Weaviate

Nov, 2024

Is your database designed for AI?

JP Hwang
Developer Educator


VECTOR DATABASES
SO HOT RIGHT NOW
imgflip.com

Vector DB Comparison

by Superlinked | Last Updated : Today

Search

Get insights

Give us a star

Vendor	About				Search			
	OSS	License	Dev Lang	VSS Launch	Filters	Hybrid Search	Facets	Geo Search
Weaviate	✅	BSD	go	2019	✅	✅	✅	✅
Vespa	✅	Apache-2.0	java c++	2017	✅	✅	✅	✅
Vectara	❌	Proprietary	-	2021	✅	✅	-	-
Vald	✅	Apache-2.0	go	2021	-	-	-	-
USearch	✅	Apa...	c++	2023	🟨	-	❌	✅
Typesense	✅	GPL-3.0	c++	2023	✅	✅	✅	✅
txtai	✅	Apache-2.0	python	2020	✅	✅	-	-
Turbopuffer	❌	Proprietary	rust	2023	✅	❌	❌	❌
Rockset	❌	Pro...	c++	2023	✅	✅	✅	✅
Redis Sea...	🟨	(i) Redis So...	c	2021				
Qdrant	✅	Apache-2.0	rust	2021				

https://superlinked.com/vector-db-comparison

Vector DB Comparison

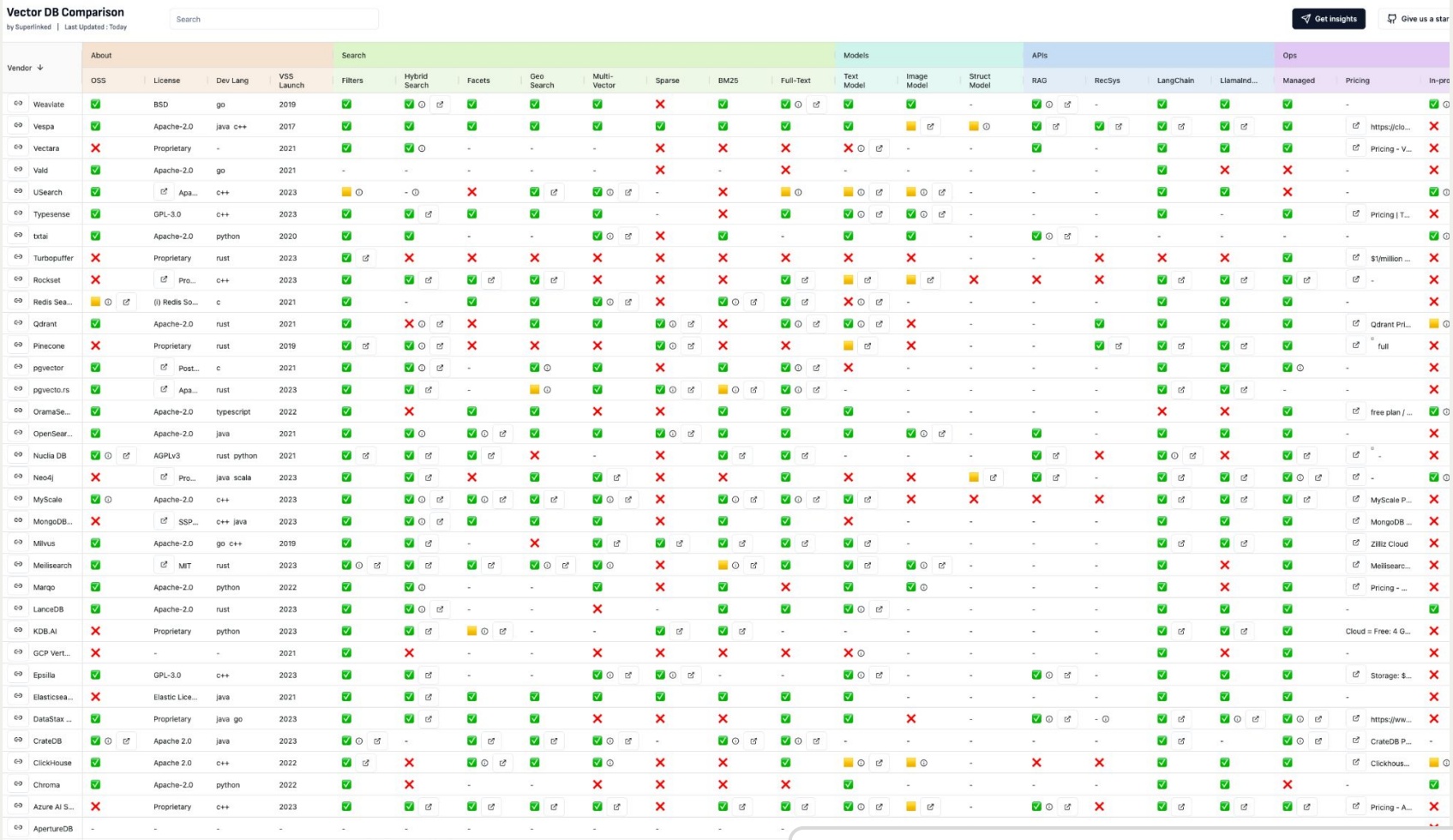
by Superlinked | Last Updated : Today

Search

Get insights | Give us a star

Vendor	About				Search								Models	
	OSS	License	Dev Lang	VSS Launch	Filters	Hybrid Search	Facets	Geo Search	Multi-Vector	Sparse	BM25	Full-Text	Text Model	Image Model
Weaviate	✅	BSD	go	2019	✅	✅	✅	✅	✅	❌	✅	✅	✅	✅
Vespa	✅	Apache-2.0	java c++	2017	✅	✅	✅	✅	✅	✅	✅	✅	✅	🟨
Vectara	❌	Proprietary	-	2021	✅	✅	-	-	-	❌	❌	❌	❌	-
Vald	✅	Apache-2.0	go	2021	-	-	-	-	-	❌	-	❌	-	-
USearch	✅	Apa...	c++	2023	🟨	-	❌	✅	✅	-	❌	🟨	🟨	🟨
Typesense	✅	GPL-3.0	c++	2023	✅	✅	✅	✅	✅	-	❌	✅	✅	✅
txtai	✅	Apache-2.0	python	2020	✅	✅	-	-	✅	❌	✅	-	✅	✅
Turbopuffer	❌	Proprietary	rust	2023	✅	❌	❌	❌	❌	❌	❌	❌	❌	❌
Rockset	❌	Prop...	c++	2023	✅	✅	✅	✅	❌	❌	❌	✅	🟨	🟨
Redis Sea...	🟨	(i) Redis So...	c	2021	✅	-	✅	✅	✅	❌	✅	✅	❌	-
Qdrant	✅	Apache-2.0	rust	2021	✅	❌	❌	✅	✅	✅	❌	✅	✅	❌
Pinecone	❌	Proprietary	rust	2019	✅	✅	❌	❌	❌	✅	❌	❌	🟨	❌
pgvector	✅	Post...	c	2021	✅	✅	-	✅	✅	❌	✅	✅	❌	-
pgvecto.rs	✅	Apa...	rust	2023	✅	✅	-	🟨	✅	✅	🟨	✅	-	-
OramaSea...	✅	Apache-2.0	typescript	2022	✅	❌	✅	✅	❌	❌	✅	✅	✅	-
OpenSear...	✅	Apache-2.0	java	2021	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅
Nuclia DB	✅	AGPLv3	rust python	2021	✅	✅	✅	❌	-	❌	✅	✅	-	-
Neo4j	❌	Prop...	java scala	2023	✅	✅	❌							
MyScale	✅	Apache-2.0	c++	2023	✅	✅	✅							

https://superlinked.com/vector-db-comparison


Vector DB Comparison
by Superlinked | Last Updated : Today
Search
Get insights
Give us a star
Vendor
About
Search
Models
APIs
Ops
OSS
License
Dev Lang
VSS Launch
Filters
Hybrid Search
Facets
Geo Search
Multi-Vector
Sparse
BM25
Full-Text
Text Model
Image Model
Struct Model
RAG
RecSys
LangChain
LlamaInd...
Managed
Pricing
In-pro
Weaviate
Vespa
Vectara
Vald
USearch
Typesense
txtai
Turbopuffer
Rockset
Redis Sea...
Qdrant
Pinecone
pgvector
pgvecto.rs
OramaSe...
OpenSear...
Nuclia DB
Neo4j
MyScale
MongoDB...
Milvus
Meilisearch
Marqo
LanceDB
KDB.AI
GCP Vert...
Epsilla
Elasticsea...
DataStax ...
CrateDB
ClickHouse
Chroma
Azure AI S...
ApertureDB
Apache S...
Apache C...
Anari AI
Activeloo...
https://superlinked.com/vector-db-comparison

Vector databases are hot!

But **why**?

AI

Why vector databases are having a moment as the AI hype cycle peaks

GenAI spurs demand for vector search startups, but database giants are also taking note

Paul Sawers / 8:00 AM PDT • April 20, 2024

Comment

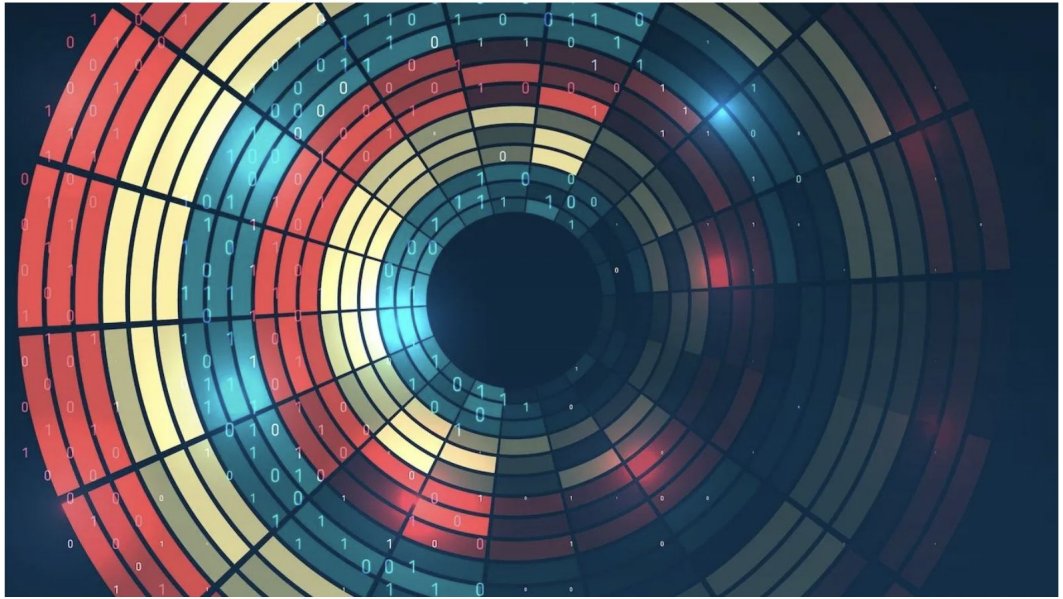
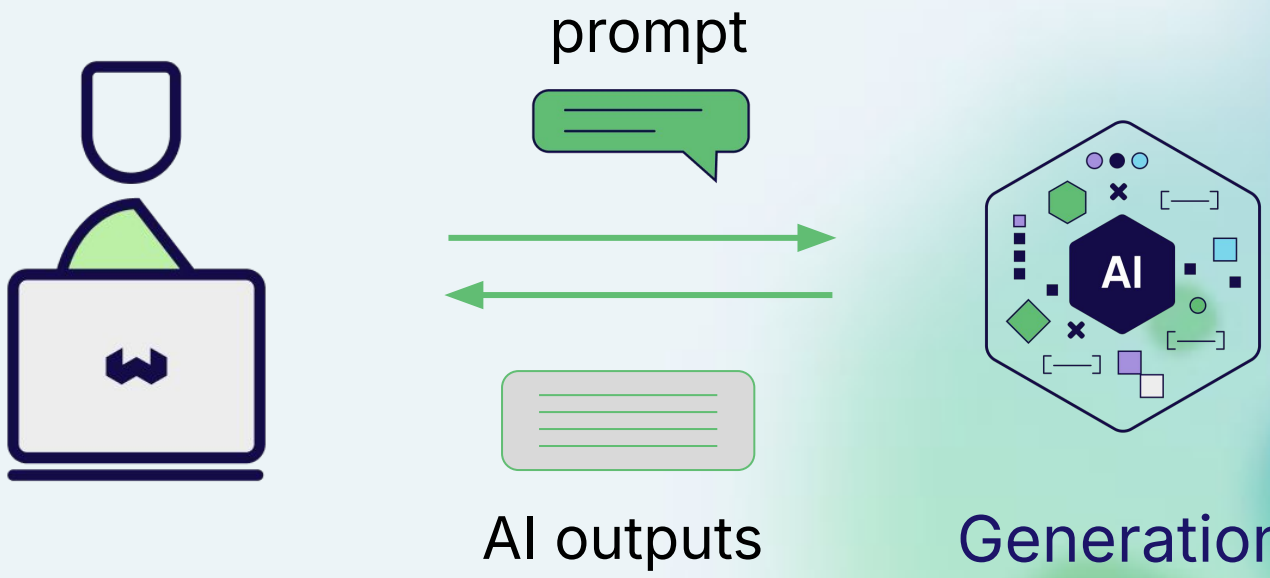


Image Credits: Getty Images

Vector databases are hot!

But **why** does AI need a new type of DB?

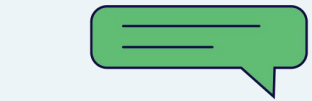
Large Language Model

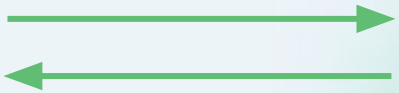


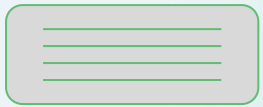
Large Language Model

Explain vector search like I am 5!

prompt



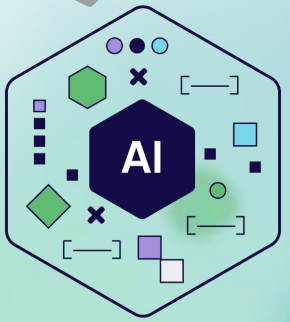




AI outputs

It's like giving each toy a special "fingerprint" ...

A magic helper that can quickly look at ALL the toys' "fingerprints" at once and tell you which ones are most similar to your query

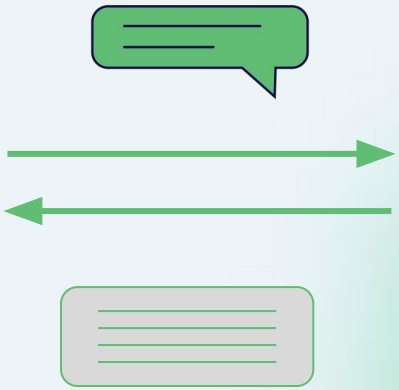


Generation

Large Language Model

How many world championships has Max Verstappen won?

prompt



AI outputs

Max Verstappen has won 3 Formula One World Championships.

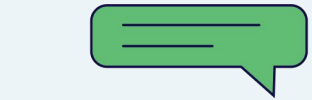
He won his first championship in 2021 ... He then dominated the 2022 and 2023 seasons.

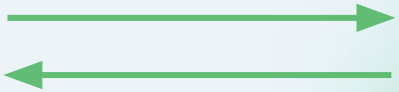
Generation

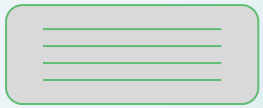
Large Language Model

Who is JP Hwang?

prompt





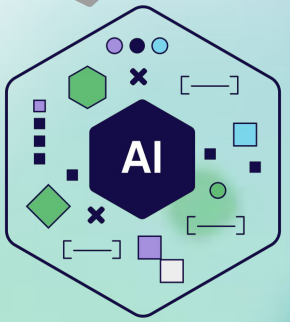


AI outputs

A South Korean-born, American entrepreneur and businessman.

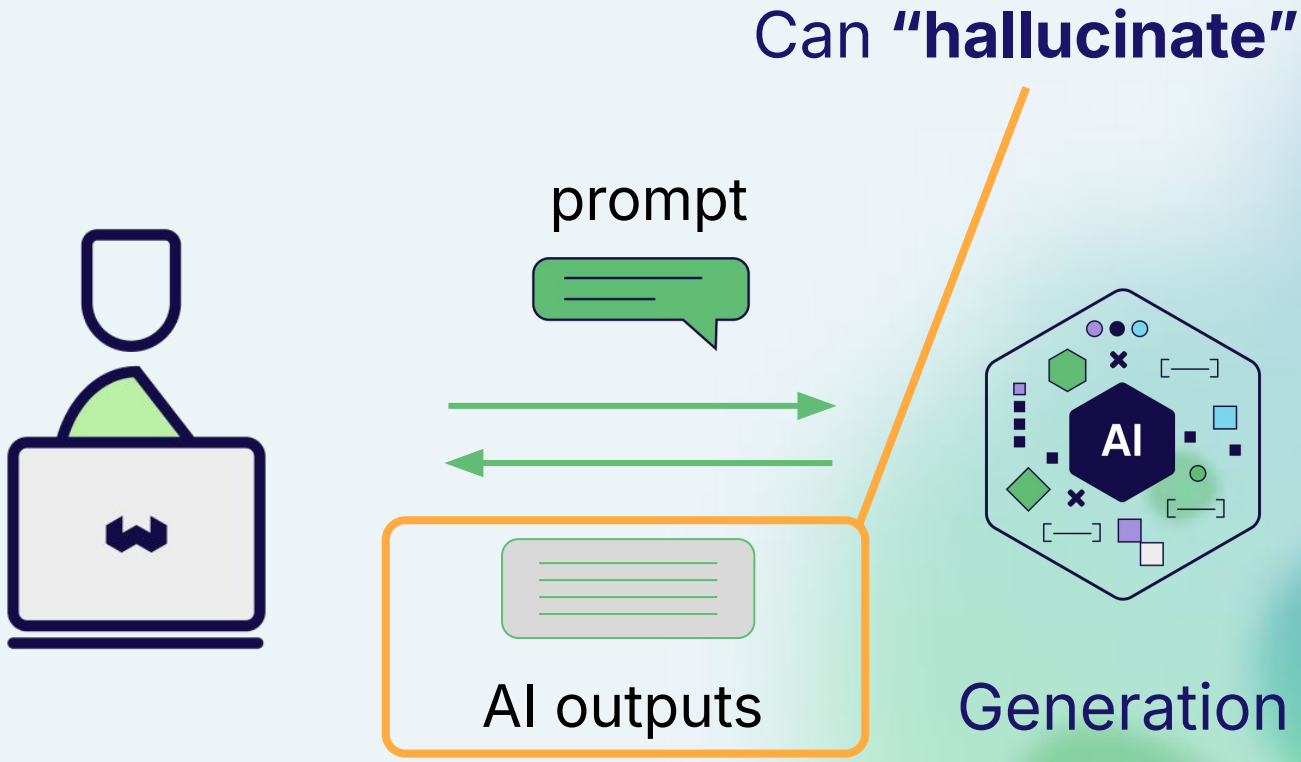
The founder and CEO of Superb Tickets, an online ticket marketplace.

...

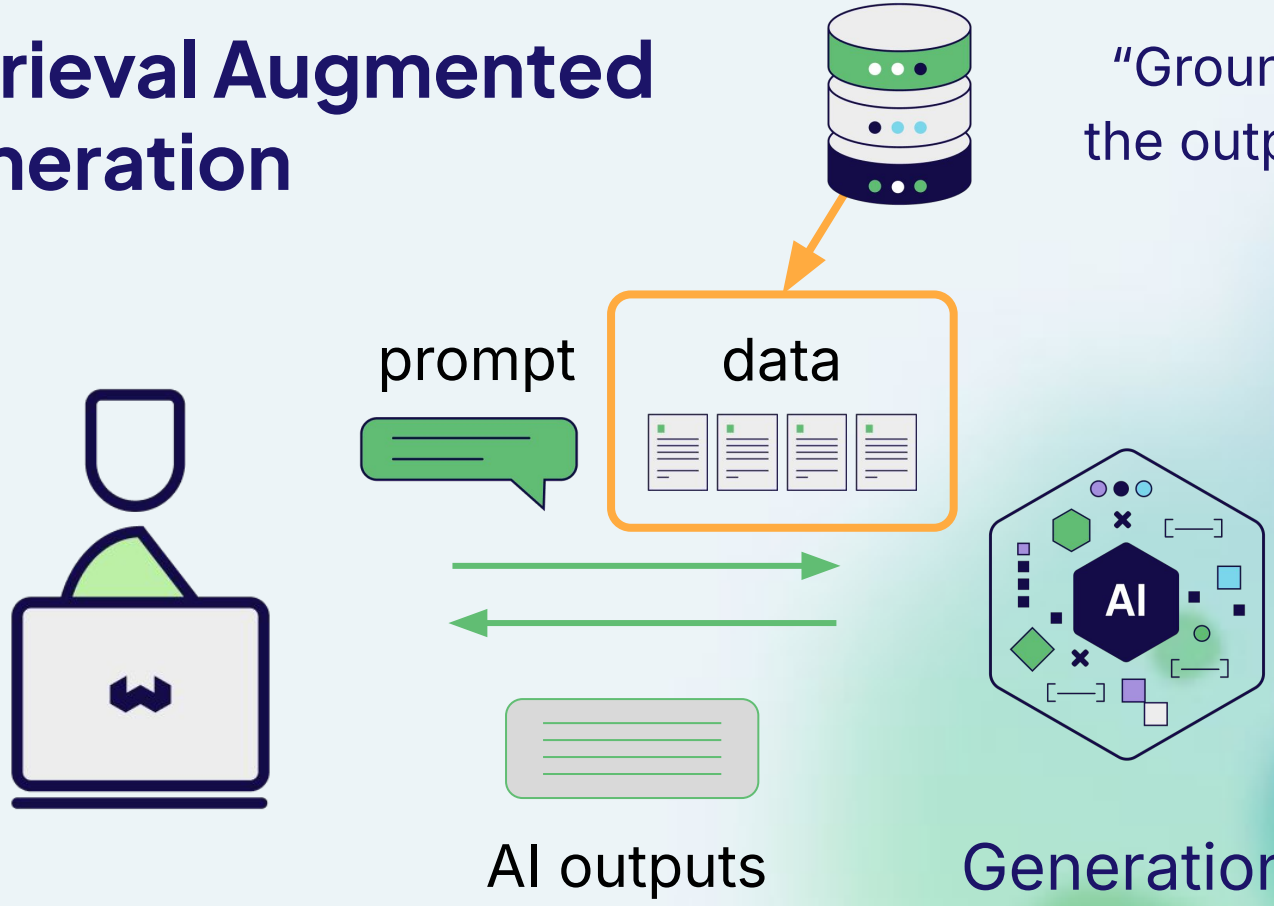


Generation

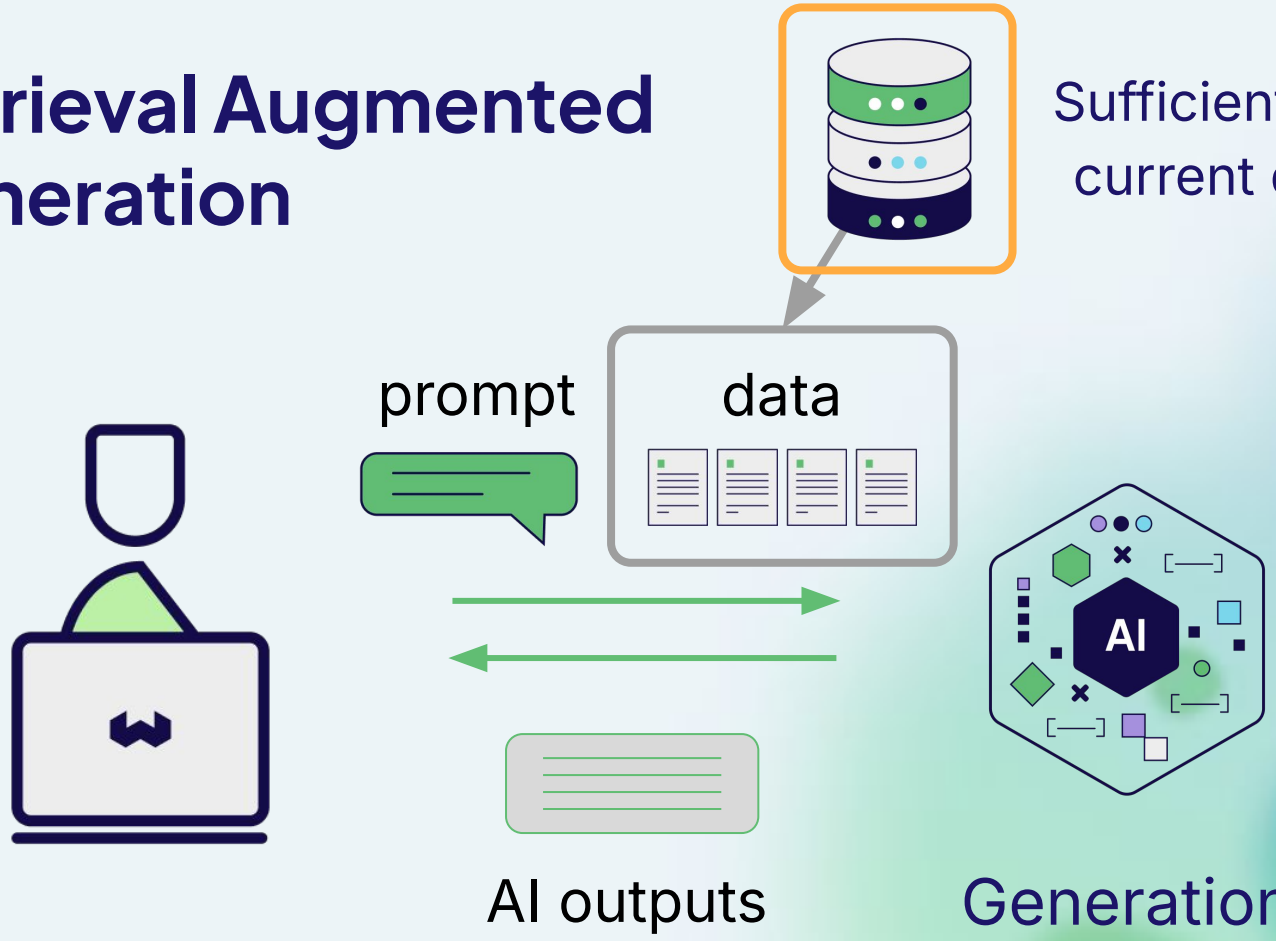
Large Language Model



Retrieval Augmented Generation



Retrieval Augmented Generation



Retrieval Augmented Generation

prompt

data

AI

AI outputs

Generation

Vector databases are hot!

But **why** does AI need a new type of DB?

Vector databases are hot!

