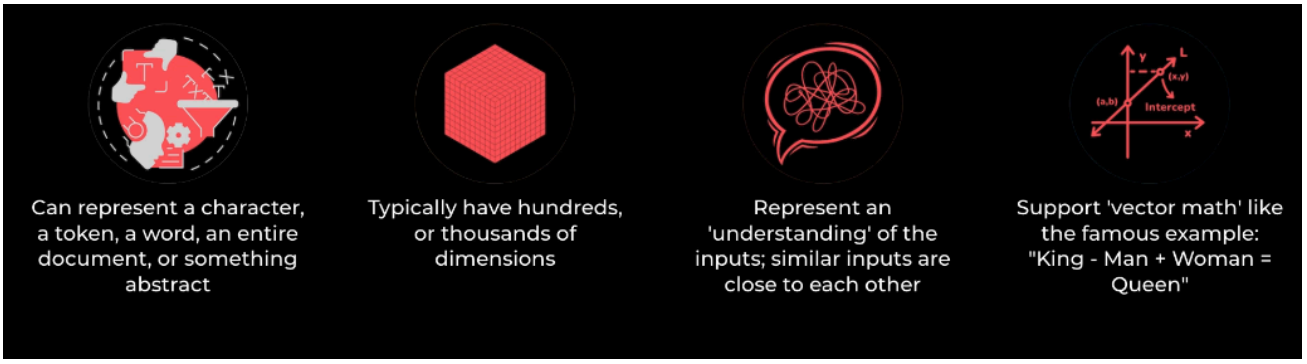
03 Intro to Vector DB

Auto-Encoding vs Auto-Regressive LLMs

- Auto-regressive LLMs predict a future token from the past - GPT3.5, Llama3.2
- Auto-encoding LLMs produce output based on the full input - BERT, RoBERTa, DistilBERT, ALBERT

Auto-encoding LLMs

- Applications include Sentiment Analysis and classification
- Also used to calculate "Vector Embeddings", representing an input as a list of numbers - i.e. a vector
- Examples include `BERT` from Google and `OpenAIEmbeddings` from OpenAI, `nomic_embed_text`



Vector embedding

Vector embedding is the process of representing objects, such as words, sentences, or entities, as vectors in a continuous vector space.

> The primary idea behind vector embedding is to capture semantic relationships between objects. In the context of word embeddings, for example, words with similar meanings are represented by vectors that are closer together in the vector space. This allows machine learning models to better understand the contextual and semantic relationships between words.

Side Note:

LLMs and vector embeddings

As a concrete example, the underlying architecture of the model for chat GTP involves the use of vectors. The model processes input data, such as text, by converting it into numerical vectors.

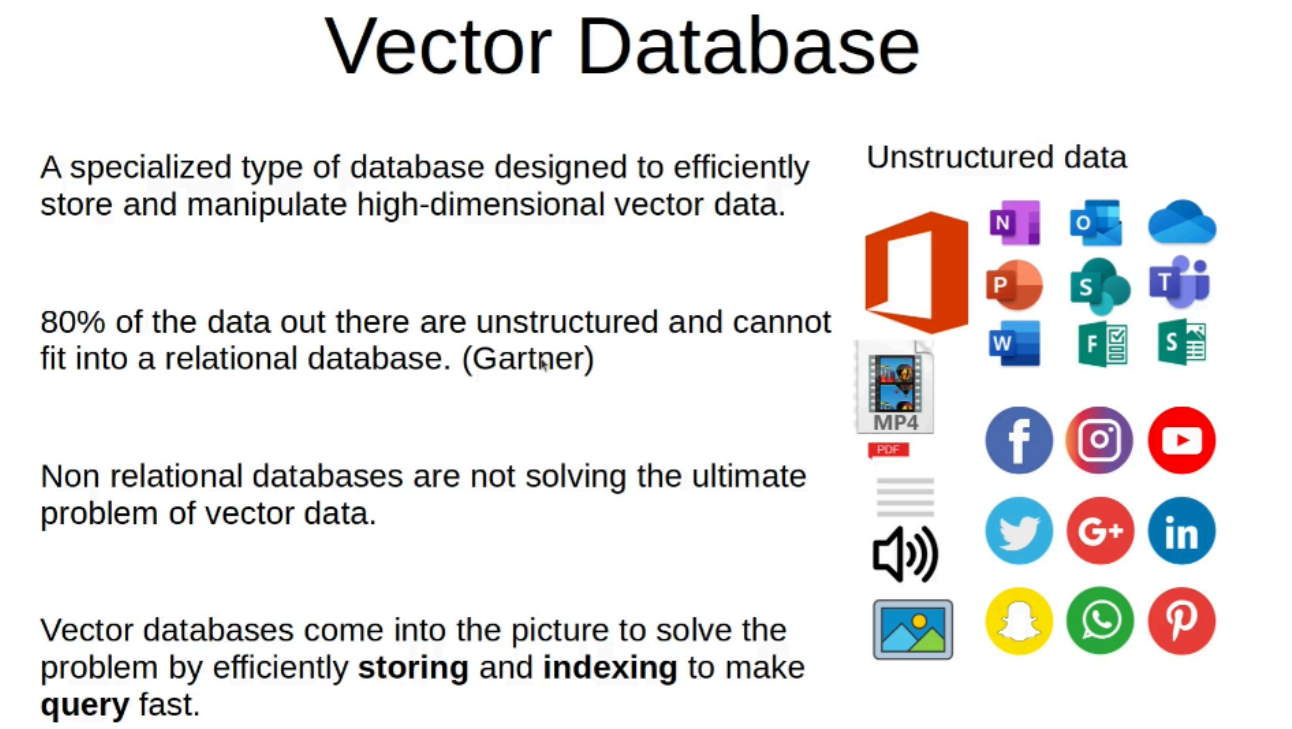
These vectors capture the semantic and contextual information of the input, allowing the model to understand and generate coherent and contextually relevant responses. The Transformer architecture, which GPT-3.5 is built upon, utilizes self-attention mechanisms to weigh the importance of different words in a sequence, further enhancing the model's ability to capture relationships and context.

