COMPARATIVE ANALISYS OF VECTOR DATABASES: Chroma, Milvus and pgVector

Academic year: 2024./2025.

Contents

1 Introduction

Traditional databases rely on exact matches or keyword-based queries. Vector databases use similarity based searches that retrieve data points based on their proximity in high-dimensional vector space, rather than exact match of values. Complex unstructured data such as text, images, audio, video and etc. is converted into its vector representations (embeddings) using embedding model, which is then stored in vector databases. Vector databases support search based on similartiy using kind of mathematical distance metric. Distance is usually measured using cosine similarity, Euclidean distance, dot product or other.

Vector databases are integral for the functioning of artificial intelligence (AI) and machine learning (ML) models, semantic search engines and personalized recommendation systems, because of their capabilities to store, manage, and quickly retrieve high-dimensional embeddings. This enables AI and ML models to perform efficient similarity searches, find patterns, and provide relevant results based on meaning rather than exact keyword matches, enhancing the accuracy and relevance of recommendations, search results, and predictions.

Figure 1. illustrates how different data types (text, images and audio) are transformed into its vector representations using transformers, which are then stored and managed in a vector database for efficient retrieval and analysis.

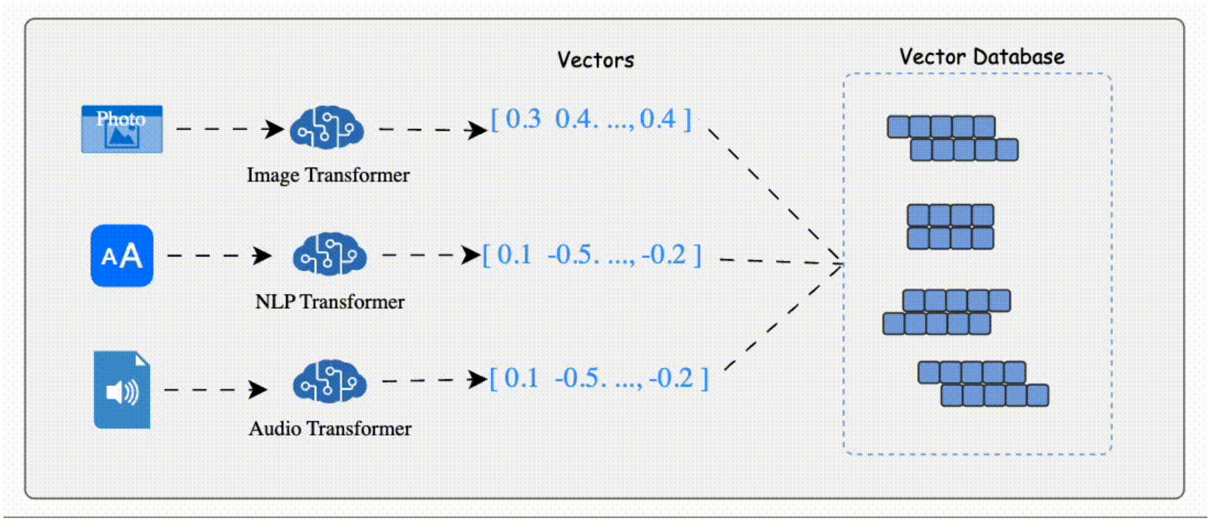

Figure 1. Data transformation and storage

2 Overview of selected databases

In this comparison analysis three databases will be evaluated : Chroma, Milvus and pgVector.

2.1 Chroma

Chroma is open-source AI application database that offers storage of embeddings, vector search, full-text search, document storage, metadata filtering and multi-modal retrieval [1.]. It's lightweight and developer focused database ideal for quickly building and testing AI applications. It supports integration with Hugging Face, OpenAI, Google Generative AI and other platforms for embedding generation, and also allows custom model pipelines. Chroma uses disk-based storage with index support. It supports horizontal scaling which makes it suitable for both small and large-scale projects. Offers multi-language client software development kits (SDKs) covering Python and JavaScript/TypeScript and others languages.

Chroma has 4 primary API functions:

- **add**: Inserts new documents and their corresponding embeddings into a collection.
- **get**: Retrieves documents or embeddings from a collection based on specified criteria.
- **update**: Modifies existing documents or embeddings within a collection.
- **delete**: Removes documents or embeddings from a collection.

which make Chroma simple and easy to use.

