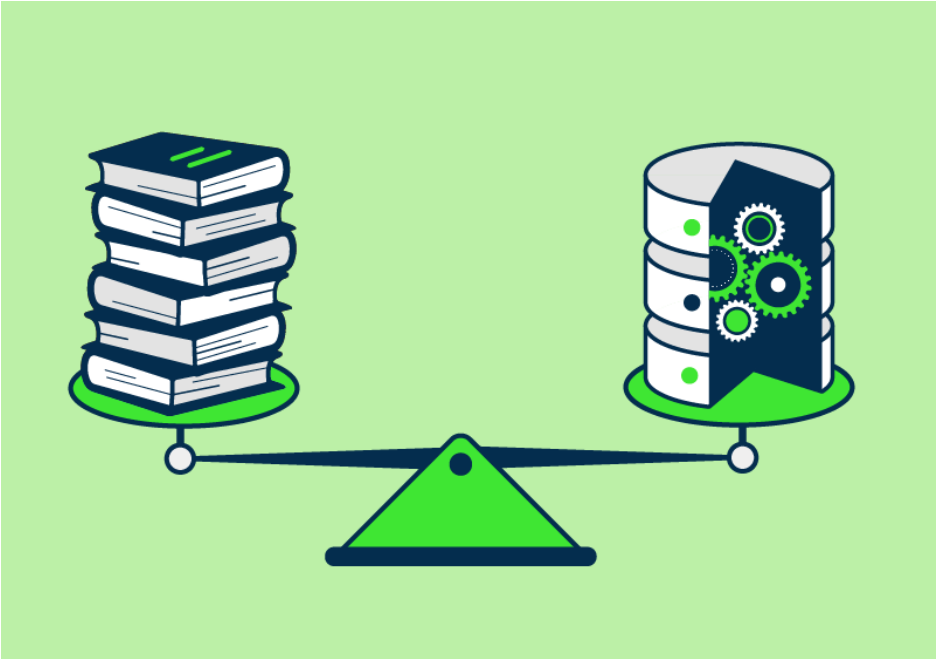
Examples of Vector Libraries

There are quite a few libraries to choose from - Facebook Faiss, Spotify Annoy, Google ScaNN, NMSLIB, and HNSWLIB. These libraries enable users to perform vector similarity search using the ANN algorithm.

The ANN algorithm has different implementations depending on the vector library. Faiss uses the clustering method, Annoy uses trees, and ScaNN uses vector compression. There is a performance tradeoff for each, which you can choose depending on your application and performance measure.

Example Use Cases

Vector libraries are commonly used for applications that do not have changing data. For example, academic information retrieval benchmarks are designed to test performance on a **static snapshot** of data. When plugging an ANN index into production-ready applications, databases offer many appealing features not found in a library.



Vector Databases

One of the core features that set vector databases apart from libraries is the ability to store and update your data. Vector databases have full **CRUD** (create, read, update, and delete) support that solves the limitations of a vector library. Additionally, databases are more focused on enterprise-level production deployments.

Store Vectors and Objects

Databases can store both the data objects and vectors. Since both are stored, you can combine vector search with structured filters. Filters allow you to make sure the nearest neighbors match the filter from the metadata. Here is an article on the effects of filtered Hierarchical Navigable Small World (HNSW) searches on recall and latency.

CRUD Support

Vector databases solve a few limitations that vector libraries have. One example is being able to add, remove, or update entries in your index after it has been created. This is especially useful when working with data that is continuously changing.

Real-time Search

Unlike vector libraries, databases allow you to query and modify your data during the import process.

As you upload millions of objects, the imported data remains fully accessible and operational, so you don't need to wait for the import to complete to start working on what is already in.

Note your queries won't return any objects that are not imported yet, as you can't query what you don't have. 🤔

Weaviate

Weaviate is an open-source vector database. If you're new to Weaviate, take a look at the Getting Started guide. 🙂

The HNSW graph is the first implementation of an ANN algorithm supported by Weaviate. Here is a benchmark that measures Weaviate's ANN performance for different use-cases.

Weaviate was built to combine the speed and capabilities of ANN algorithms with the features of a database such as backups, real-time queries, persistence, and replication (part of the v1.17 release). Weaviate can be accessed through GraphQL, REST, and client libraries in multiple programming languages.

Example Use Cases

Vector databases are great to use for your application if your data is constantly changing. You can use vector search engines for e-commerce recommendations, image search, semantic similarity, and the list goes on. Weaviate just released a new module that introduces a way to represent a user's interests by drawing a graph of cross-references. For more information about this new feature, read this blog post by Connor Shorten.

Weaviate also has a bunch of example use cases on GitHub. Find your favorite example, give it a star, and try to recreate it yourself!

Feature Comparison - Library versus Database

The table below summarizes the differences between vector libraries and databases. This is by no means an exhaustive list of features, and not every library or database has the same features.

Feature	Vector Library	Vector Database (Weaviate as an example)
Filtering (in combination with Vector Search)	No	Yes
Updatability (CRUD)	No (some do, e.g. hnswlib)	Yes
Incremental importing, concurrent reading while importing	No (some do, e.g. hnswlib)	Yes
Stores objects and vectors	No	Yes
Speed	Typically faster than full-blown database	Typically slower than pure library
Performance optimized for	In-memory similarity search	End2end-callstack, including: vector search, object retrieval from persistent storage, optional inverted index filtering, network requests, etc.
Durability, crash recovery	No	Yes
Persistence	Only at explicit snapshot	Immediate (after each insert, update or delete)
Sharding (i.e. distribute dataset among more than one machine, automatically merge/aggregate results at query time across machines)	No (there are some 3rd-party projects that wrap sharding around existing libraries)	Yes

Feature	Vector Library	Vector Database (Weaviate as an example)
Replication	No	Yes (from v1.17)
Automated backups (i.e. to and from cloud storage)	No	Yes
Deployment ecosystem (Docker, K8s, Helm, SaaS)	No (you have to build it yourself)	Yes
SDKs / language clients	No (although most have python bindings)	Yes (Python, Java, JS, Golang)
Execution	Embedded (can be turned into standalone service if you build a simple wrapper app around it)	Standalone service
Communication with app	Foreign Function Interface (FFI) / Language Bindings	Network calls (HTTP, gRPC, etc.)
Multi-tenancy	No	Yes
Arbitrary aggregations of stored objects	No	Yes
Module ecosystem (e.g. vectorization, QnA)	No (have to build it yourself)	Yes
Hybrid BM25+dense vector search	No	Yes (from v1.17)

Resources

If you're interested in learning more about vector databases and vector libraries, check out the resources below:

- Listen to this podcast with Meta AI Scientist Matthijs Douze and Abdel Rodriguez, Etienne Dilocker, and Connor Shorten from Weaviate! They talk about Facebook Faiss, product quantization for ANN search, and more.

What's next

Check out Getting Started with Weaviate, and begin building amazing apps with Weaviate.

You can reach out to us on Slack or Twitter, or join the community forum.

Tags: concepts search

Edit this page