## What is a vector database?

A vector database stores, manages and indexes high-dimensional vector data. Data points are stored as arrays of numbers called “vectors,” which are clustered based on similarity. This design enables low-latency queries, making it ideal for AI applications.

Vector databases are growing in popularity because they deliver the speed and performance needed to drive generative artificial intelligence (AI) use cases and applications. According to Gartner®, by 2026, more than 30% of enterprises will have adopted vector databases to build their foundation models with relevant business data.[1]

## Vector databases versus traditional databases

Unlike traditional relational databases with rows and columns, data points in a vector database are represented by vectors with a fixed number of dimensions. Because they use high-dimensional vector embeddings, vector databases are better able to handle unstructured datasets.

The nature of data has undergone a profound transformation. It's no longer confined to structured information easily stored in traditional databases. Unstructured data—including social media posts, images, videos, audio clips and more—is growing 30% to 60% year over year.[2]

Relational databases excel at managing structured and semistructured datasets in specific formats. Loading unstructured data sources into a traditional relational database to store, manage and prepare the data for artificial intelligence (AI) is a labor-intensive process, especially with new generative use cases such as similarity search.

Traditional search typically represents data by using discrete tokens or features, such as keywords, tags or metadata. Traditional searches rely on exact matches to retrieve relevant results. For example, a search for "smartphone" would return results containing the word "smartphone."

Opposed to this, vector search represents data as *dense vectors*, which are vectors with most or all elements being nonzero. Vectors are represented in a continuous *vector space*, the mathematical space in which data is represented as vectors.

Vector representations enable similarity search. For example, a vector search for “smartphone” might also return results for “cellphone” and “mobile devices.”

Each dimension of the dense vector corresponds to a latent feature or aspect of the data. A *latent feature* is an underlying characteristic or attribute that is not directly observed but inferred from the data through mathematical models or algorithms.

Latent features capture the hidden patterns and relationships in the data, enabling more meaningful and accurate representations of items as vectors in a high-dimensional space.

## What are vectors?

Vectors are a subset of *tensors*, which in machine learning (ML) is a generic term for a group of numbers—or a grouping of groups of numbers—in *n*-dimensional space. Tensors function as a mathematical bookkeeping device for data. Working up from the smallest element:

- A *scalar* is a zero-dimensional tensor, containing a single number. For example, a system modeling weather data might represent a single day's high temperature (in Fahrenheit) in scalar form as *85*.

- Then, a *vector* is a one-dimensional (or *first-degree* or *first-order*) tensor, containing *multiple* scalars of the same type of data. For example, a weather model might use the low, mean and high temperatures for a single day in vector form: *62*, *77*, *85*. Each scalar component is a *feature*—that is, a dimension—of the vector, representing a feature of that day's weather.

Vector numbers can represent complex objects such as words, images, videos and audio generated by an ML model. This high-dimensional vector data, containing multiple features, is essential to machine learning, natural language processing (NLP) and other AI tasks. Some example uses of vector data include:

- *Text:* Chatbots need to understand natural language. They do this by relying on vectors that represent words, paragraphs and entire documents.

- *Images:* Image pixels can be described by numerical data and combined to make up a high-dimensional vector for that image.

- *Speech or audio:* Like images, sound waves can also be broken into numerical data and represented as vectors, enabling AI applications such as voice recognition.

## What are vector embeddings?

Vector embeddings are numerical representations of data points that convert various types of data—including nonmathematical data such as words, audio or images—into arrays of numbers that ML models can process.

Artificial intelligence (AI) models, from simple linear regression algorithms to the intricate neural networks used in deep learning, operate through mathematical logic.

Any data that an AI model uses, including unstructured data, needs to be recorded numerically. Vector embedding is a way to convert an unstructured data point into an array of numbers that expresses that data's original meaning.

Here's a simplified example of word embeddings for a very small corpus (2 words), where each word is represented as a 3-dimensional vector:

- cat [0.2, -0.4, 0.7]
- dog [0.6, 0.1, 0.5]

In this example, each word ("cat") is associated with a unique vector ([0.2, -0.4, 0.7]). The values in the vector represent the word's position in a continuous 3-dimensional vector space.

Words with similar meanings or contexts are expected to have similar vector representations. For instance, the vectors for "cat" and "dog" are close together, reflecting their semantic relationship.