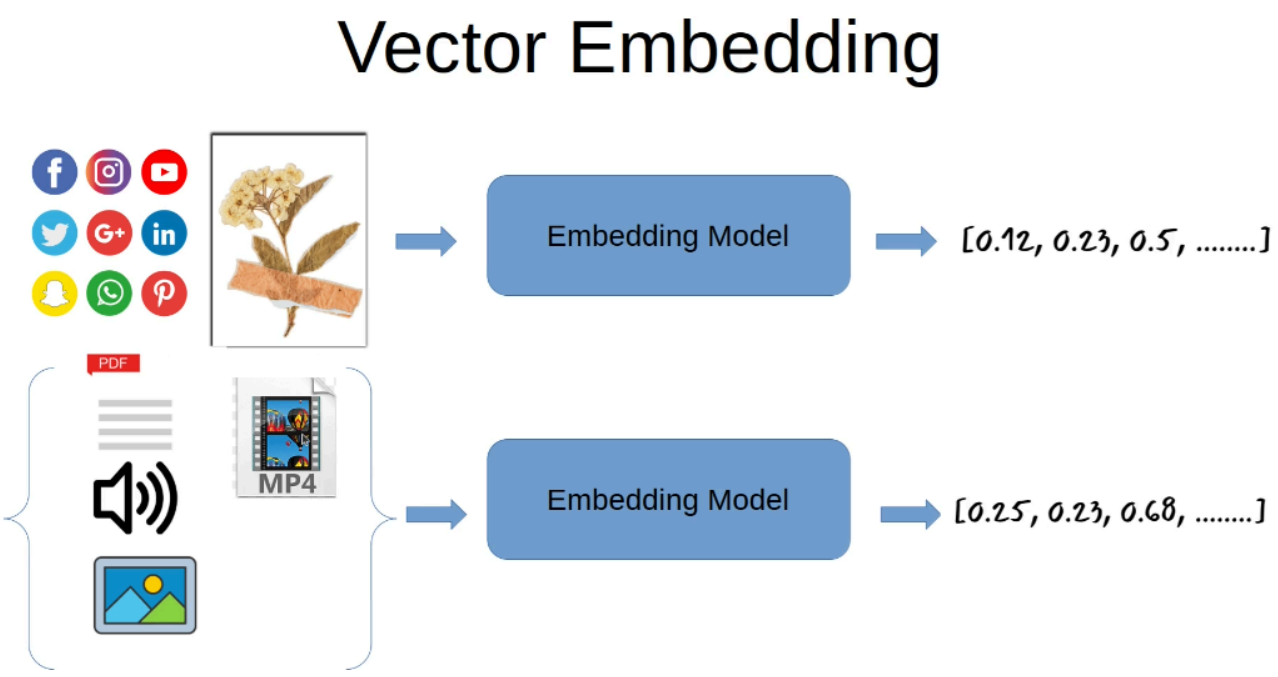
"Self-attention" refers to the model's capability to assign varying degrees of importance to different words within the input sequence.

