Page 1: Introduction to Vector Databases

Vector databases are specialized database systems designed to store, index, and query high-dimensional vector data efficiently. Unlike traditional databases that work with structured data like numbers and strings, vector databases are optimized for similarity search operations on dense vector representations.

The rise of vector databases has been driven by the explosion of AI applications, particularly in natural language processing, computer vision, and recommendation systems. These applications generate embeddings - dense vector representations of data that capture semantic meaning in a mathematical format.

Key characteristics of vector databases include support for approximate nearest neighbor (ANN) search, horizontal scalability, and integration with machine learning pipelines. Popular vector databases include Pinecone, Weaviate, Chroma, and Qdrant.

Page 2: Similarity Search and Distance Metrics

Similarity search is the core functionality of vector databases. When you query a vector database, you're essentially asking: "Find me the vectors that are most similar to this query vector." This similarity is measured using distance metrics.

The most common distance metrics include:

Cosine similarity measures the cosine of the angle between two vectors, making it ideal for high-dimensional data where magnitude matters less than direction. Euclidean distance calculates the straight-line distance between two points in vector space. Manhattan distance sums the absolute differences between vector components.

Each metric has different characteristics and use cases. Cosine similarity is particularly popular in NLP applications because it focuses on the orientation of vectors rather than their magnitude, making it robust to document length variations.

Page 3: Applications in RAG Systems

Retrieval Augmented Generation (RAG) systems represent one of the most important applications of vector databases in modern AI. RAG systems combine the power of large language models with external knowledge retrieval to provide more accurate and up-to-date responses.

In a typical RAG workflow, documents are first converted into embeddings using models like OpenAI's text-embedding-ada-002 or sentence transformers. These embeddings are then stored in a vector database with metadata linking back to the original text chunks.

When a user asks a question, the query is embedded using the same model, and the vector database performs a similarity search to find the most relevant document chunks. These chunks are then provided as context to a language model, which generates a response grounded in the retrieved information.

Page 4: Performance and Optimization

Vector database performance depends on several factors including indexing algorithms, dimensionality of vectors, and query patterns. Most modern vector databases use approximate nearest neighbor algorithms like HNSW (Hierarchical Navigable Small World) or IVF (Inverted File) to achieve sub-linear search times.

Optimization strategies include dimensionality reduction techniques, proper index configuration, and efficient batching of operations. Some systems also support filtering capabilities, allowing you to combine vector similarity search with traditional database filtering on metadata.

The choice of embedding model significantly impacts both accuracy and performance. Smaller models like text-embedding-3-small offer faster inference and lower storage requirements, while larger models may provide better semantic understanding at the cost of increased computational overhead.