Bengaluru | Feb 29 2024

PGConf India, 2024

Unlocking the Power of Vector Similarity Search with pgvector

Sachin Khanna
Lead Database Consultant
AWS GCCI India

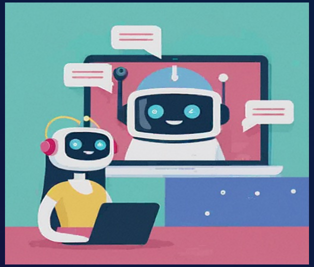
Jitender Kumar
Sr. Lead Database Consultant
AWS GCCI India

Agenda

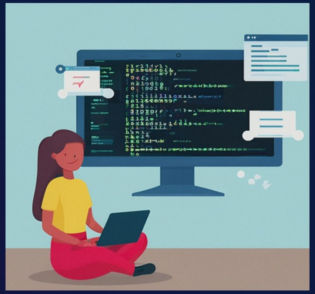
- Generative AI Overview
- The role of vectors in generative AI
- PostgreSQL – pgvector Extension
- Optimizing search with IVFFlat and HNSW indexes
- Best practices for pgvector
- Demo
- Q & A

What is Generative AI ?

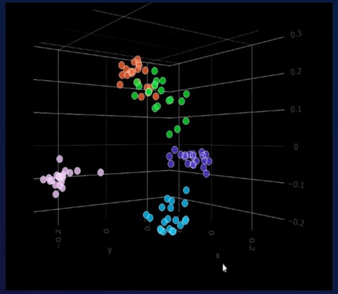
Content Generation



Chatbots and Virtual Assistant



Code Generation



3D Modelling

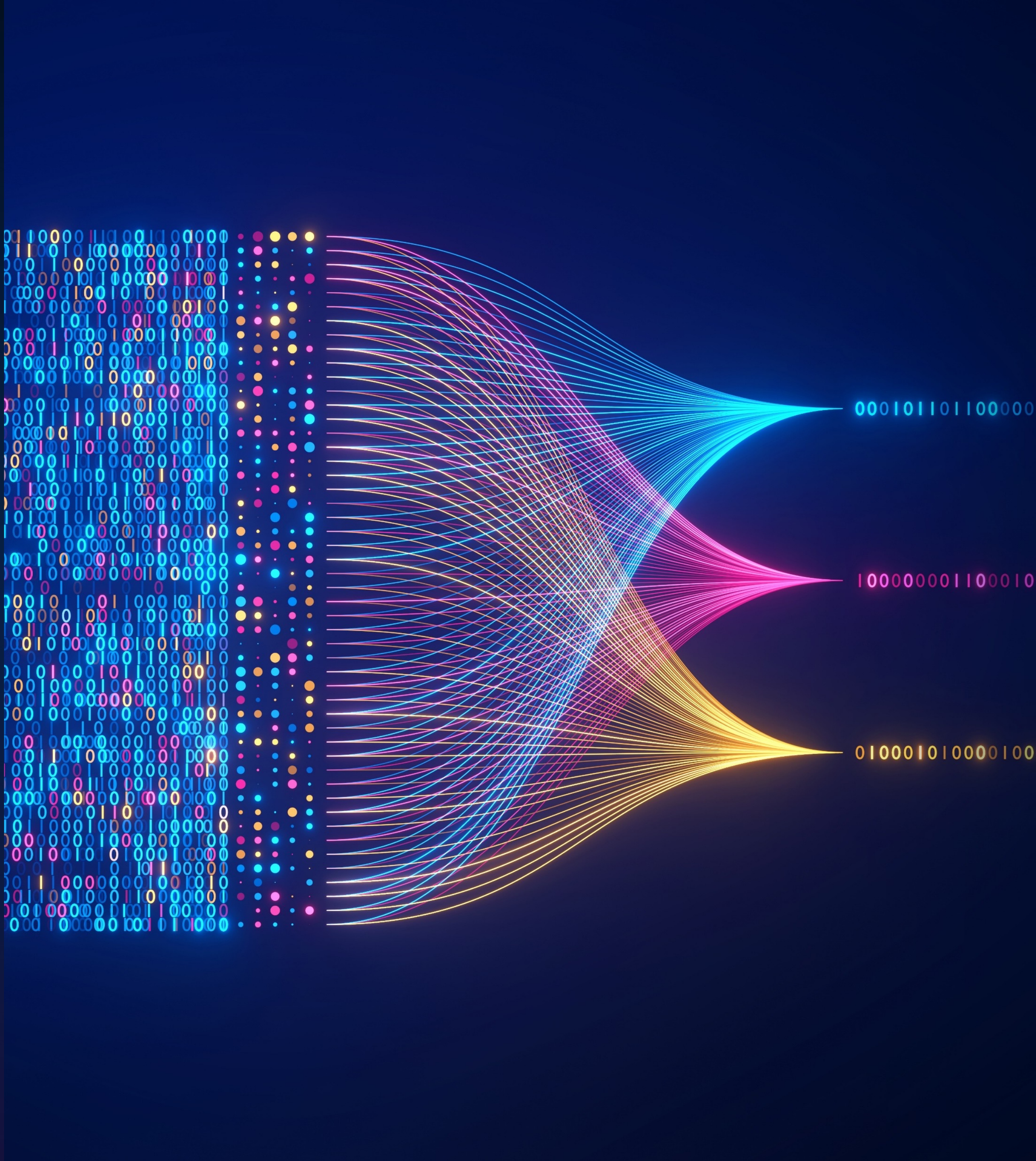
Generative AI is powered by foundation models

