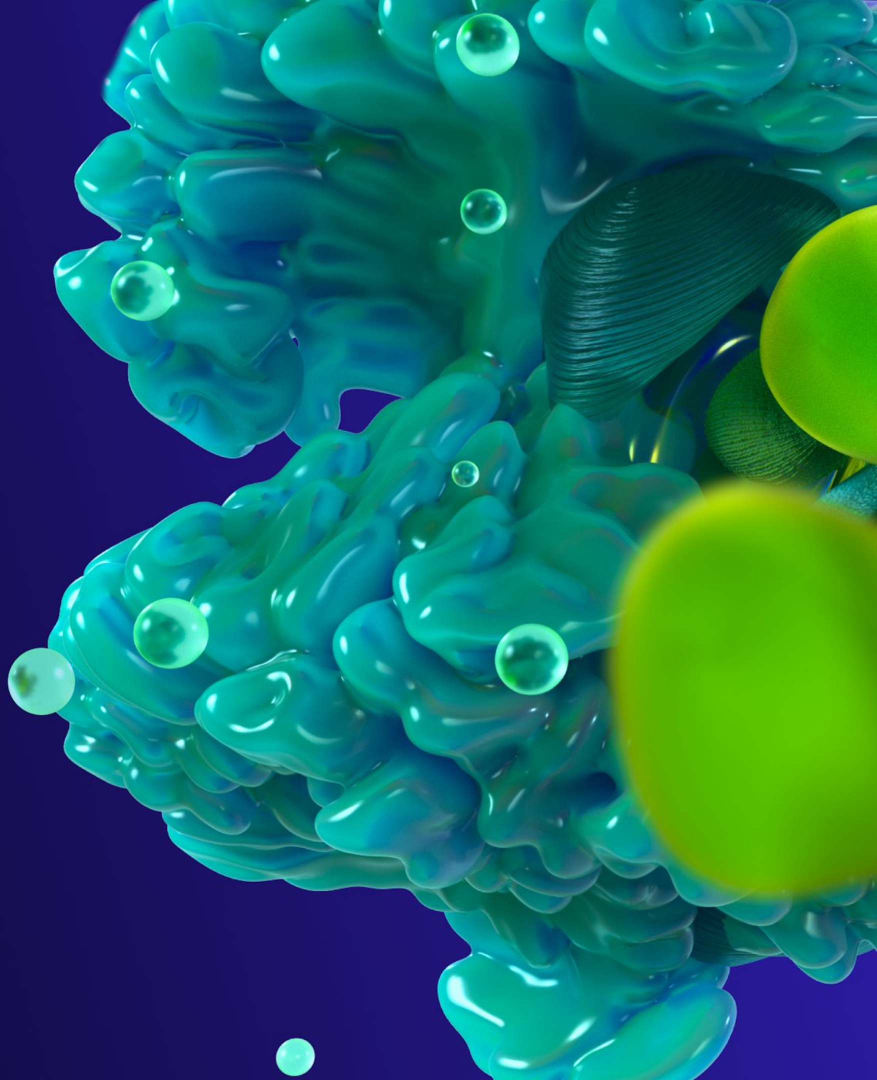
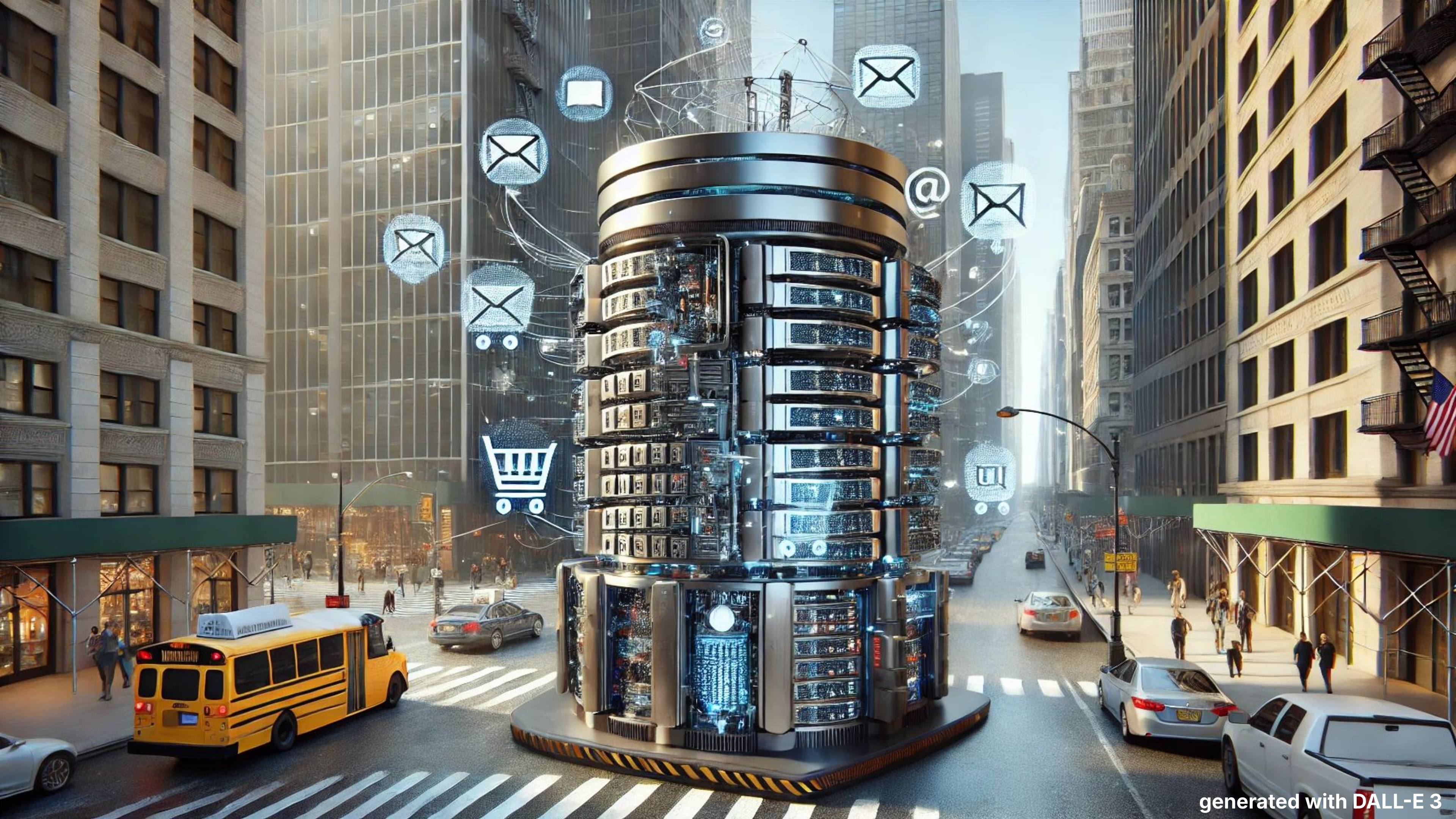
weaviate

