

What is a Vector Database & How Does it Work?

Use Cases + Examples

By Roie Schwaber-Cohen | May 3, 2023 | Pinecone

What is a Vector Database?

A vector database indexes and stores vector embeddings for fast retrieval and similarity search, with capabilities like CRUD operations, metadata filtering, horizontal scaling, and serverless.

We're in the midst of the AI revolution. It's upending any industry it touches, promising great innovations - but it also introduces new challenges. Efficient data processing has become more crucial than ever for applications that involve large language models, generative AI, and semantic search.

All of these new applications rely on vector embeddings, a type of vector data representation that carries within it semantic information that's critical for the AI to gain understanding and maintain a long-term memory they can draw upon when executing complex tasks.

Embeddings are generated by AI models (such as Large Language Models) and have many attributes or features, making their representation challenging to manage. In the context of AI and machine learning, these features represent different dimensions of the data that are essential for understanding patterns, relationships, and underlying structures.

That is why we need a specialized database designed specifically for handling this data type. Vector databases like Pinecone fulfill this requirement by offering optimized storage and querying capabilities for embeddings. Vector databases have the capabilities of a traditional database that are absent in standalone vector indexes and the specialization of dealing with vector embeddings, which traditional scalar-based databases lack.

The challenge of working with vector data is that traditional scalar-based databases can't keep up with the complexity and scale of such data, making it difficult to extract insights and perform real-time analysis. That's where vector databases come into play – they are intentionally designed to handle this type of data and offer the performance, scalability, and flexibility you need to make the most out of your data.

With a vector database, we can add knowledge to our AIs, like semantic information retrieval, long-term memory, and more.

How Vector Databases Work - Overview

1. First, we use the embedding model to create vector embeddings for the content we want to index.
2. The vector embedding is inserted into the vector database, with some reference to the original content the embedding was created from.

- When the application issues a query, we use the same embedding model to create embeddings for the query and use those embeddings to query the database for similar vector embeddings.

What's the difference between a vector index and a vector database?

Standalone vector indices like FAISS (Facebook AI Similarity Search) can significantly improve the search and retrieval of vector embeddings, but they lack capabilities that exist in any database. Vector databases, on the other hand, are purpose-built to manage vector embeddings, providing several advantages over using standalone vector indices:

- Data management:** Vector databases offer well-known and easy-to-use features for data storage, like inserting, deleting, and updating data.
- Metadata storage and filtering:** Vector databases can store metadata associated with each vector entry. Users can then query the database using additional metadata filters for finer-grained queries.
- Scalability:** Vector databases are designed to scale with growing data volumes and user demands, providing better support for distributed and parallel processing.
- Real-time updates:** Vector databases often support real-time data updates, allowing for dynamic changes to the data to keep results fresh.
- Backups and collections:** Vector databases handle the routine operation of backing up all the data stored in the database.
- Ecosystem integration:** Vector databases can more easily integrate with other components of a data processing ecosystem, such as ETL pipelines, analytics tools, and visualization platforms.
- Data security and access control:** Vector databases typically offer built-in data security features and access control mechanisms to protect sensitive information.

How does a vector database work?

We all know how traditional databases work (more or less)—they store strings, numbers, and other types of scalar data in rows and columns. On the other hand, a vector database operates on vectors, so the way it's optimized and queried is quite different.

In traditional databases, we are usually querying for rows in the database where the value usually exactly matches our query. In vector databases, we apply a similarity metric to find a vector that is the most similar to our query.

A vector database uses a combination of different algorithms that all participate in Approximate Nearest Neighbor (ANN) search. These algorithms optimize the search through hashing, quantization, or graph-based search.

Common pipeline for a vector database:

- Indexing:** The vector database indexes vectors using an algorithm such as PQ, LSH, or HNSW. This step maps the vectors to a data structure that will enable faster searching.
- Querying:** The vector database compares the indexed query vector to the indexed vectors in the dataset to find the nearest neighbors.

3. Post Processing: In some cases, the vector database retrieves the final nearest neighbors from the dataset and post-processes them to return the final results.

Serverless Vector Databases

Serverless represents the next evolution of vector databases. The above architectures get us to a vector database architecture that is accurate, fast, scalable, but expensive. This architecture is what we see in first-generation vector DBs. With the rise of AI use cases where cost and elasticity are increasingly important, a second generation of serverless vector databases is needed.