Pretrained on vast amounts of unstructured data

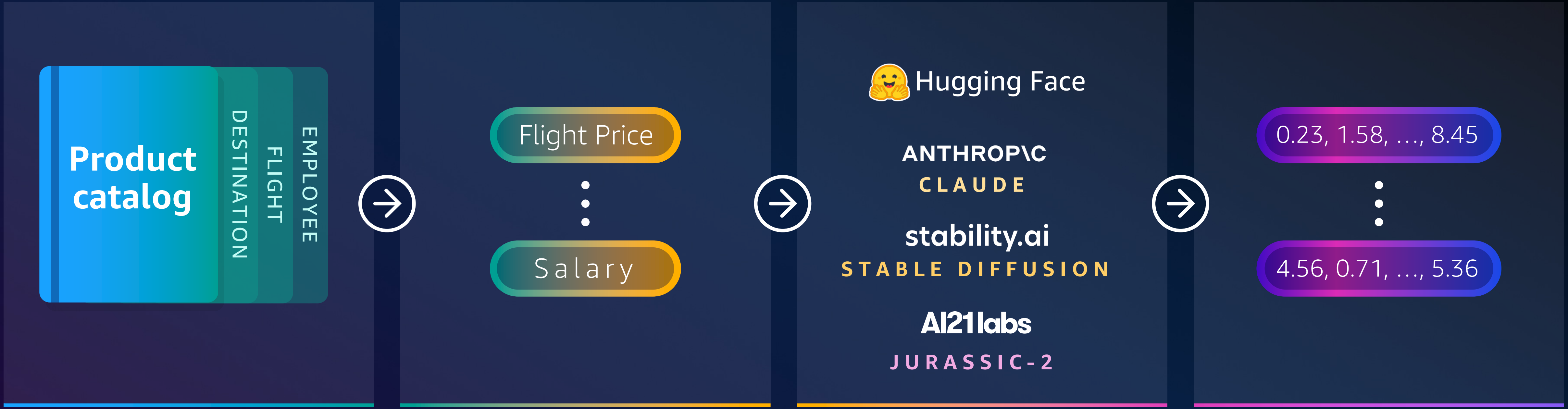
Contain large number of parameters that make them capable of learning complex concepts

