Indexing Techniques for Vector Search

Why do it?

Database indexes help retrieve relevant vectors without scanning everything. They are data structures that help search and retrieve results fast.

Traditional databases have indexes built on B+ Trees. NoSQL databases use Sorted String Tables.

Vector databases use vector indexes (what a surprise 😛).

Indexing techniques

1. IVF – Inverted File Index

Cluster the vector space into regions using K-Means. At query time, find the nearest cluster centroids, then only scan vectors within those clusters.

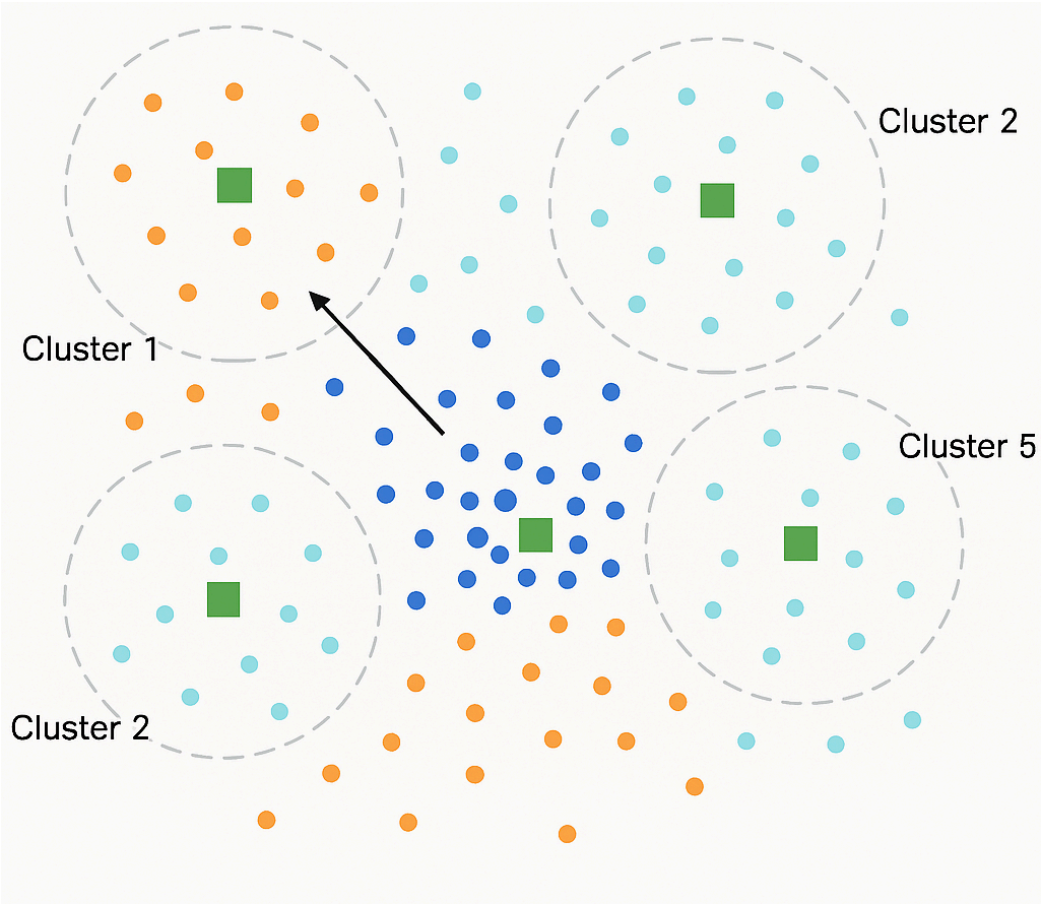

Figure 1: Inverted File Index builds clusters and finds the nearest cluster for a query during search.

Tunable parameters:

- Number of clusters
- Number of clusters to search at runtime

Trade-off:

Lots of clusters → better recall and slower search.
Few clusters → fast, less accurate search.
Very large number of clusters → Slow, brittle recommendations.
Very few clusters → Full table scan.