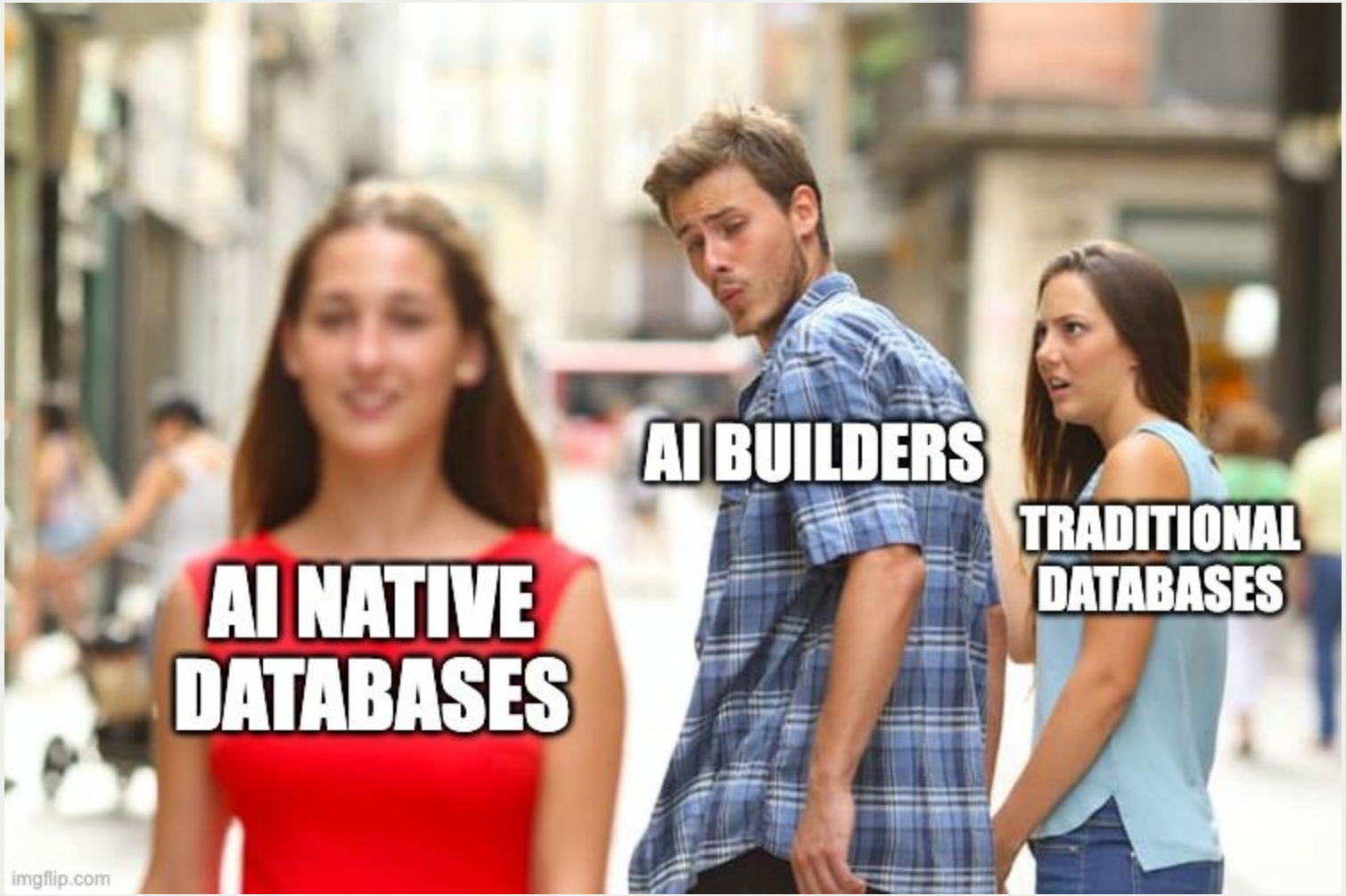
But **why** does AI need a new type of DB?

What makes a database **AI-native**?

What is AI-Native?

AI-native:
Remove AI model →
Nothing!

AI-enabled:
Remove AI model →
Lose functionality


AI BUILDERS
AI NATIVE DATABASES
TRADITIONAL DATABASES
imgflip.com

AI-native databases are hot!

But **why** does AI need a new type of DB?

What makes a database **AI-native**?

Will it be **useful for me?**

Let’s find out by…

Building our own solution

(At least **conceptually**!)

Conceptually build an AI-native DB

- Start small
- Add components
  - Learn their jobs
- Why do this?
  - Knowledge!
  - Tool selection and usage

Vectors:
A refresher / An introduction

A vector is

A vector is a set of numbers

Like

```
[1, 0]
```

or

```
[0.513, 0.155, 0.983, 0.001, 0.932]
```

or

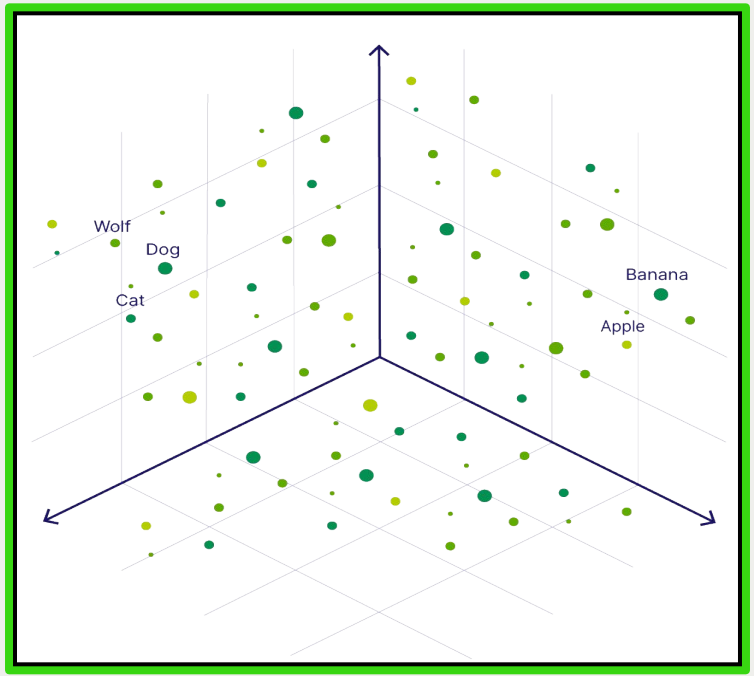
```
[0.0009420722, 0.020158706, -0.03939992,
-0.025480185, 0.018441677, 0.0023035712,
-0.012281344, -0.025270471, -0.056622636, …]
```

Why does this matter? 🤔

Vectors can **represent meaning**.

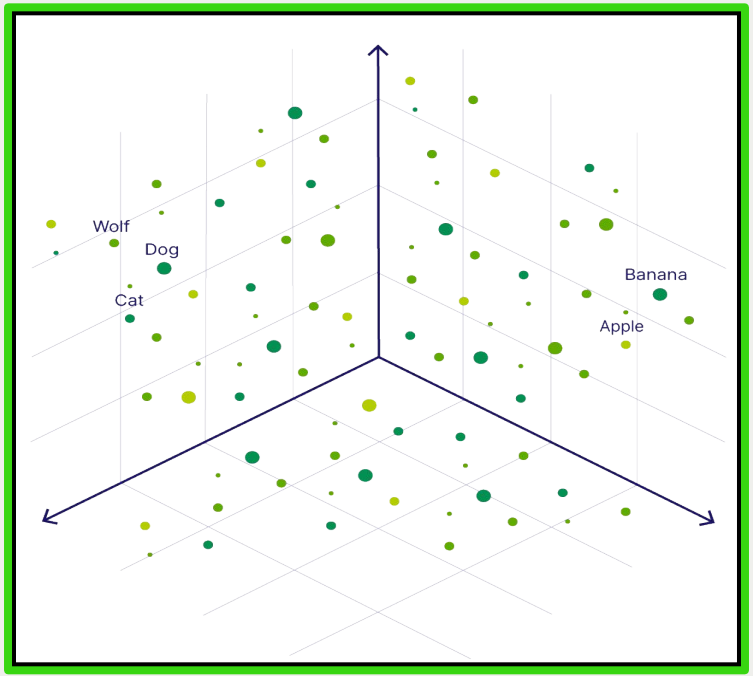
Why vectors?

- Vectors close if meaning close
- **Vector search is similarity search**



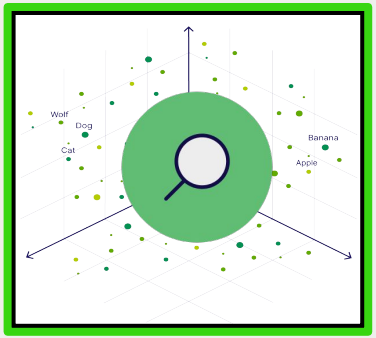
Why vectors?

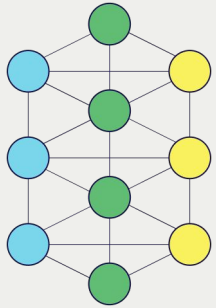
- Vectors close if meaning close
- Vector search is similarity search
- **Great for finding contextually relevant info**



Why vectors?

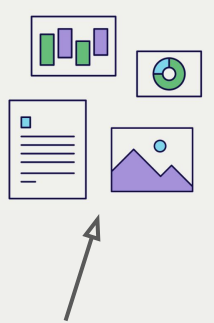
- Vectors close if meaning close
- Vector search is similarity search
- Great for finding contextually relevant info
    - **For RAG**



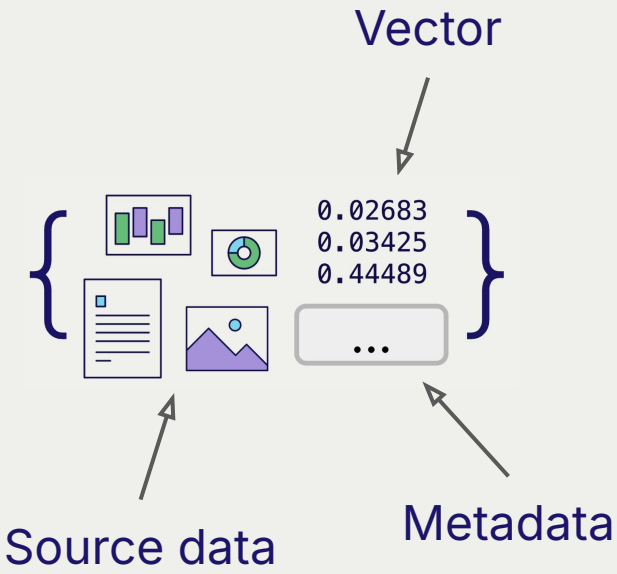


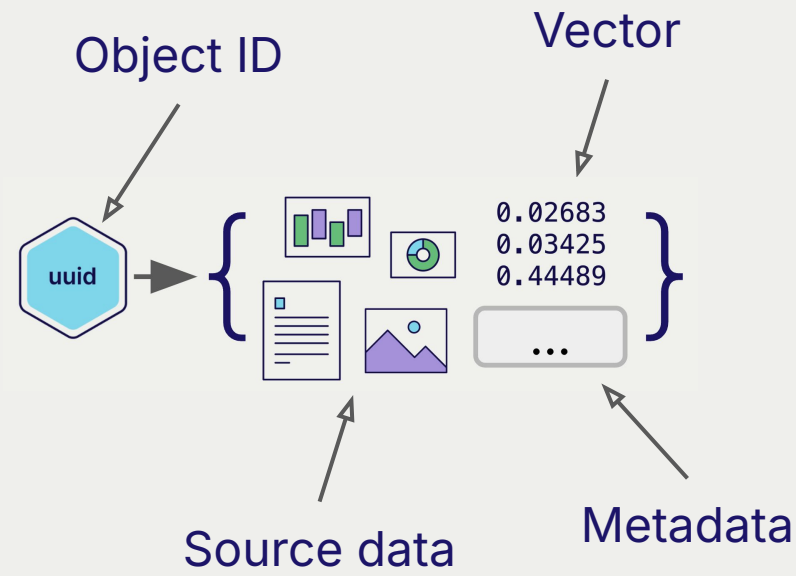
Generative

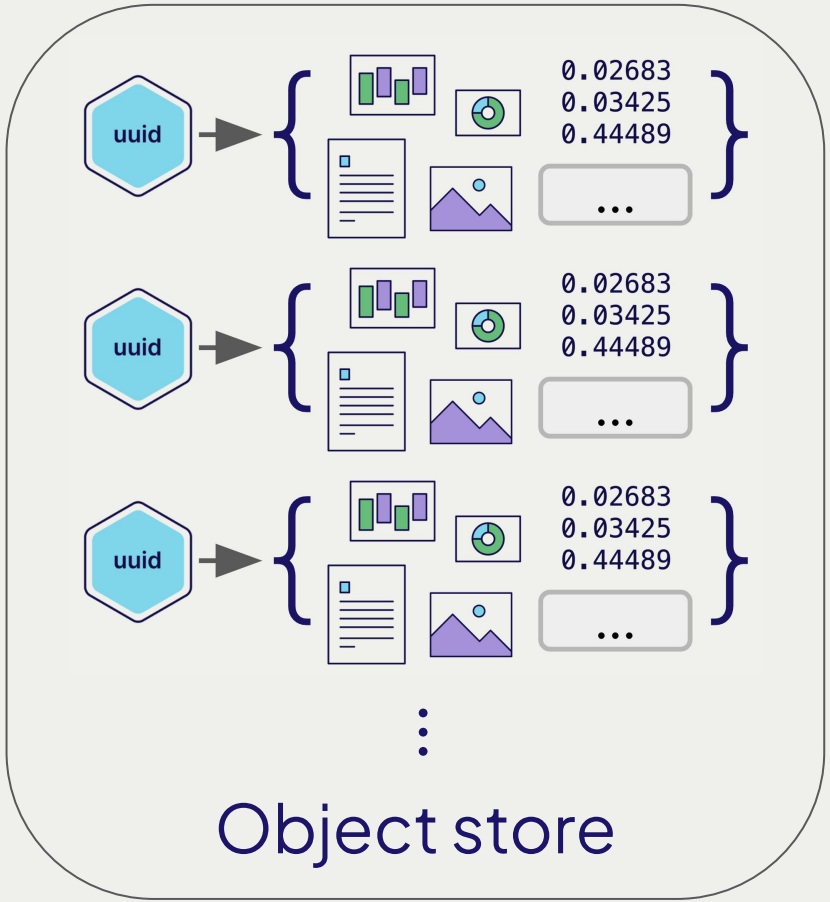
Let’s build a vector store



Source data


Vector
0.02683
0.03425
0.44489
...
Source data
Metadata

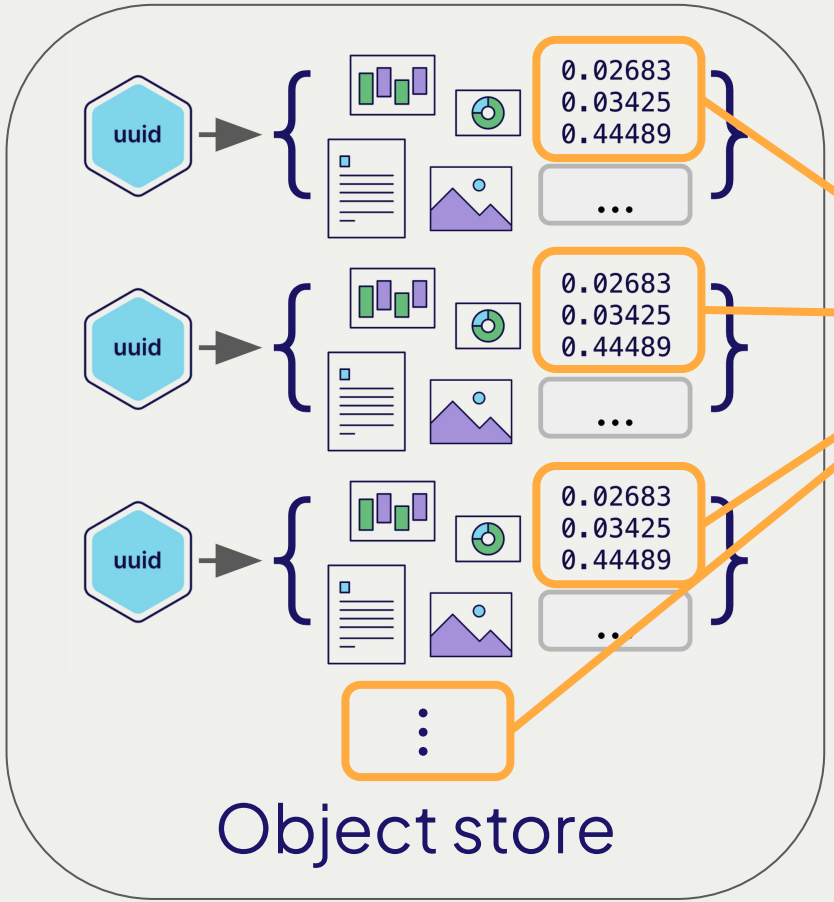

Object ID
Vector
uuid
0.02683
0.03425
0.44489
...
Source data
Metadata



User perspective:

- Can it save desired:
  - Data types
  - Vectors
  - Metadata

(Like many other key/value stores)

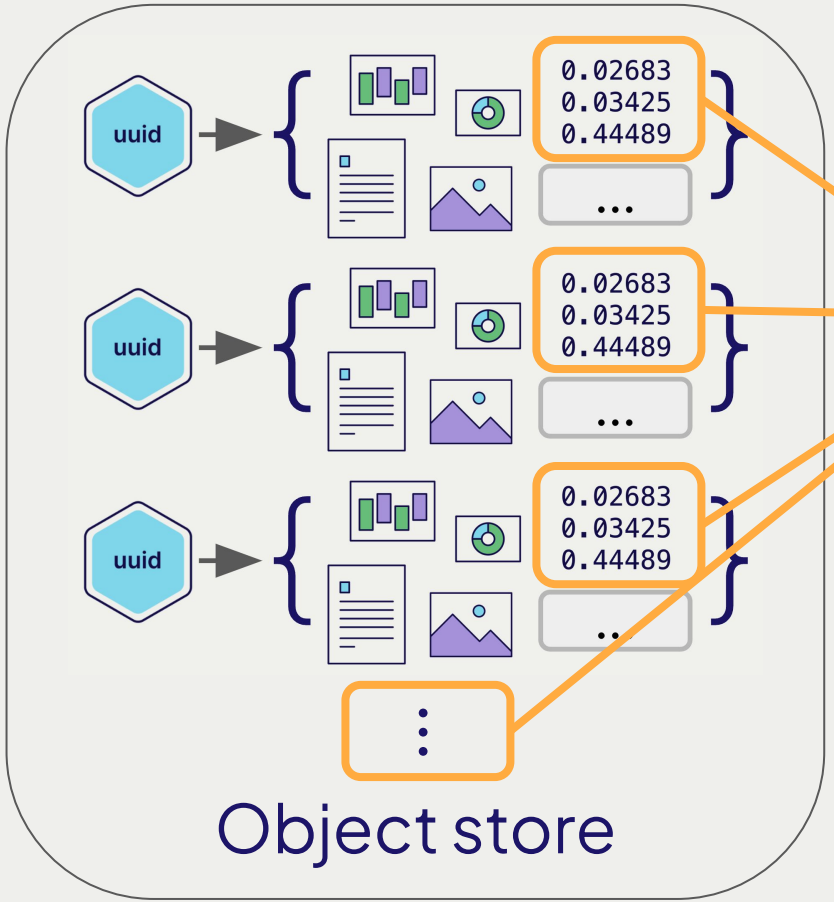


Solution:

- Calculate vector similarity vs query

Limitation:

- Retrieve the right information



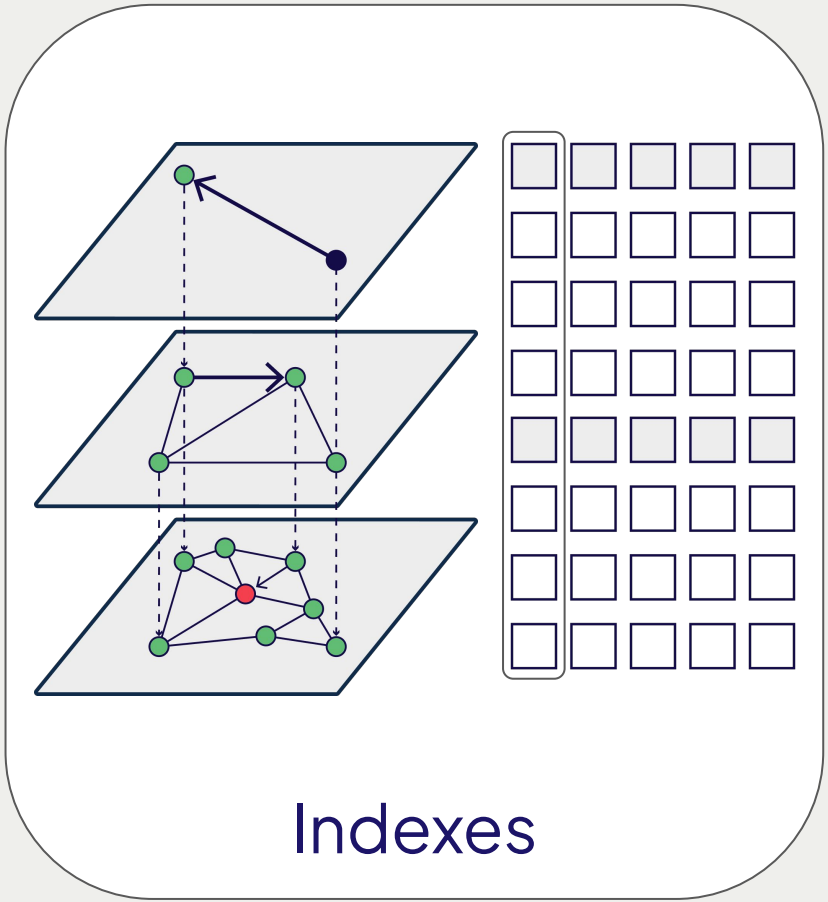
Solution:

- Calculate vector similarity vs query
- **Expensive!**

Limitation:

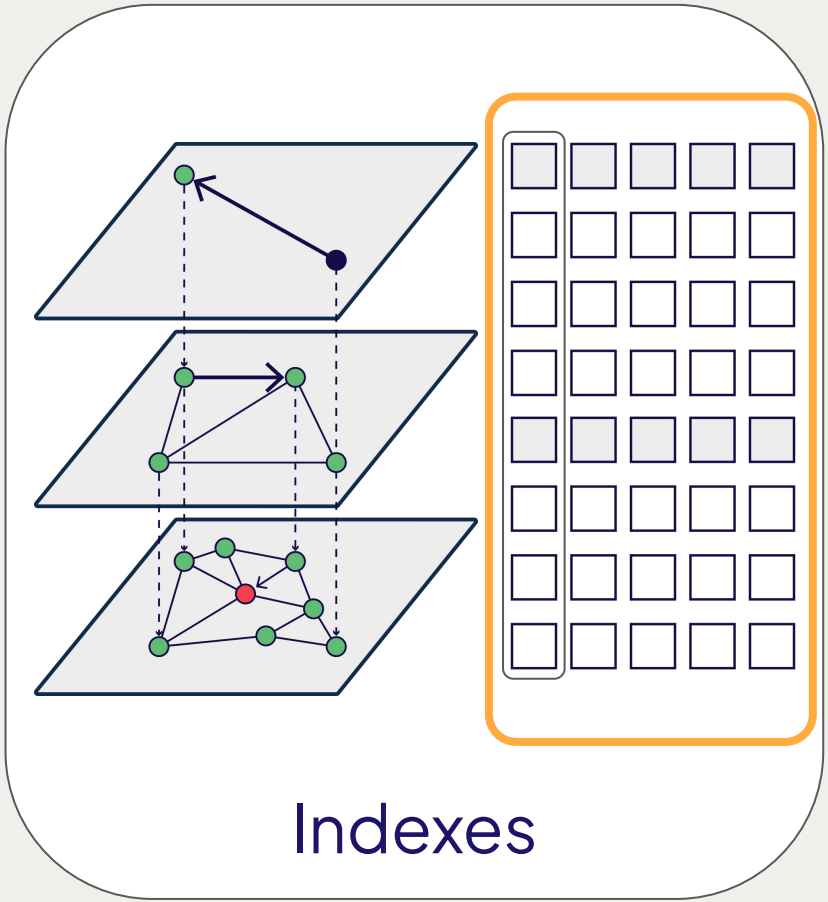
- Retrieve the right information

How do we scale search?



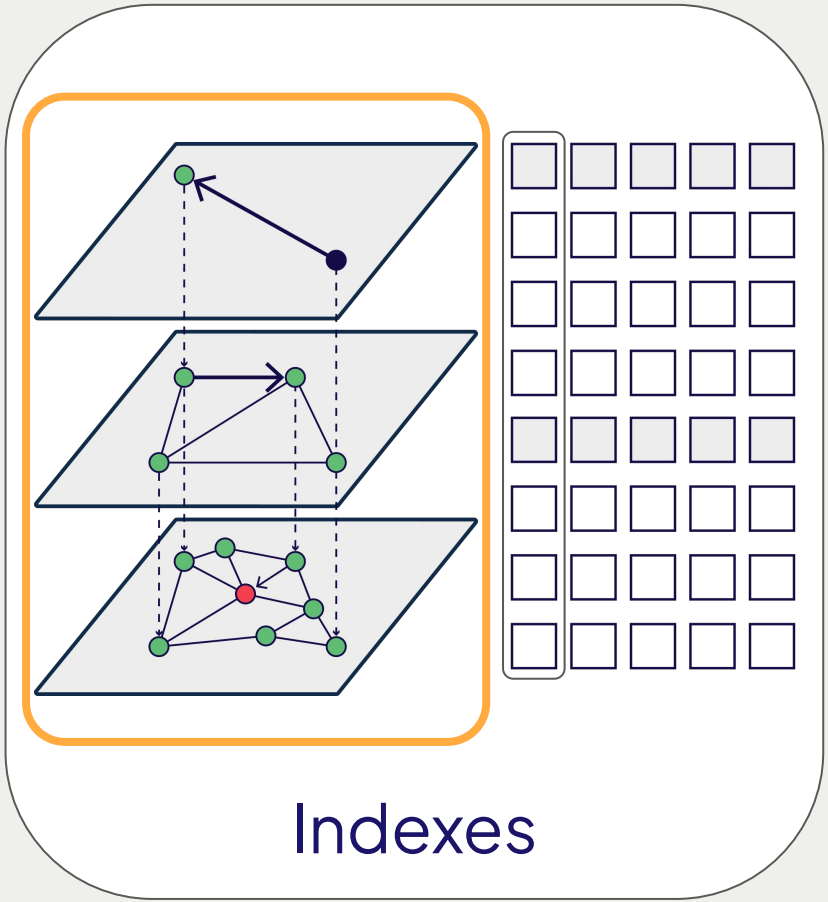
Indexes:

- Catalogue of data
  - Can speed up search
- Exist in RDBMS



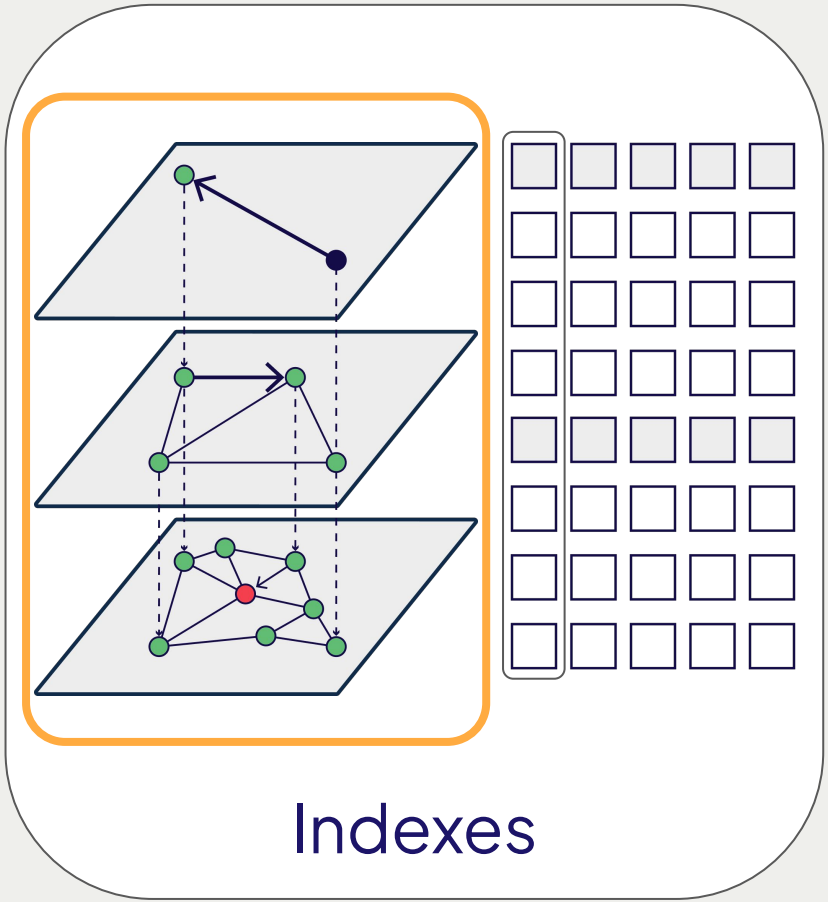
Inverted indexes:

- Organized information, by:
  - Property
  - Metadata (e.g. timestamp)



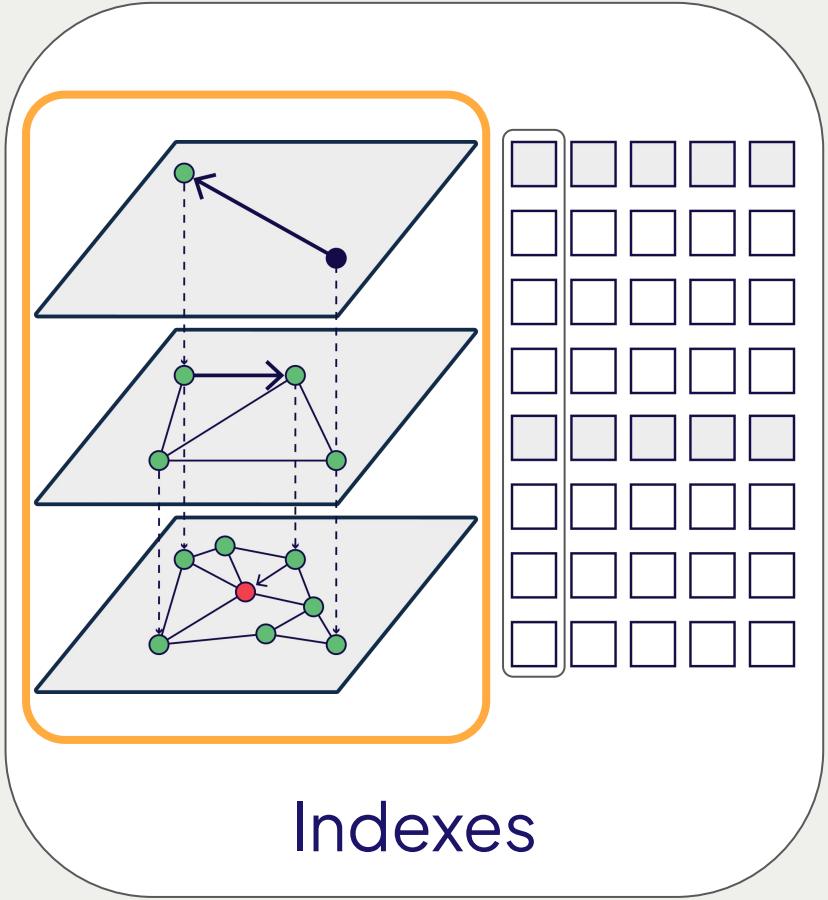
Vector indexes:

- Vectors capture meaning



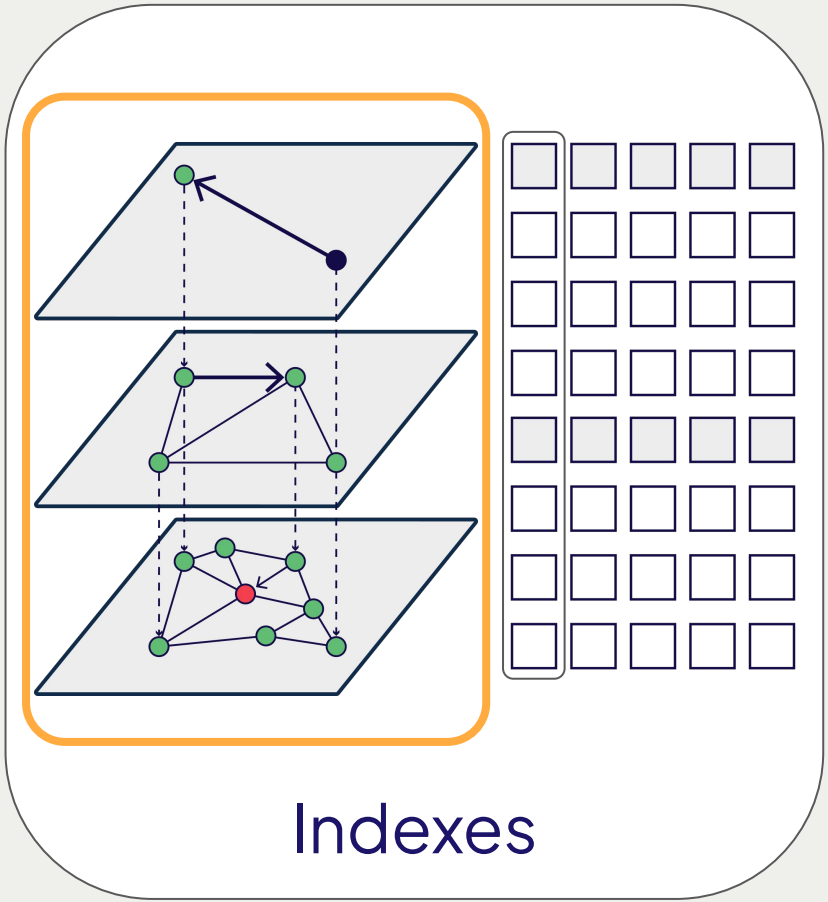
Vector indexes:

- Must be organised by *similarity*



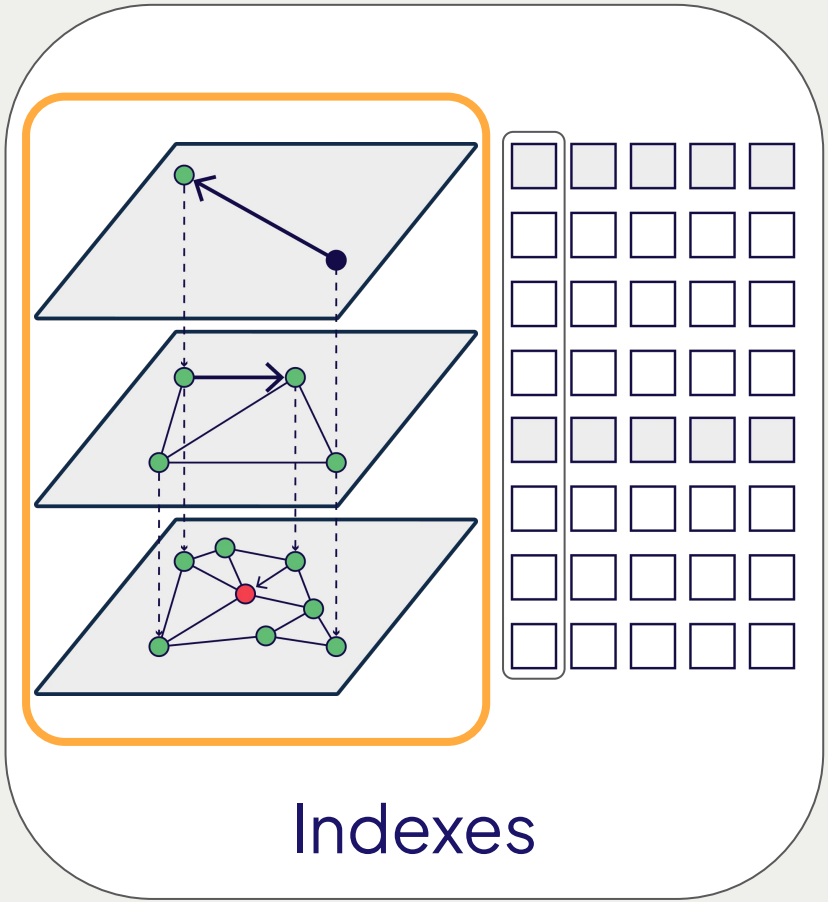
Vector indexes:

- Must be organised by *similarity*
- Types:
  - Flat
  - Graph-based (HNSW)
  - Tree-based
  - Others



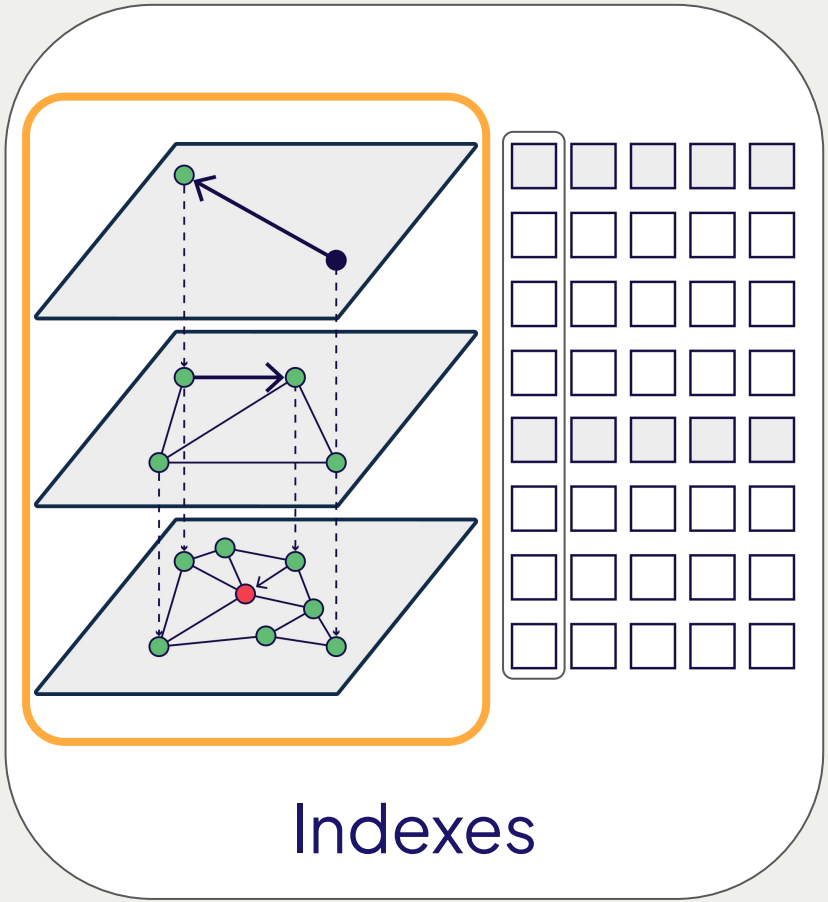
Vector indexes:

- “Approximate” searches
- Trade-offs
  - Speed
  - Accuracy
  - Memory



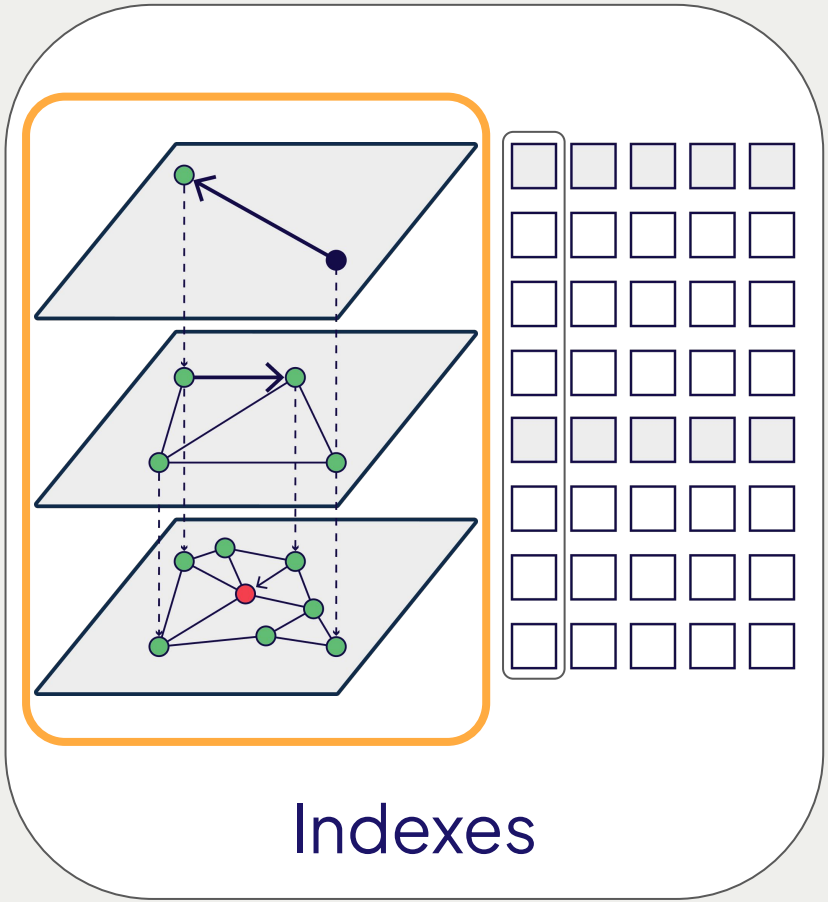
Flat indexes:

- Low overhead
- Linear search
- Good for small sets
- Does not scale well - O(n)



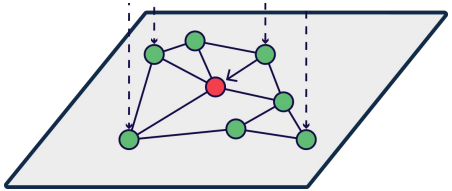
Graph-based indexes:

- Common
  - HNSW most common
- Scalable
- Typical limitation:
  - Memory footprint



Other index types:

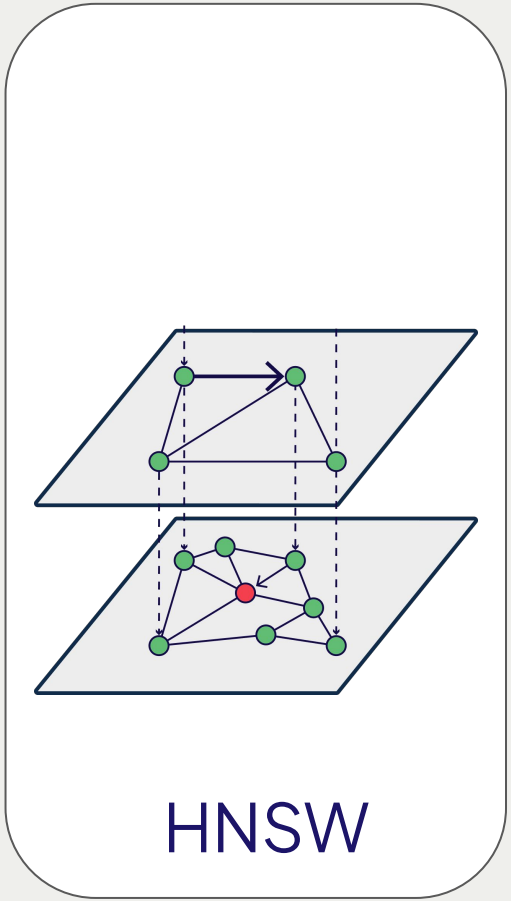
- Numerous (ANNOY, IVF)
- Have some benefits
- Generally not as scalable as HNSW
- Not as popular



HNSW

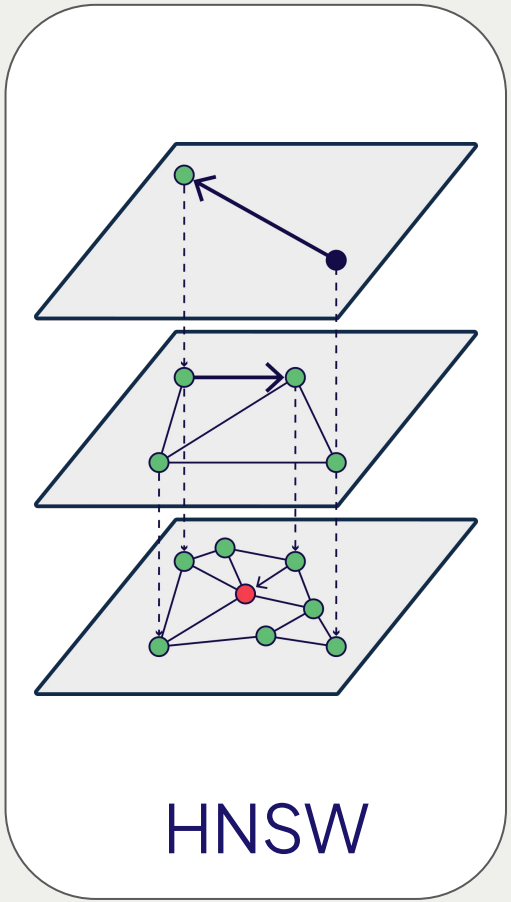
HNSW in focus

- Layered graphs
- Lowest layer: all vectors (nodes)
  - Nodes connected to neighbors



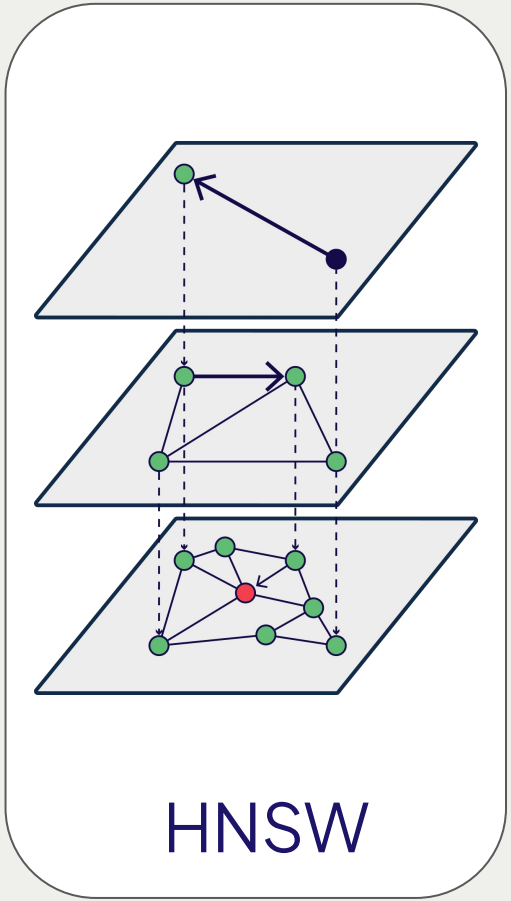
HNSW in focus

- Layered graphs
- Lowest layer: all vectors (nodes)
  - Nodes connected to neighbors
- Upper layers: subsets of nodes

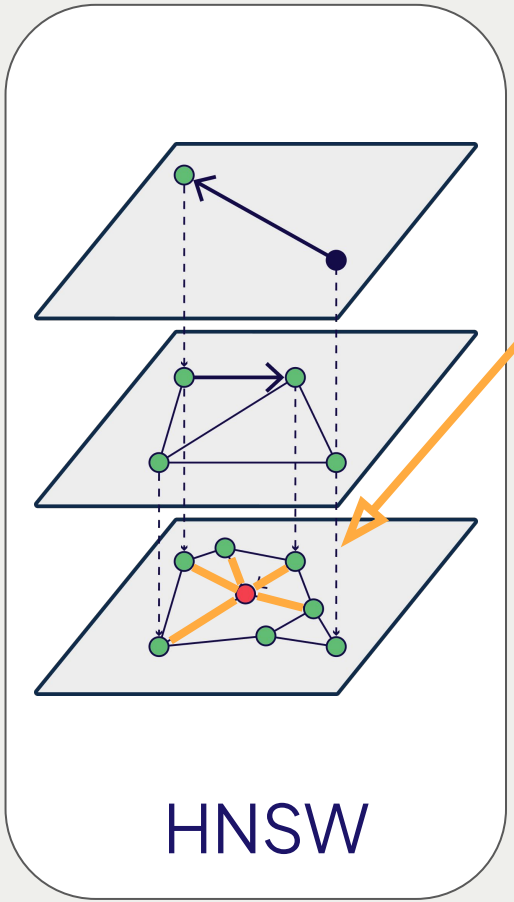


HNSW in focus

- Layered graphs
- Lowest layer: all vectors (nodes)
  - Nodes connected to neighbors
- Upper layers: subsets of nodes
  - Higher ↔ fewer nodes
  - Higher ↔ faster traversal
- Multi-dimensional skip lists

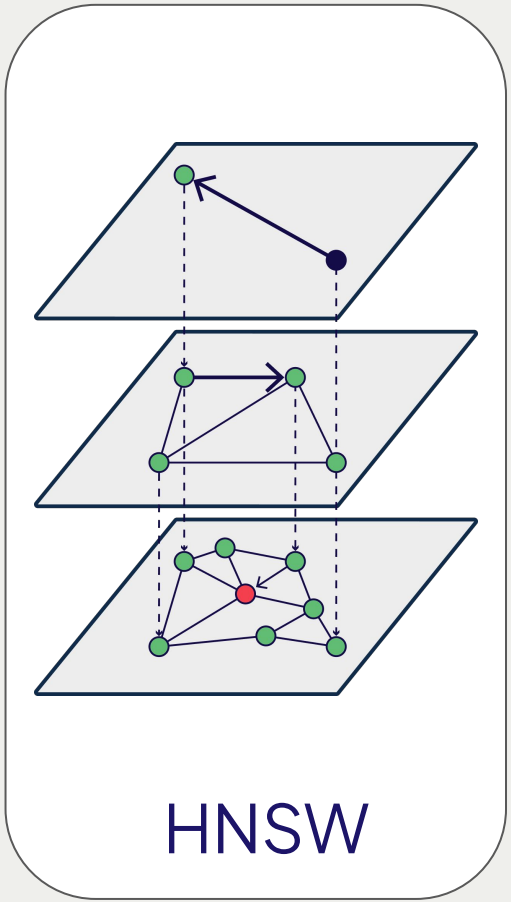


HNSW parameters



HNSW parameters

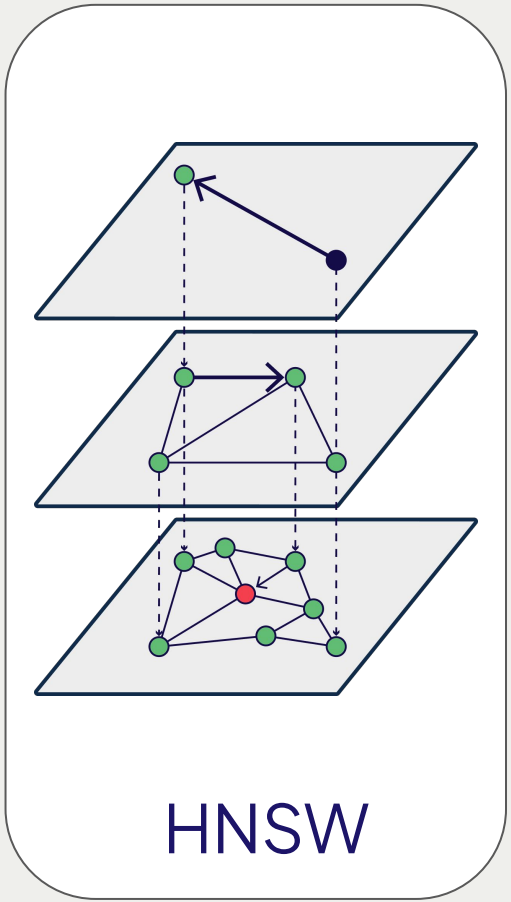
- **maxConnections** (m)
  - N (node → node) connections
    - e.g. 32
  - Note: bottom layer → m * 2



HNSW parameters

- maxConnections (m)
- **ef** → size of list during search
  - Higher → better
  - Cost: slower

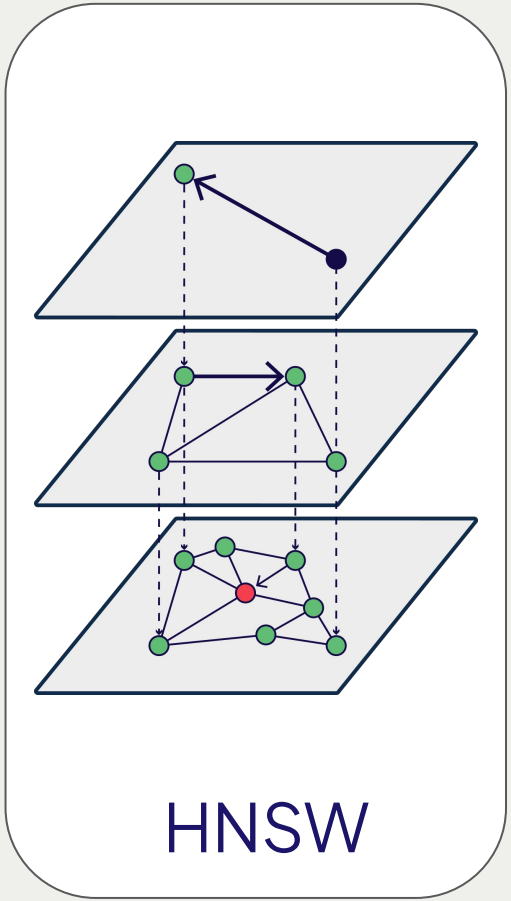
rank	object	distance
1	<uuid>	0.0514
2	<uuid>	0.0642
…	…	…
…	…	…
…	…	…
…	…	…
max(ef, limit)	<uuid>	0.3532



HNSW parameters

- maxConnections (m)
- ef
- **efConnections**
  - For graph construction
  - Higher → better

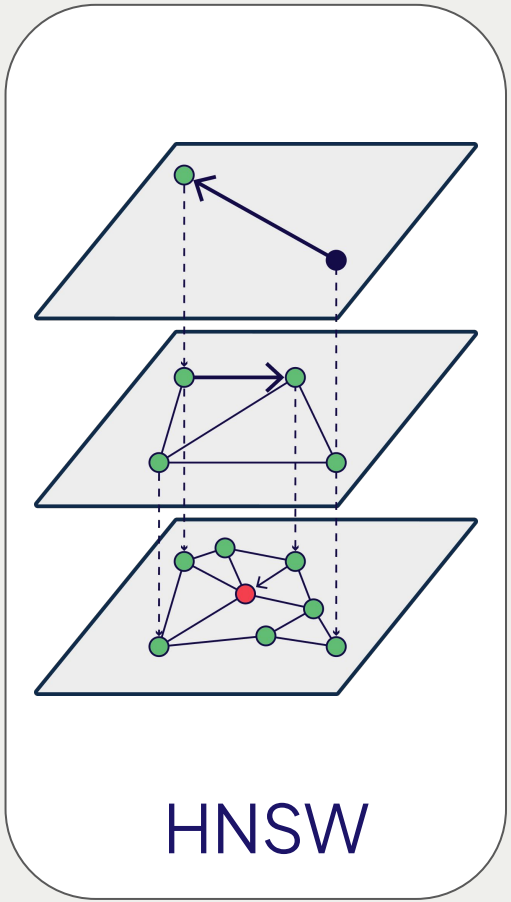
    Slower insertion



HNSW parameters

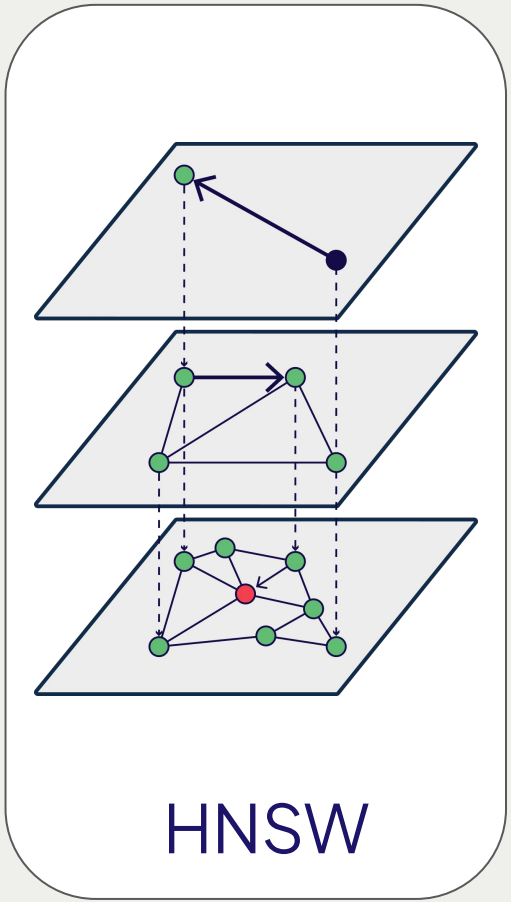
- maxConnections (m)
- ef
- efConnections

Tune speed vs recall



HNSW in focus

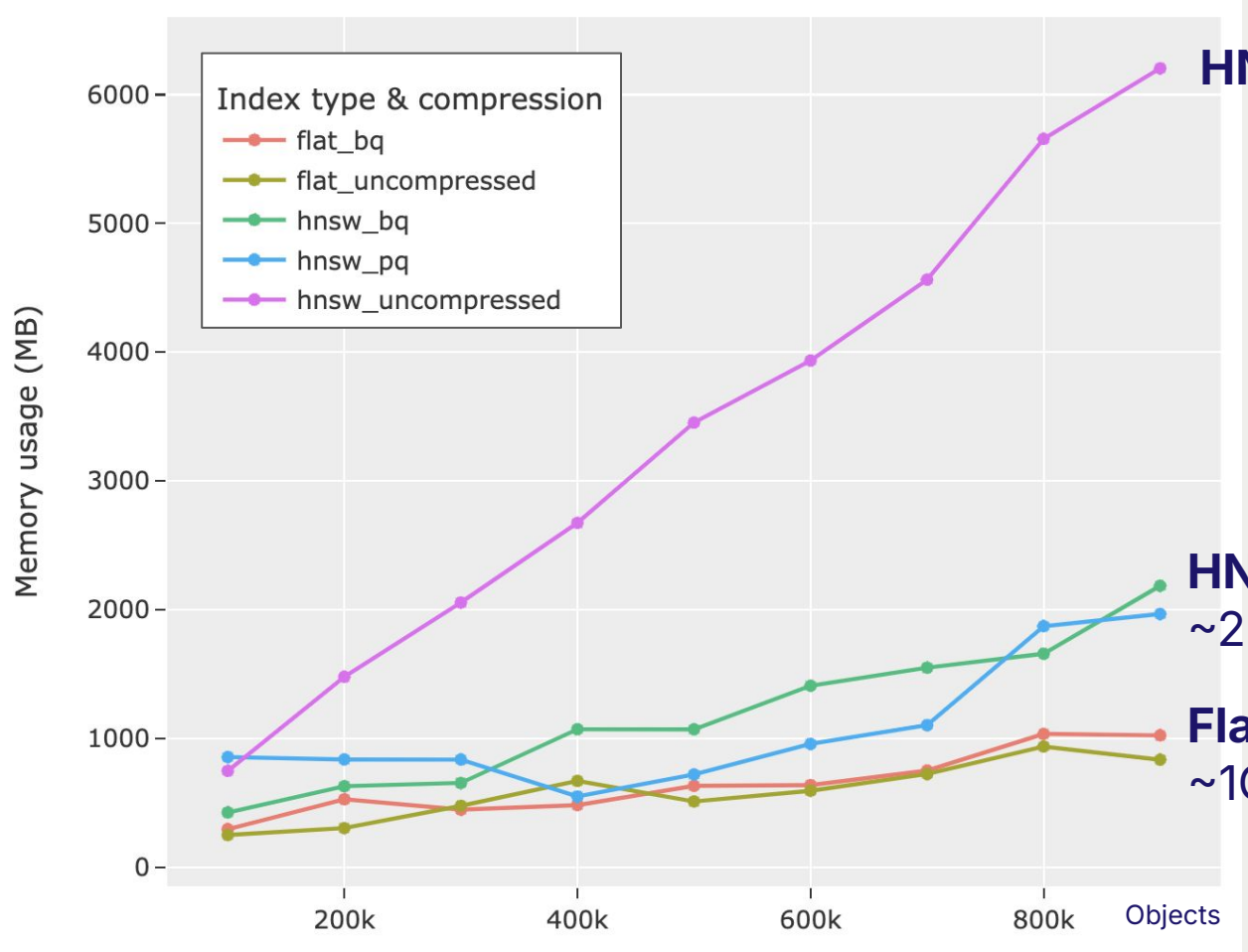
- In-memory data structure
- Typical limitation: memory footprint
  - Connections (~10B/ea @ 32-64)
  - Vectors (~4B/dim @ ~2000 dims)
  - @10M objects → ~65GB
  - @100M → ~650GB, etc...