So, in essence, the GPT-3.5 model operates on vectorized representations of language to perform various natural language understanding and generation tasks. This vector-based approach is a key factor in the model's success across a wide range of language-related applications.

Vector databases

A vector database is a specialized database designed to store and query vectorized data rapidly. Unlike conventional databases, which organize data in tables, a vector database represents data as vectors in a multi-dimensional space. These vectors encapsulate essential attributes of the items they represent, making vector databases ideal for tasks requiring similarity searches, nearest neighbor queries, and assessing distances or similarities between vectors.



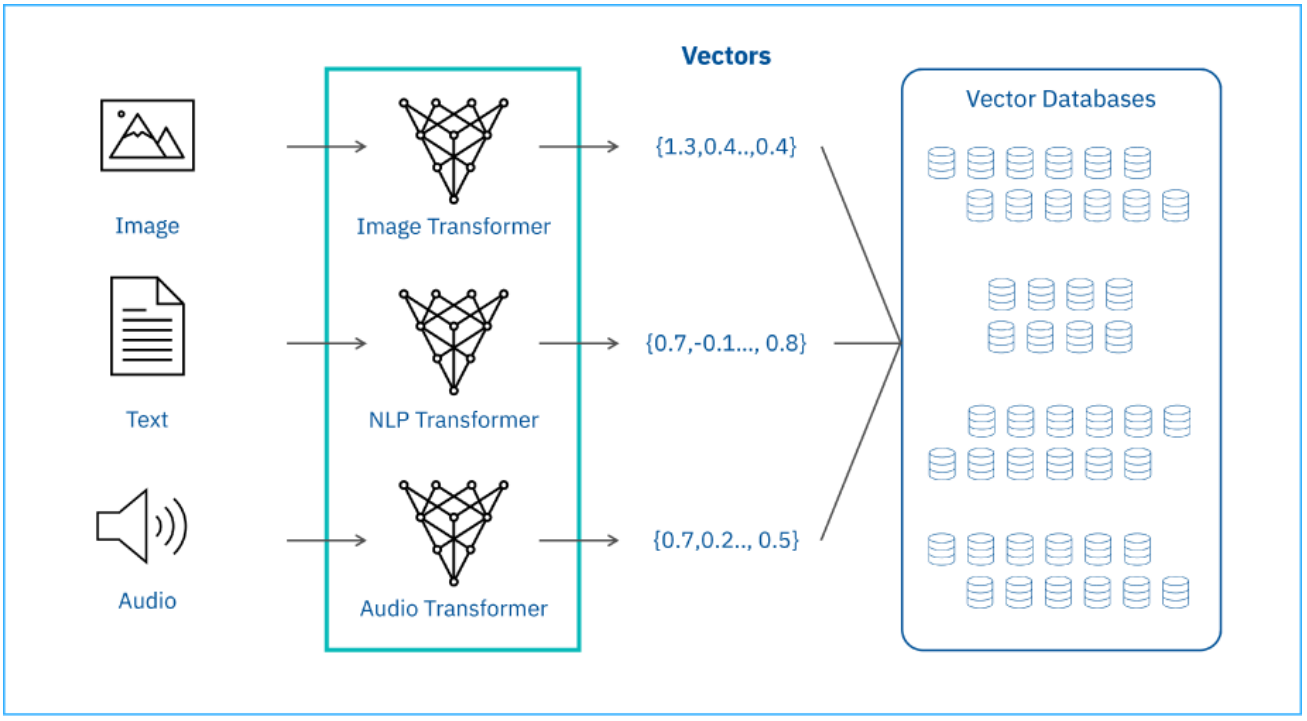


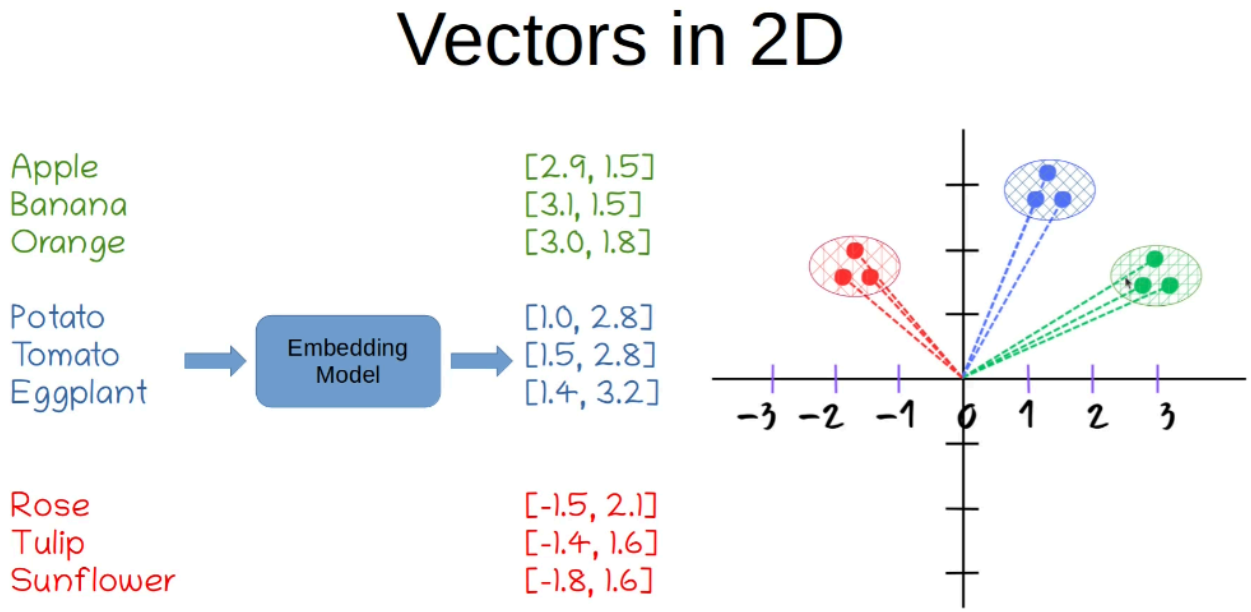
Vector databases data storage

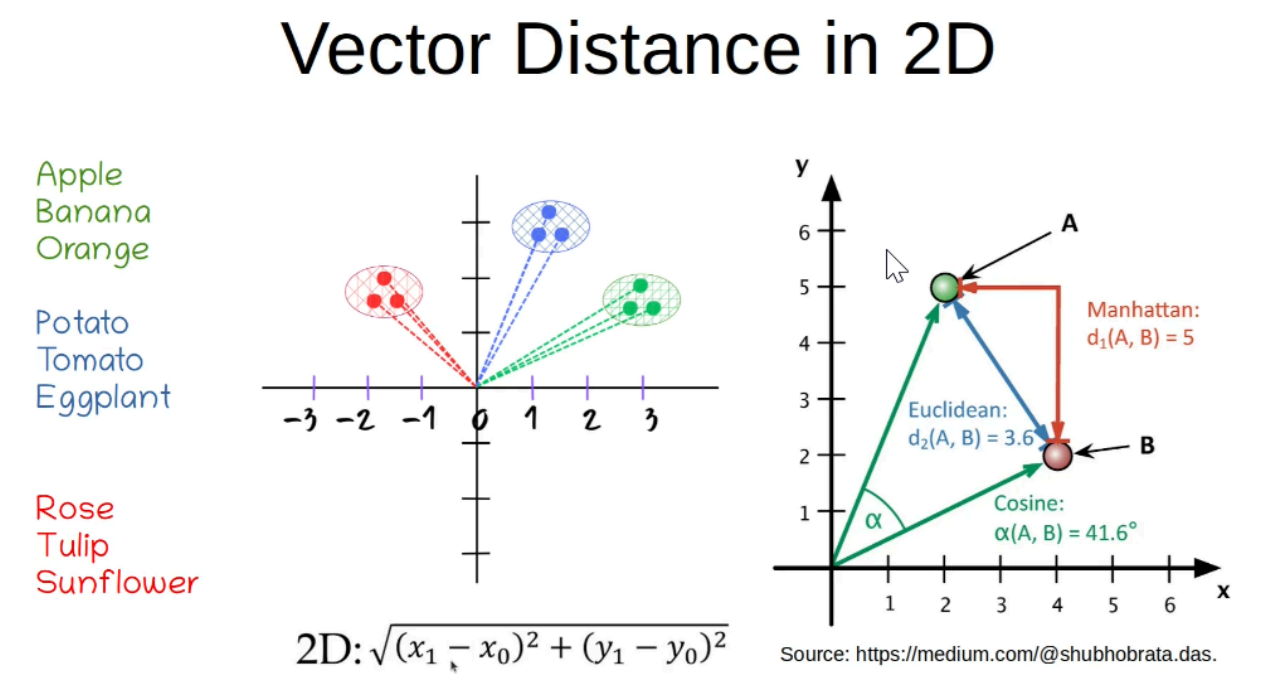
Vector databases store data formatted so that numerical vectors depict each data item. These numbers signify various attributes or features of the object, with each vector dimension corresponding to a specific attribute. For instance, in an image database, each image may be represented as a vector of pixel values, while a text database might represent each piece of text as a vector of word frequencies.