Can be applied in a wide range of contexts

Customize FMs using your data for domain specific tasks

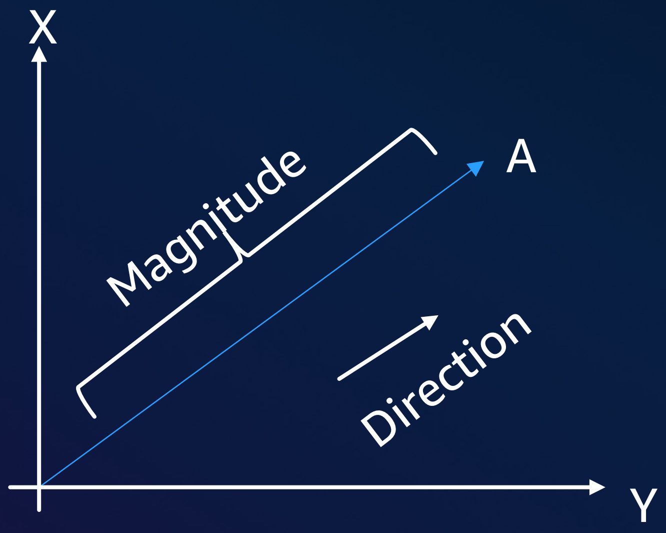


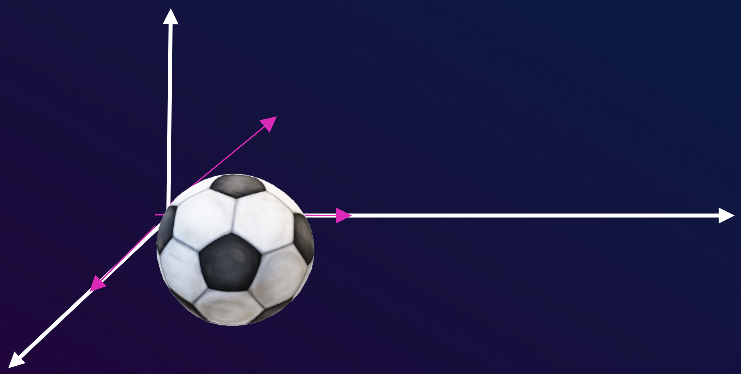
What are vector embeddings?



Unstructured data has to be vectorized into vectors to be used in generative AI applications

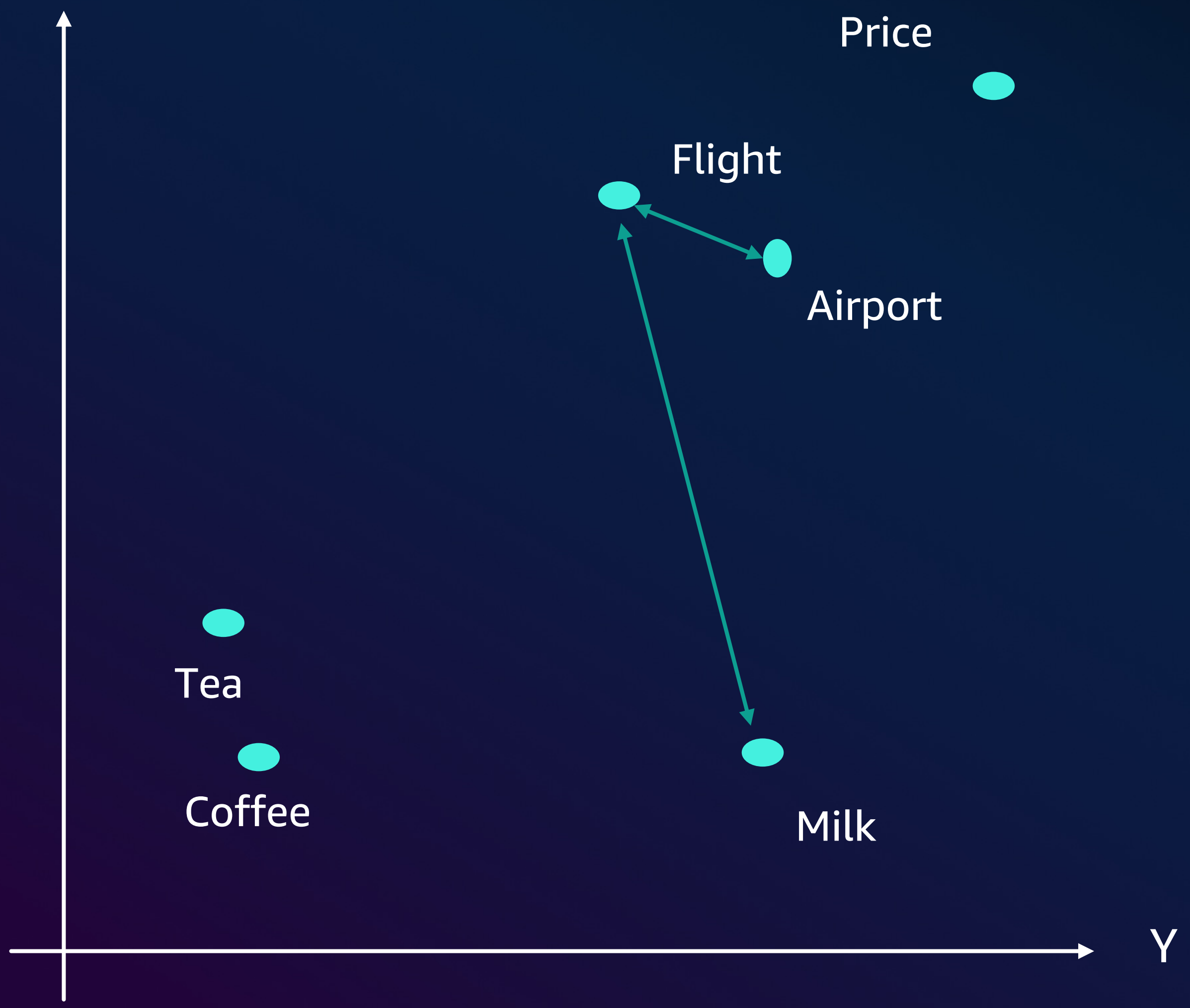
What are Vectors ?





