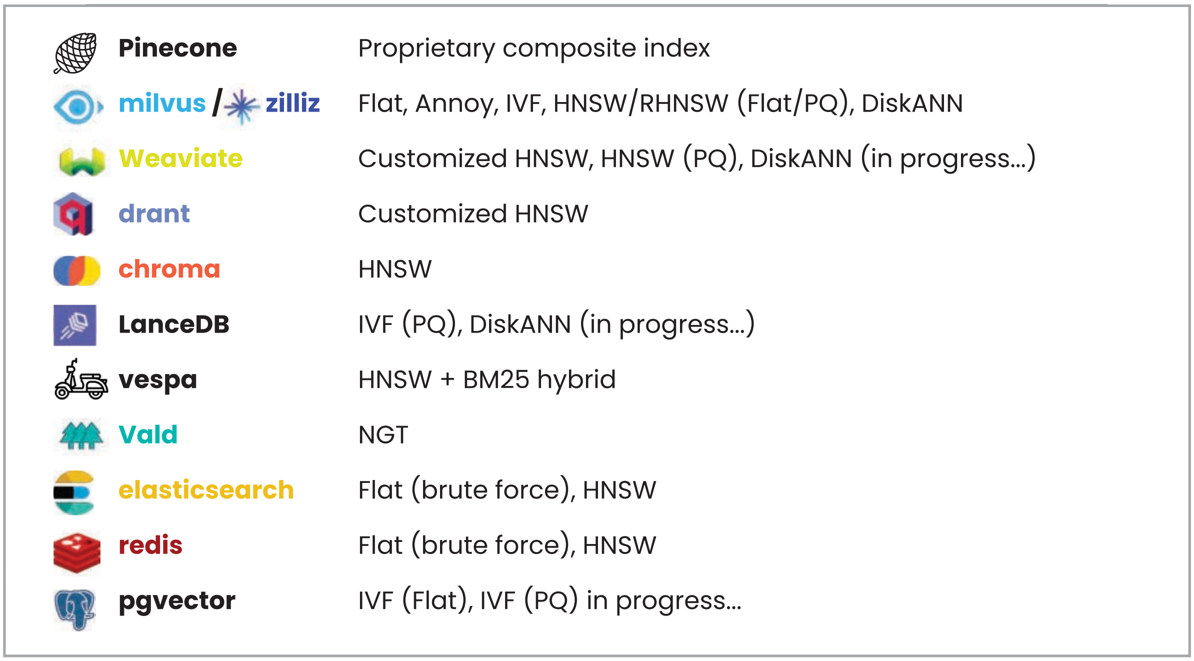
4.3

CHOOSING THE PERFECT VECTOR DATABASE

Once embeddings are generated, they are stored in a vector database. The vector database indexes these embeddings, organizing them for efficient similarity searches. In this chapter, we'll do a deep exploration of vector databases and how we can choose the right one for our use case.

A vector database is a specialized database management system designed to store, index, and query high-dimensional vectors efficiently. Unlike traditional relational databases that primarily handle structured data, vector databases are optimized for managing unstructured and semi-structured data, such as images, text, and audio represented as numerical vectors in a high-dimensional space.

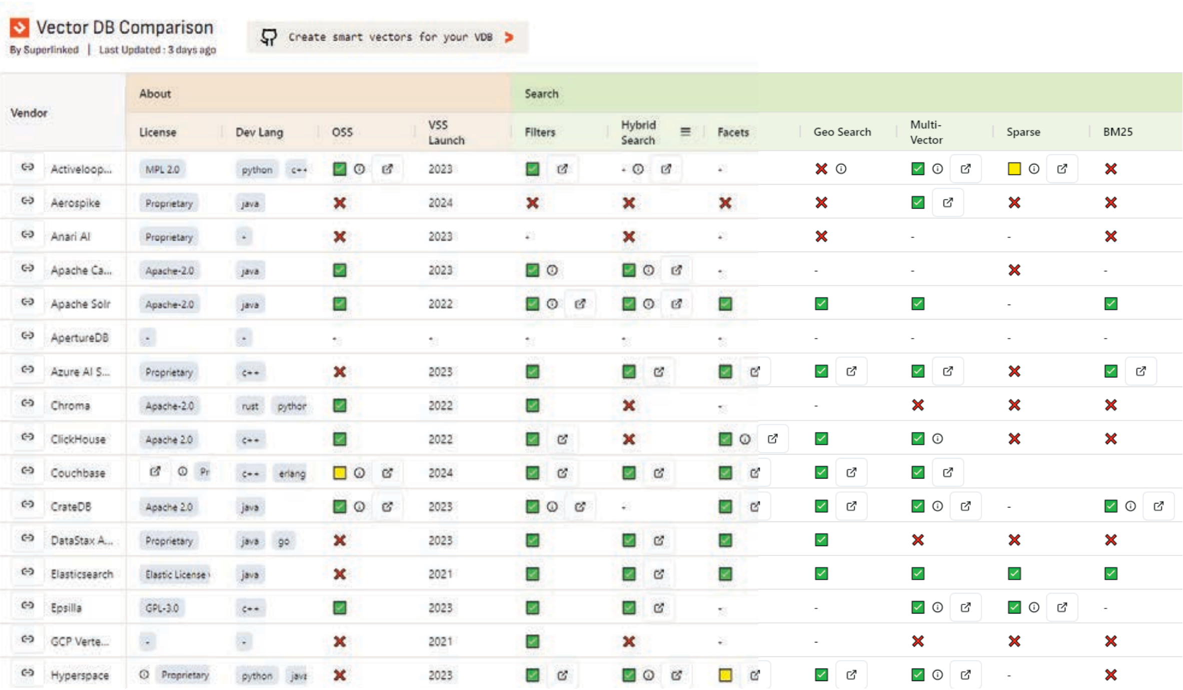
These vectors capture the inherent structure and relationships within the data. This helps in sophisticated similarity search, recommendation, and data analysis tasks.