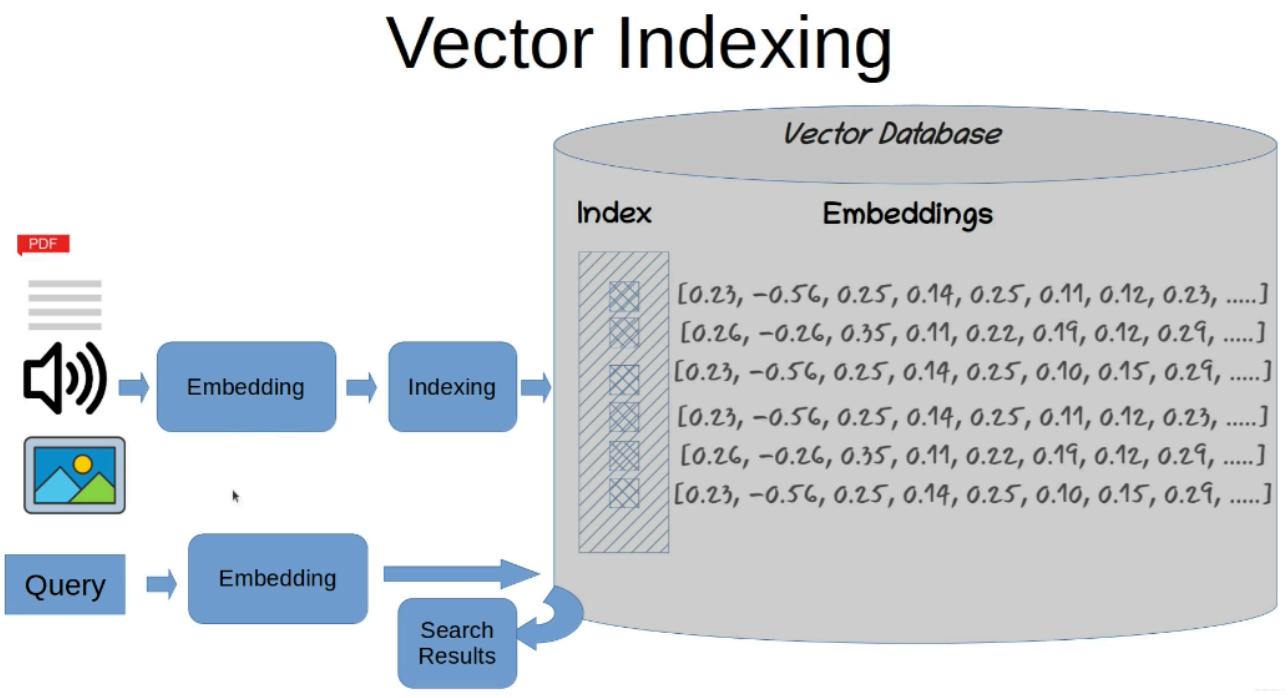
King = `[.8, .7, .9, .6, ...]`

> Vector databases have full CRUD (create, read, update, and delete) capabilities.









Vector Database Use Cases

Vector databases offer powerful solutions for various tasks, especially in AI, machine learning, and data analysis. Below are some of the key use cases:

1. Long-term Memory for Large Language Models (LLMs)

- **Problem:** LLMs often "hallucinate" or generate incorrect information.
- **Solution:** Using vector databases as long-term memory to store source knowledge, helping the model reference accurate data and reduce hallucinations in its responses.

2. Semantic Search & Similarity Search

- **Applications:**
  - **Text**: Searching for semantically similar content within large text corpora.

