**Vector databases for advanced queries on complex data**

Vector databases are designed to store and query data with many attributes. Vector databases are a big part of the modern data ecosystem because they can show similarities in high-dimensional data. They do this by measuring the closeness of vectors based on distance metrics, like cosine similarity.

Let’s say you documented your entire life in a single photo album. Photos have a ton of attributes— people, locations, colors, times of day, and camera settings. If you wanted to find every photo of your best friend, you’d have to do a lot of manual searching and sorting. On the other hand, if your photo album were stored in a vector database, you’d simply tell the database what you’re looking for, and it’d find it faster than any human could.

This is possible because the vector database stores all attributes as dimensions in a vector space, which is a list of numbers in a unique pattern. Photos, videos, and audio files are often put in vector databases because they hold so much data that you’d otherwise not search efficiently.

To get a little more technical, vectors are created by embeddings—a process where raw data is transformed into high-dimensional vector space. Techniques like deep learning make the semantic and contextual essence clear in the vector. Vector databases are often integrated with machine learning pipelines, where the ML models create the embeddings and ingest them into the vector database for querying and analysis.

Since many vector databases support both real-time ingestion and batch processing, they’re popular in real-time use cases. For example, vector databases help Netflix recommend your next binge-worthy show, ChatGPT to answer your prompts, and your credit card company to predict fraudulent transactions.

***One downside to vector databases is their high demand for computation power. To reduce this, popular vector databases like***[***Pinecone***](https://www.pinecone.io/)***and [Weaviate](https://weaviate.io/) often use Approximate Nearest Neighbor (ANN) algorithms for searches on large datasets. ANN algorithms trade a small amount of accuracy for a huge boost in speed.***

**Graph databases for powerful relationship mapping**

Graph databases are designed to represent and store data as graphs. This makes it easy to represent people, products, and events along with what ties them together. Search engines, logistics businesses, and social networks typically use graph databases to understand connections in their data.

Graph databases stand out for their unique ability to represent connections between datasets. They are based on mathematical graph theory and consist of two key parts: nodes and edges.

Nodes are the primary entities in a graph database. Each node holds all data about a person, product, business, event, or another entity.

Edges are the connecting parts of graph databases. They show similarities, relationships, and commonalities. You can define the properties and weights of edges to fit your purpose.

Graph databases analyze the structures and patterns that connect data points. They uncover influential nodes in a network, detect anomalies in transaction data, and easily adapt as data grows and evolves. This is what makes graph databases suited for dynamic datasets and applications.

*The power of a fine-tuned graph database can be impressive. For example, Facebook uses its graph database to suggest new friends based on common friends and interests. The*[*Facebook database*](https://neo4j.com/news/how-facebook-matured-its-data-structure-and-stepped-into-the-graph-world/)*also powers*[*effective cross-channel advertising*](https://www.facebook.com/business/goals/maximize-media-performance-with-data)*with ever-evolving automation on Meta platforms. And, it lets Facebook discover data as belonging to non-users that it then creates*[*shadow profiles*](https://en.wikipedia.org/wiki/Shadow_profile)*for.*

**When to use vector databases vs. graph databases**

Both vector and graph databases offer valuable solutions for business-specific challenges – not least for real-time data. Graph databases bring you the full power of relationships in data, while vector databases are best suited for managing and querying high-dimensional data in use cases that require similarity searches and machine learning.

While both graph and vector databases offer powerful capabilities, they also come with drawbacks. You should consider your data, queries, and specific objectives – technical as well as business – before choosing one, the other, or an integration. To make the decision easier, here are the main pros and cons of each.

**Advantages of vector databases**

**Technical**

Efficient high-dimensional data handling: Ideal for complex data types like images, text, or audio.

Advanced similarity searches: Quick to identify data points close to a given query vector in the multi-dimensional space.

Integration with machine learning: Seamless storage and query solution for embeddings from ML models.

**Business**

Improved user engagement and revenue: Finding intricate patterns in big data to support content discovery, personalized user experiences, recommendation systems e-commerce, streaming platforms, and more.

Scalability, timely insights, and maintained system performance: Businesses can scale without compromising speed, availability, and other system performance, as the database expands.

Better decisions and automation: Many machine learning models output embeddings that work well in vector databases. This informs better decisions and helps automate processes.

**Disadvantages of vector databases**

Curse of dimensionality: Search efficiency can decrease, and data can get sparse with increased dimensionality. Vector databases employ techniques to mitigate it, but it’s still an issue.

Precision trade-off: Fast search times come at the cost of accuracy.

High memory and storage requirements: Storing high-dimensional vectors can be memory-intensive – especially with large datasets.