*Embedding models* are trained to convert data points into vectors. Vector databases store and index the outputs of these embedding models. Within the database, vectors can be grouped together or identified as opposites based on semantic meaning or features across virtually any data type.

Vector embeddings are the backbone of recommendations, chatbots and generative apps such as ChatGPT.

For example, take the words "car" and "vehicle." They have *similar* meanings but are spelled differently. For an AI application to enable effective semantic search, the vector representations of "car" and "vehicle" must capture their semantic similarity. In machine learning, embeddings represent high-dimensional vectors that encode this semantic information.

How are vector databases used?

Vector databases serve three key functions in AI and ML applications:

- **Vector storage**
- **Vector indexing**
- **Similarity search based on querying or prompting**

In operation, vector databases work by using multiple algorithms to conduct an approximate nearest neighbor (ANN) search. The algorithms are then gathered in a pipeline to quickly and accurately retrieve and deliver data neighboring the vector that is queried.

For example, an ANN search could look for products that are visually similar in an e-commerce catalog. Additional uses include anomaly detection, classification and semantic search. Because a dataset runs through a model just once, results are returned within milliseconds.

## Vector storage

Vector databases store the outputs of an embedding model algorithm, the vector embeddings. They also store each vector's metadata—including title, description and data type—which can be queried by using metadata filters.

By ingesting and storing these embeddings, the database can facilitate fast retrieval of a similarity search, matching the user's prompt with a similar vector embedding.

## Vector indexing

Vectors need to be indexed to accelerate searches within high-dimensional data spaces. Vector databases create indexes on vector embeddings for search functions.

The vector database indexes vectors by using an ML algorithm. Indexing maps the vectors to new data structures that enable faster similarity or distance searches, such as nearest neighbor searches, between vectors.

Vectors can be indexed by using algorithms such as hierarchical navigable small world (HNSW), locality-sensitive hashing (LSH) or product quantization (PQ).

- *HNSW* is popular as it creates a tree-like structure. Each node of the tree shows a set of vectors complete with the hierarchies in each. Similarities between vectors are shown at the edges between the nodes.
- *LSH* indexes content by using an approximate nearest-neighbor search. For extra speed, the index can be optimized by returning an approximate, but nonexhaustive result.
- *PQ* converts each dataset into a short, memory-efficient representation. Only the short representations are stored, rather than all of the vectors.

## Similarity search based on querying or prompting

Query vectors are vector representations of search queries. When a user queries or prompts an AI model, the model computes an embedding of the query or prompt. The database then calculates distances between query vectors and vectors stored in the index to return similar results.

Databases can measure the distance between vectors with various algorithms, such as nearest neighbor search. Measurements can also be based on various similarity metrics, such as cosine similarity.

The database returns the most similar vectors or nearest neighbors to the query vector according to the similarity ranking. These calculations support various machine learning tasks, such as recommendation systems, semantic search, image recognition and other natural language processing tasks.

## Advantages of vector databases

Vector databases are a popular way to power enterprise AI-based applications because they can deliver many benefits:

- **Speed and performance**
- **Scalability**
- **Lower cost of ownership**
- **Data management**
- **Flexibility**

### Speed and performance

Vector databases use various indexing techniques to enable faster searching. Vector indexing and distance-calculating algorithms such as nearest neighbor search can help optimize performance when searching for relevant results across large datasets with millions, if not billions, of data points.

One consideration is that vector databases provide approximate results. Applications requiring greater accuracy might need to use a different kind of database at the cost of a slower processing speed.

### Scalability

Vector databases can store and manage massive amounts of unstructured data by scaling horizontally with additional nodes, maintaining performance as query demands and data volumes increase.

### Lower cost of ownership

Because they enable faster data retrieval, vector databases speed the training of foundation models.

### Data management

Vector databases typically provide built-in features to easily update and insert new unstructured data.

### Flexibility

Vector databases are built to handle the added complexity of using images, videos or other multidimensional data.

Given the multiple use cases ranging from semantic search to conversational AI applications, vector databases can be customized to meet business and AI requirements. Organizations can start with a general-purpose model such as IBM® Granite™ series models, Meta's Llama-2 or Google's Flan models, and then provide their own data in a vector database to enhance the output of the models and AI applications.

## Considerations for vector databases and data strategy

Organizations have a breadth of options when choosing a vector database capability. To find one that meets their data and AI needs, many organizations consider:

- **Types of vector databases**
- **Integration with a data ecosystem**
- **When vector indexing is not optimal**
- **Tools for creating and deploying vector databases**

## Types of vector databases

There are a few alternatives to choose from.