  - **Images**: Searching for similar images based on visual features.
  - **Audio & Video**: Finding similar audio clips or video content through semantic matching.
- **Benefit:** Enables content-based searching and matching across multiple modalities.

3. Recommendation Systems

- **Use Case:** Suggest products, movies, music, or services based on user preferences and behaviors.
- **Benefit:** Provides personalized recommendations by analyzing user activity and comparing it to a vector space of items.

Personalized Recommendation

Deep Learning based Recommendation Systems uses vector embedding/vector data base:

1. DLRM by Facebook
2. Neural Collaborative Filtering (NCF)
3. Variational Auto-encoder Collaborative Filtering (VAE-CF)
4. Wide and Deep

Source: Nvidia

Deep learning based CF learns the user and item embeddings (latent feature vectors) based on user and item interactions with a neural network.

4. Machine Learning

- **Clustering**: Grouping similar data points based on vector representations.
- **Classification**: Categorizing data by learning from vector representations of features.
- **Benefit:** Efficient storage and retrieval of feature vectors for fast machine learning tasks.

5. Anomaly Detection

- **Use Case:** Identifying outliers or unusual patterns in data.
- **How It Works:** Vector representations help capture normal patterns, making deviations from these patterns easier to detect.

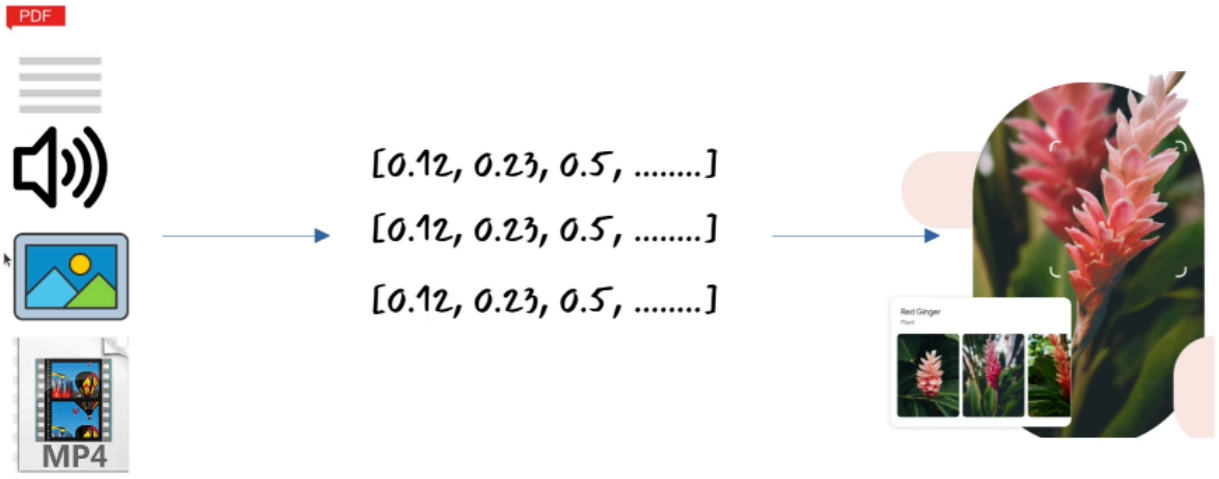
6. Graph Analysis

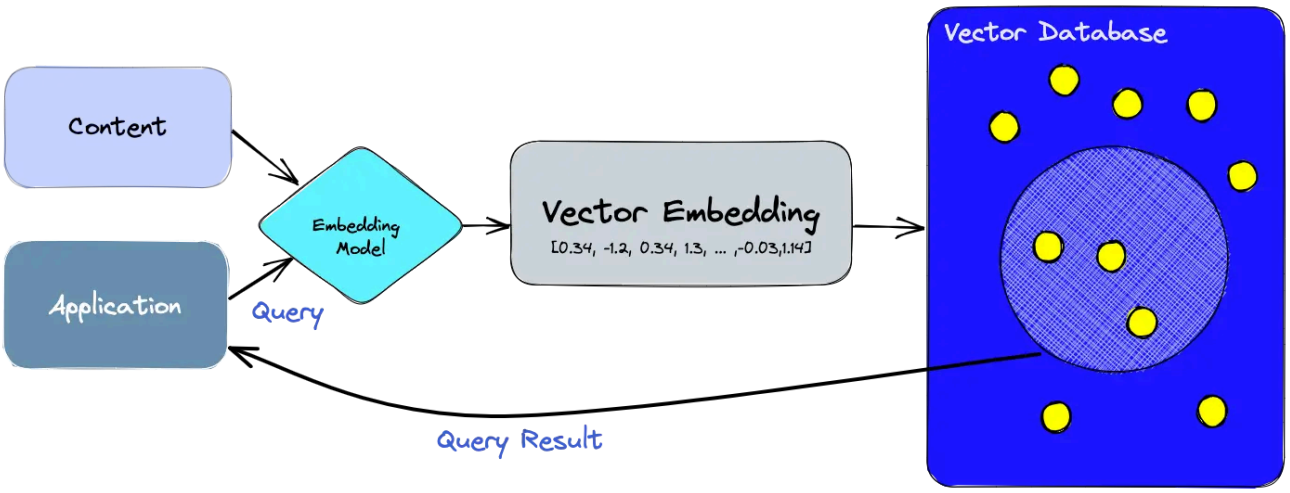
- **Use Case:** Analyze and model relationships and structures in data, such as social networks, transport networks, and knowledge graphs.
- **Benefit:** Use vector representations of nodes and edges to explore complex relationships in large-scale graphs.