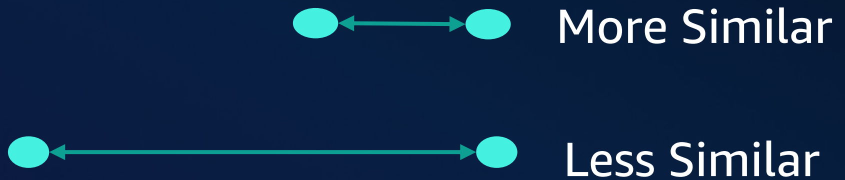
- Vector is a unit which has two dimensions: magnitude and direction.
- Example of vector are velocity , acceleration and displacement.
- Greater dimensions imply more properties and characteristics.
- In Language Model, a word like ‘flight’ could be represented as [0.2, 0.1, 0.3] in a 3-dimensional vector space.
- In case of LLMs/FMs, each object is represented in high dimensional space, capturing the objects features and characteristics.

What is vector similarity search?



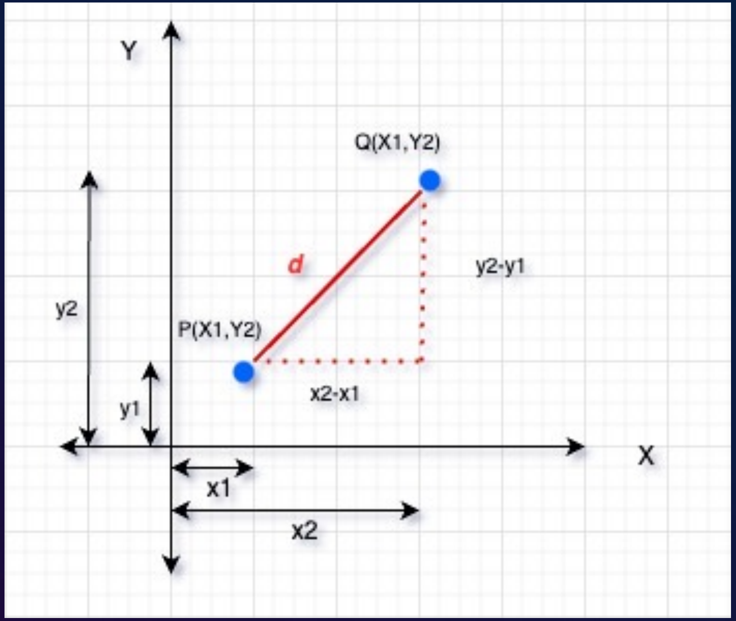
Flight
[0.1,0.2, 0.3]