- Stand-alone, proprietary, fully vectorized databases such as Pinecone.
- Open-source solutions such as Weaviate or Milvus, which provide built-in RESTful APIs and support for Python and Java programming languages.
- Data lakehouses with vector database capabilities integrated, such as IBM watsonx.data™.
- Vector database and database search extensions such as PostgreSQL's open source pgvector extension, which provides vector similarity search capabilities. An SQL vector database can combine the advantages of a traditional SQL database with the power of a vector database.

## Integration with a data ecosystem

Vector databases should not be considered as stand-alone capabilities, but rather a part of a broader data and AI ecosystem.

Many offer APIs, native extensions or can be integrated with databases. Because vector databases are built to use enterprise data to enhance models, organizations must also have proper

data governance and security in place to help ensure that the data used to train large language models (LLMs) can be trusted.

Beyond APIs, many vector databases use programming-language-specific software development kits (SDKs) that can wrap around the APIs. Using the SDKs, developers often find it easier to work with the data in their apps.

## When vector indexing is not optimal

Using a vector store and index is well suited for applications that are based on facts or fact-based querying, such as extracting specific information from complex documents.

However, asking for a summary of topics would not work well with a vector index. In this case, an LLM would go through all the different possible contexts on that topic within the data.

A faster option would be to use a different kind of index, such as a list index rather than a vector index, because a list index would immediately fetch the first element in each listing.

## Tools for creating and deploying vector databases

To optimize vector database development, LangChain is an open-source orchestration framework for developing applications that use LLMs.

Available in both Python-based and JavaScript-based libraries, LangChain's tools and APIs simplify the process of building LLM-driven apps such as chatbots and virtual agents. LangChain provides integrations for over 25 different embedding methods, and for over 50 different vector stores (both cloud-hosted and local).

To power enterprise-grade AI, a data lakehouse might be paired with an integrated vector database. Organizations can unify, curate and prepare vectorized embeddings for their generative AI applications at scale across their trusted, governed data. This enhances the relevance and precision of their AI workloads, including chatbots, personalized recommendation systems and image similarity search applications.

## Vector database use cases

The applications for vector databases are vast and growing. Some key use cases include:

- **Retrieval-augmented generation (RAG)**
- **Conversational AI**
- **Recommendation engines**
- **Vector search**

## Retrieval-augmented generation (RAG)

Retrieval-augmented generation (RAG) is an AI framework for enabling large language models (LLMs) to retrieve facts from an external knowledge base. Vector databases are key to supporting RAG implementations.

Enterprises are increasingly favoring RAG in generative AI workflows for its faster time-to-market, efficient inference and reliable output. The framework is particularly helpful in use cases such as customer care, HR and talent management.

RAG helps ensure that a model is linked to the most current, reliable facts and that users have access to the model's sources so that its claims can be verified. Anchoring the LLM in trusted data can help reduce model hallucinations.

RAG uses high-dimensional vector data to enrich prompts with semantically relevant information for in-context learning by foundation models. RAG requires effective storage and retrieval during the inference stage, which handles the highest volume of data.

Vector databases excel at efficiently indexing, storing and retrieving these high-dimensional vectors, providing the speed, precision and scale needed for applications such as recommendation engines and chatbots.

### Conversational AI

Vector databases, particularly when used to implement RAG frameworks, can help improve virtual agent interactions by enhancing the agent's ability to parse relevant knowledge bases efficiently and accurately. Agents can provide real-time contextual answers to user queries, along with the source documents and page numbers for reference.

### Recommendation engines

E-commerce sites, for instance, can use vectors to represent customer preferences and product attributes. This enables them to suggest items similar to past purchases, based on vector similarity, enhancing user experience and increasing retention.

### Vector search

This search technique is used to discover similar items or data points, typically represented as vectors, in large collections. Vector search can capture the semantic relationships between elements, enabling effective processing by machine learning models and artificial intelligence applications.

These searches can take several forms.

- *Semantic search:* Perform searches based on the meaning or context of a query, enabling more precise and relevant results. Because both words and phrases can be represented as vectors, semantic vector search functions understand user intent better than general keywords.

- *Similarity search and applications:* Find similar images, audio, video or text data to support advanced image and speech recognition and natural language processing. Images and video can be indexed and retrieved based on similarity.