A Story of Two Extremes:

Large-Scale Vector Search in E-Commerce and Email RAG

Etienne Dilocker
Co-Founder & CTO




generated with DALL-E 3

How can we serve two very different billion-scale use cases with the same vector db?

e-commerce ⇔ email RAG

single-tenant ⇔ multi-tenant

billions of vectors per dataset ⇔ millions of datasets

in-memory indexes ⇔ disk indexes

query latency ⇔ cost per tenant

high-frequency updates ⇔ writes across tenants

E-commerce Search

How does the user interact with the app?

A **prominently placed search bar** starts the user journey.

A search can be **pure text or faceted** (e.g. within price ranges, specific attributes, etc.)

The user **finds the most relevant products** for their (semantic) query.

generated with DALL-E 3

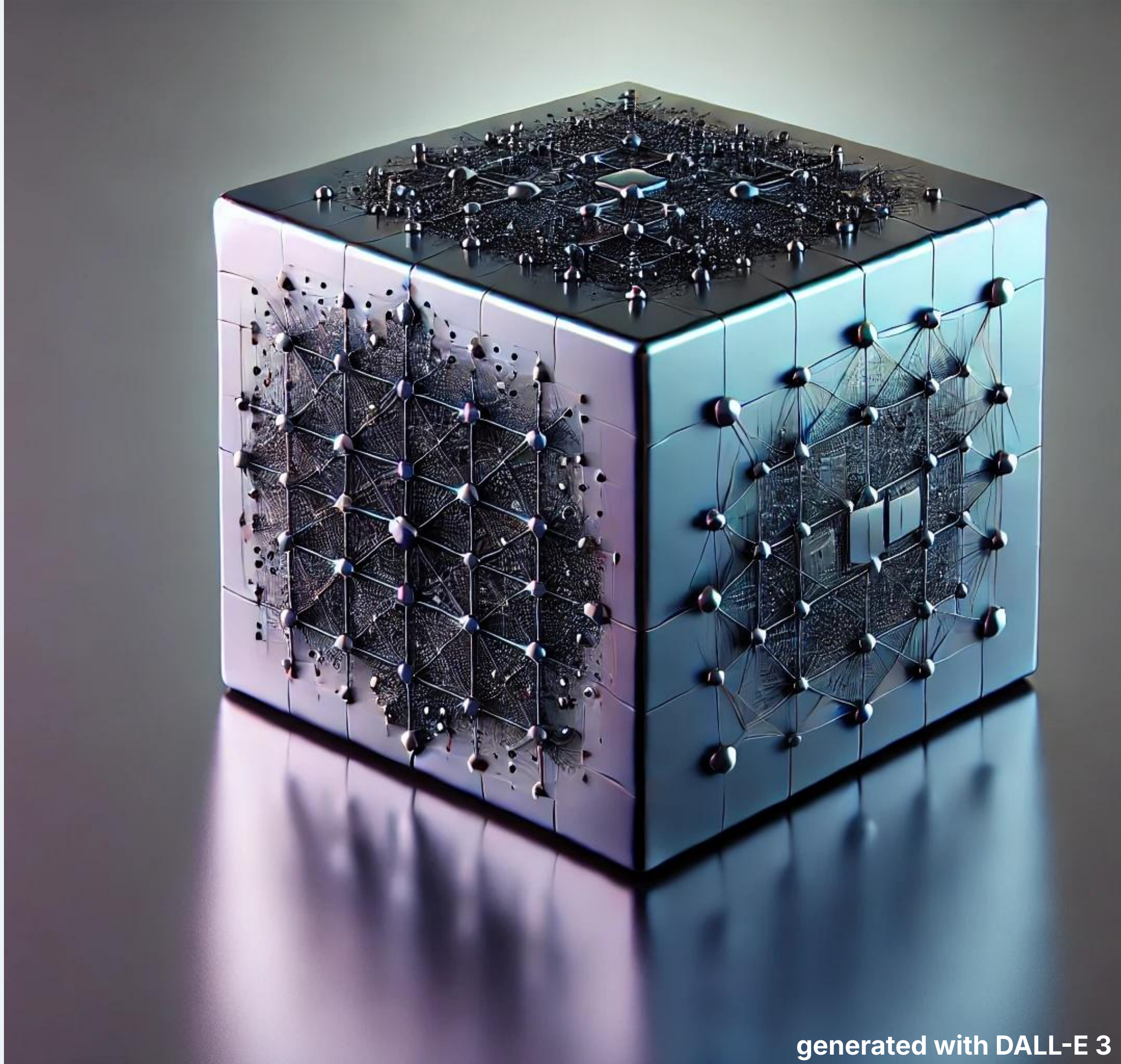
What does the overall dataset look like?

There are **no or few natural partitions** in the dataset.

An incoming query is likely to hit an **unpredictable subsection of the data**.

Examples:

- “colorful summer dress”
- “professional video camera”



How does data flow?

Product changes are **streamed in real-time** from various external sources producing **millions of updates every day**.

Users query a **single, unified dataset** agnostic of where the data originally came from.

High amount of **concurrent imports and queries** – on a single dataset.

What is the scale? What are our targets?