Airport
[0.2,0.3,0.1]



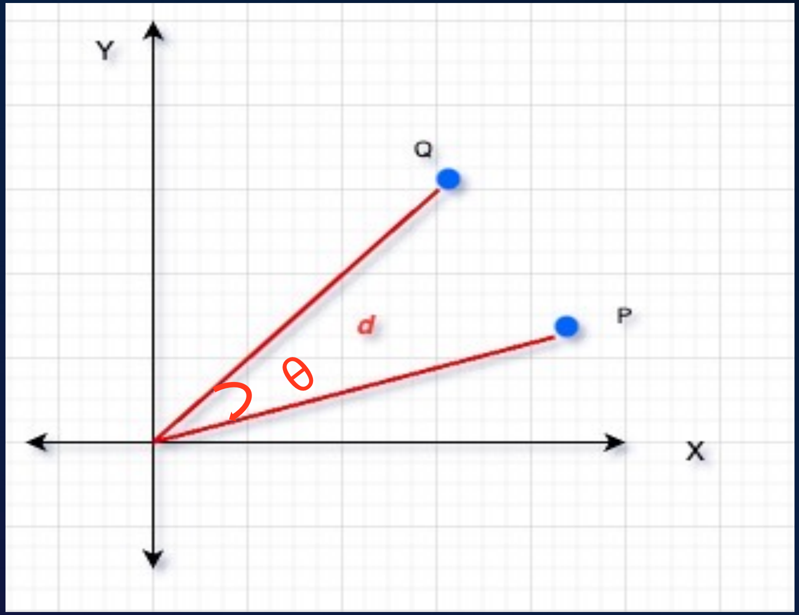
Vector Distance Measurements Algorithms

Euclidean Similarity



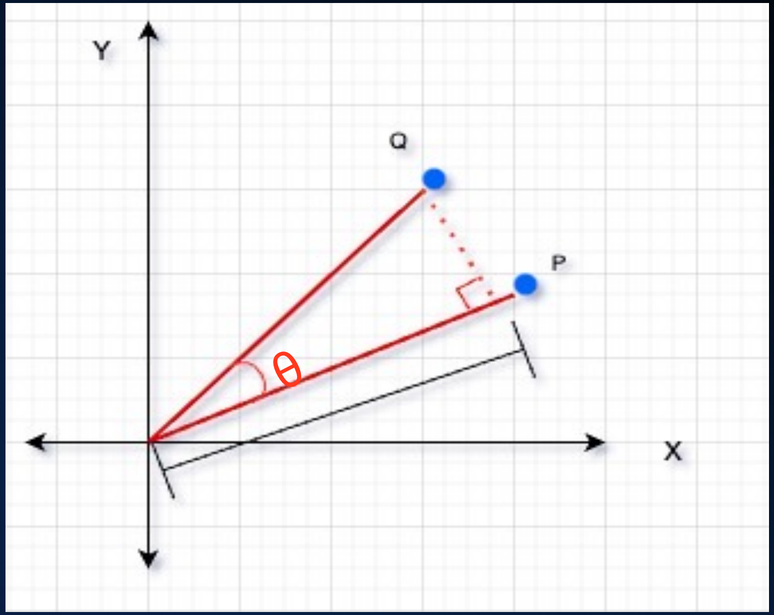
Useful for
Product recommendation system