Pinecone	Proprietary composite index
milvus / zilliz	Flat, Annoy, IVF, HNSW/RHNSW (Flat/PQ), DiskANN
Weaviate	Customized HNSW, HNSW (PQ), DiskANN (in progress...)
drant	Customized HNSW
chroma	HNSW
LanceDB	IVF (PQ), DiskANN (in progress...)
vespa	HNSW + BM25 hybrid
Vald	NGT
elasticsearch	Flat (brute force), HNSW
redis	Flat (brute force), HNSW
pgvector	IVF (Flat), IVF (PQ) in progress...

Fig 4.3.1: Various Vector DB(s)

The vector database you choose for your RAG system (see the list and comparisons in Fig 4.3.1 and Fig 4.3.2) will have a major impact on your RAG performance. Vector databases have emerged as a powerful solution for efficiently storing, indexing, and searching through unstructured data. In this guide, we'll look at key factors to consider when selecting a vector database for your Enterprise RAG system.



Vector DB Comparison

By Superlinked | Last Updated : 3 days ago

Vendor	About				Search						
	License	Dev Lang	OSS	VSS Launch	Filters	Hybrid Search	Facets	Geo Search	Multi-Vector	Sparse	BM25
Activeloop...	MPL 2.0	python c++	✅ ⓘ ↗	2023	✅ ↗	- ⓘ ↗	-	❌ ⓘ	✅ ⓘ ↗	🟨 ⓘ ↗	❌
Aerospike	Proprietary	java	❌	2024	❌	❌	❌	❌	✅ ↗	❌	❌
Anari AI	Proprietary	-	❌	2023	-	❌	-	❌	-	-	❌
Apache Ca...	Apache-2.0	java	✅	2023	✅ ⓘ	✅ ⓘ ↗	-	-	-	❌	-
Apache Solr	Apache-2.0	java	✅	2022	✅ ⓘ ↗	✅ ⓘ ↗	✅	✅	✅	-	✅
ApertureDB	-	-	-	-	-	-	-	-	-	-	-
Azure AI S...	Proprietary	c++	❌	2023	✅	✅ ↗	✅ ↗	✅ ↗	✅ ↗	❌	✅ ↗
Chroma	Apache-2.0	rust pythor	✅	2022	✅	❌	-	-	❌	❌	❌
ClickHouse	Apache 2.0	c++	✅	2022	✅ ↗	❌	✅ ⓘ ↗	✅	✅ ⓘ	❌	❌
Couchbase	↗ ⓘ Pr	c++ erlang	🟨 ⓘ ↗	2024	✅ ↗	✅ ↗	✅ ↗	✅ ↗	✅ ↗		
CrateDB	Apache 2.0	java	✅ ⓘ ↗	2023	✅ ⓘ ↗	-	✅ ↗	✅ ↗	✅ ⓘ ↗	-	✅ ⓘ ↗
DataStax A...	Proprietary	java go	❌	2023	✅	✅ ↗	✅	✅	❌	❌	❌
Elasticsearch	Elastic License	java	❌	2021	✅	✅ ↗	✅	✅	✅	✅	✅
Epsilla	GPL-3.0	c++	✅	2023	✅	✅ ↗	-	-	✅ ⓘ ↗	✅ ⓘ ↗	-
GCP Verte...	-	-	❌	2021	✅	❌	-	-	❌	❌	❌
Hyperspace	ⓘ Proprietary	python java	❌	2023	✅ ↗	✅ ⓘ ↗	🟨 ↗	✅ ↗	✅ ⓘ ↗	-	❌

Fig 4.3.2: Comparison of vector databases

Sourced from superlinked.com/vector-db-comparison

Key Factors

Open-Source (OSS)

Open-source vector databases provide you with transparency, flexibility, and community-driven development. They often have active communities contributing to their improvement and may be more cost-effective for you if you have limited budgets. Examples include Milvus, Annoy, and FAISS.