Number of Objects/Vectors

1-10 billion

Updates per day

10s of millions

Desired query latency

p50=200ms

p99=500ms

Tenants / dataset partitions

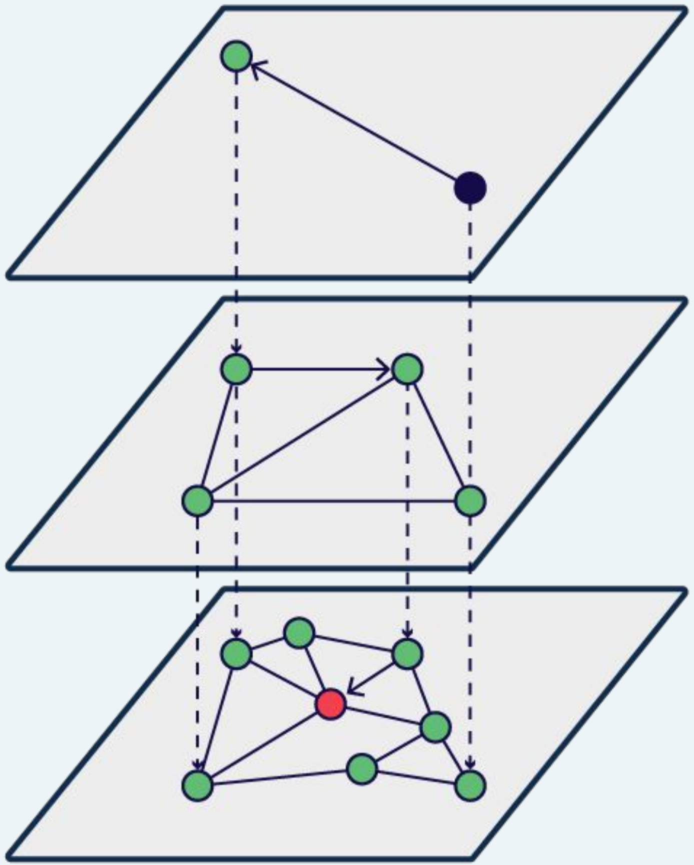
1 or few

How do we fit the right tech?

HNSW is a graph-based **approximate nearest neighbor** (ANN) vector index.

It is optimized for **very low latency and high throughput**.

It is somewhat costly to build and requires all **vectors in memory** to serve queries.



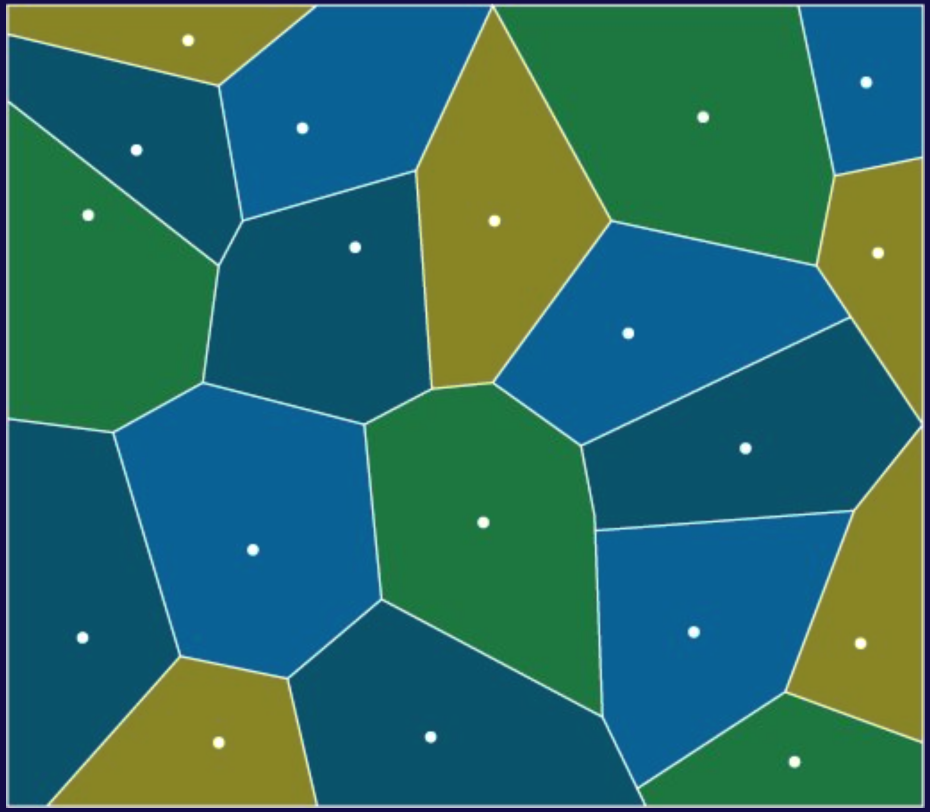
What does it mean to hold 10B vectors in memory?

Example: text-embedding-3-small with 1536d

```
1e10 * 1536 * 4Byte = 55TiB
```

Product Quantization is a compression technique that can **reduce the memory footprint** of common vector embeddings 8-fold.

While using **lossful compression**, accuracy can be restored through **disk-based rescoring** of candidate vectors.