Cosine Similarity



Useful for
Semantic search and Document Classification

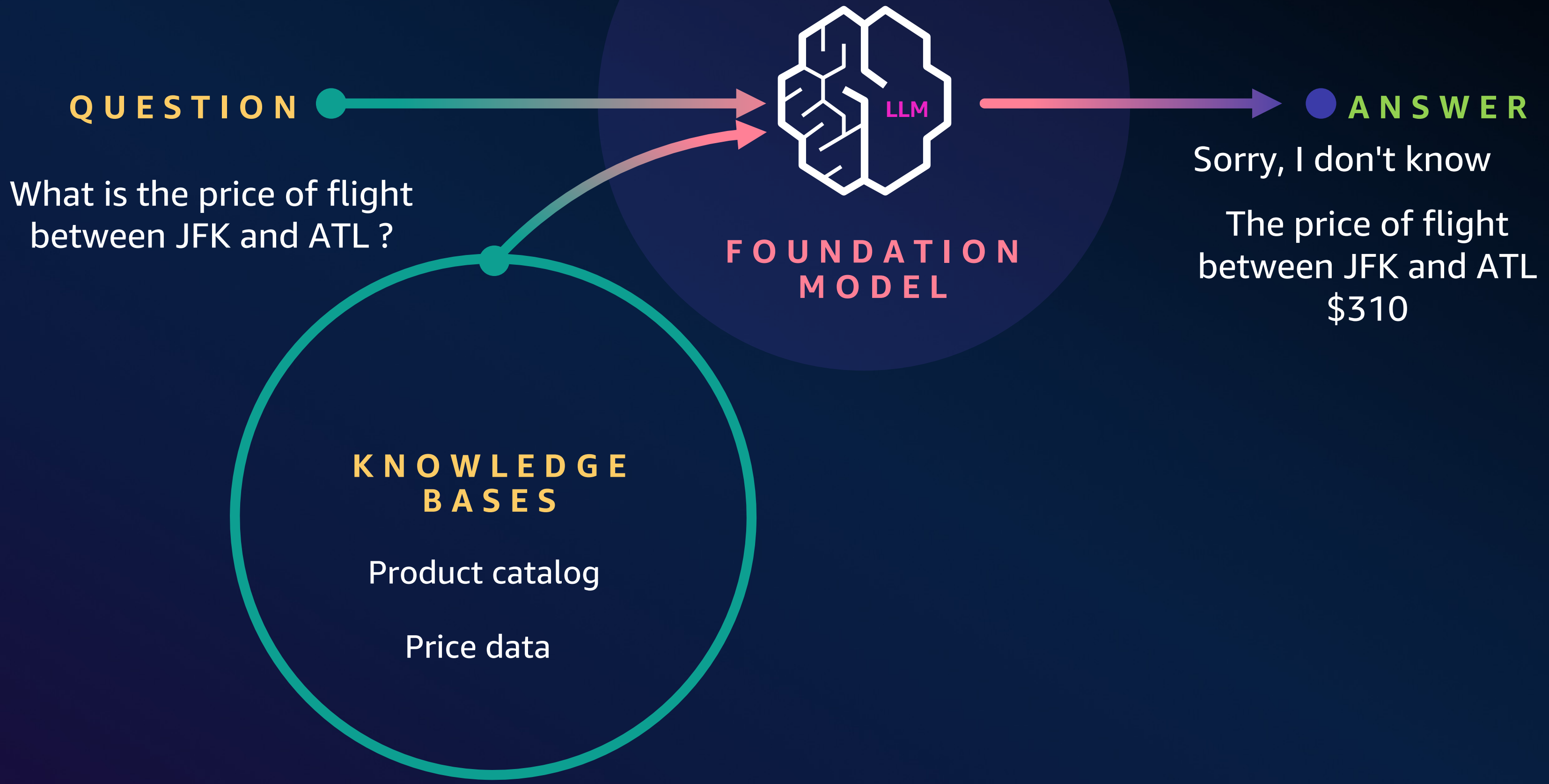
Dot Product Similarity