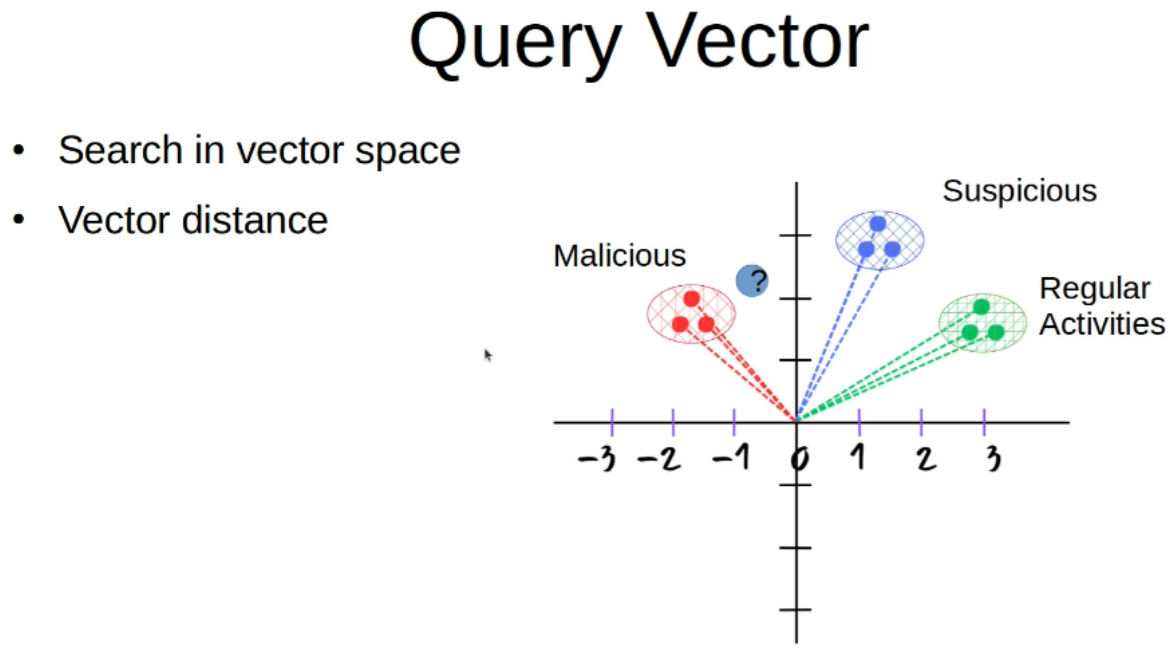
Example of Vector Databases

Why Vector Database?

The Vector Space







Vector databases

A vector database is a specialized database designed to store and query vectorized data rapidly. Unlike conventional databases, which organize data in tables, a vector database represents data as vectors in a multi-dimensional space. These vectors encapsulate essential attributes of the items they represent, making vector databases ideal for tasks requiring similarity searches, nearest neighbor queries, and assessing distances or similarities between vectors.

Vector databases data storage

Vector databases store data formatted so that numerical vectors depict each data item. These numbers signify various attributes or features of the object, with each vector dimension corresponding to a specific attribute.

Relational database organization

A relational database organizes data into tables, utilizing rows and columns. Relational databases use structured query language (SQL) for data querying and manipulation, adhering to the relational model.