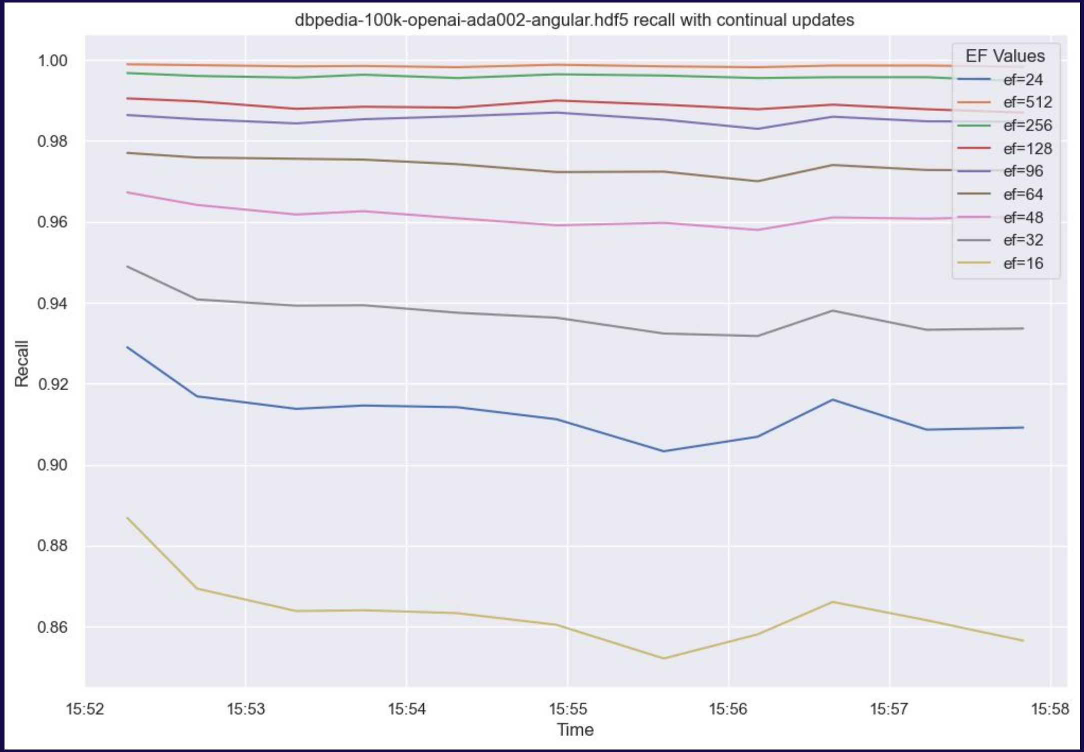


Can HNSW handle so many updates and deletes?

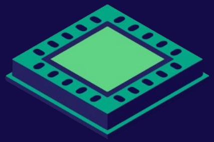
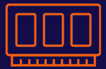
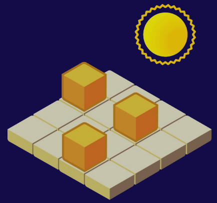
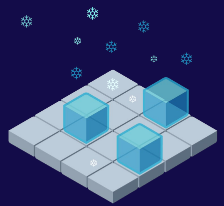
Will it stay “fresh”?

Weaviate's HNSW implementation makes use of **an in-place batch repair job**.

As a result, it can tackle **millions of updates and deletes per hour** without sacrificing result quality.



Tech Summary

 vCPUs	●●●●○	Constant updates / Freshness High concurrent query throughput Low-latency billion-scale search
 Hot  Memory	●●●○○	In-memory vector index (HNSW) Memory footprint reduced by compression (PQ)
 Warm  SSD Drives	●●○○○	Store and serve billions of objects Auxiliary disk-based indexes (filtering, ranges, fuzzy matching)
 Cold  Cloud Storage	○○○○○	All caches always “hot”, no need to offload to cloud storage.

Email RAG

How does the user interact with the app?

generated with DALL-E 3

Email RAG is a classical **AI-assistant or "Ask AI"** style of feature.

The user pulls up the AI assistant on demand to **gain insights into a personal dataset** (in this case their email mailbox).

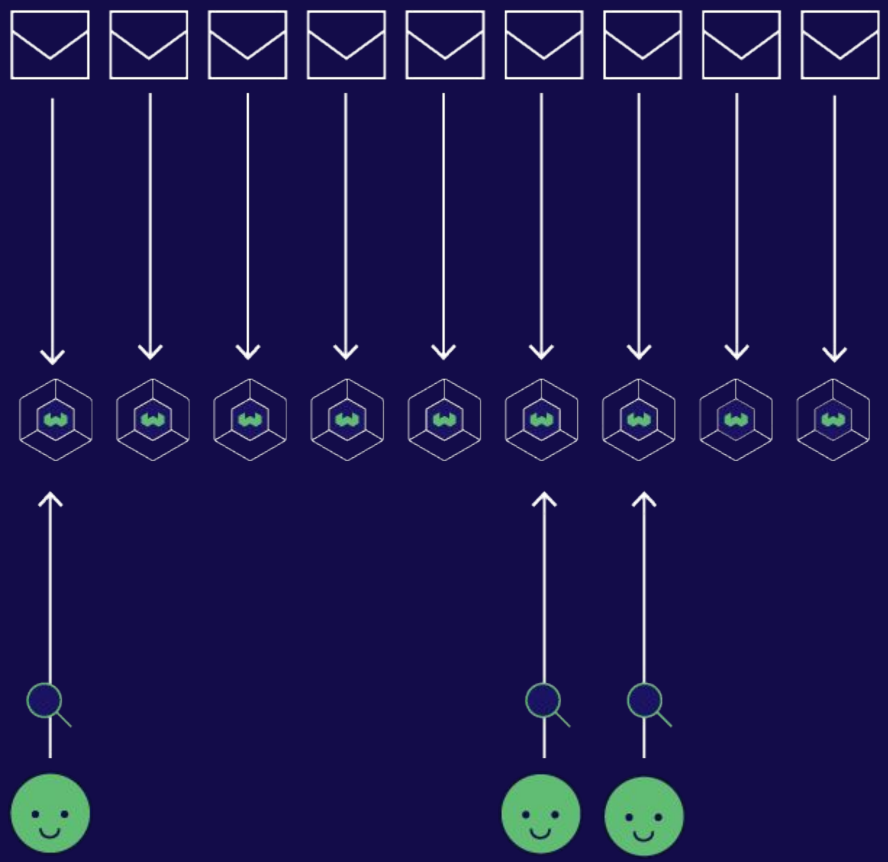
What does the overall dataset look like?

The **dataset has many natural partitions**. A typical query can be narrowed down to **exactly one** such partition.

Example:

Jane searches through her emails with “when does my flight leave from SFO?”

How does data flow?



Nearly every mailbox receives **at least one email per day**.

The total number of **emails per mailbox is low**, but the **overall volume is massive**.

User **behavior is sporadic**. Follows business hours, many users have days where they don't query at all.

What is the scale?
What are our targets?

