Vector Search in a Transactional Database

Enabling Mission-Critical AI-Powered Apps.

Edward Pollack
Microsoft Data Platform MVP

Ed Pollack

Microsoft Data Platform MVP

Published author of:

- Dynamic SQL: Applications, Performance, and Security, 2nd Edition
- Analytics Optimization with Columnstore Indexes in SQL Server
  - 2nd edition released this year!
- Expert Performance Indexing in SQL Server, 4th Edition
- Published in Expert T-SQL Functions in SQL Server, 3rd Edition

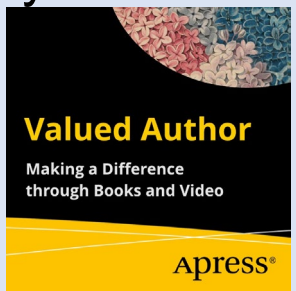
Author on Simple Talk.

Organizes:

- Day of Data Albany
- SQL Saturday New York City
- Capital Area SQL Server User Group

Speaker at many data events

Simple Talk



Day of Data Albany

redgate COMMUNITY AMBASSADOR 2024

MVP Microsoft® Most Valuable Professional



Agenda

- Transactional databases: Definition and optimal uses.
- Brief intro to vectors and vector embedding.
- Vector use in semantic search.
- Demos of vector search in action!
- Conclusion

Transactional Databases

- Intended for **real-time application processing**.
- Enforces ACID guarantees on transactions:
  - Atomic
  - Consistent
  - Isolated
  - Durable
- **Row-oriented**
- Intended for high **concurrency** and many small reads/writes

Transactional Database Usage

Great for:

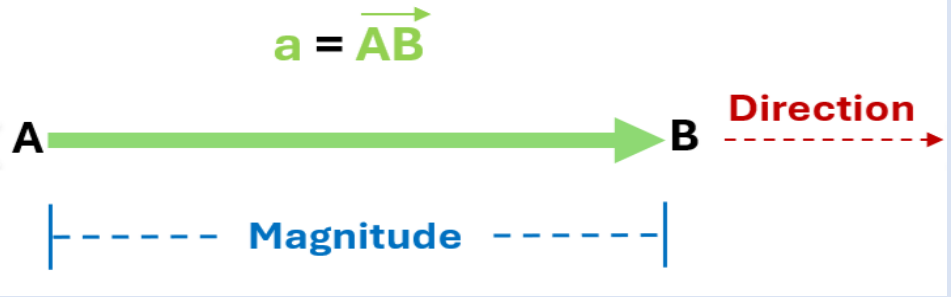
- E-commerce & order processing
- Monetary transactions (banks, financial institutions)
- Healthcare and medical data
- Inventory management

Not so great for:

- Analytics
- Bulk data processing

Vectors

- Mathematical representation of data, most commonly:
  - Spatial
  - Textual
- Math classes typically illustrate 2D and 3D objects with magnitude and direction.



- Vector embedding **dimensions** vary by model.
- **Decimal precision** within each vector varies by configuration.
- Model demonstrated today uses 1,536 dimensions per text.
  - And is VERY cost-effective!

How Text Becomes a Vector Embedding

An embedding model turns text into a list of numbers that captures its meaning

1 Start with text

"The cat sat on the mat"

Any sentence, title, or document

2 Split into tokens

The | cat | sat | on | the | mat

Words (or pieces of words)