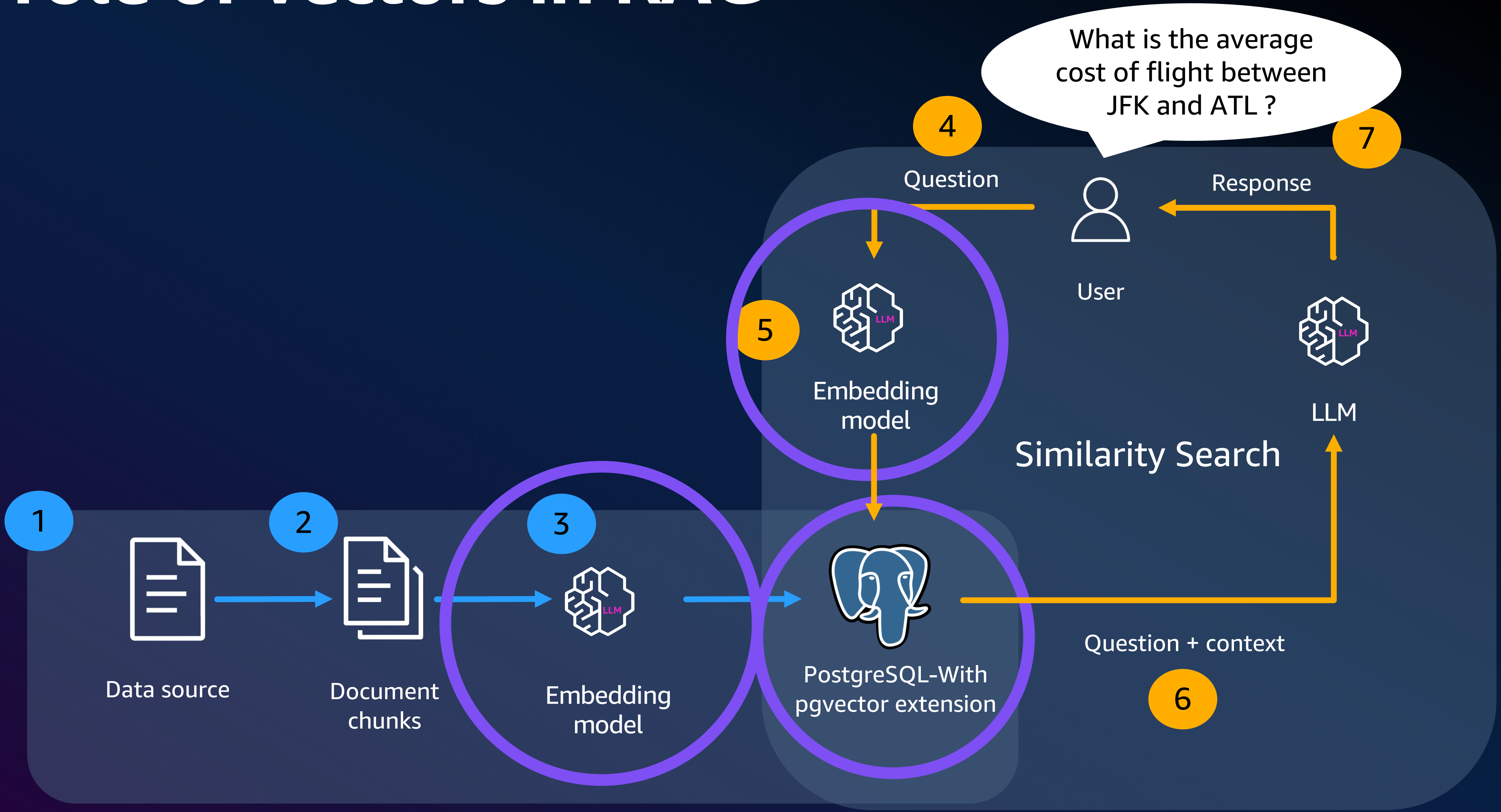
Useful for
Collaborative Filtering in Recommendation system

Retrieval Augmented Generation (RAG)