First-generation vector DBs have three critical pain points that serverless solves:

- **Separation of storage from compute:** To optimize costs, compute should only be used when needed. That means decoupling the index storage from queries and searching only what is needed.
- **Multitenancy:** Handling namespaces in indexes to ensure infrequently queried namespaces do not increase costs.
- **Freshness:** A vector DB needs to provide fresh data, meaning within a few seconds of inserting new data, it is queryable.

Algorithms

Several algorithms can facilitate the creation of a vector index. Their common goal is to enable fast querying by creating a data structure that can be traversed quickly. They will commonly transform the representation of the original vector into a compressed form to optimize the query process.

Random Projection

The basic idea behind random projection is to project the high-dimensional vectors to a lower-dimensional space using a random projection matrix. We create a matrix of random numbers. The size of the matrix is going to be the target low-dimension value we want. We then calculate the dot product of the input vectors and the matrix, which results in a projected matrix that has fewer dimensions than our original vectors but still preserves their similarity.

Product Quantization

Product quantization (PQ) is a lossy compression technique for high-dimensional vectors. It takes the original vector, breaks it up into smaller chunks, simplifies the representation of each chunk by creating a representative "code" for each chunk, and then puts all the chunks back together - without losing information that is vital for similarity operations.

The process of PQ can be broken down into four steps:

1. **Splitting** - The vectors are broken into segments.
2. **Training** - We build a "codebook" for each segment using k-means clustering.
3. **Encoding** - The algorithm assigns a specific code to each segment.

4. **Querying** - The algorithm breaks down vectors into sub-vectors and quantizes them.

Locality-Sensitive Hashing (LSH)

Locality-Sensitive Hashing (LSH) is a technique for indexing in the context of an approximate nearest-neighbor search. It is optimized for speed while still delivering an approximate, non-exhaustive result. LSH maps similar vectors into "buckets" using a set of hashing functions.

Hierarchical Navigable Small World (HNSW)

HNSW creates a hierarchical, tree-like structure where each node of the tree represents a set of vectors. The edges between the nodes represent the similarity between the vectors. The algorithm starts by creating a set of nodes, each with a small number of vectors.

Similarity Measures

Similarity measures are mathematical methods for determining how similar two vectors are in a vector space. Several similarity measures can be used:

- **Cosine similarity:** measures the cosine of the angle between two vectors. It ranges from -1 to 1, where 1 represents identical vectors.
- **Euclidean distance:** measures the straight-line distance between two vectors. It ranges from 0 to infinity, where 0 represents identical vectors.
- **Dot product:** measures the product of the magnitudes of two vectors and the cosine of the angle between them.

Filtering

Every vector stored in the database also includes metadata. In addition to the ability to query for similar vectors, vector databases can also filter the results based on a metadata query. To do this, the vector database usually maintains two indexes: a vector index and a metadata index.

- **Pre-filtering:** Metadata filtering is done before the vector search. This can help reduce the search space but may cause the system to overlook relevant results.
- **Post-filtering:** Metadata filtering is done after the vector search. This ensures all relevant results are considered but may introduce additional overhead.

Database Operations

Unlike vector indexes, vector databases are equipped with a set of capabilities that makes them better qualified to be used in high scale production settings.

Performance and Fault Tolerance

To ensure both high performance and fault tolerance, vector databases use sharding and replication:

- **Sharding:** Partitioning the data across multiple nodes. When a query is made, it is sent to all the shards and the results are retrieved and combined (scatter-gather pattern).
- **Replication:** Creating multiple copies of the data across different nodes. This ensures that even if a particular node fails, other nodes will be able to replace it.

Monitoring

Monitoring is critical for detecting potential problems, optimizing performance, and ensuring smooth production operations. Key aspects include: resource usage, query performance, and system health.

Access Control

Access control is the process of managing and regulating user access to data and resources. It is a vital component of data security, ensuring that only authorized users have the ability to view, modify, or interact with sensitive data stored within the vector database.

Summary

The exponential growth of vector embeddings in fields such as NLP, computer vision, and other AI applications has resulted in the emergence of vector databases as the computation engine that allows us to interact effectively with vector embeddings in our applications.

Vector databases are purpose-built databases that are specialized to tackle the problems that arise when managing vector embeddings in production scenarios. For that reason, they offer significant advantages over traditional scalar-based databases and standalone vector indexes.

In this post, we reviewed the key aspects of a vector database, including how it works, what algorithms it uses, and the additional features that make it operationally ready for production scenarios.

Source: <https://www.pinecone.io/learn/vector-database/>

© Pinecone Systems, Inc.