3 Embedding model



AI model that captures meaning

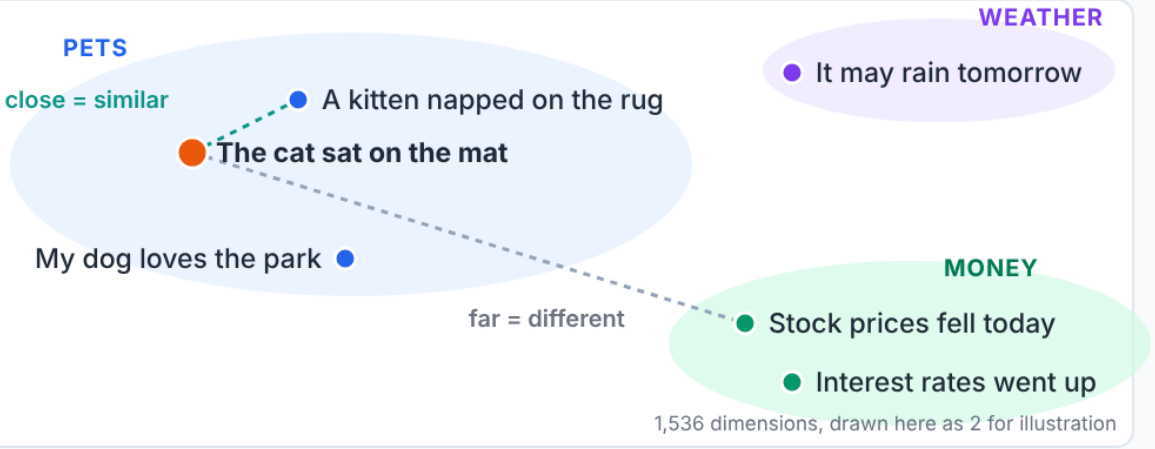
4 Get a vector

```
[0.021, -0.134, 0.087,
…, 0.042]
```



1,536 numbers per text

5 Similar meanings land close together



1,536 dimensions, drawn here as 2 for illustration

Why it matters

- **Search by meaning,** not exact words
- **Closer vectors =** more similar text
- **Works for search, recommendations,** grouping, and finding duplicates

Search: **"pet sleeping on a carpet"**

Top match: **"A kitten napped on the rug"**

No words in common, same meaning

Vector Search: Benefits

- Provides similarity based on meaning.
- Typos are no problem
- Allows ranked searches
- Vector embedding model can be adjusted to improve:
  - Performance
  - Accuracy
  - Relevance to specific domains

Vectors: Why in a Database?

- Can store vector embeddings alongside other related data.
- Native vector search is FAST!
- Native vector indexes ensure optimal performance.
- Ideal for those already using OLTP databases for related purposes.
- Available vector functions provide flexibility:
  - VECTOR_SEARCH
  - VECTOR_DISTANCE
  - VECTOR_NORM
  - VECTORPROPERTY

Demo: Vector Embedding & Semantic Search

Optimizations Based on Applications

- Cache search terms for a given time period.
- Adjust model dynamically based on data volume needed
  - New embedding models and API updates | OpenAI
  - Speed is excellent across models. Data volume dictates true latency.
- Dynamically adjust to a non-AI solution for simplistic searches
- Adjust **decimal precision** based on proximity of search results.
- For large batches of vector embedding, consider **modulus** breakdown of table into multiple parallel threads.
- Dynamic row count returned based on search ***distance***, which also indicates search ***confidence***. Or simply a confidence threshold.

Conclusion

- Transactional databases are optimized for **set-based operations** against application data.
- Vector searching in **bulk** is extremely efficient!
- **Diskann vector indexing** provides exceptional performance against direct searches against native vector data. Open source info:
  - https://github.com/Microsoft/DiskANN
  - Index size is typically 0.1%-0.5% of the data size, despite its complexity.
- Vectors **cannot be faked with strings** or other data types!
- Small models are **cost effective** & **fast**! Start here before scaling up.

Questions? Thank You!

Find me here:

- Ed Pollack | LinkedIn
- Edward Pollack | Most Valuable Professionals
- https://sessionize.com/edward-pollack/

Find my content here:

- EdwardPollack (Ed Pollack) (github.com)
- Edward Pollack, Author at Simple Talk (red-gate.com)
- Ed Pollack, Author at SQL Shack - articles about database auditing, server performance, data recovery, and more