



Bases de Datos Vectoriales

Introduction

1. Introducción a las Bases de Datos Vectoriales

Concepto

- Una **base de datos vectorial** almacena **vectores de alta dimensión** que representan información en forma numérica (*embeddings*).
- Se utilizan principalmente en **búsqueda semántica** y **sistemas RAG**, donde se busca información basada en **significado**, no solo palabras exactas.

💡 Analogía:

Imagina una biblioteca donde los libros están organizados **por significado** en lugar de por título o autor. Así, libros sobre temas relacionados están cerca, aunque no usen las mismas palabras.



Diferencia con bases de datos tradicionales

Característica	Relacional (SQL)	Vectorial
Tipo de datos	Texto, números, fechas	Vectores de alta dimensión
Búsqueda	Coincidencia exacta	Similitud semántica
Ejemplo	“Buscar por título”	“Buscar por significado”

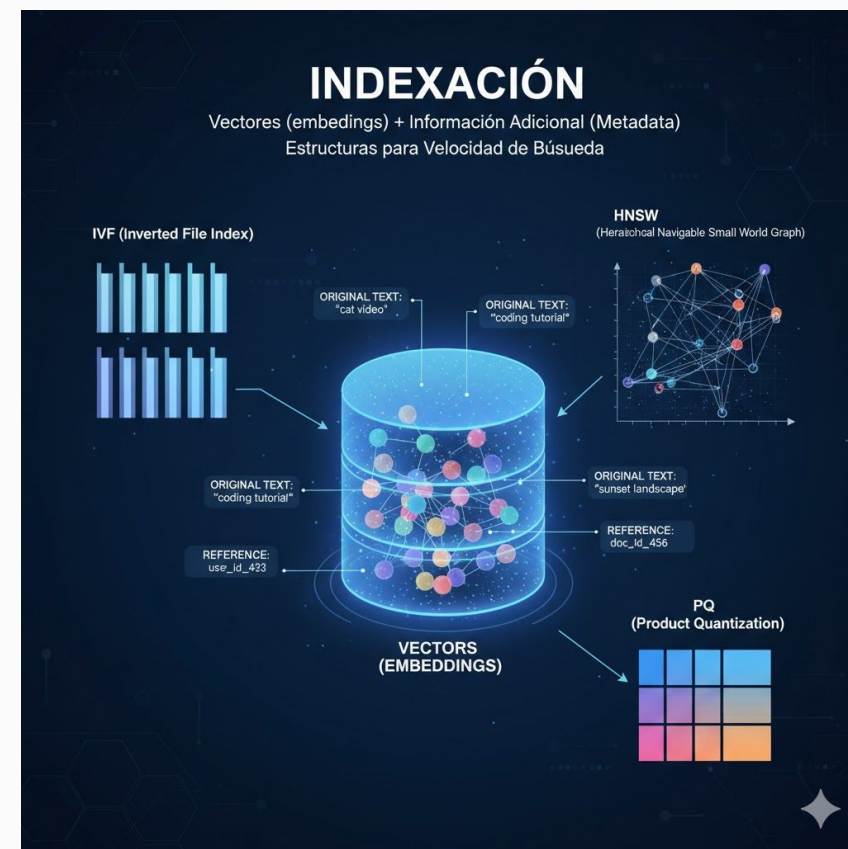
Ejemplos de bases de datos vectoriales

- **Pinecone**: servicio gestionado en la nube, fácil de escalar.
- **ChromaDB**: open source, integración sencilla con Python y LLMs.
- **Milvus, Weaviate, FAISS**: otras opciones populares.

2. Indexación y consulta de vectores

Indexación

- Los vectores (embeddings) se **almacenan en la base** junto con información adicional (*metadata*), como el texto original o referencias.
- Para mejorar la **velocidad de búsqueda**, se usan estructuras de indexación como:
 - **IVF (Inverted File Index)**
 - **HNSW (Hierarchical Navigable Small World Graph)**
 - **PQ (Product Quantization)**