Figure 2 illustrates how Chroma is used in conjunction with a LLM to enhance question-answering or retrieval-augmented generation (RAG) workflows.

User submits a query to the system. The query is passed through an embedding model (such as OpenAI's embedding API or another ML model) to convert the text into a vector representation (embedding).

Chroma stores documents (text data, such as "Call me Ishmael...") and embeddings. Alongside each document, Chroma stores its vector embedding ([1.0, 2.1, 3.4...]).

The query embedding is used to perform a similarity search in Chroma, retrieving documents whose embeddings are most similar to the query.

The most relevant documents are fed to the LLM to improve the quality and relevance of its answers. The LLM uses both the original query and the contextual information from Chroma to generate a more accurate and informed answer.

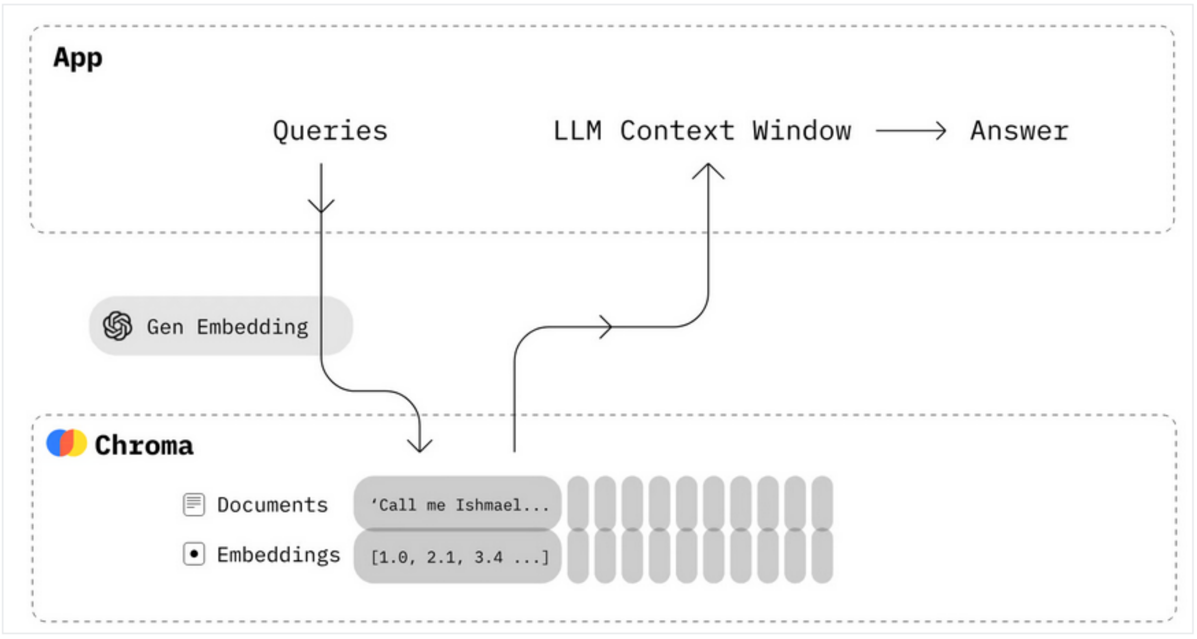

Figure 2. Chroma conjuction with LLM [2.]

2.2 Milvus

Milvus is purpose-built, industrial grade vector database which utilizes a distributed system. It is focused on efficient management of large-scale vector data and uses its distributed architecture for scalability. It is designed for handling billions and trillions of vectors, supports advanced indexing methods, making it well suited for enterprise level deployments and large-scale production systems. Milvus uses multi-layered architecture to efficiently manage and scale vector data. Features separate computing and storage components which means that processing power is separate from where data is kept.

Milvus architecture has 4 main layers:

1. **Access layer** – entry point for users, handles client connections, API requests, load balancing and basic validation
2. **Coordinator service layer** – manages data through coordinators, querying and indexing, handling data nodes and metadata
3. **Worker nodes layer** – executes tasks like insertion, carries out commands from coordinators, dynamically scale based on system load
4. **Object storage layer** – provides persistent storage (metadata, logs, object files) and system reliability

Database also supports wide range of indexing algorithms, offers robust multi-language SDKs covering Python, Java, C# etc. and leverages GPU acceleration to further speed up index building.