Number of Objects/Vectors

1-10 billion

Inserts per day

low millions

Desired query latency

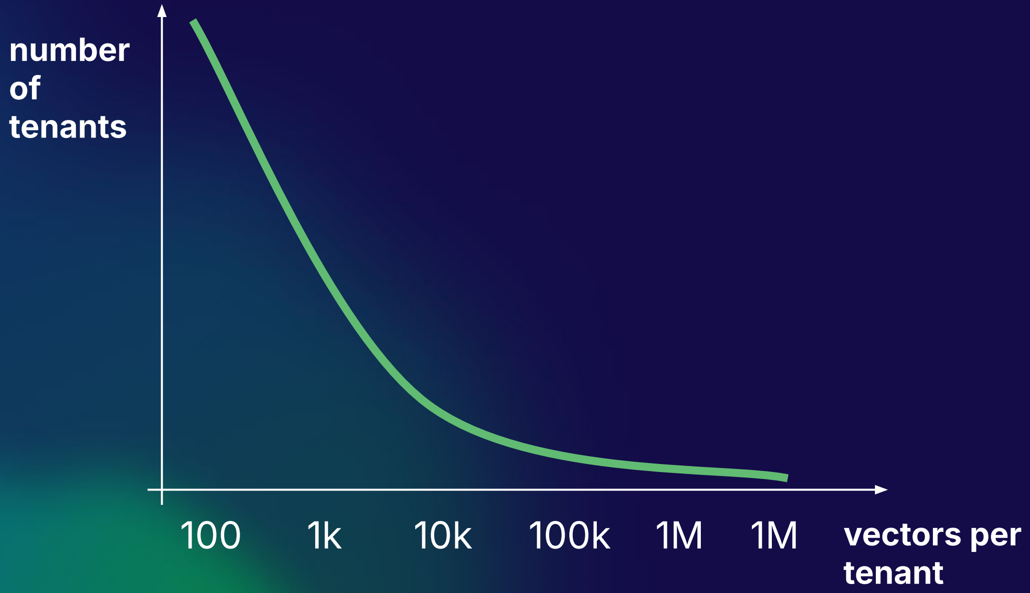
p50=200ms

p99=1000ms

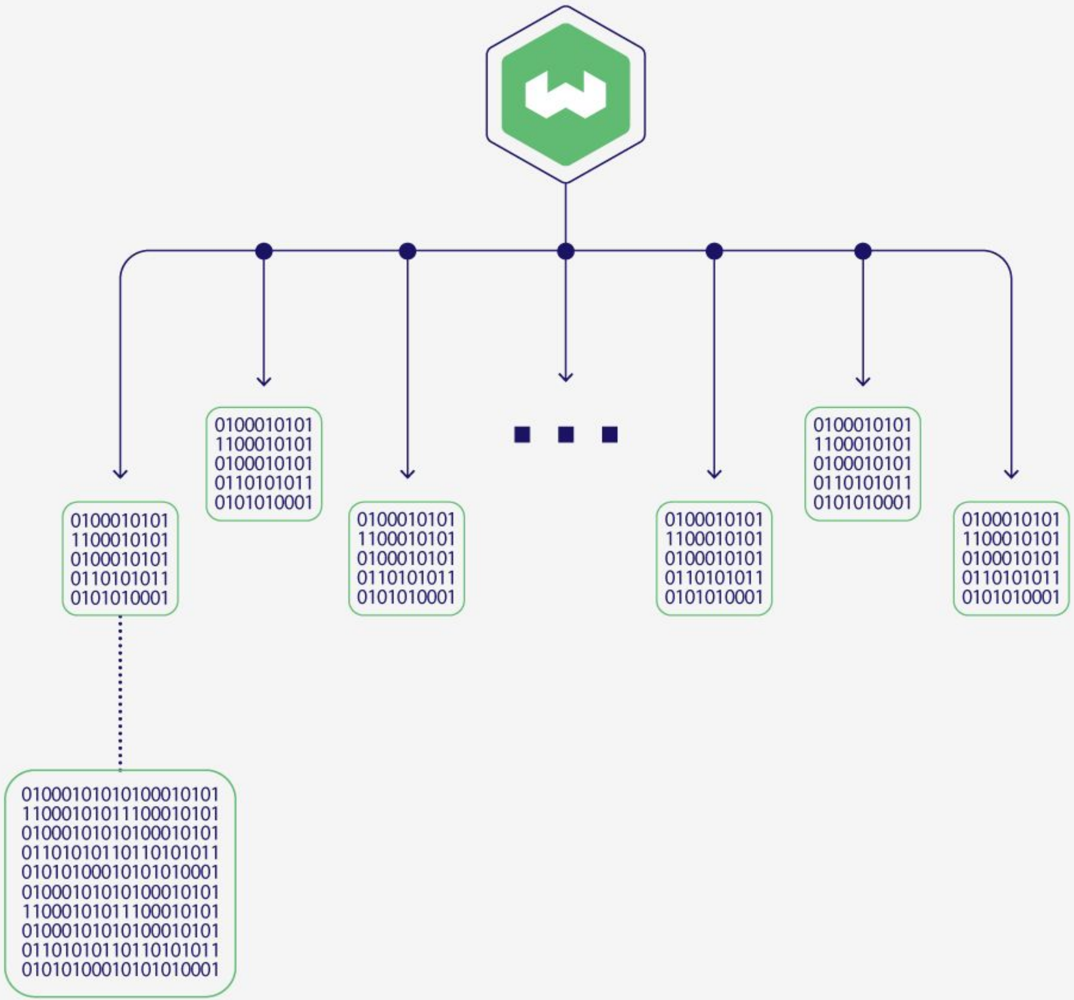
Tenants / dataset partitions

100s of thousands

Tenant Size Distribution



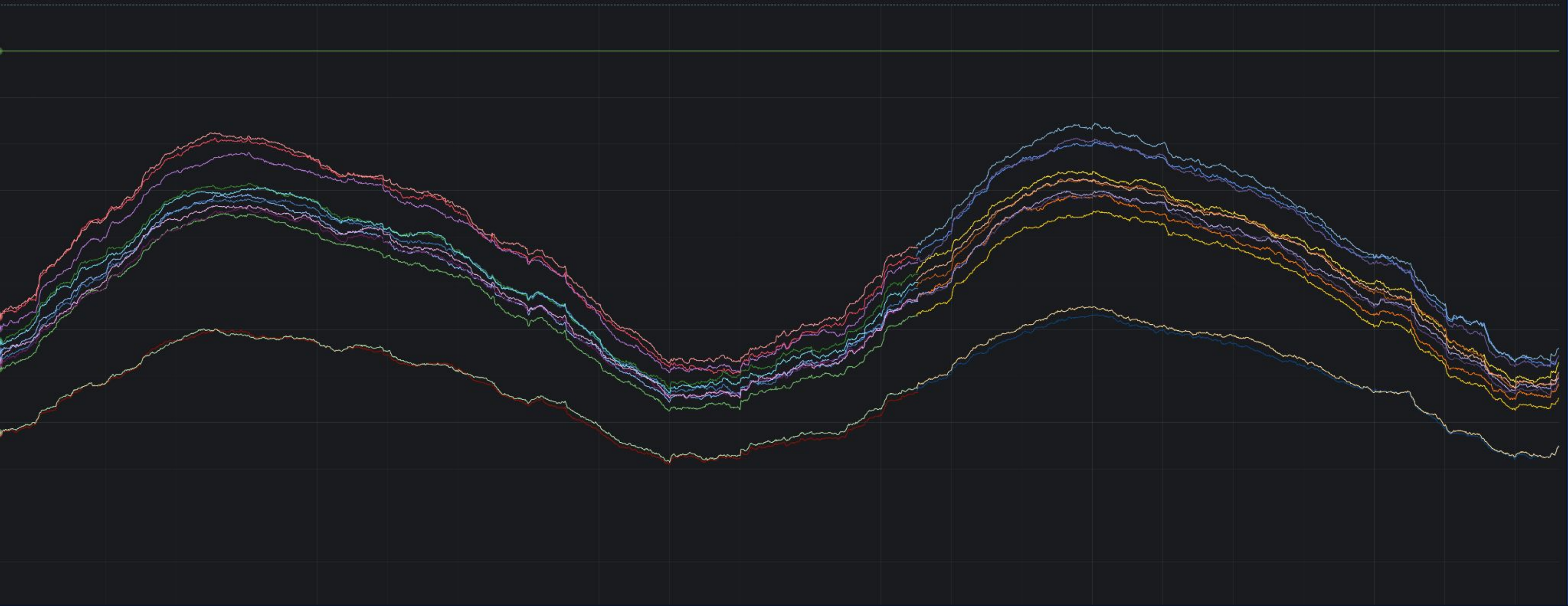

```
p50 = 50k
p90 = 100k
p95 = 300k
p99 = 1M
max = 3M
```



The **flat index** is a disk-based kNN index that can be combined with compression techniques such as **Binary Quantization** to turn it into an aNN index.

It excels with relatively small datasets, such as 1–5M objects.

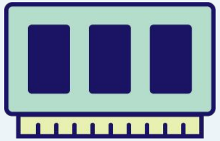
Tenant (mailbox) activity over 48h



Storage Tiers and their cost

$$$ fast

$ slow



Memory (HOT)
very fast, but very expensive