---

2. HNSW – Hierarchical Navigable Small World Graph

A. Build a graph where nodes are vectors and edges connect to “close” vectors.

B. The graph has multiple levels: The Top levels are sparse, the lower ones are dense.

C. During construction, nodes with a high degree (highly connected nodes) are chosen to be promoted to an upper layer. This is done recursively, till a few nodes are on the top layer.

D. Search is like climbing down a mountain: Start high, zoom into the nearest zones layer by layer.

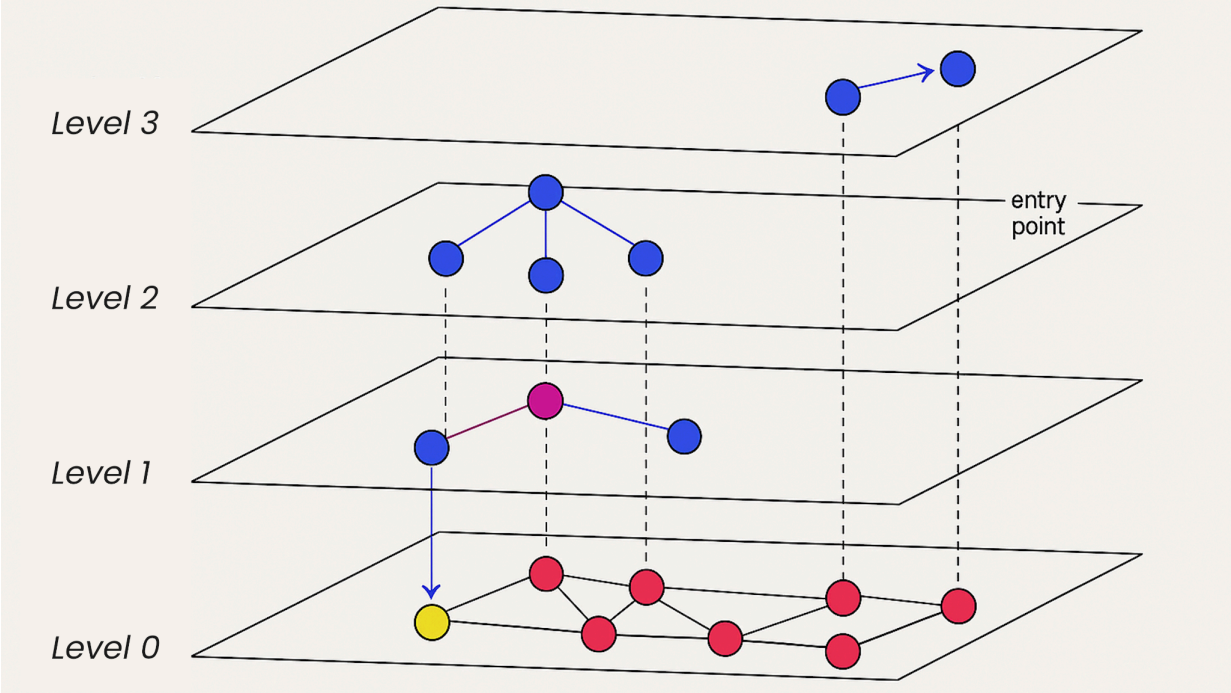

Figure 2: HNSW builds layers and finds the nearest vector for a search query layer by layer.

Performance:

- Very high recall with low latency.
- Outperforms IVF for many CPU-bound use cases.

Comparison

Feature	IVF	HNSW
Speed	Fast	Fast
Accuracy	Needs to be manually configured	High
Memory usage	Low	High
Hardware use	GPU-friendly	CPU-friendly
Construction	Fast	Slow (due to graph building)

Why don’t we use QuadTrees or R-Trees?

Because they work well for 2D or 3D. But in a 100D+ vector space, they suffer from the *curse of dimensionality*. The space becomes too sparse, and partitioning doesn’t help.

Key takeaways

- Indexing makes vector search practical at scale.
- IVF splits the vector space into clusters.
- HNSW builds a graph and uses multi-level traversal.