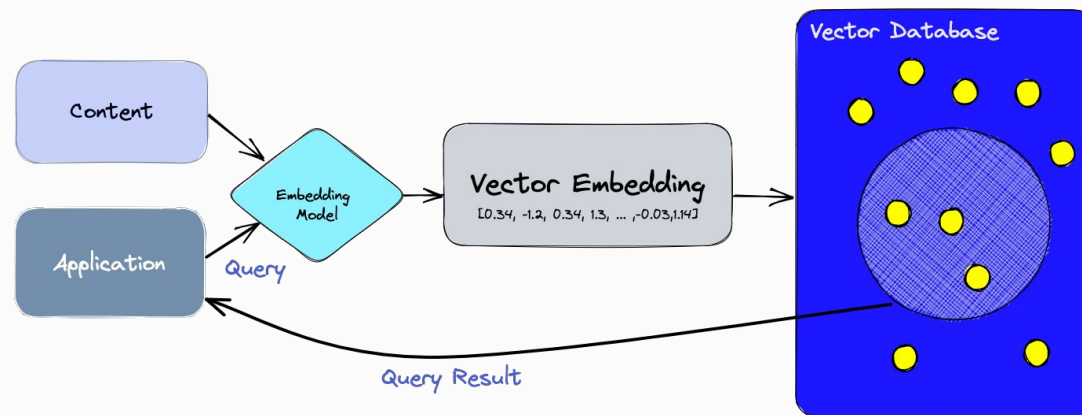


Consulta de vectores

- El proceso consiste en:
 1. Transformar la **consulta del usuario** en un *embedding*.
 2. Buscar en la base los vectores más cercanos usando **distancia de coseno, Euclidiana o Manhattan**.
 3. Recuperar los documentos o información asociada a esos vectores.

Ejemplo práctico:

Si un usuario pregunta “beneficios del ejercicio físico”, el sistema busca vectores cercanos en significado, aunque los documentos no contengan exactamente esas palabras.



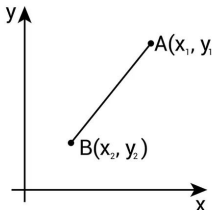
Métricas de similitud

- **Cosine Similarity:** mide la **dirección** de los vectores, ignorando magnitud.
- **Euclidean Distance:** mide la **distancia directa** en el espacio vectorial.
- **Dot Product:** útil cuando los vectores están normalizados.



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Distance Formula

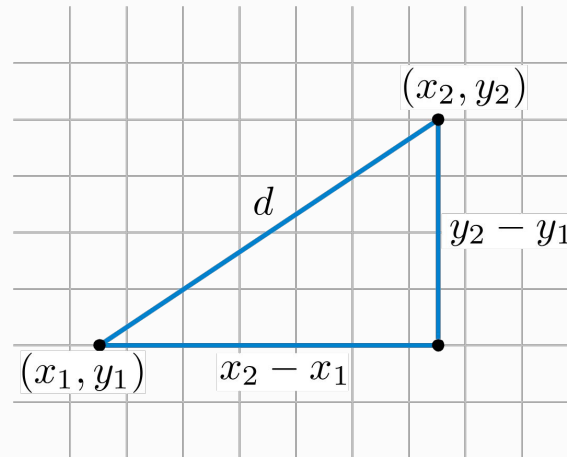
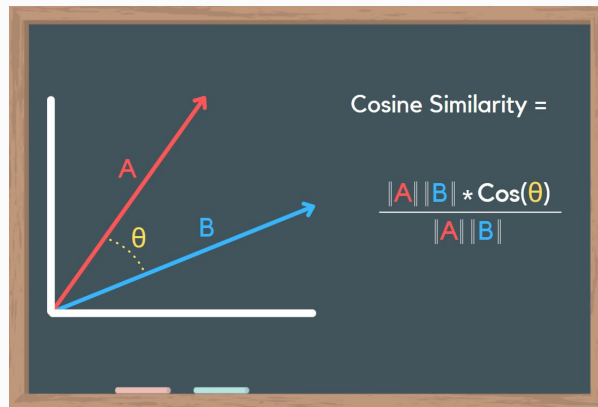


$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\vec{A} \cdot \vec{B} = |A| |B| \cos \theta$$

Or

$$\vec{A} \cdot \vec{B} = A_x B_x + A_y B_y$$



3. Arquitectura interna de una Base de Datos Vectorial

Componentes principales:

- **Storage Engine:** almacena los vectores y metadatos.
- **Index Builder:** crea los índices (HNSW, IVF, PQ).
- **Query Engine:** maneja las búsquedas y calcula las similitudes.
- **Metadata Store:** guarda información textual o etiquetas asociadas a cada vector.
- **API o cliente:** interfaz que permite insertar, buscar y administrar los vectores.