SSD Drives (WARM)
medium speed, medium cost

Cloud Storage (COLD)
cheap, but slow

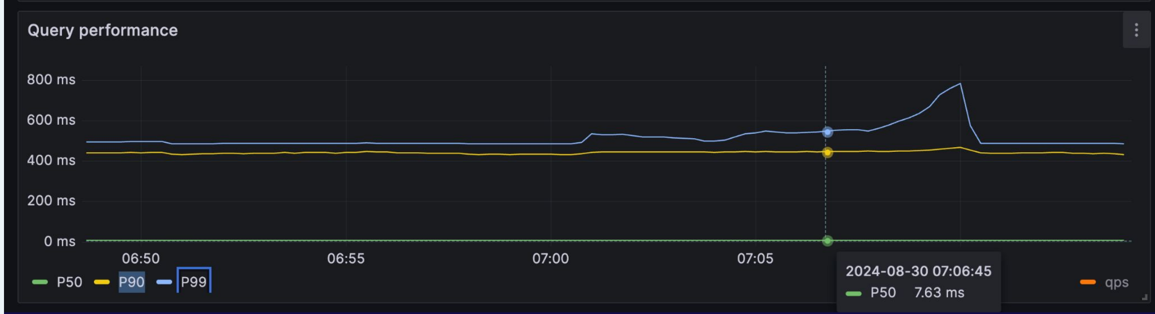
Tech Summary

vCPUs	●○○○○	Individual indexes are small (~1M per tenant) Importing/update is cheap (no graph updates required)
Hot Memory	●●○○○	Memory primarily acts as a cache to improve latency. Sweet spot at fairly low numbers.
Warm SSD Drives	●●●●○	SSDs are the main workhorses here. Best balance between import and query load.
Cold Cloud Storage	●●○○○	Inactive or infrequently queried tenants reside in cold (cloud) storage.

Where do the cases overlap?

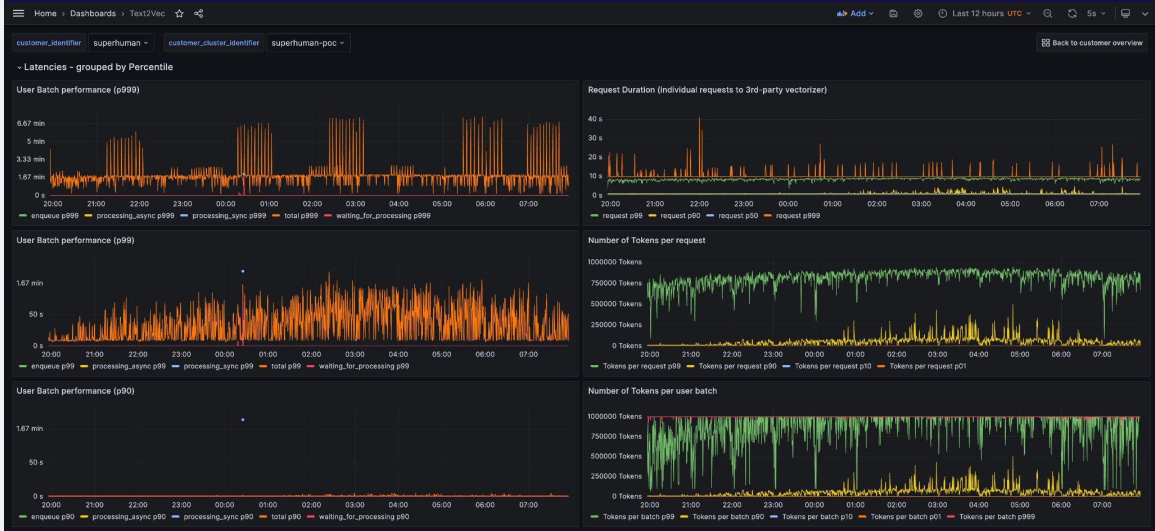
The overall amount of objects and vectors in both cases is somewhat similar (**1-10bn range**), yet the composition is very different.