**Advantages of graph databases**

**Technical**

Optimized supply chains and fraud detection: Businesses get actionable relationship insights that help them understand customer behavior, smooth out supply chains, and detect suspicious activity.

Fast and flexible: With flexible data modeling, businesses can adapt faster to changing needs, integrate new data sources, and keep data infrastructure aligned with business goals.

Make better decisions, faster: With fast query times come faster decisions based on valuable insights.

**Business**

Native relationship handling: Graph databases efficiently span connections between data points without the need for demanding join operations of relational databases.

Flexible schema: Schema-less or schema-optional data modeling helps you avoid extensive restructuring as data or data types evolve.

Optimized for complex queries: Multi-step, intricate queries make it easy to find the shortest path, detect cycles, or identify clusters.

**Disadvantages of graph databases**

Scalability challenges: Historically hard to scale across multiple nodes compared to traditional relational databases. Luckily, this is improving with modern graph databases.

Steep learning curve: Query languages for graph databases (like Cypher for Neo4j) can be different from standard SQL and take time to learn.

Unnecessary overhead: Datasets that don’t make use of relationship-focused technology will have lower efficiency because of the unneeded overhead in managing relationships.

As you go over the strong pros and cons of both technologies, you might start to find the thought of integrating them appealing. Let's take a quick look at what that solution would entail.

**Benefits of combining graphs and vectors in a database**

It can be hard to make the choice and go with either graph or vector technology for your database. With generative AI, large language models (LLM), and real-time data playing an increasing part in modern applications, we’re seeing an increase in combined solutions.

This is why Neo4j recently added the ability to perform vector similarity search. They aim to make more sense of data and combat LLM hallucinations by blending similar feature vectors to input vectors found through lookups in the knowledge graph.

Combining graphs and vectors is novel and demanding, but it can offer some clear benefits.

Richer data representation: Graphs give you a structured view of relationships, while vectors allow for a deep understanding of data content and semantics. Together they provide a holistic data representation.

Enhanced query options: Hybrid queries can uncover relations and similarities for efficient searches that bring back more insights.

Improved recommendation systems: Recommendations can be spookily precise when they’re based on both user interactions and the actions of similar users.

Scalable knowledge graphs: Knowledge graphs combined with deep insights from embeddings can answer complex queries considering both relationships and semantics.

Unified data management: Manage unstructured and structured data types with streamlined operations and simpler infrastructure.

**The challenges of integrating vector and graph databases**

As we mentioned previously, the big advantages and disadvantages of graph and vector databases make the thought of integrating them appealing to many. However, the technologies bring their issues along when they move in together.

Graphs are costly and often slow to update because graphs take up a lot more memory and compute capacity than a pure vector database designed to store vectors. It’s common to require the setup of well-designed streams that update nodes and relationships individually to increase throughput. You might also want to consider an order-dependent design.

Whatever approach you decide to follow, a good first step is to get familiar with the different tools you can use to implement them. Happily enough, we've listed those for you here.

**Popular vector database tools**

When you need to perform similarity searches on high-dimensional data, vector databases are the obvious choice. But there’s still a lot to consider before you pick a vector database tool to work with. A tool that fits the size and nature of your data, your specific operations, integration needs, and other key factors will improve scalability, efficiency, and your operations.

Facebook AI Similarity Search (Faiss) library: [Faiss](https://faiss.ai/) is a library developed by Facebook for similarity search in vector databases with large datasets. It supports GPU acceleration and is known for high speeds fit for real-time search.

Pinecone managed database: [Pinecone](https://www.pinecone.io/) is a fully managed vector database designed for high-performance AI applications that need to scale with ease.

**Popular graph database tools**

As we mentioned for the vector database tools, you’ll make the best choices when you consider your use case in detail. How complex are the queries you’ll be executing? What does the volume of data and relationships look like? Which integrations do you need? Would you like to have support from communities around the tools? Here’s what you can use. Neo4j graph database: One of the most well-known graph databases. [Neo4j](https://neo4j.com/) is known for popularizing the Labeled Property Graph model (LPG). LPG is also used by others, but Neo4j has stayed focused on this approach to modeling graphs with their powerful query language [Cypher](https://neo4j.com/product/cypher-graph-query-language/). Neo4j also offers visualization tools, many integrations, and a high level of security.

ArangoDB multi-model database: The [ArangoDB](https://www.arangodb.com/) database supports graph, document, and key/value data models. Its query language AQL supports complex graph traversal and aggregation. It also offers SmartGraphs to optimize sharded graph queries in cluster deployments and high scalability.

Memgraph open-source graph database: [Memgraph](https://memgraph.com/) is a popular graph database. It’s marketed as a Neo4j-compatible graph database without the Neo4j complexity and costs.