2.3 pgVector

Unlike standalone vector databases, pgVector operates as an extension for PostgreSQL, bringing vector similarity search capabilities to one of the most popular relational database systems. This approach allows organizations to leverage their existing PostgreSQL infrastructure for vector operations rather than deploying and maintaining a separate vector database. PostgreSQL leverages the relational database's storage and query engine to support hybrid queries but is constrained by PostgreSQL's scalability. PgVector may face challenges with extremely large vector datasets compared to purpose-built vector databases. Its performance is tied to PostgreSQL's underlying architecture [3.].

3 Comparison

Milvus stands out for its enterprise-grade scalability and performance, designed specifically for massive vector datasets and production workloads. Chroma DB excels in developer experience and ease of implementation, making it ideal for smaller projects and rapid development cycles. PgVector offers a pragmatic approach for PostgreSQL users and brings vector capabilities to an established relational database system. Table 1 highlights some of the key design characteristics and key components.

Database System	Architecture Type	Key Components	Design
Chroma	Vector database	Index support for disk-stored data	Optimized for larger datasets
Milvus	Purpose-built vector database	Distributed system, Graphics Processing Unit (GPU) acceleration, Log-Structured Merge-tree (LSM) based structure	Efficient management of large-scale vector data
pgVector	Generalized vector database	Integration with PostgreSQL	Support vector operations within a relational database

Table 1 Comparison by key components and design characteristics

<u>Use case suitability</u>

- **Milvus:** Ideal for enterprise-scale applications requiring robust production performance, low latency, and massive scale. Suitable for large organizations with datasets exceeding tens of millions of vectors with expectations of significant growth of vector data needs and complex applications. Provides the most robust scaling capabilities [4.].
- **Chroma DB:** Best for developers prioritizing rapid implementation and ease of use, particularly for applications with moderate-sized datasets (under one million vectors). Well-suited for startups, research projects, and development environments. For smaller datasets or proof-of-concept projects, Chroma DB offers a more streamlined implementation experience [4.].
- **pgVector:** Optimal for organizations already invested in PostgreSQL infrastructure who need to add vector search capabilities without adopting an entirely new database system. Particularly valuable when vector data needs to be queried alongside relational data [3.].

Integration with Existing Infrastructure

- **Milvus:** operates as a standalone specialized database requiring dedicated infrastructure but offers comprehensive API support
- **Chroma DB:** provides smooth integration with popular ML frameworks like TensorFlow and PyTorch
- **pgVector:** leverages existing PostgreSQL deployments, potentially reducing infrastructure complexity for organizations already using this database

Data storage

- **Milvus**: employs distributed storage with separation of computing and storage
- **Chroma:** uses disk-based storage with index support
- **pgVector:** integrates with PostgreSQL storage

The comparison of pgVector, Milvus, and Chroma DB reveals that each solution offers distinct advantages for different use cases and operational environments. The optimal choice depends on specific requirements regarding integration needs, scale, performance priorities, and existing infrastructure.

PgVector presents a compelling option for organizations already using PostgreSQL that want to incorporate vector search capabilities without adding infrastructure complexity. Its integration with existing data and familiar SQL interface provides a seamless experience, though potentially at the cost of specialized performance optimization for very large vector datasets.

Milvus stands out for its scalability and deployment flexibility, making it suitable for a wide range of applications from prototyping to enterprise-scale deployments. Its specialized design for vector operations and ability to handle billions of vectors position it as a robust solution for large-scale AI applications.

Chroma DB's in-memory architecture and user-friendly API make it an attractive option for developers seeking rapid implementation of vector search capabilities, particularly for applications integrated with language models that prioritize query speed over massive scale.

4 Conclusion

As vector search continues to grow in importance for AI applications, these databases represent different approaches to solving the fundamental challenge of efficient vector similarity search. Organizations should evaluate their specific requirements around data volume, query performance, integration needs, and operational constraints when selecting among these options.

5 References

[1.] Chroma documents, introduction **https://docs.trychroma.com/docs/overview/introduction**

[2.] Chroma DB tutorial: A Step-By-Step Guide, **https://www.datacamp.com/tutorial/chromadb-tutorial-step-by-step-guide**

[3.] pgvector Tutorial: Integrate Vector Search into PostgreSQL, **https://www.datacamp.com/tutorial/pgvector-tutorial?dc_referrer=https%3A%2F%2Fwww.perplexity.ai%2F**

[4.] Choosing a Vector Database: Milvus vs. Chroma DB, **https://zilliz.com/blog/milvus-vs-chroma**