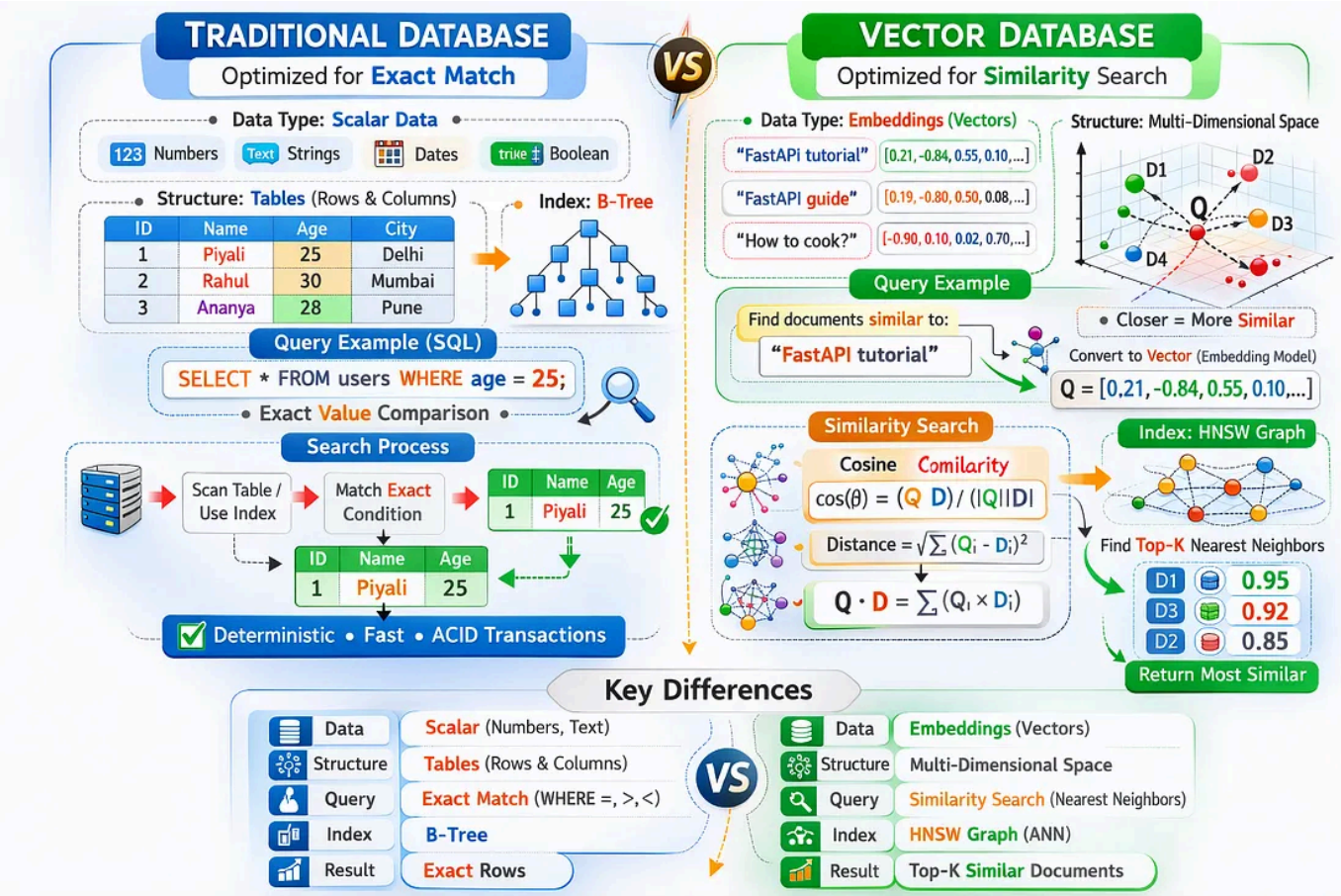
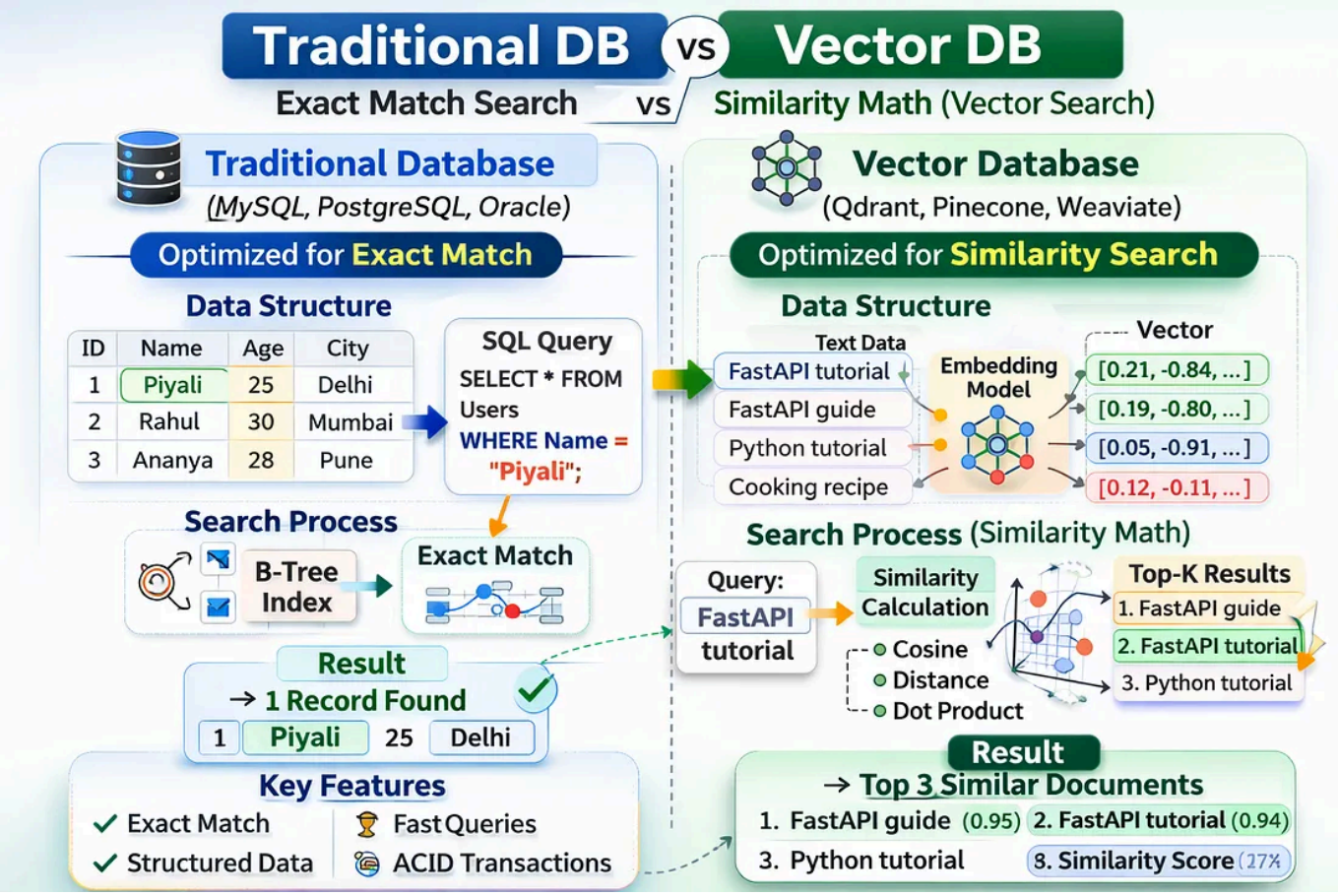
AI : Why Vector Databases use Numbers?