Private

Proprietary vector databases offer additional features, dedicated support, and may be better suited for you if you have specific requirements or compliance needs. Examples include Elasticsearch, DynamoDB, and Azure Cognitive Search.

Language Support

You'll need to make sure that the vector database supports the programming languages commonly used within your organization. Look for comprehensive client libraries and SDKs for languages such as Python, Java, JavaScript, Go, and C++. This helps ensure seamless integration with your existing applications and development frameworks.

Below is a small exercise you can undertake.

- First, identify the primary programming languages used in your organization.
- Choose a vector database (e.g., Milvus, FAISS, Elasticsearch) that we looked at in the previous point. Consider what works best for you, i.e., OSS or private vector database.
- Go through the client libraries and SDKs provided by the vector database for the programming languages you identified.
- Optional: Write a small script in one of the primary languages to connect to the vector database, insert a sample vector, and retrieve it.

License

After completing the exercise, move to evaluate the vector database's licensing model. This is to check its compatibility with your organization's policies and objectives. Common licenses include Apache License 2.0, GNU General Public License (GPL), and commercial proprietary licenses. You'll need to list and understand any restrictions, obligations, or usage limitations imposed by the license. Here's a quick exercise for you to complete.

- Select a vector database and review its licensing terms (e.g., Apache License 2.0, GPL, proprietary).
- Then, compare the license terms with your organization's legal and operational requirements.
- Identify any restrictions or obligations that may impact your usage and look at ways you can address them. In the end, create a summary of your findings.

Maturity

After summarizing your findings with respect to the licensing models, the next important step would be to assess the vector database's maturity by considering factors like development, adoption, and community support. Look for databases with a proven track record of stability, reliability, and scalability. Also, consider factors such as release frequency, community activity, and longevity in the market. Here's an exercise for you to complete.

Create a comparison matrix to help you understand the maturity of each database you have shortlisted. Below is a reference you can use.

Criteria	Weightage	Database X (Score)	Database X (Weighted)	Database Y (Score)	Database Y (Weighted)	Database Z (Score)	Database Z (Weighted)
Release History	20%	5	1.00	4	0.80	4	0.80
Version Stability	20%	4	0.80	5	1.00	4	0.80
Frequency of Updates	10%	5	0.50	3	0.30	4	0.40
Community Activity	20%	5	1.00	3	0.60	4	0.80
Industry Adoption	20%	4	0.80	4	0.80	5	1.00
Language Support	10%	4	0.40	3	0.30	5	0.50
Total Score			4.50		3.80		4.30

Table 4.3.1: Comparison matrix template to evaluate Vector DB(s) on different parameters

Let's explore key enterprise features that you should consider when evaluating vector databases for their complex data management needs.

Enterprise Features

Regulatory Compliance Open-Source (OSS)

First and foremost, you'll need to ensure that the vector databases comply with industry standards and regulations, such as SOC-2 certification. This ensures that data management practices meet stringent security and privacy requirements.

SSO

Single Sign-On (SSO) integration allows users to access the vector database using their existing authentication credentials from other systems, such as Google, Microsoft, or LDAP. SSO streamlines user access management, enhances security, and improves user experience by eliminating the need for multiple logins.

Rate Limits

Rate limits are thresholds or constraints imposed on the rate of incoming requests or operations within a specified timeframe. By setting predefined limits on the number of queries, inserts, updates, or other operations, you can prevent system overload, prioritize critical tasks, and maintain optimal performance.

Multi-tenancy

Multi-tenant support enables efficient resource sharing and isolation for multiple users or clients within a single database instance, including user authentication, access control, and resource allocation policies. It enhances scalability and resource utilization in multi-user environments.

Role-based Control

Role-based control mechanisms enable administrators to define access privileges and permissions based on user roles and responsibilities. This ensures that only authorized personnel can access, modify, or delete sensitive data within the vector database. Role-based access control (RBAC) enhances security, mitigates risks, and facilitates compliance with regulatory mandates such as GDPR and HIPAA.