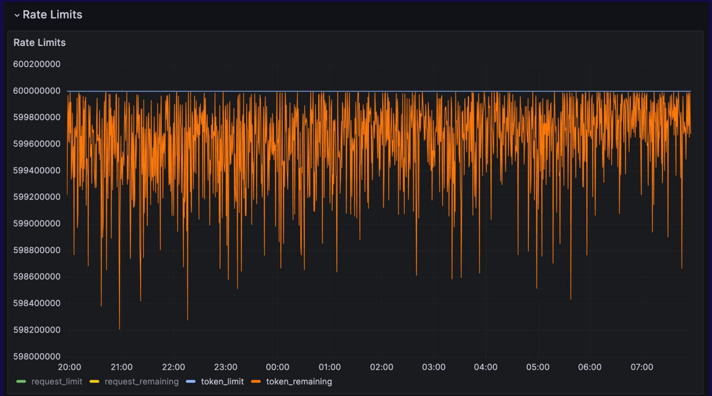
Both cases require **sub-second real-time latency** to serve millions of users.



Weaviate makes it very easy to integrate with other vendors in the embedding ecosystem.

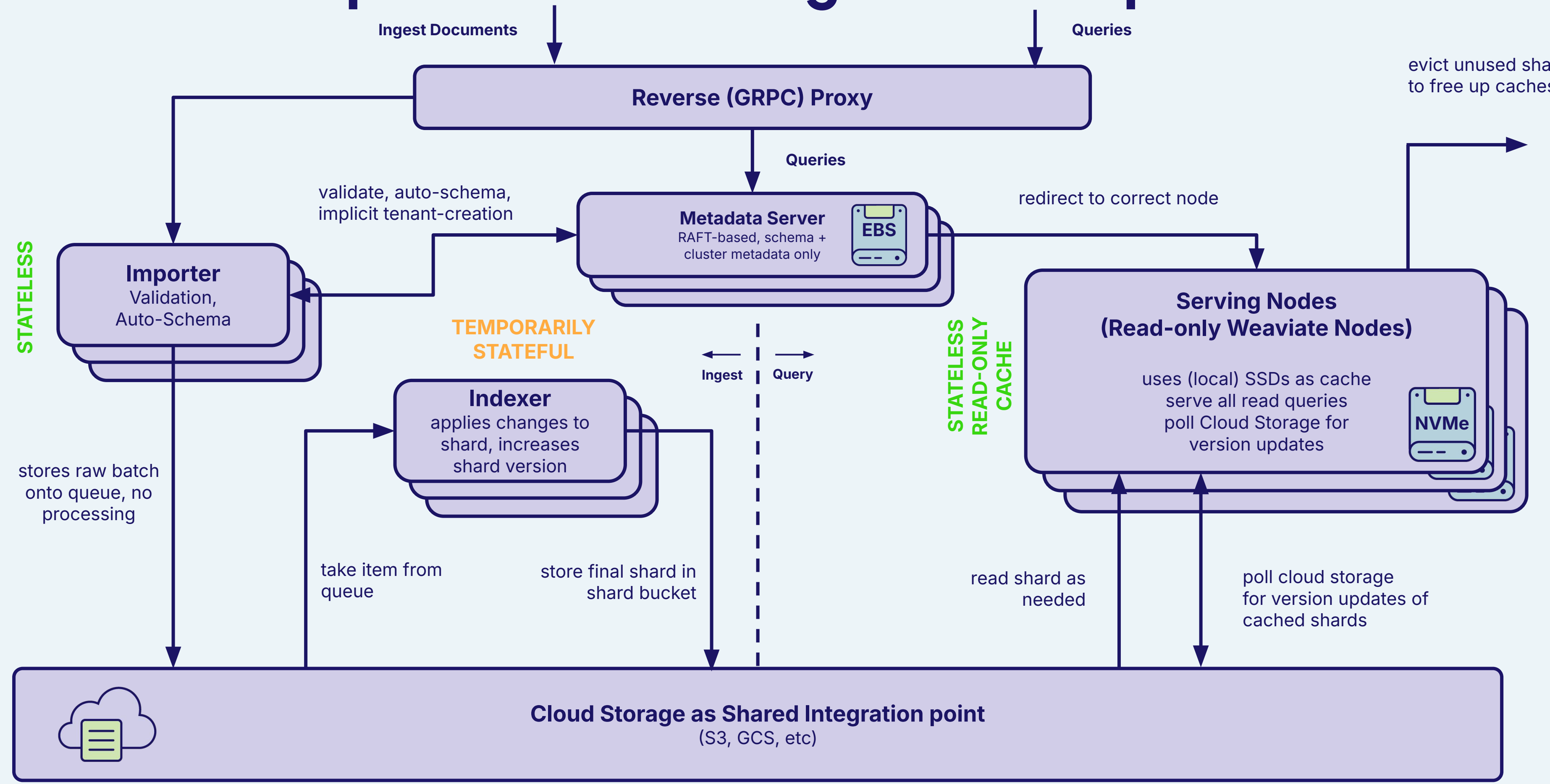
For example, turning **text to vectors** using 3rd-party vectorizers like **OpenAI, Cohere, Ollama**, etc works out of the box – optimized to max out your specific rate limit and provide **great observability**.





What's next?

Full separation of storage and compute



Recap

We’ve looked at two billion-scale use cases that both require real-time latencies.

The e-commerce case has a single dataset, requiring a high-throughput, low-latency in-memory index. We used compression to reduce the memory footprint considerably.

The email RAG case had 100s of thousands of partitions each with relatively small (millions) of objects. This allowed us to use separate indexes that are all disk-based.

How can you use these learnings for your case?

The first question to ask is:

What are the natural partitions in your dataset?