Quantization

- Compression
- Reduce precision from floats
  - To integers, or even binary
- Reduce number of dimensions
  - Product quantization

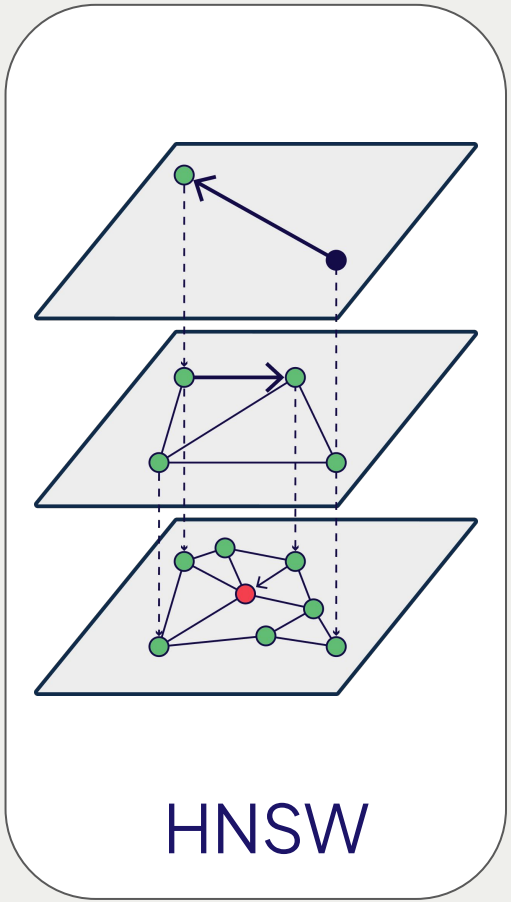
Example: Index type & quantization vs memory footprint



HNSW: ~6GB

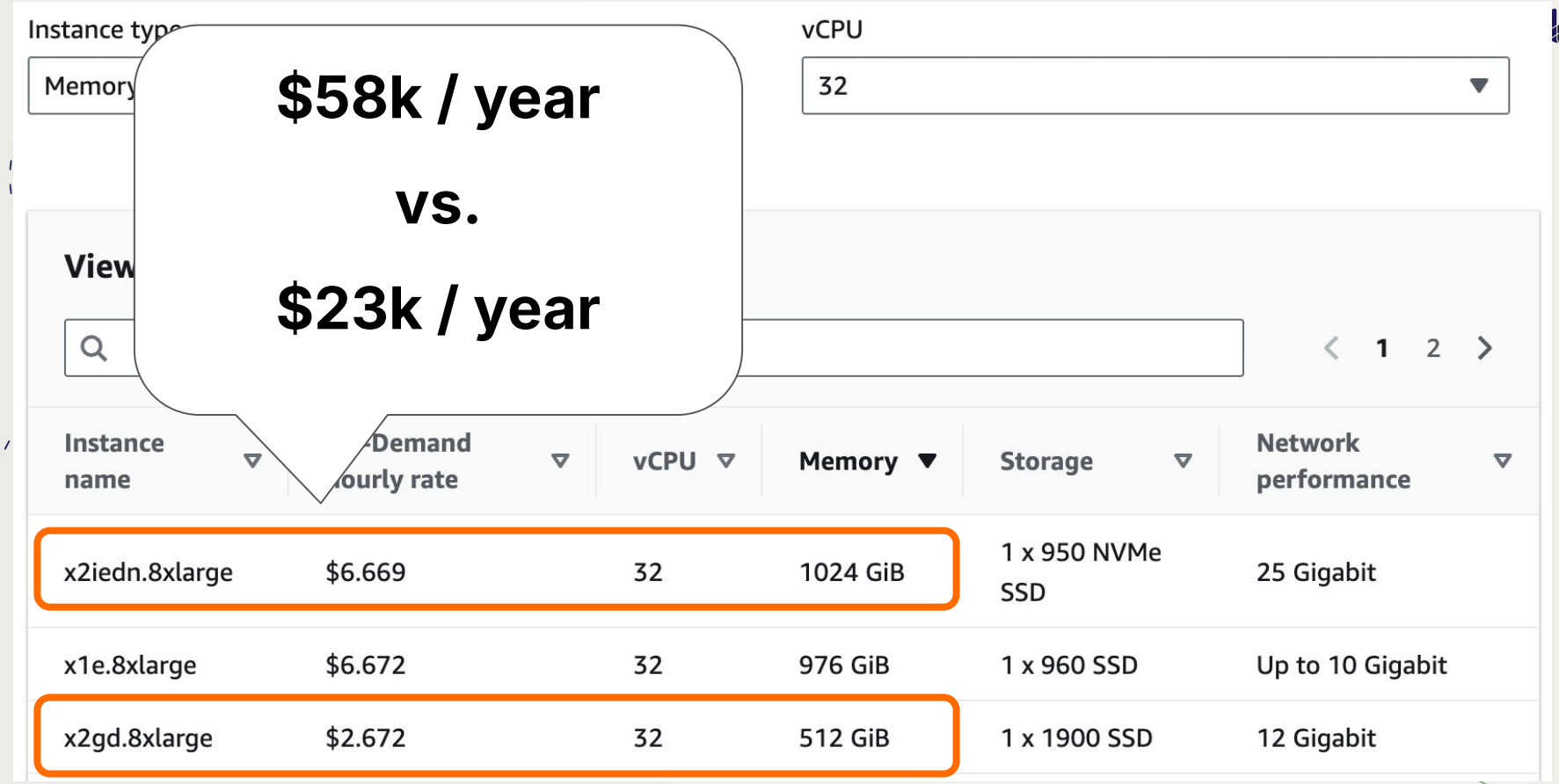
HNSW + BQ/PQ: ~2GB

Flat (+BQ): ~1GB

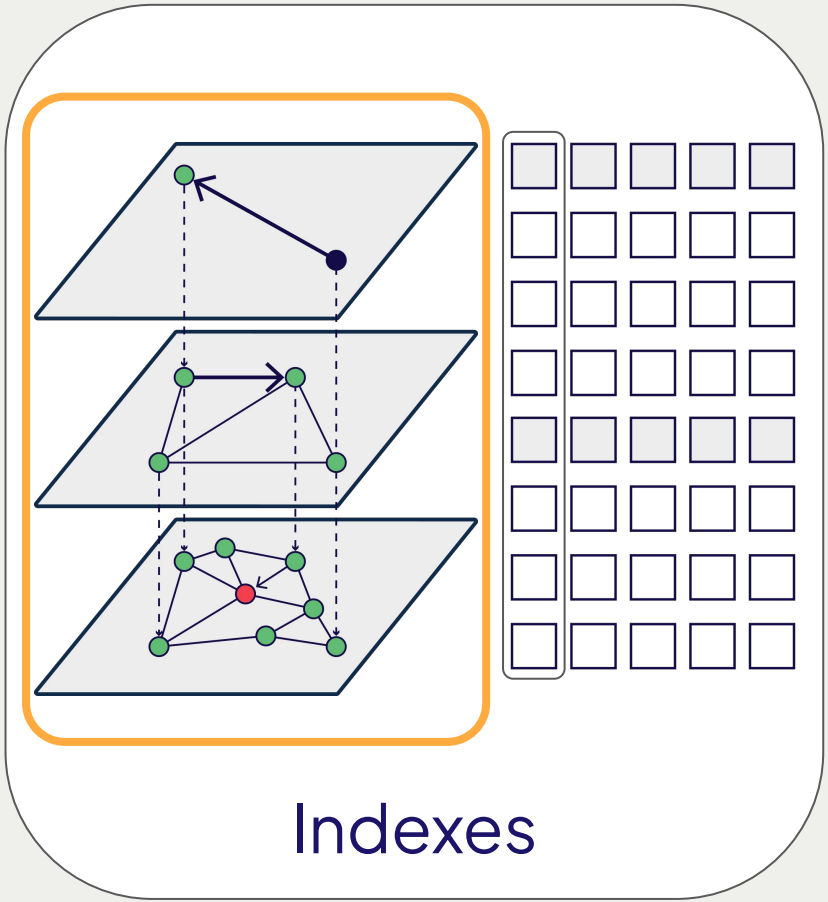


HNSW in focus

- In-memory data structure
- Typical limitation: memory footprint
  - Connections (~10B/ea @ 32-64)
  - Vectors (~4B/dim @ ~2000 dims)
  - @10M objects → ~65GB
  - **@100M → ~650GB, etc...**



AWS Spot pricing, Nov 2024



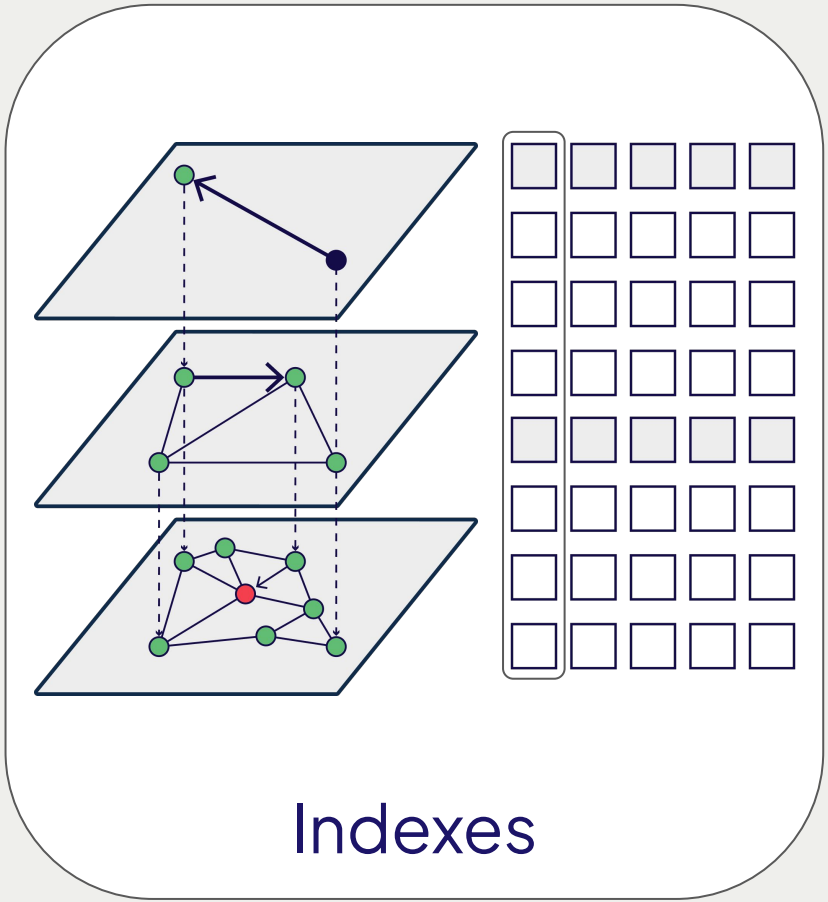
User perspective:

- Two indexes
- Many options to tune

  Speed, recall & memory
  - Index choice
  - Index parameters
  - Quantization

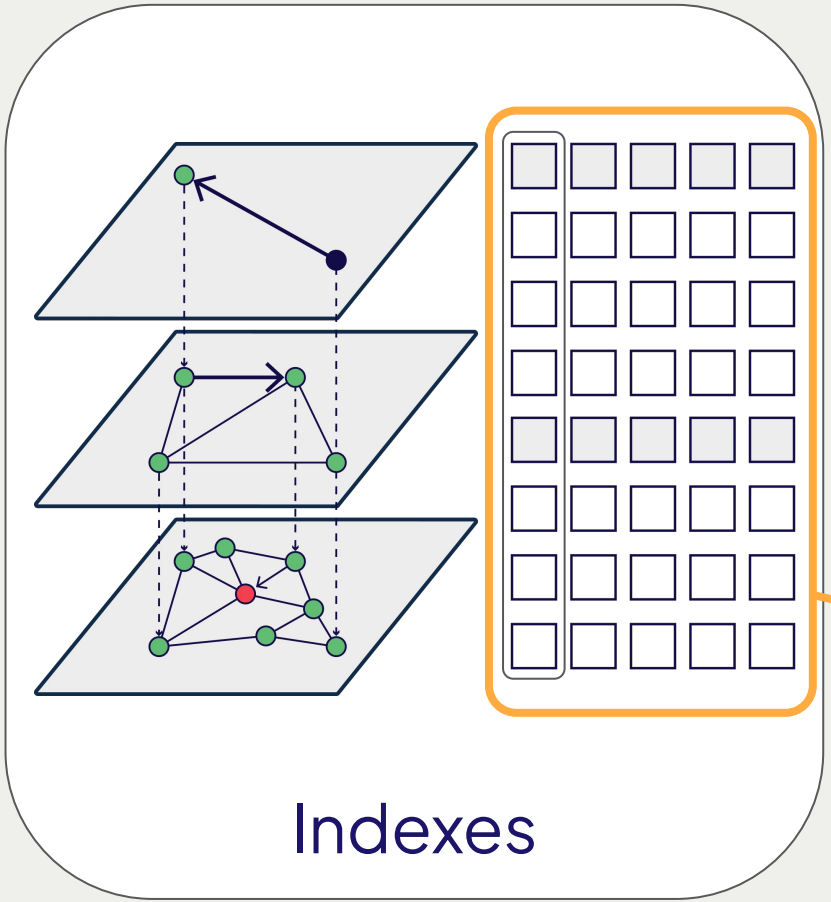
    (precision)

Tricks with both indexes



Filtering:

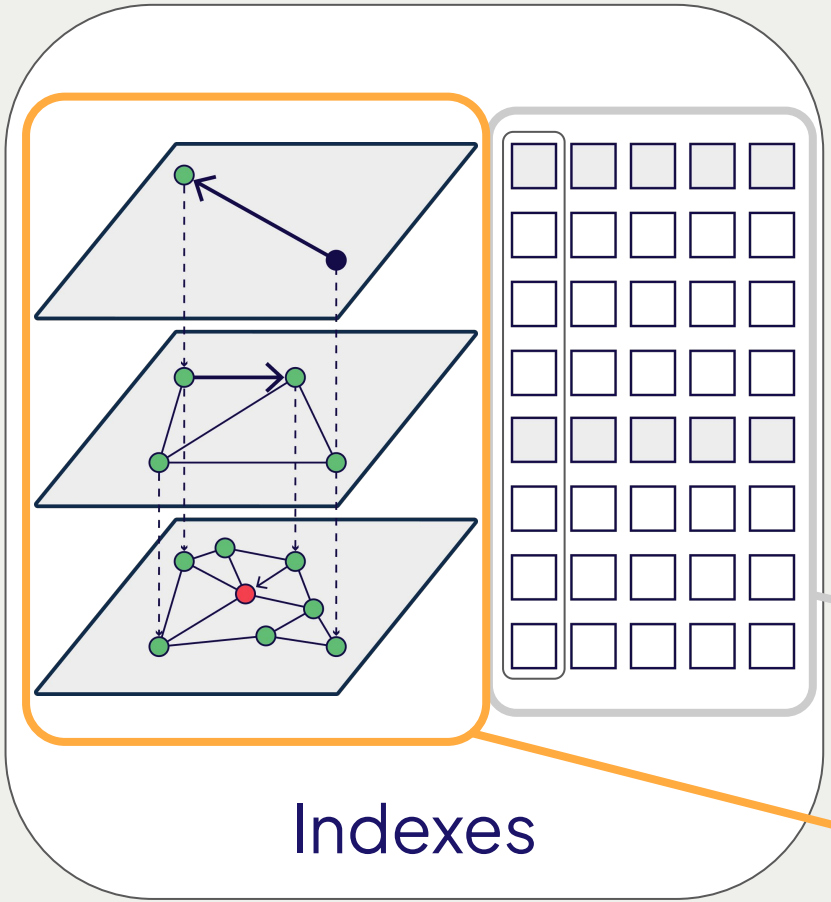
- Search with criteria, e.g.
  - `where ‘source’ == ‘Wiki’`



Filtering:

- Search with criteria, e.g.
  - `where 'source' == 'Wiki'`

① Build “allow list”

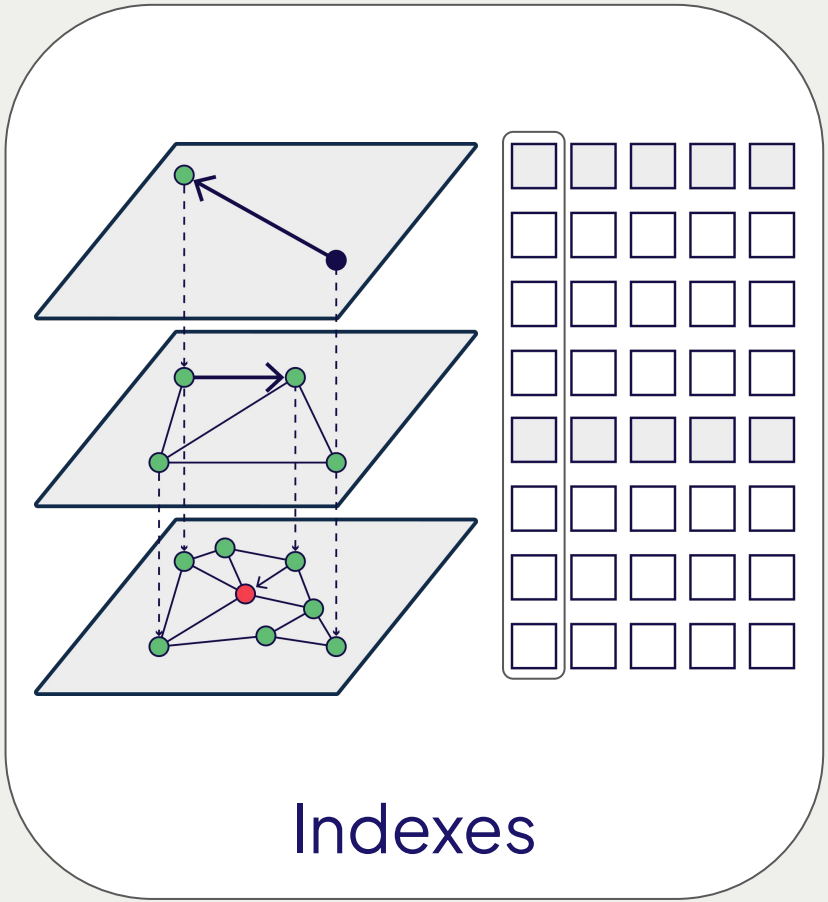


Filtering:

- Search with criteria, e.g.
  - `where 'source' == 'Wiki'`

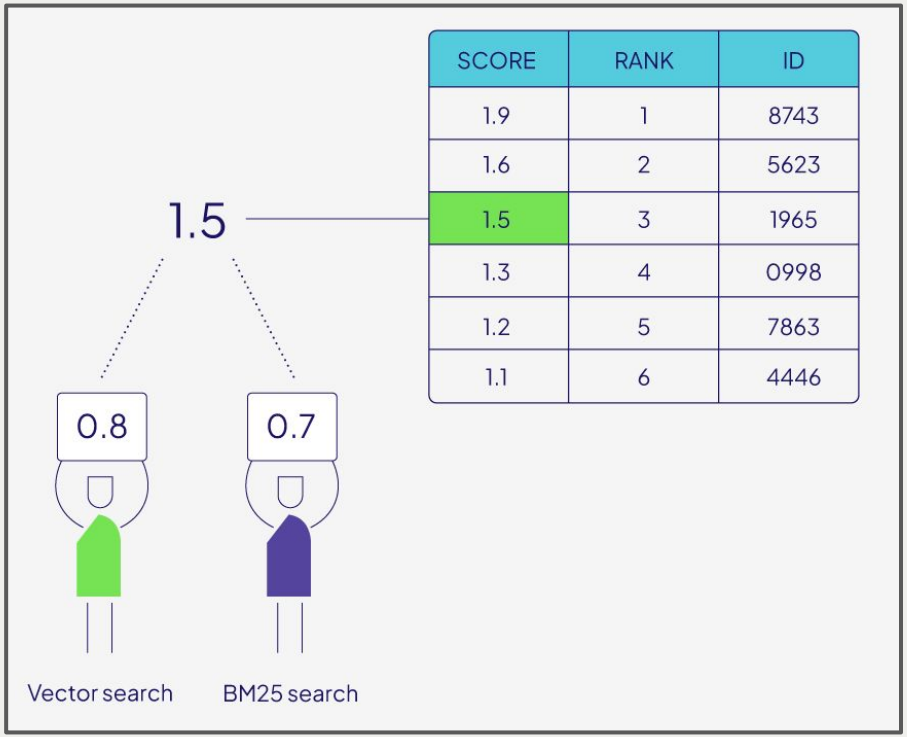
① Build “allow list”

② Vector search with “allow list” context



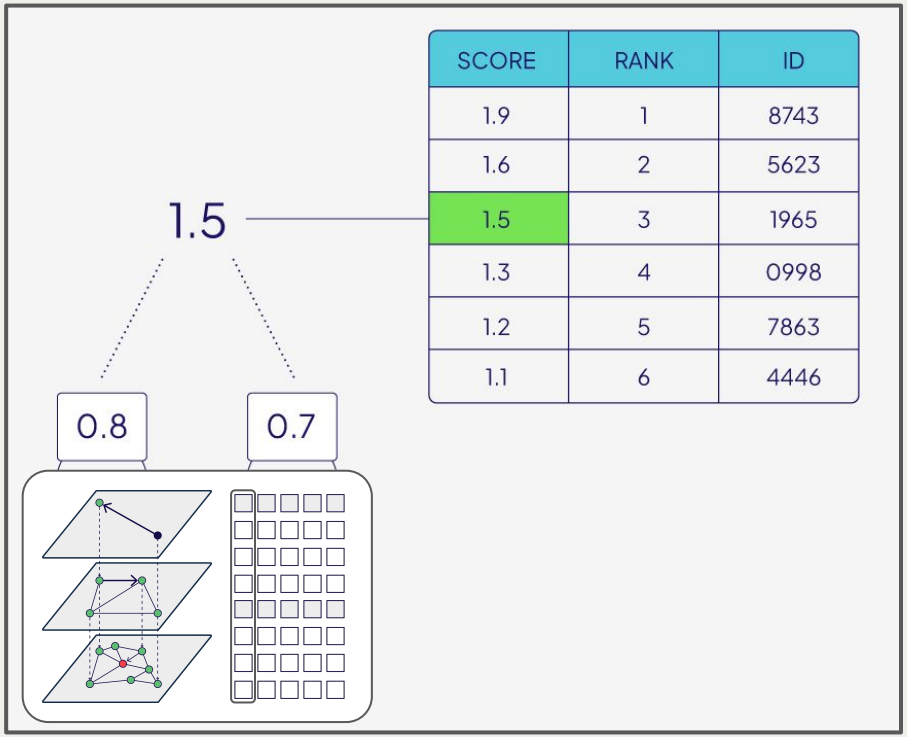
Filtering:

- Search with criteria, e.g.
  - `where ‘source’ == ‘Wiki’`
- Requires speed
  - Allow list:v big/v small
  - Separate filterable index
    - Roaring bitmaps



Hybrid search:

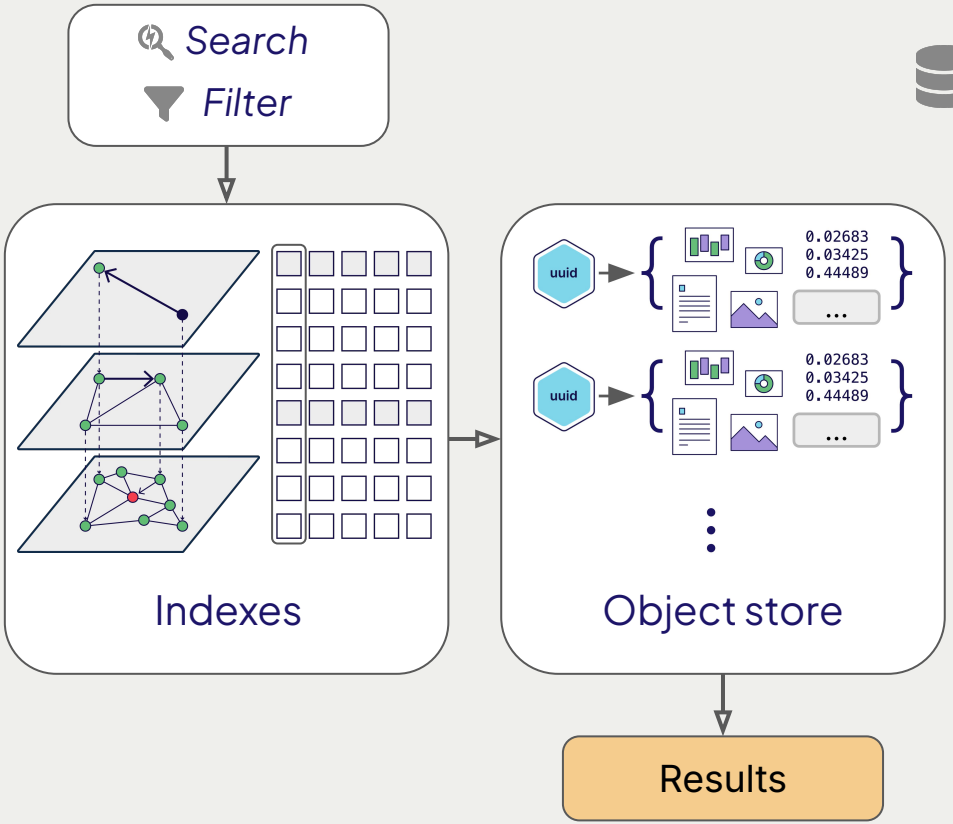
- Combine keyword & vector search results



Hybrid search:

- Combine keyword & vector search results
- Each search uses a different index

Progress update / recap

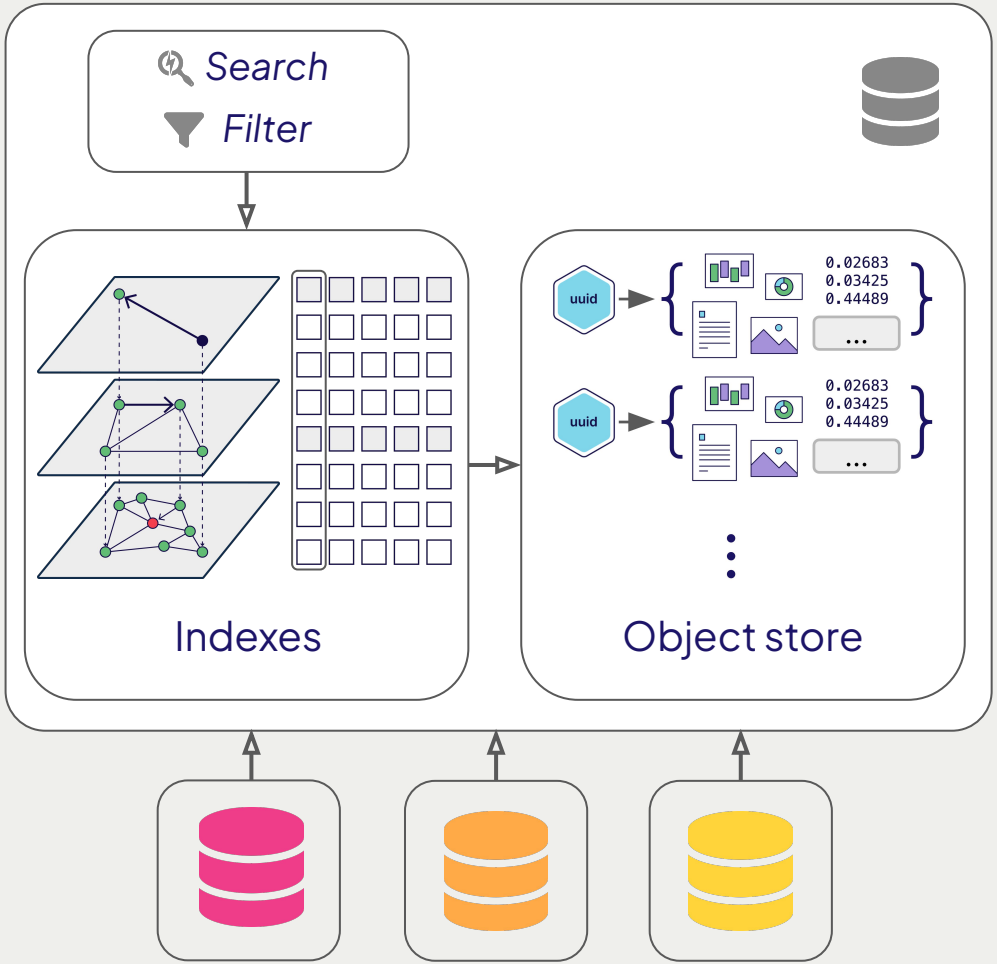


Mini-database:

- Store data
- Fast searches & filters

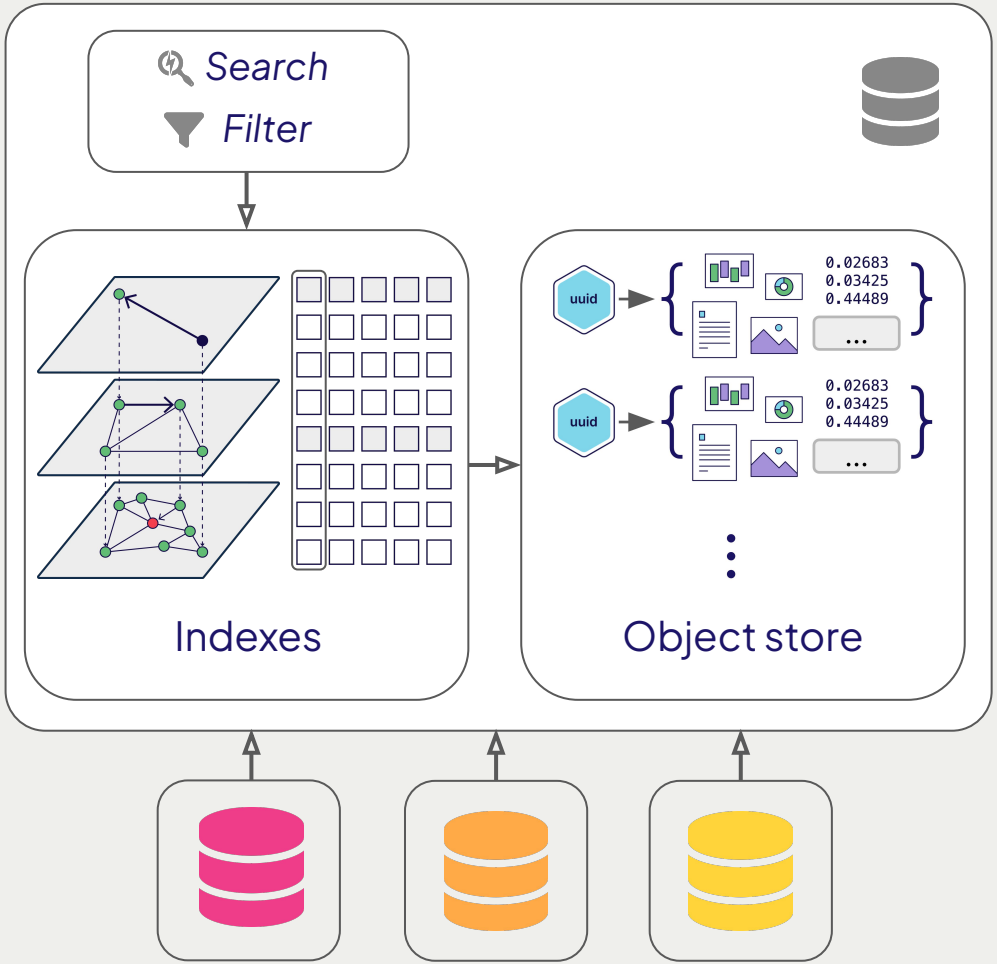
Limitation:

- Scale
  - More data
  - Multiple datasets



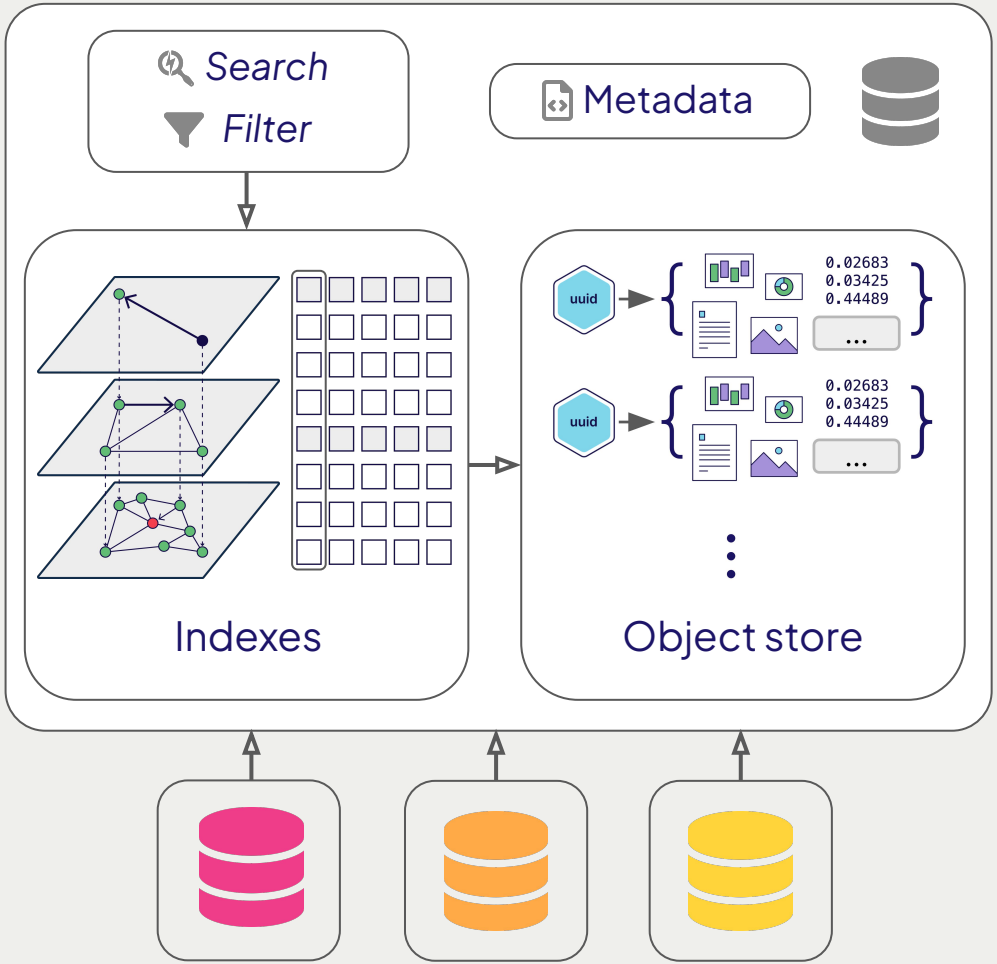
Mini-database:

- What if they could be **multiplied?**



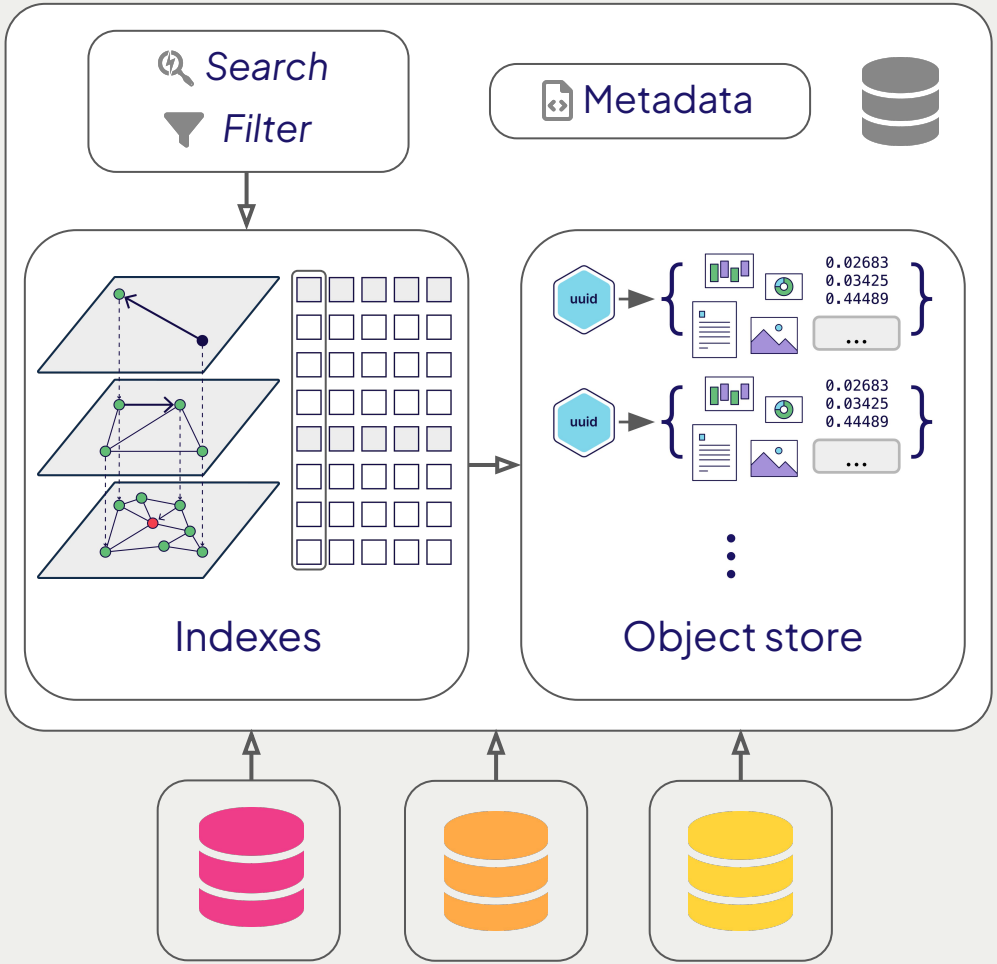
Mini-database:

- What if they could be **multiplied?**
- Could be added across multiple devices



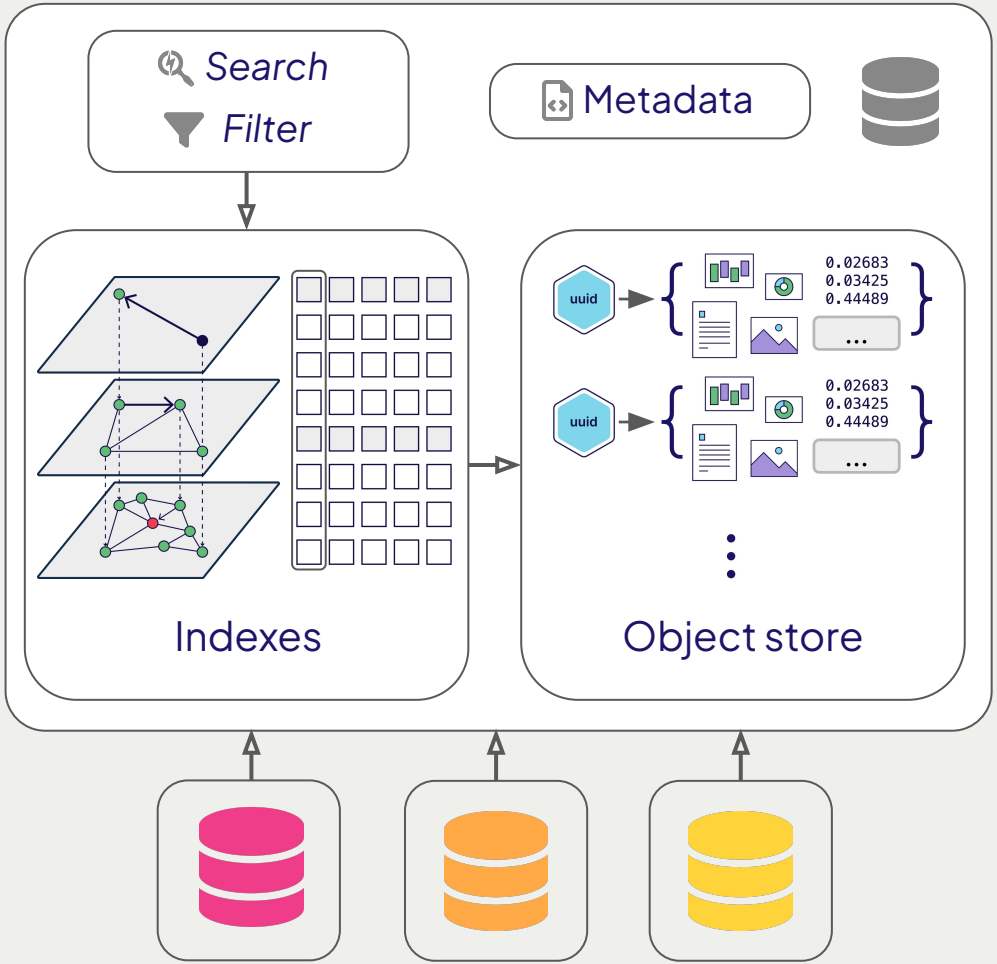
Mini-database:

- What if they could be **multiplied?**
- Could be added across multiple devices
- Called “shards” in distributed databases



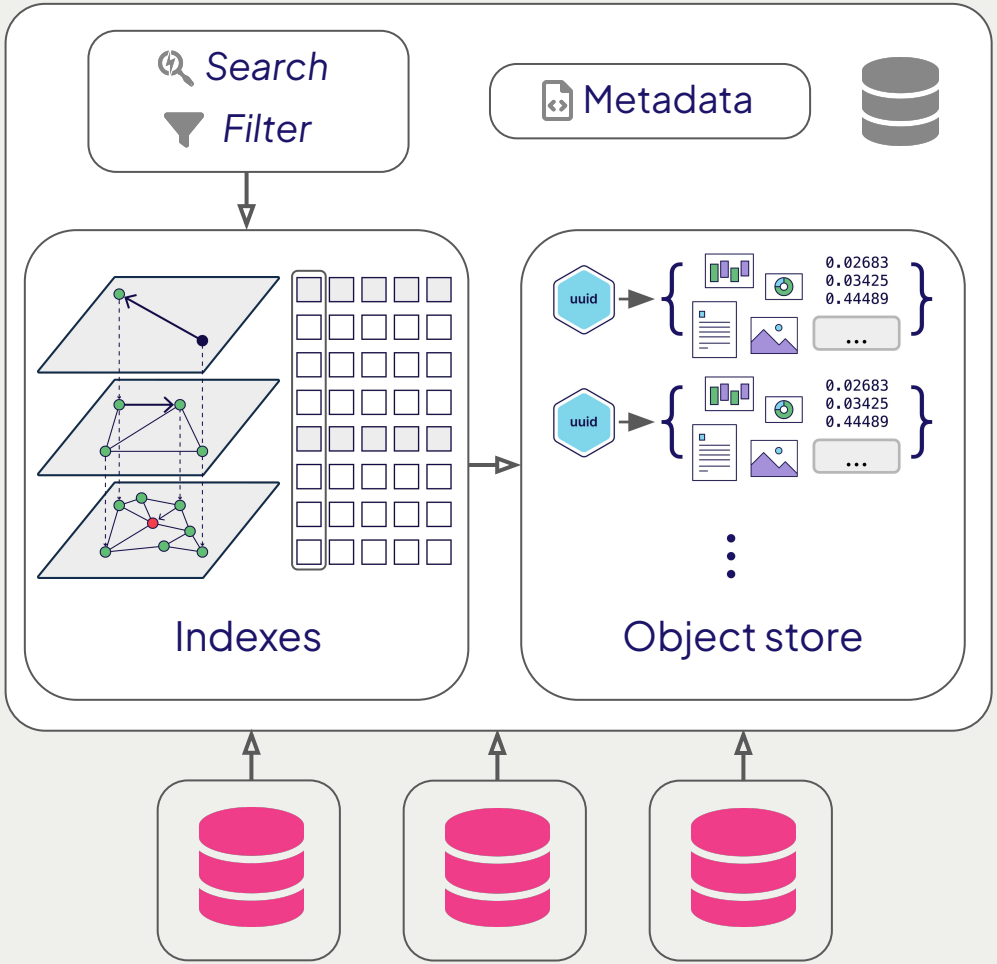
Mini-database:

- What if they could be **multiplied?**
- Could be added across multiple devices
- Called “shards” in distributed databases
- Solves a lot of issues!



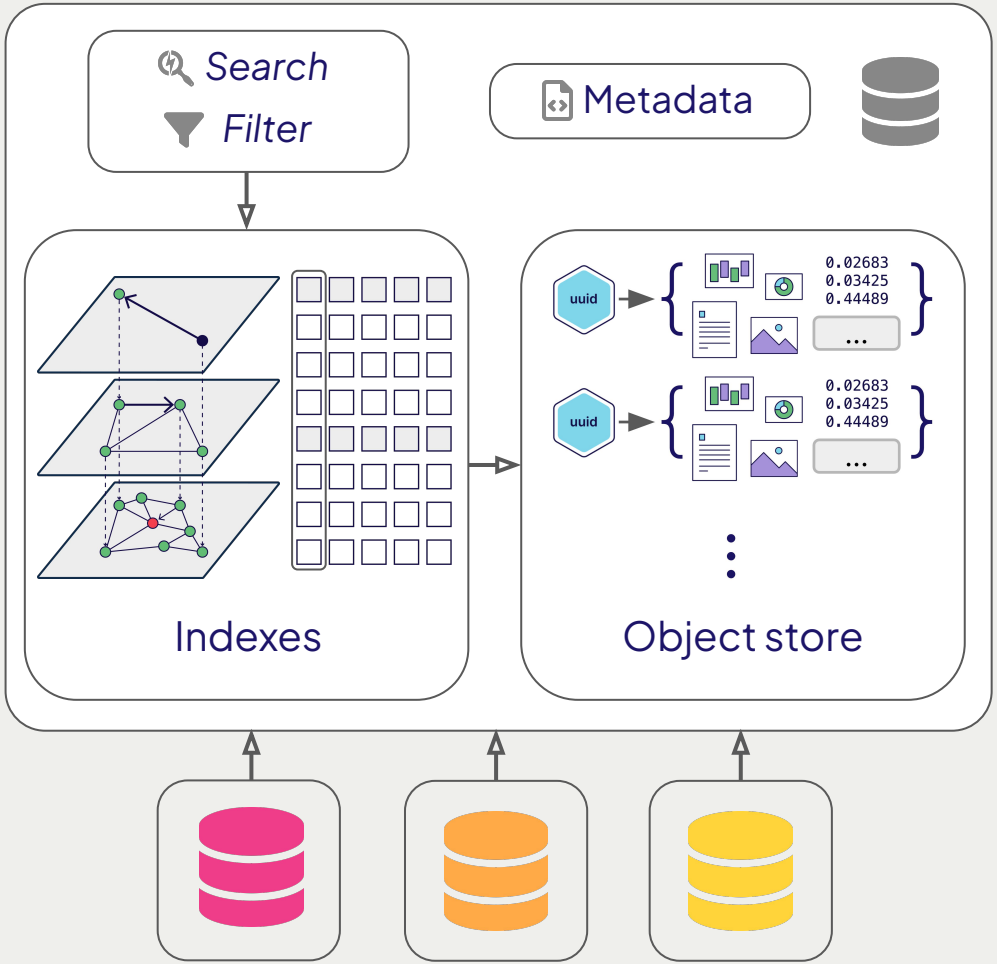
Each shard can:

- Be part of a collection
  - Horizontal scaling
- Each search goes to every shard
- Results combined



Each shard can:

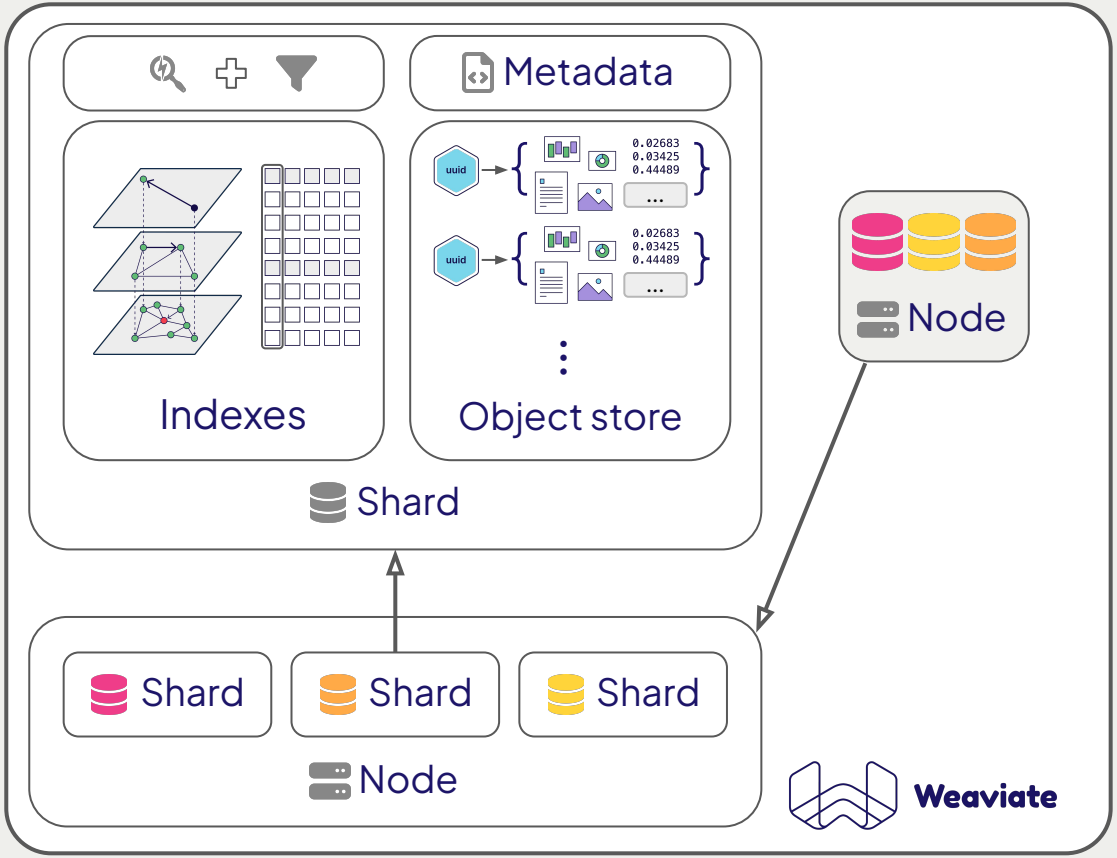
- Have “clones”
  - Replication
- Coordinate database schema
- Keep objects consistent



Each shard has:

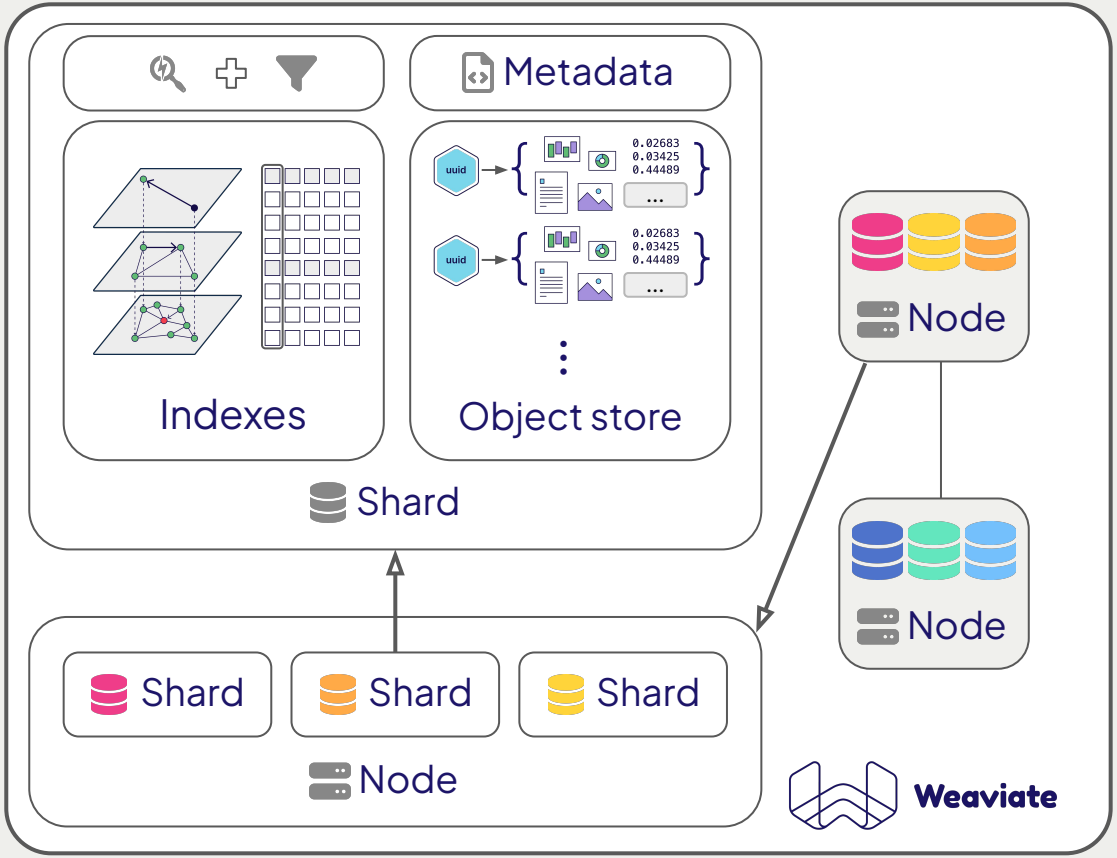
- An object store
- A vector index
- Inverted indexes
  - Searchable (keyword)
  - Filterable

How do we get BIGGER?