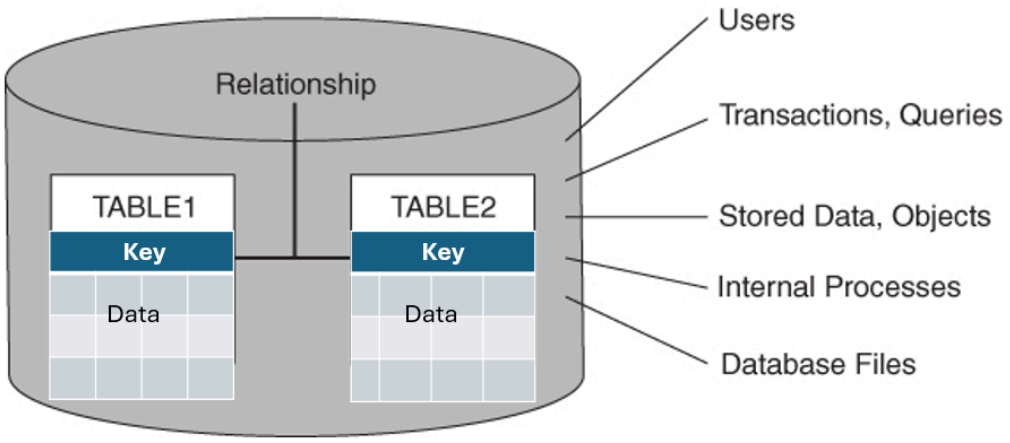
Relational databases excel in managing structured data where relationships between different entities are well-defined.

Relational database data storage

Relational databases store data in tables, each representing a distinct entity or relationship.

Each row in a table corresponds to a record, while each column represents a property or attribute. Tables contain data and get connected, having relationships. with other tables using keys, such as main and foreign keys.

In a relational database, users can peform transactions, queries, and internal processes on stored data or objects (database files) within rows and columns of tables



Vector VS relational databases

Function	Traditional databases	Vector databases
Data Representation	Traditional databases organize data in a structured format using tables, rows, and columns, ideal for relational data.	Vector databases represent data as multi-dimensional vectors, efficiently encoding complex and unstructured data like images, text, and sensor data.
Data Search and Retrieval	SQL queries are suited for traditional databases with structured data.	Vector databases specialize in similarity searches and retrieving vectorized data, facilitating tasks like image retrieval, recommendation systems, and anomaly detection.
Indexing	Traditional databases employ indexing methods like B-trees for efficient data retrieval.	Vector databases use indexing structures like metric trees and hashing suited for high-dimensional spaces, enhancing nearest-neighbor searches and similarity assessments.
Scalability	Scaling traditional databases can be challenging, often requiring resource augmentation or data sharding.	Vector databases are designed for scalability, especially in handling large datasets and similarity searches, using distributed architectures for horizontal scaling.

Function	Traditional databases	Vector databases
Applications	Traditional databases are pivotal in business applications and transactional systems where structured data is processed.	Vector databases shine in analyzing vast datasets, supporting fields like scientific research, natural language processing, and multimedia analysis.

Recap

- A vector database is a specialized database representing data numerically as vectors in a multi-dimensional space.
- While traditional databases are adept at managing structured data and transactional operations, vector databases handle high-dimensional data and perform rapid similarity searches.
- Vector libraries read and update data; however, vector databases can perform create, read, update, and delete (CRUD) functions.
- Vector databases store data formatted so that numerical vectors depict each data item.
- Relational databases organize data into tables, utilizing rows and columns.
- Traditional databases use SQL queries, and vector databases use similarity search to retrieve data.

Potential Bottlenecks and Solutions

Bottleneck 1: Serialization and Deserialization Overhead

Issue: Converting vectors to and from binary format can introduce overhead. **Solution**: Using more efficient serialization libraries like `pyarrow` or `protobuf` can mitigate this.

Bottleneck 2: Query Performance

Issue: The query performance might degrade with a large number of vectors due to the linear search approach. **Solution**: Implementing more sophisticated indexing mechanisms like HNSW (Hierarchical Navigable Small World) or leveraging SQLite extensions such as `sqlite3_vss` can significantly enhance performance, which we will cover in the next blog.

Bottleneck 3: Storage Space

Issue: Storing high-dimensional vectors can consume considerable space. **Solution**: Applying vector quantization techniques can reduce the storage footprint while maintaining search accuracy.