Connect with us!

weaviate.io

weaviate/weaviate

@etiennedi

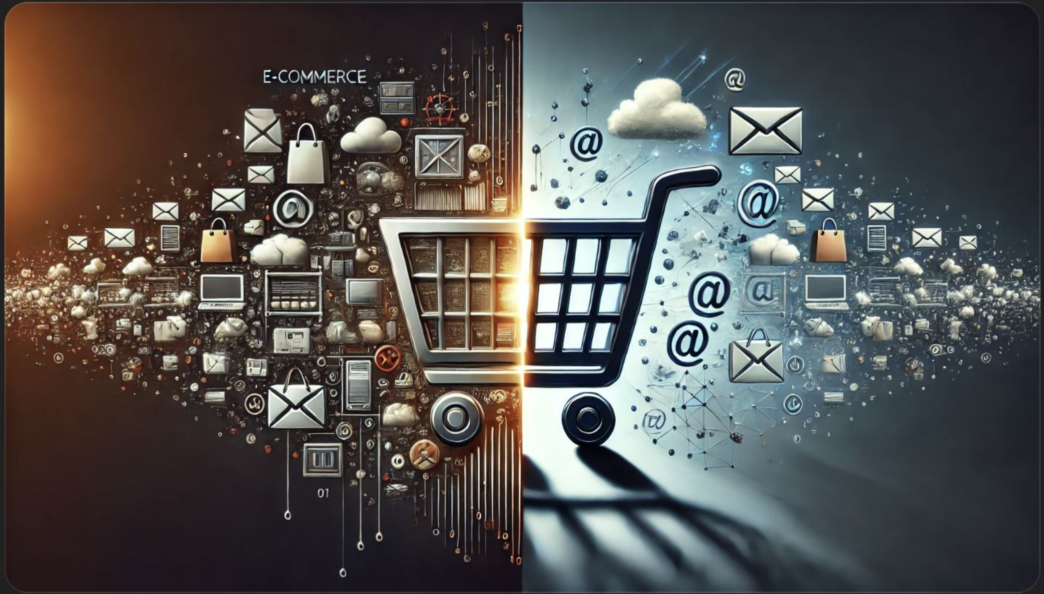
@weaviate_io

All illustrations and visualizations that were not created by our awesome design team (thank you!) were generated with DALL-E 3.

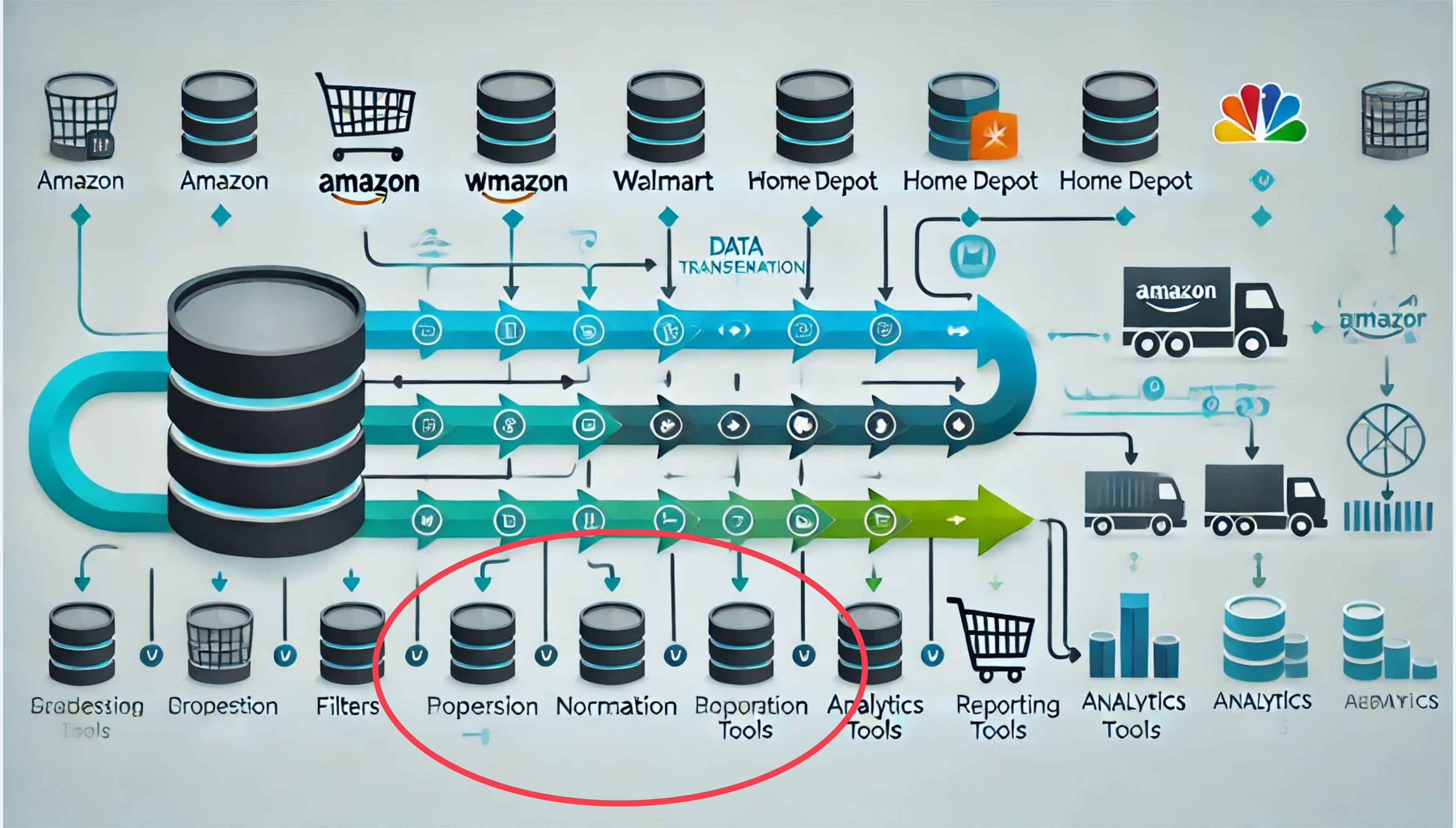
GenAI blooper reel

Here is the revised mockup with the words "E-commerce Search" mentioned only once in a central place above the search bar. The design language is closer to the original style, with the text minimized and the overall layout cleaned up. If you have any more requests or further adjustments, feel free to let me know!

It is better, however, there is still a large shopping cart in the center. This should not be a shopping cart, but rather a database or a generic machine



The shopping cart in the center has been replaced with a neutral, abstract, and futuristic machine symbolizing a database. This new visual maintains the stark contrast between "e-commerce" and "email RAG" while highlighting the unifying power of a vector database. I hope this version better fits your vision for the title slide!


Amazon
Amazon
amazon
wmazon
Walmart
Home Depot
Home Depot
Home Depot
DATA
TRANSENATION
amazon
Filters
Popersion
Normation
Boporation
Tools
Analytics
Tools
Reporting
Tools
ANALYTICS
Tools
ANALYTICS

Connect with us!

weaviate.io

weaviate/weaviate

@etiennedi

@weaviate_io

All illustrations and visualizations that were not created by our awesome design team (thank you!) were generated with DALL-E 3.