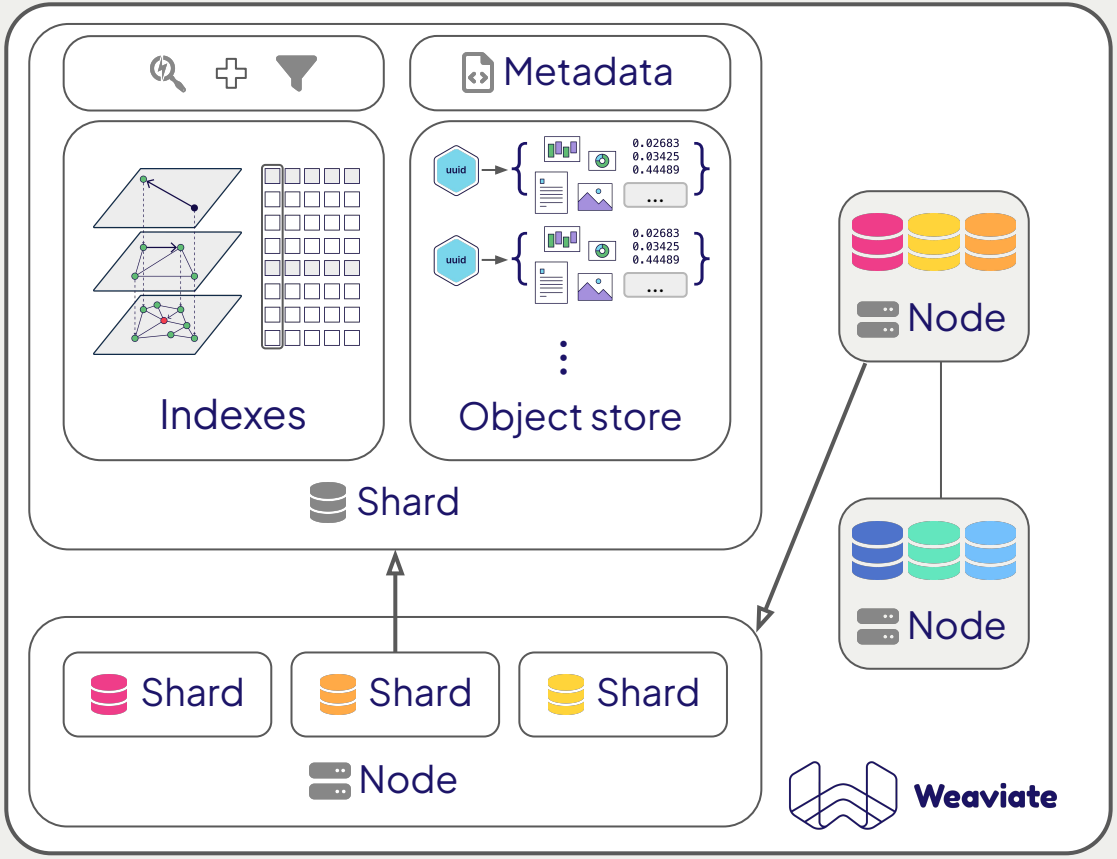
Each node:

- Can have multiple shards



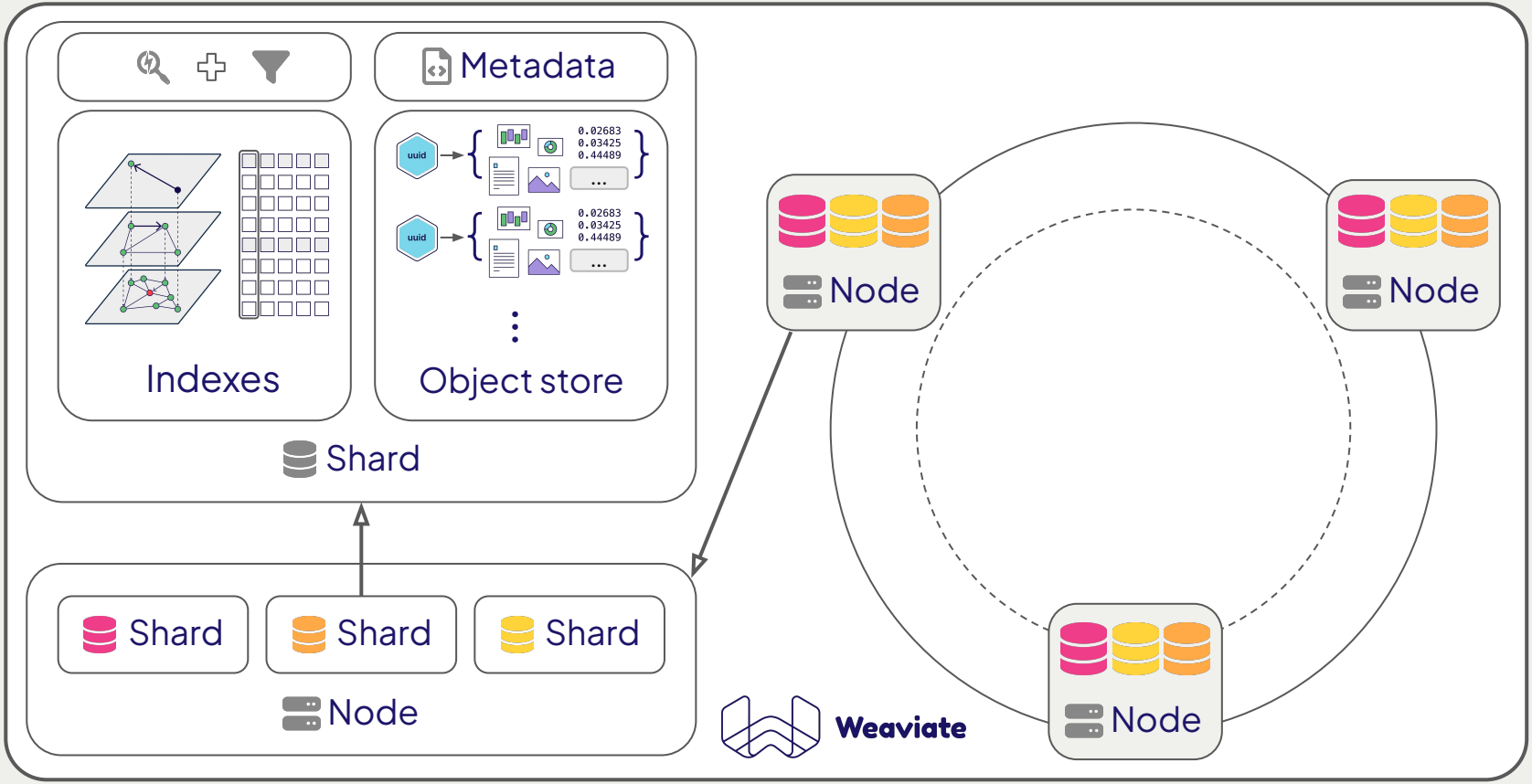
Each node:

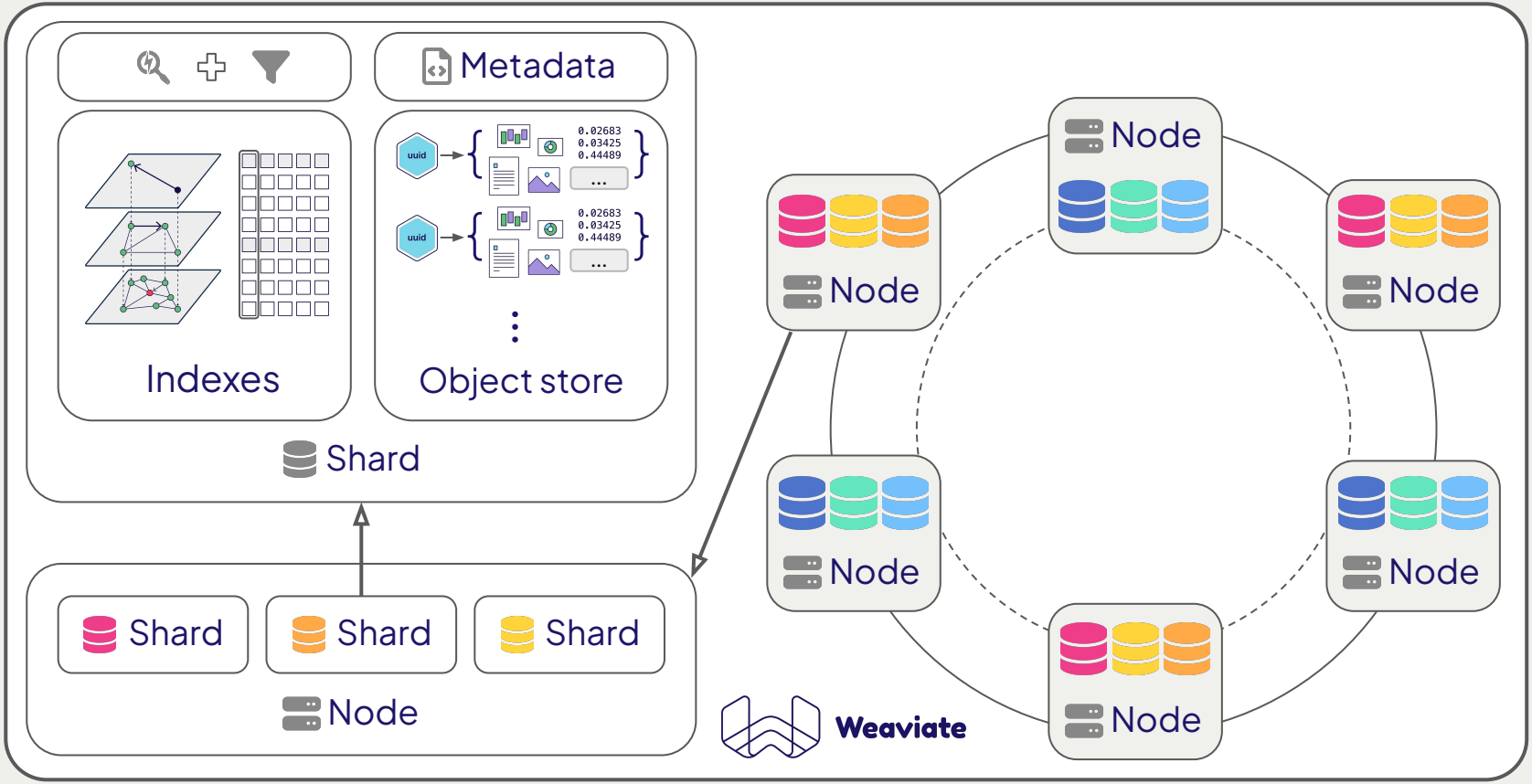
- Can have multiple shards
- Can also talk to other nodes

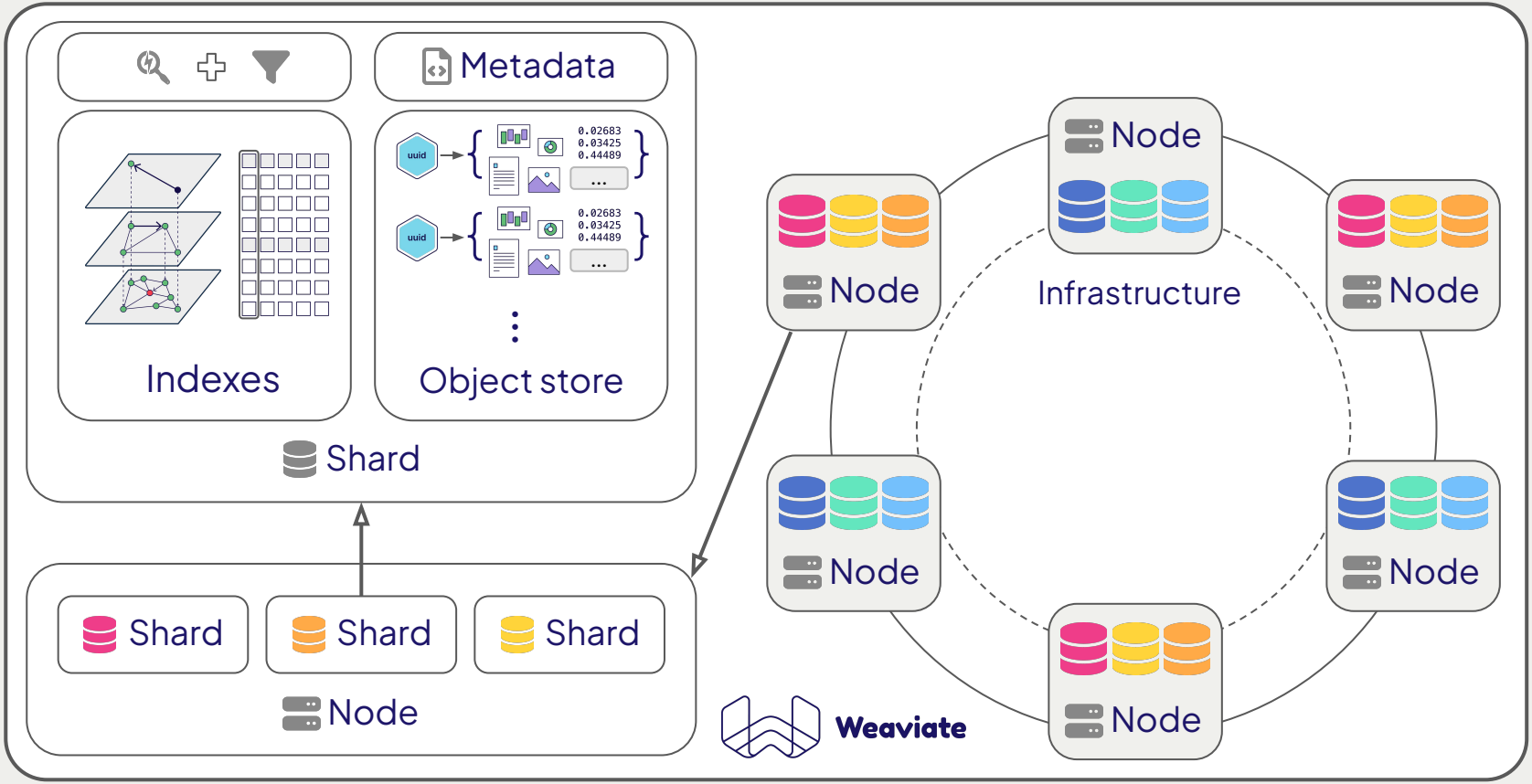


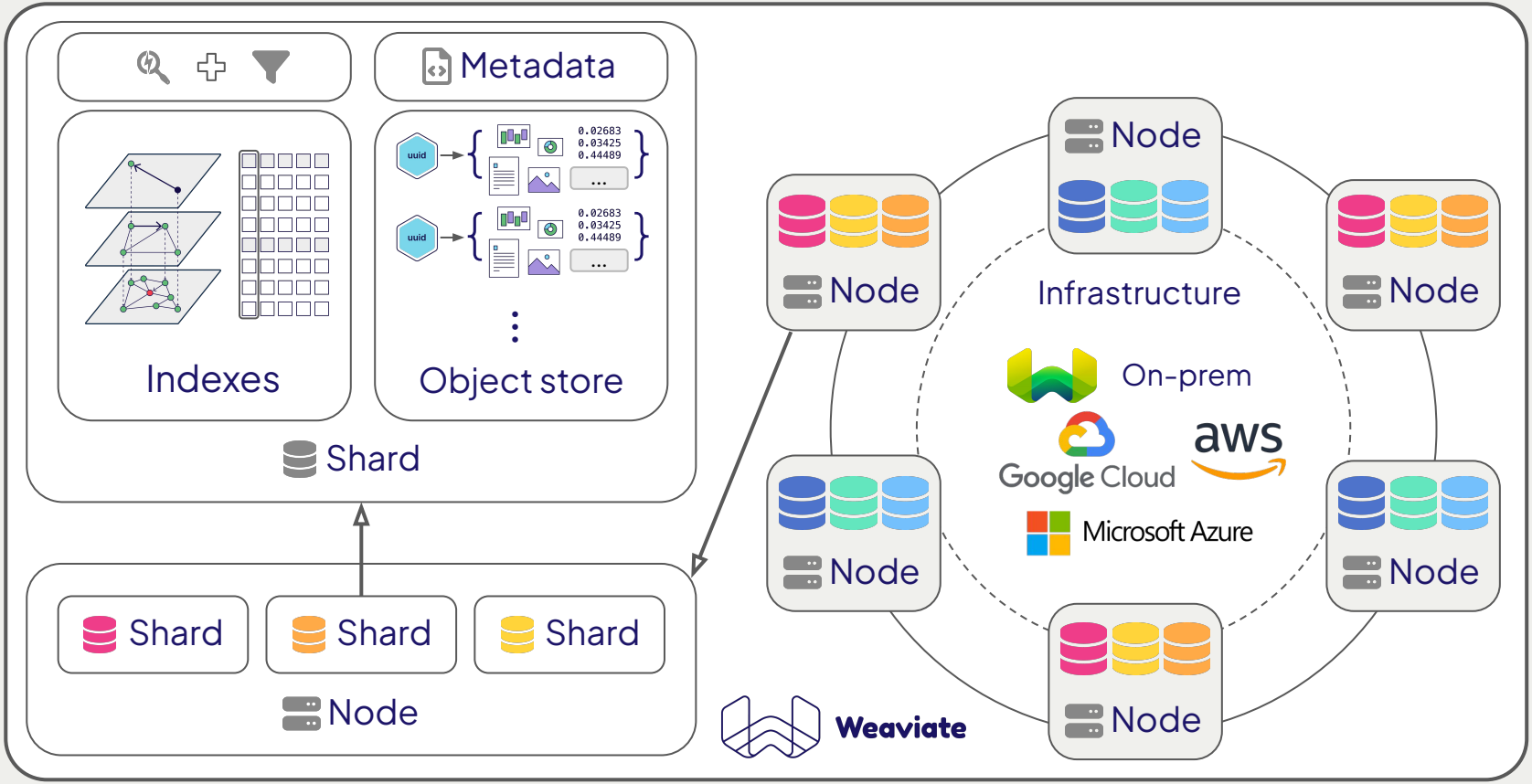
User perspective:

- Resource planning
  - How many replicas?
  - How many objects?
  - What/any compression?


Metadata
Indexes
uuid
0.02683
0.03425
0.44489
...
uuid
0.02683
0.03425
0.44489
...
Object store
Shard
Shard
Shard
Shard
Node
Node
Node
Node
Weaviate

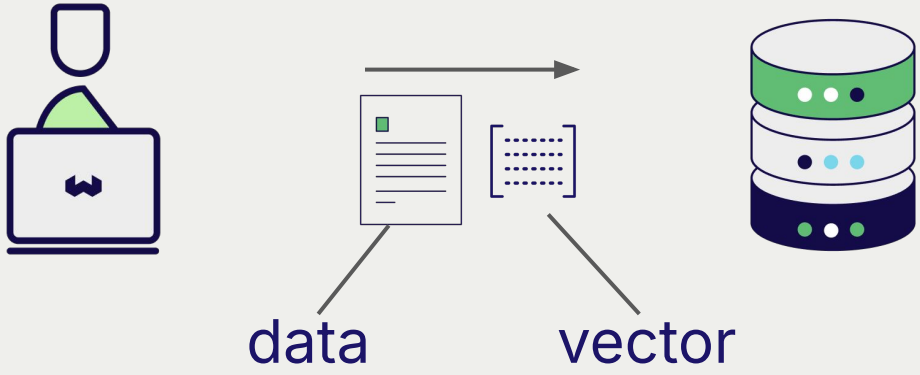

Metadata
Indexes
uuid
0.02683
0.03425
0.44489
...
uuid
0.02683
0.03425
0.44489
...
Object store
Shard
Shard
Shard
Shard
Node
Node
Node
Node
Node
Node
Node
Weaviate


Metadata
Indexes
Object store
uuid
0.02683
0.03425
0.44489
...
Shard
Shard
Shard
Shard
Node
Node
Infrastructure
Node
Node
Node
Node
Node
Weaviate

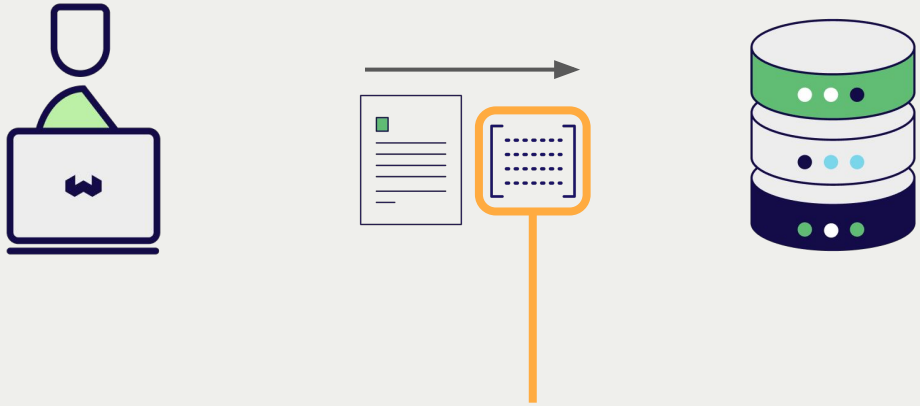

Metadata
Indexes
uuid
0.02683
0.03425
0.44489
...
Object store
Shard
Shard
Shard
Shard
Node
Node
Infrastructure
Node
Node
Node
Node
Node
On-prem
Google Cloud
aws
Microsoft Azure
Weaviate

But wait… there’s more!

Ingest data

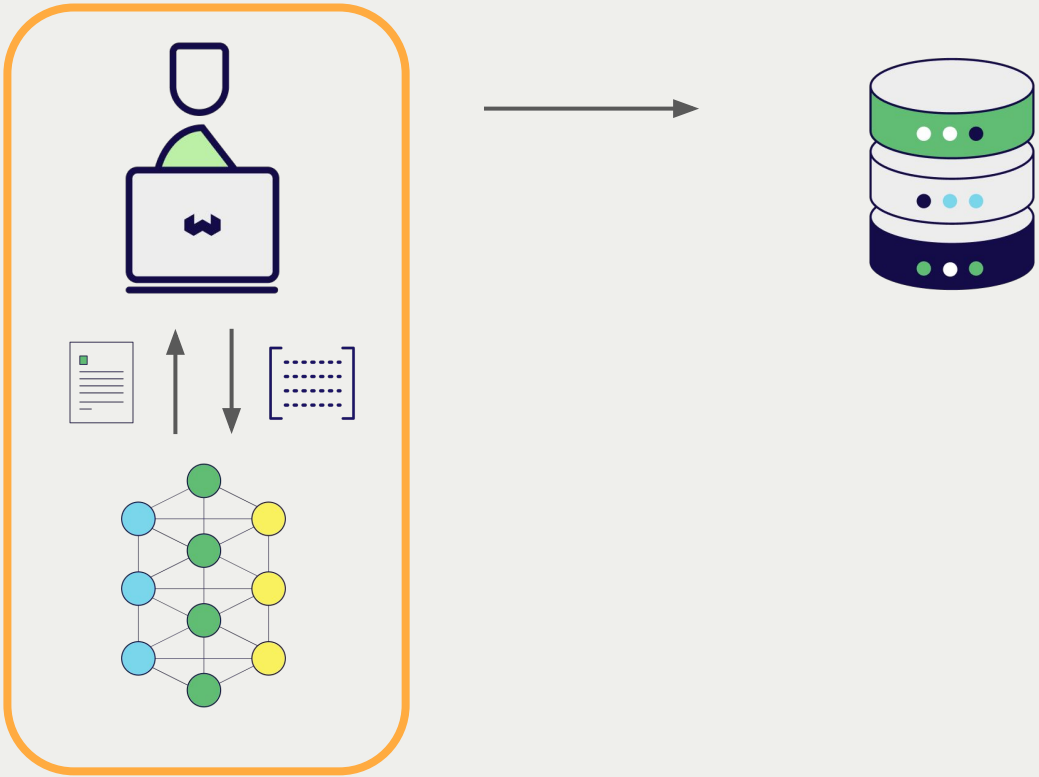


Ingest data

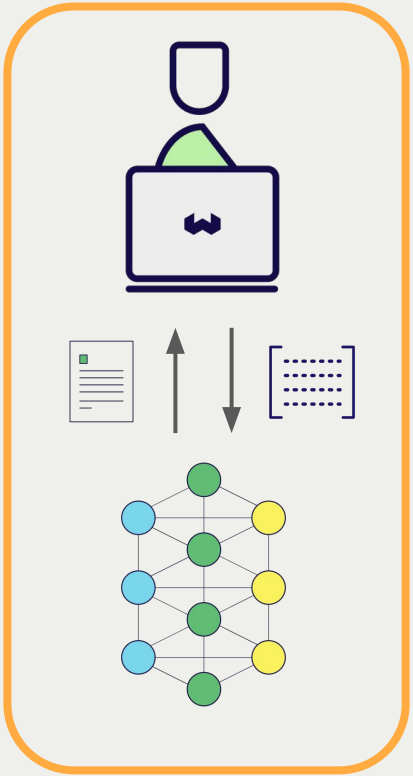


Where do they come from??