4. Flujo completo de una consulta vectorial (paso a paso)

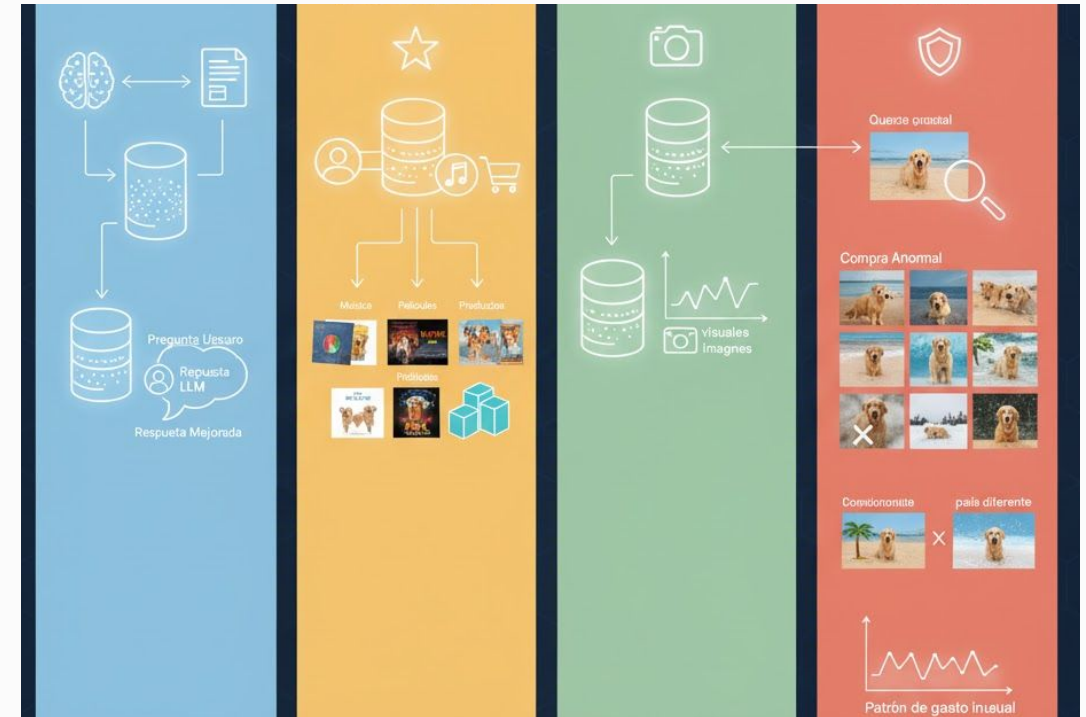
Flujo típico:

1. **Entrada del usuario:** pregunta o texto.
2. **Cálculo del embedding:** convertir la entrada en un vector.
3. **Búsqueda de vecinos más cercanos (k-NN):** encontrar los vectores más similares.
4. **Filtrado por metadatos:** (opcional) limitar resultados por categoría o fuente.
5. **Recuperación del texto original:** devolver el documento asociado.
6. **(En RAG):** el LLM genera una respuesta basada en los resultados.



5. Casos de uso

1. **Sistemas RAG:** usar embeddings de documentos para mejorar respuestas de LLMs.
2. **Recomendaciones personalizadas:** música, películas o productos similares según preferencia.
3. **Búsqueda de imágenes:** encontrar imágenes similares usando embeddings visuales.
4. **Detección de fraude o patrones:** comparar vectores de transacciones o comportamientos.



6. Ventajas de las Bases de Datos Vectoriales

- **Búsqueda semántica rápida y escalable.**
- **Alta dimensionalidad:** puede manejar cientos o miles de dimensiones.
- **Integración con IA:** ideal para LLMs, embeddings de texto, audio o imágenes.
- **Flexibilidad:** permite almacenar metadatos junto con vectores para consultas avanzadas.