Piyali Das 3 min read · 1 hour ago



traditional_vs_vector_database1



traditional_vs_vector_database2

🔹 Is storing numbers always faster than storing strings?

👉 Short answer: No, not always.

It depends on:

- Data size
- Index type
- Query type
- Database engine

✅ Why numbers are usually faster than strings

1. **Fixed size**

- `INT` = fixed 4 bytes
- `FLOAT` = fixed 4/8 bytes
- `VARCHAR` = variable length

2. **Comparison speed**

- Comparing `25 == 25` (integer) is very fast (CPU-level operation).
- Comparing `"Piyali" == "Piyali"` requires character-by-character comparison.

3. **Index efficiency**

- B-tree indexes work very efficiently on numeric values.

So yes — for **exact comparisons**, numbers are usually faster than strings.

🔹 Numbers are usually faster than strings. Is that why vector databases use numbers?

👉 **No. This is NOT the main reason.**

Vector databases use numeric vectors because:

🎯 They represent meaning, not text.

When you convert text into embeddings using models like:

- OpenAI models

- Ollama
- SentenceTransformers

You don’t convert text to numbers for speed.

You convert text to numbers because:

> Math works only on numbers.

🔹 Why vectors must be numeric

Similarity search uses mathematical operations like:

- Cosine similarity
- Dot product
- Euclidean distance

Example:

```
"FastAPI tutorial"
→ [0.12, -0.98, 0.44, ...]
```

Then database calculates:

```
cosine(query_vector, stored_vector)
```

You cannot compute cosine similarity on raw strings.

That's the real reason.

🔥 Important Clarification

> Vector databases are NOT faster because: "Numbers are faster than strings"
>
> They are faster because: They use specialized indexing algorithms like HNSW (Approximate Nearest Neighbor search)

Example vector DB:

- Qdrant

They use:

- HNSW graph index
- ANN search
- Optimized high-dimensional search structures

Traditional DB = optimized for exact match
Vector DB = optimized for similarity math

Completely different problem.