Obtain Embeddings

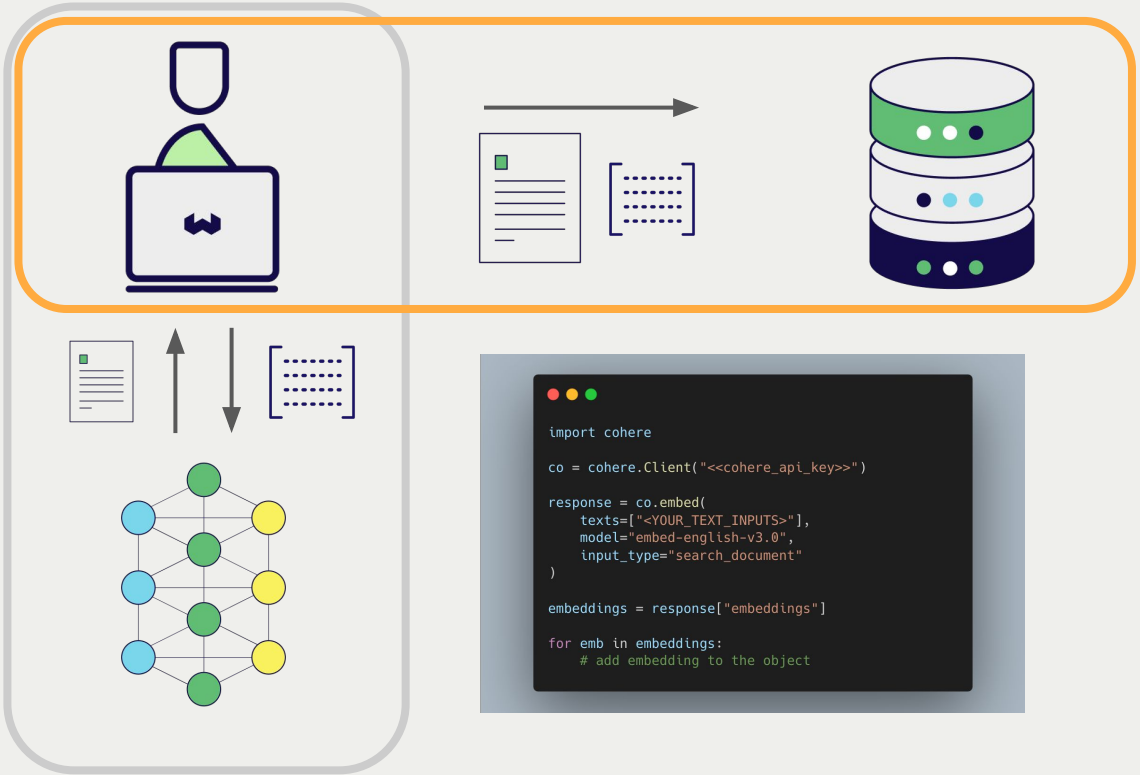


Obtain Embeddings



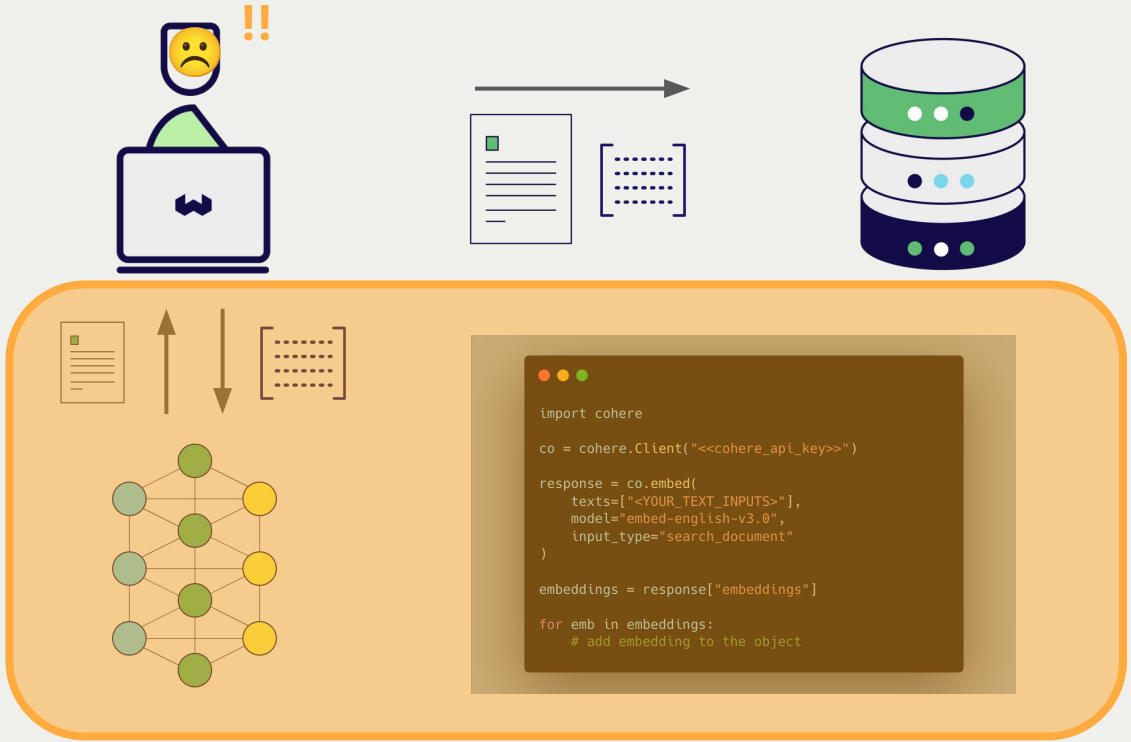
```python
import cohere

co = cohere.Client("<<cohere_api_key>>")

response = co.embed(
    texts=["<YOUR_TEXT_INPUTS>"],
    model="embed-english-v3.0",
    input_type="search_document"
)

embeddings = response["embeddings"]

for emb in embeddings:
    # add embedding to the object
```

Obtain Embeddings

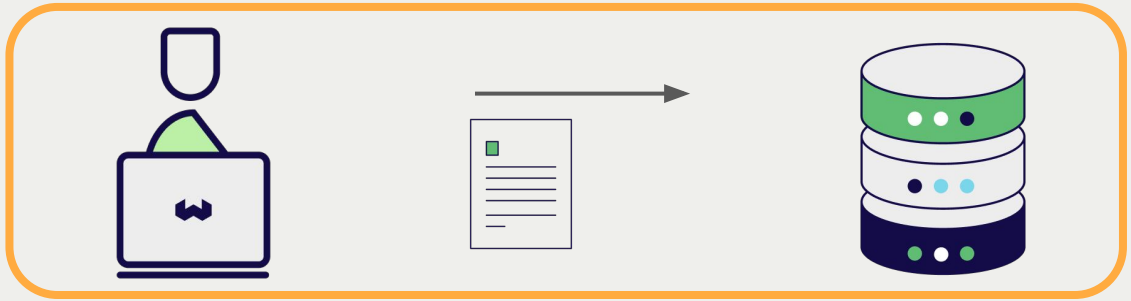


```python
import cohere

co = cohere.Client("<<cohere_api_key>>")

response = co.embed(
    texts=["<YOUR_TEXT_INPUTS>"],
    model="embed-english-v3.0",
    input_type="search_document"
)

embeddings = response["embeddings"]

for emb in embeddings:
    # add embedding to the object
```

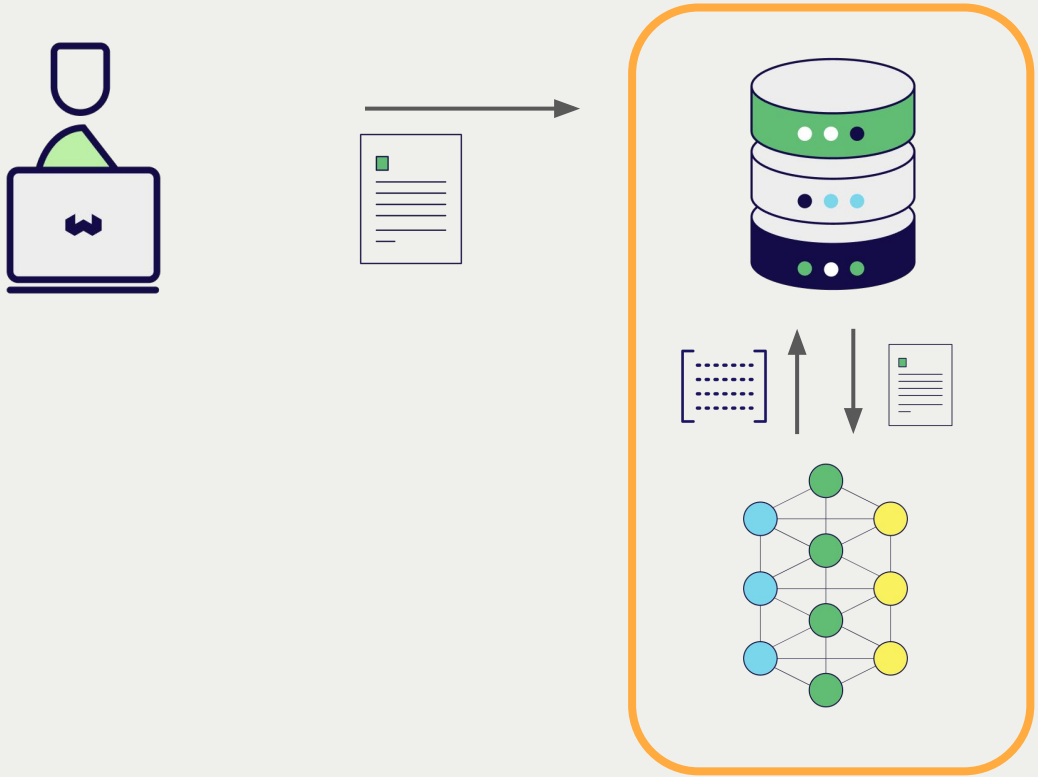
Overhead


!!
import cohere
co = cohere.Client("<<cohere_api_key>>")
response = co.embed(
texts=["<YOUR_TEXT_INPUTS>"],
model="embed-english-v3.0",
input_type="search_document"
)
embeddings = response["embeddings"]
for emb in embeddings:
# add embedding to the object

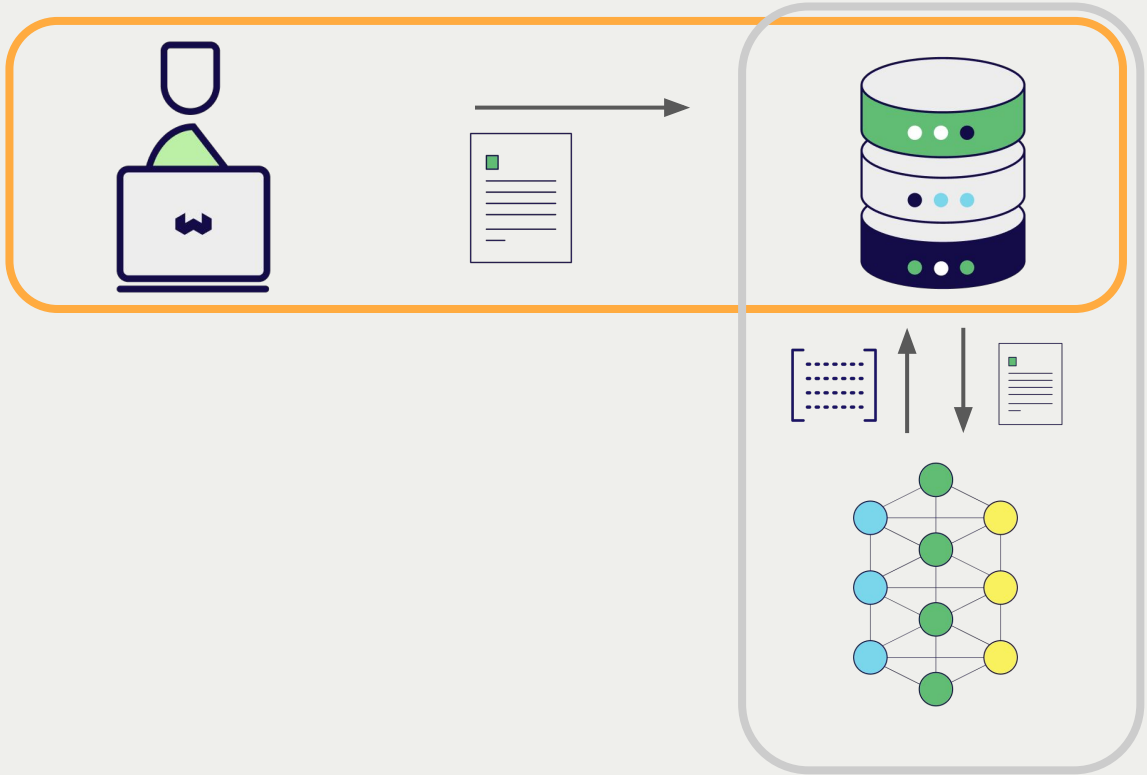
Ingest data



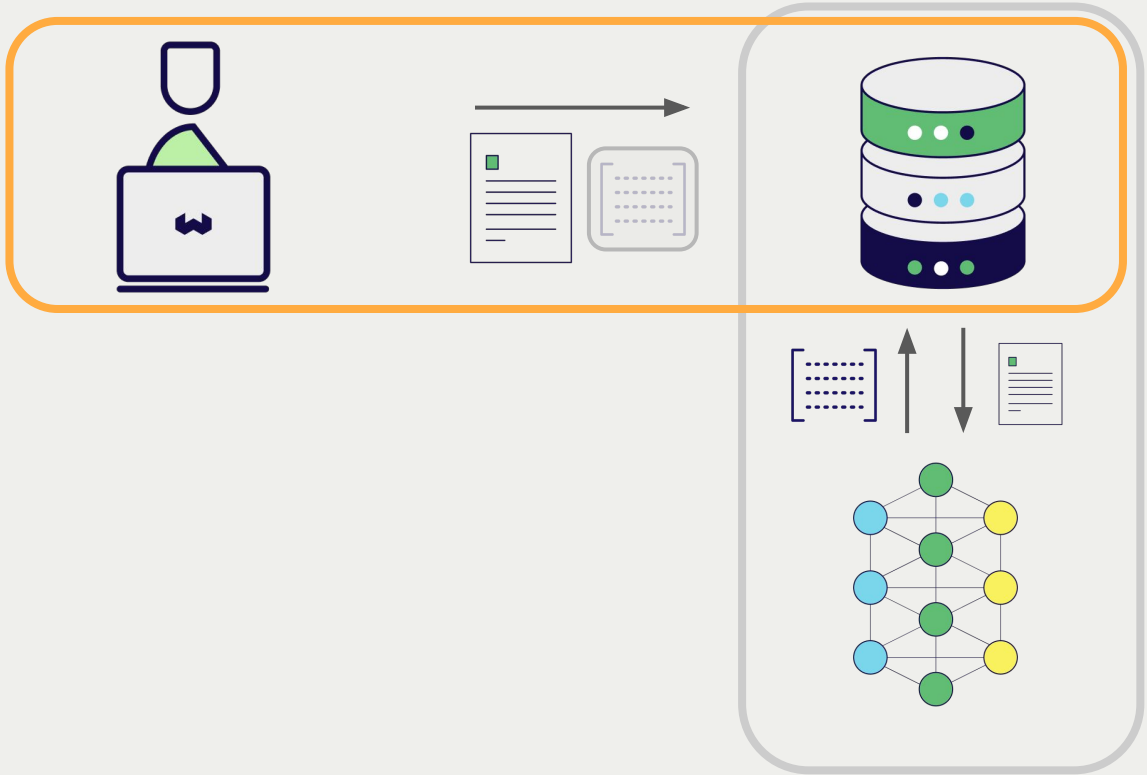
Have DB obtain embeddings



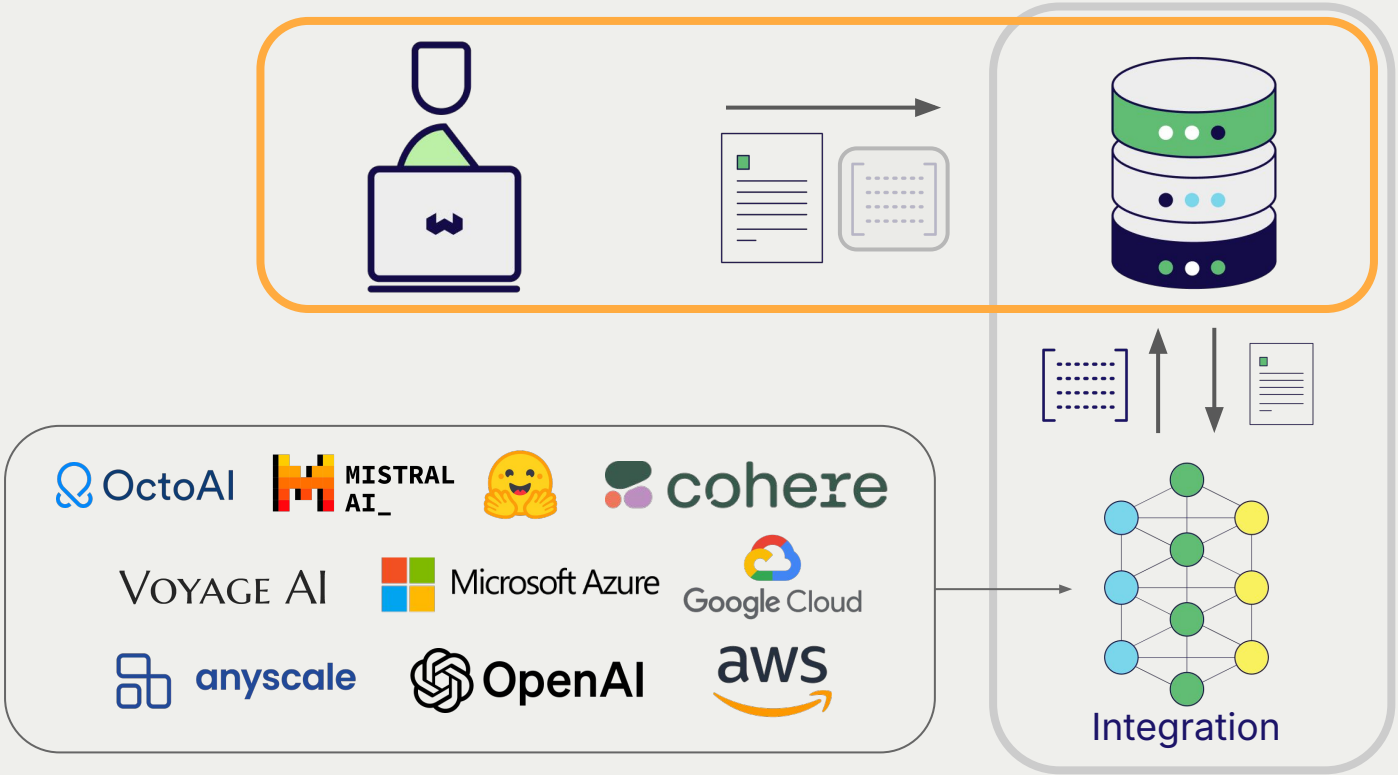
Ingest data (embeddings in background)



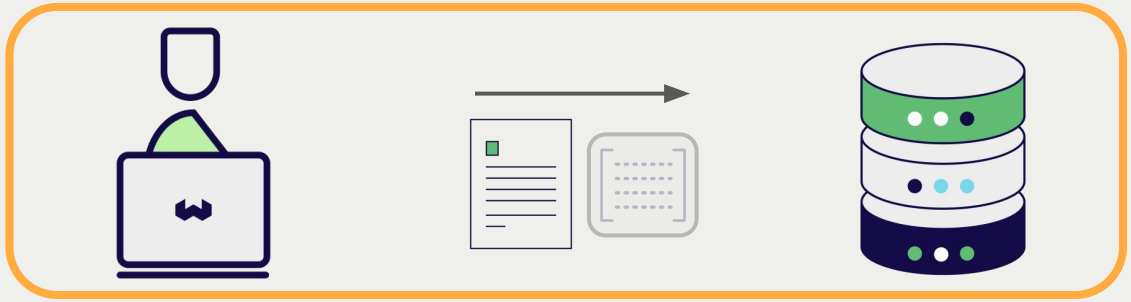
Ingest data (embeddings optional)



Ingest data (with provider integrations)



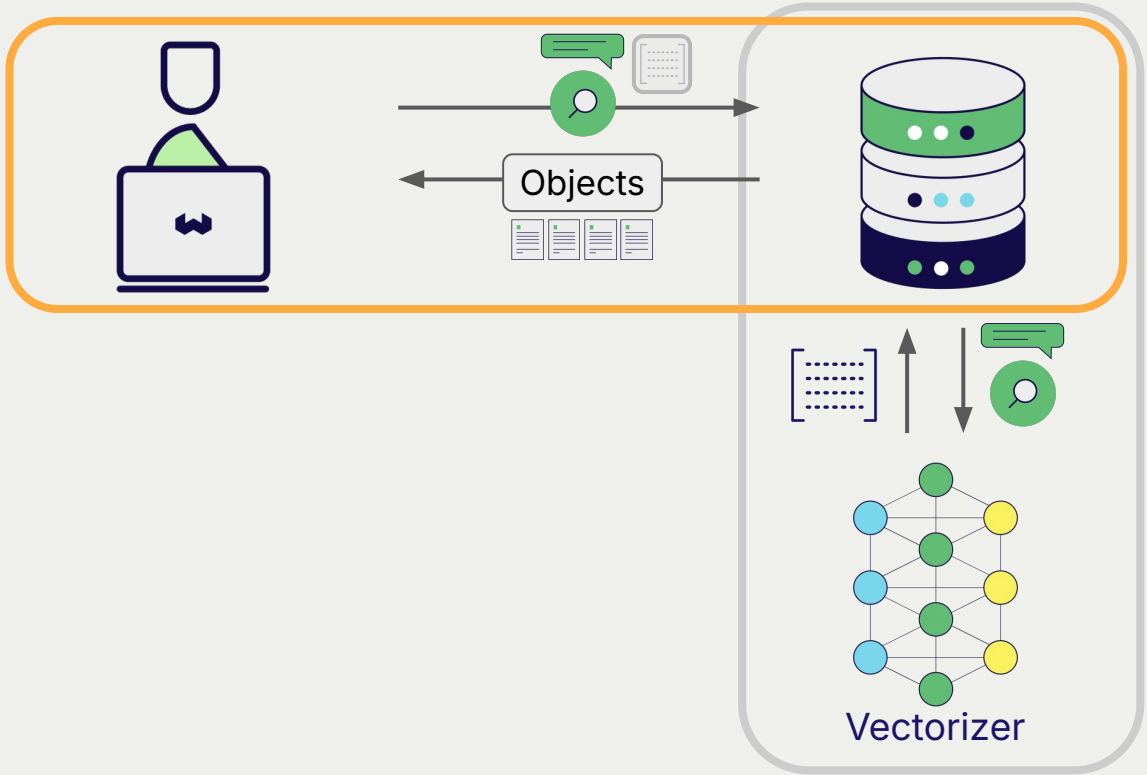
Simplified Experience

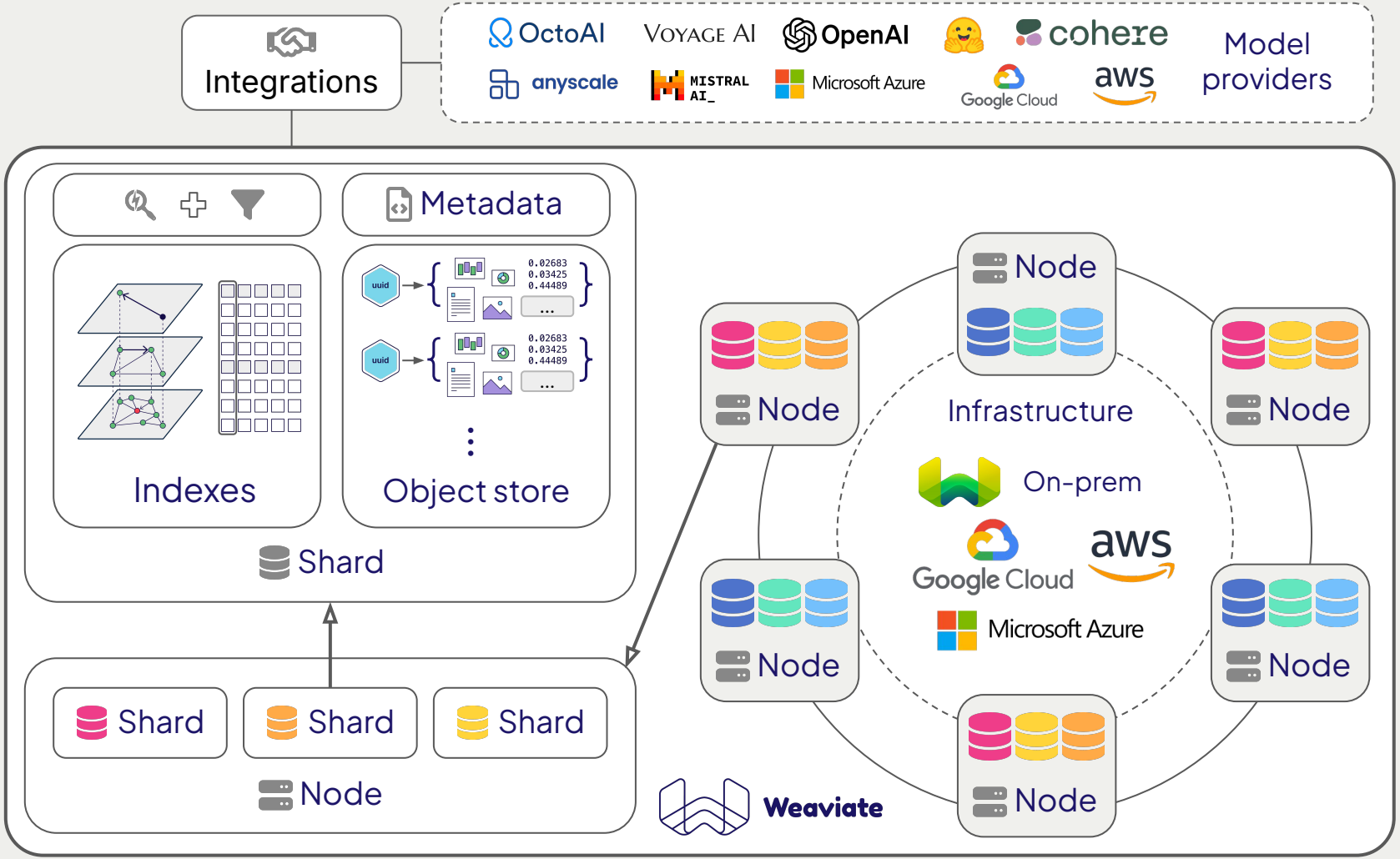


User experience

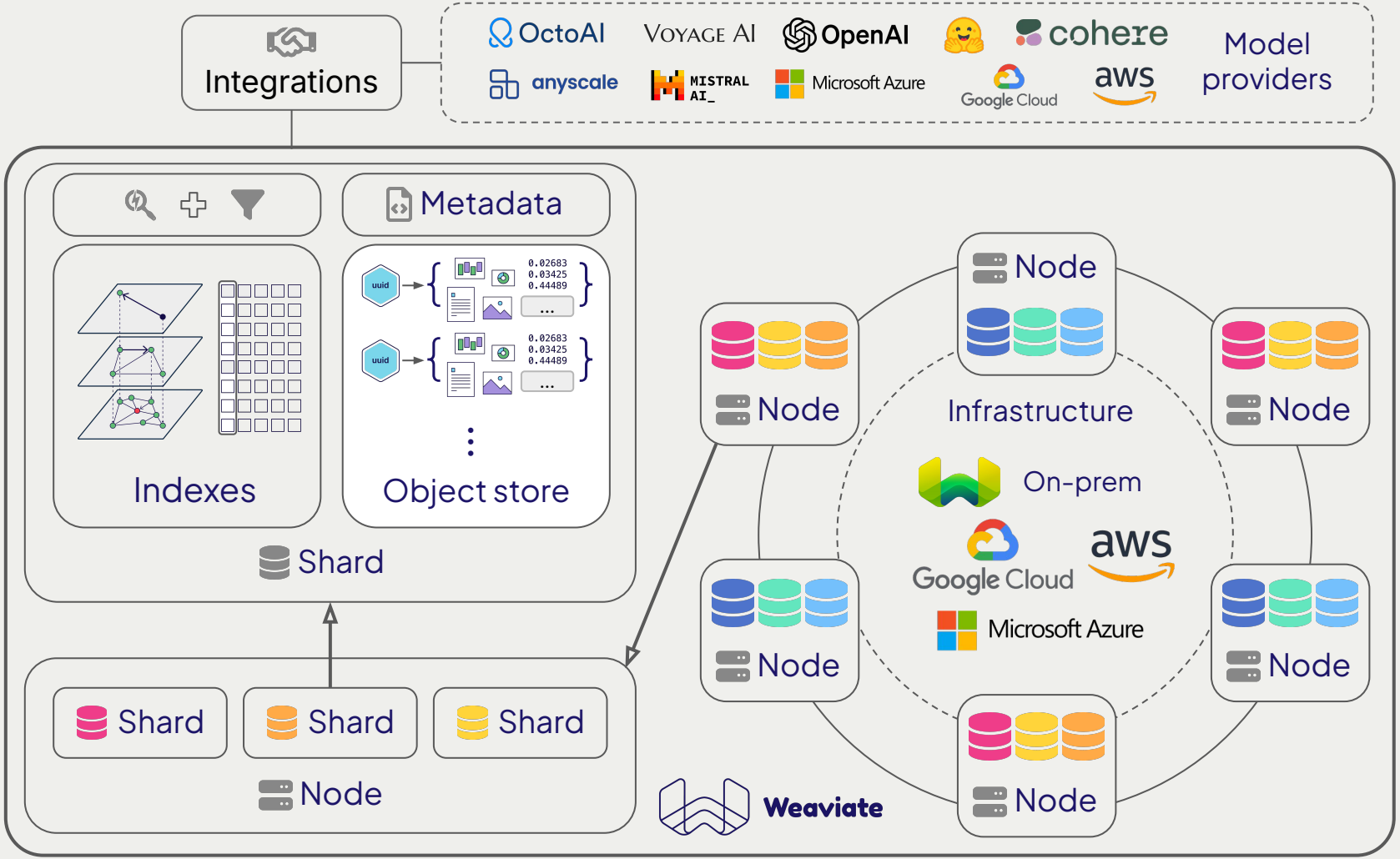
Under the hood

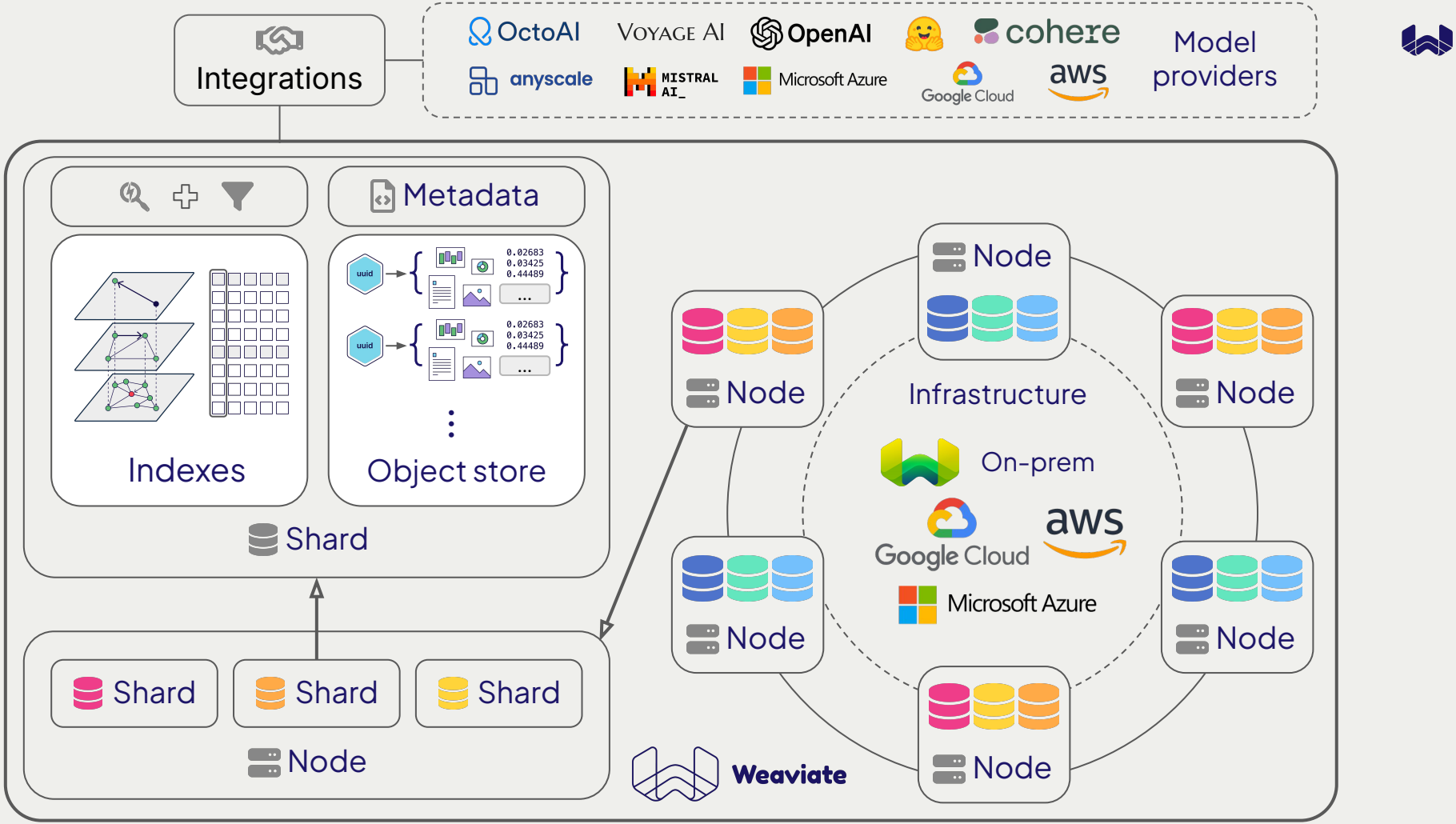
Query

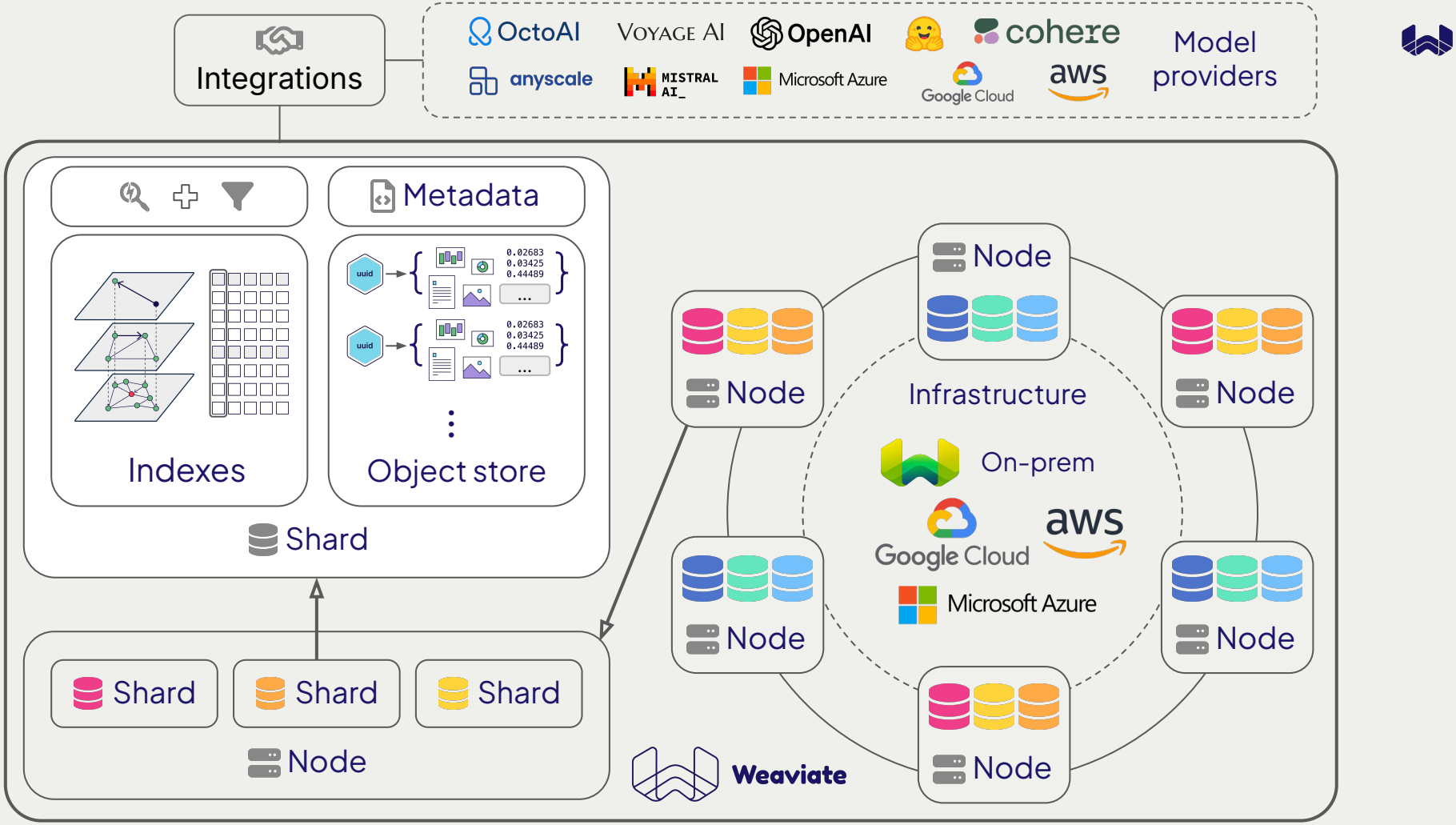


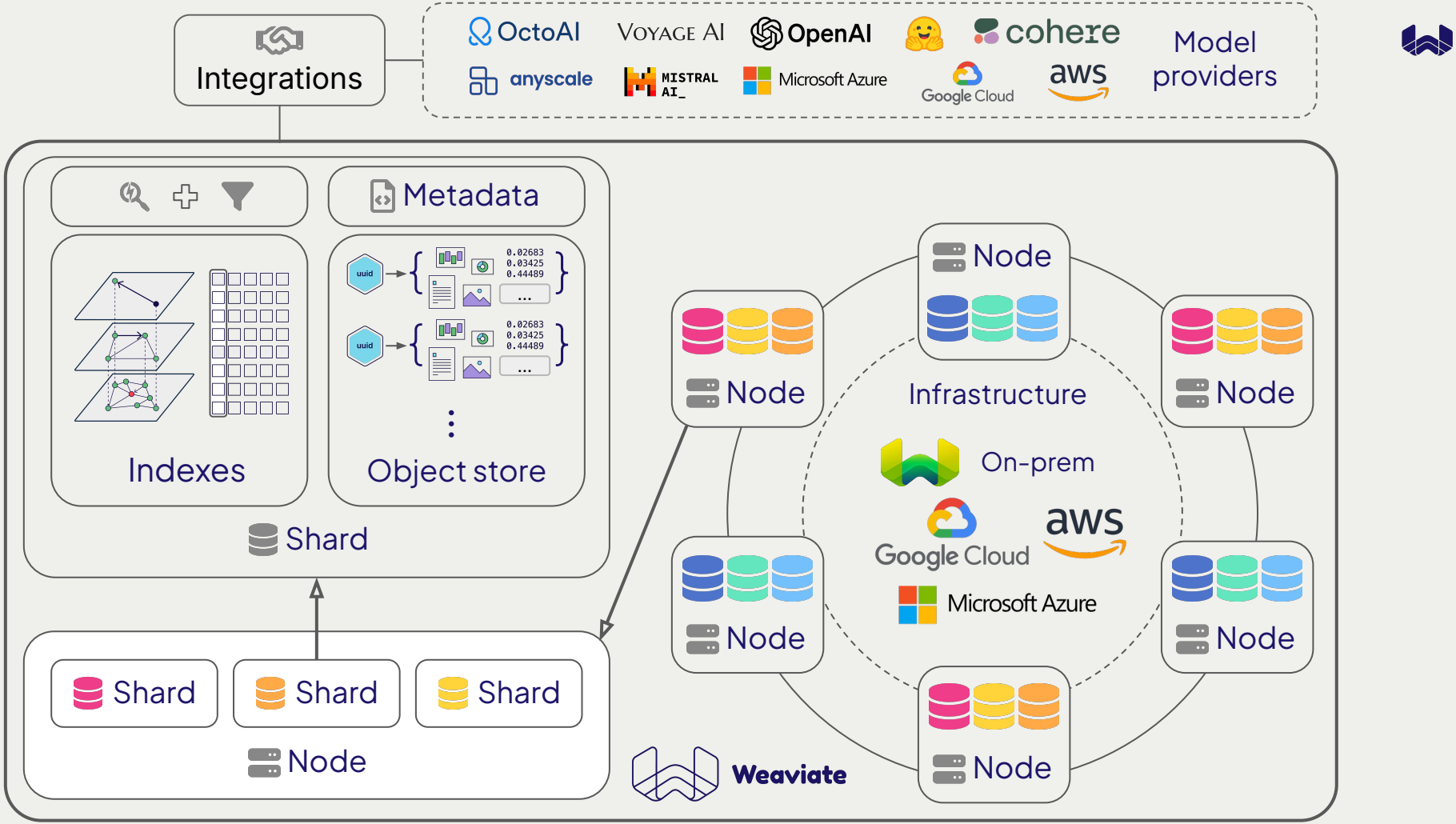

Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
Indexes
Object store
uuid
0.02683
0.03425
0.44489
Shard
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure
Weaviate

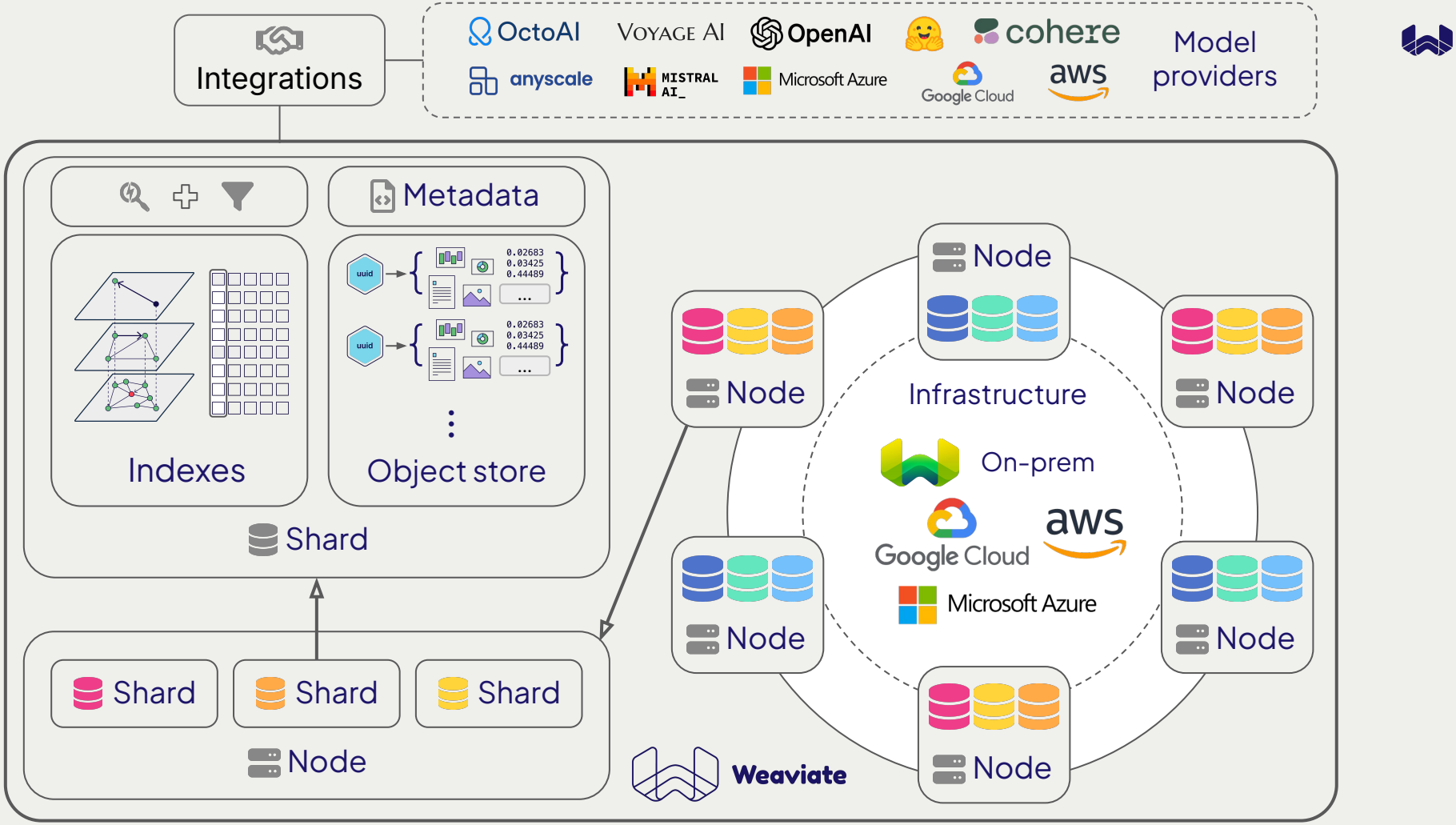
What have we built? (Full recap)

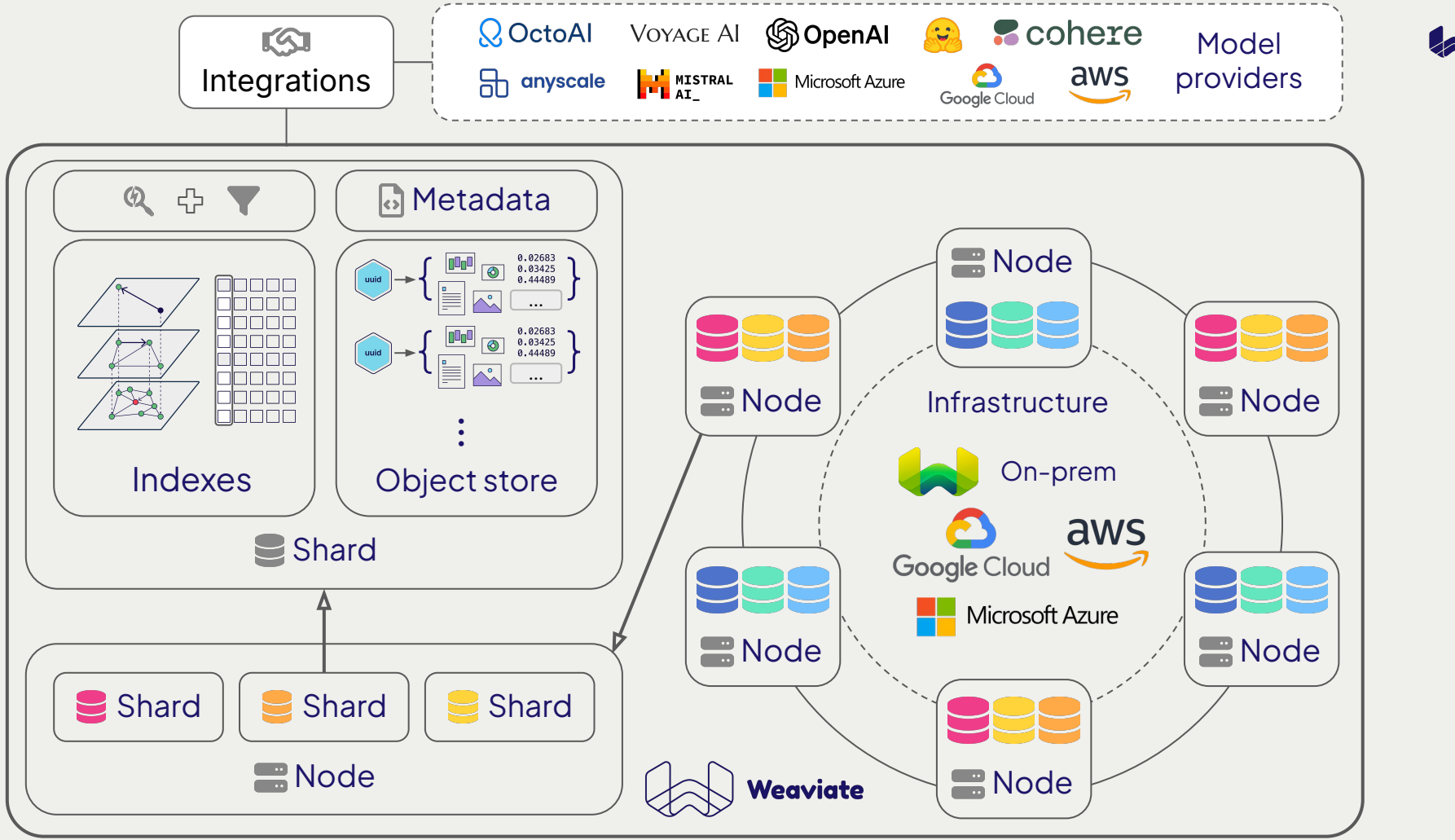

Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
Indexes
Object store
uuid
0.02683
0.03425
0.44489
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure

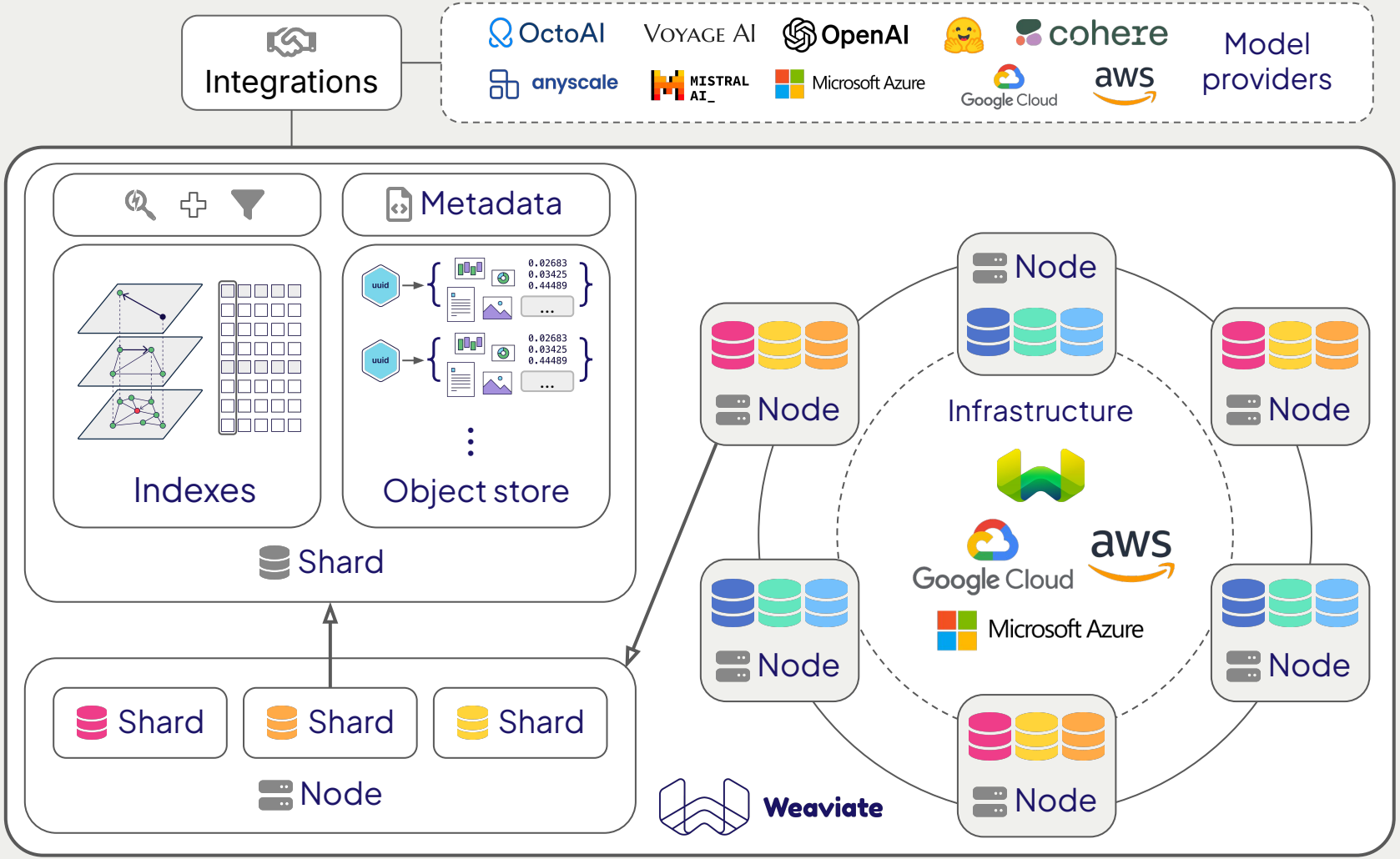

Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
Indexes
Object store
uuid
0.02683
0.03425
0.44489
Shard
Shard
Shard
Shard
Node
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure
Weaviate

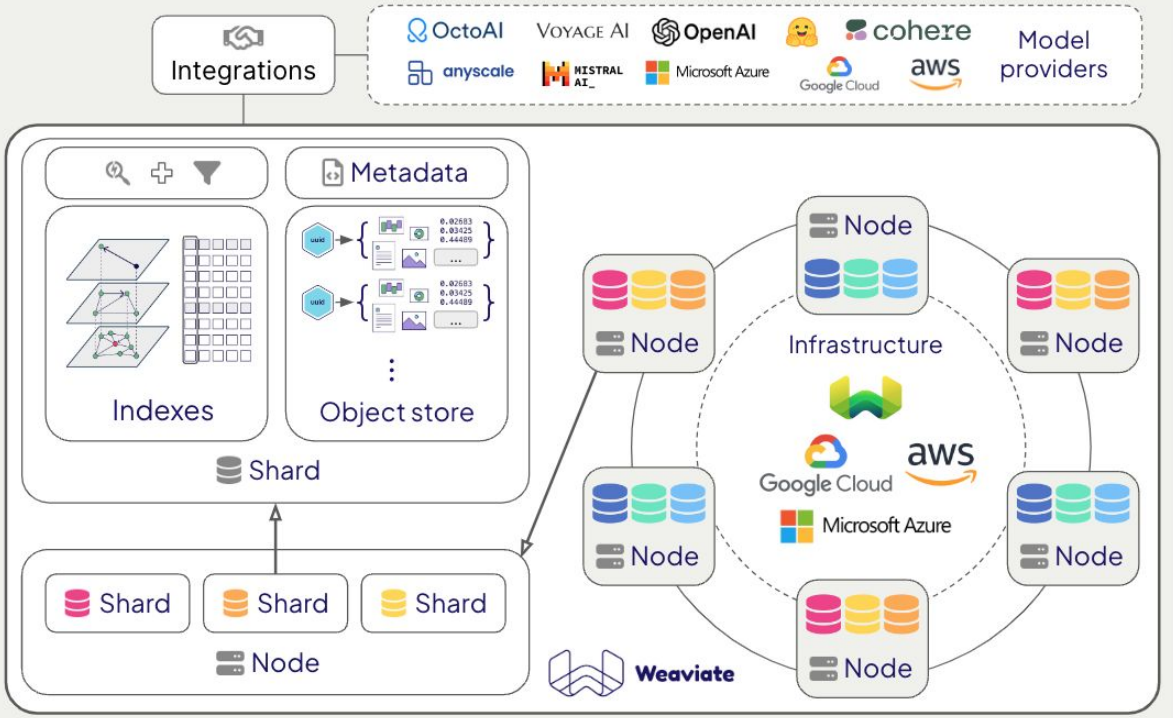

Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
uuid
0.02683
0.03425
0.44489
...
Indexes
Object store
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure


Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
uuid
0.02683
0.03425
0.44489
...
Indexes
Object store
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure


Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
uuid
0.02683
0.03425
0.44489
...
Indexes
Object store
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure


Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
uuid
0.02683
0.03425
0.44489
...
Indexes
Object store
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
On-prem
Google Cloud
aws
Microsoft Azure


Integrations
OctoAI
Voyage AI
OpenAI
cohere
anyscale
Mistral AI_
Microsoft Azure
Google Cloud
aws
Model providers
Metadata
uuid
0.02683
0.03425
0.44489
...
Indexes
Object store
Shard
Shard
Shard
Shard
Node
Weaviate
Node
Node
Node
Node
Node
Node
Infrastructure
Google Cloud
aws
Microsoft Azure



Try

https://hubs.ly/Q02tMNDH0