🔹 Simple Analogy

Traditional DB:

> Find person whose name = “Piyali”

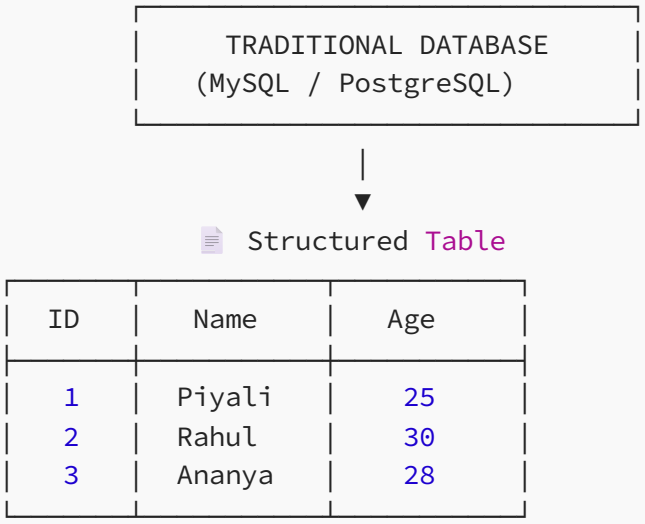
Vector DB:

> Find documents similar in meaning to “How to build FastAPI RAG system”

Different tasks → different data representation.

Traditional DB = Optimized for Exact Match
Vector DB = Optimized for Similarity Math

🟦 Traditional Database (Exact Match Search)




```
SQL Query:
SELECT * FROM users WHERE age = 25;

🔍 Search Type:
✓ Exact value comparison
```

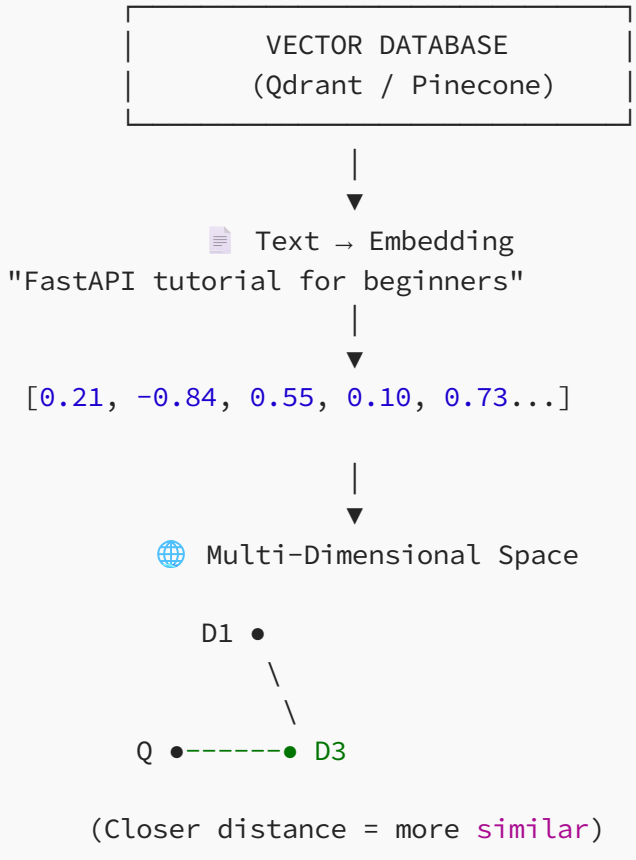
```
✓ B-Tree index
✓ Deterministic result
✓ ACID transactions

Result:
🎯 Exact row returned
```

Color Concept:

- 🔵 Blue = Structured

🟢 Vector Database (Similarity Search)



```
Similarity Math:
Cosine(Q, D)
Euclidean Distance
Dot Product

🔍 Search Type:
✓ Nearest Neighbor Search
✓ HNSW Graph
✓ Approximate search
✓ Returns Top-K similar results

Result:
🎯 Most semantically similar documents
```

Color Concept:

- 🟢 Green = AI / ML
- 🌐 Space = Multi-dimensional geometry
- 📐 Math-based similarity

```
┌──────────────────────┬──────────────────────────┐
│   🔵 Traditional DB  │   🟢 Vector Database     │
├──────────────────────┼──────────────────────────┤
│ Data: Scalar         │ Data: Embeddings         │
│ Structure: Tables    │ Structure: N-D Space     │
│ Query: WHERE =       │ Query: Similarity Math   │
│ Index: B-Tree        │ Index: HNSW Graph        │
│ Result: Exact Match  │ Result: Closest Meaning  │
└──────────────────────┴──────────────────────────┘
```

> *Traditional DB: “Find exact value.”*
> *Vector DB: “Find closest meaning.”*