Configure FM to interact with your company data

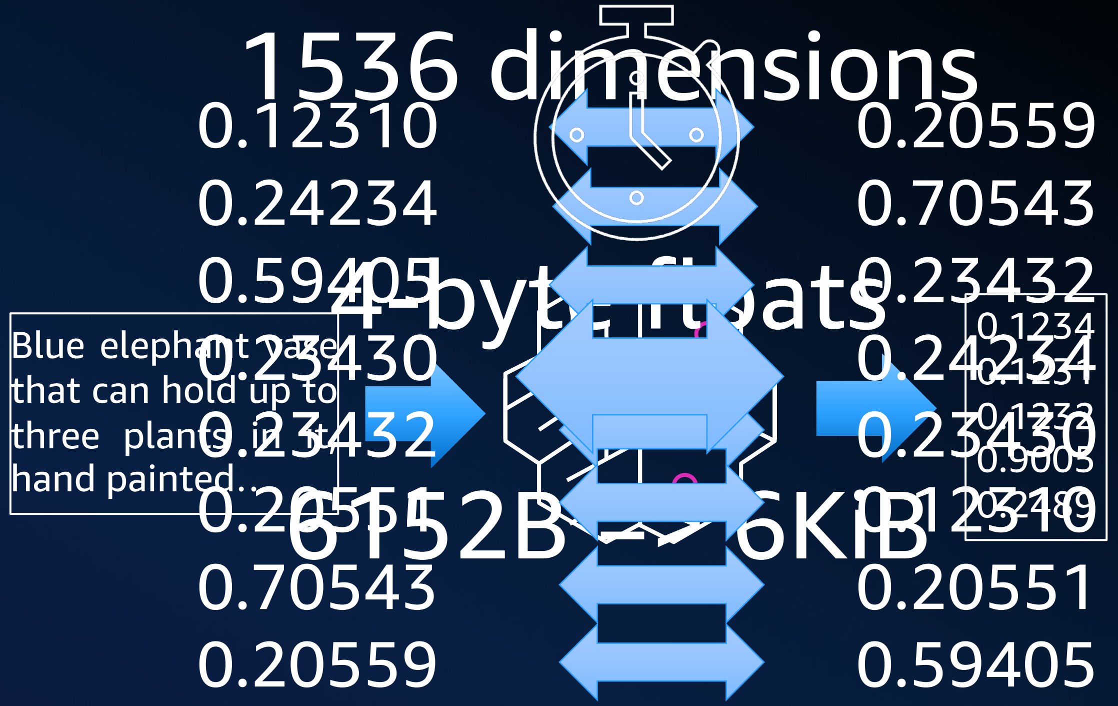

QUESTION
What is the price of flight between JFK and ATL ?
LLM
FOUNDATION MODEL
ANSWER
Sorry, I don't know
The price of flight between JFK and ATL $310
KNOWLEDGE BASES
Product catalog
Price data

The role of vectors in RAG



Challenges with vectors

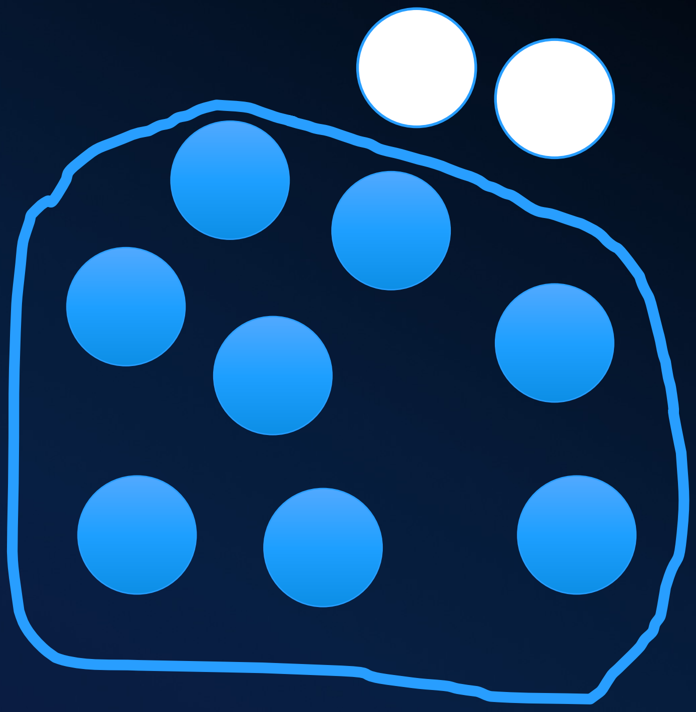
- Time to generate embeddings
- Embedding size
- ~~Compression~~
- Query time



1,000,000 => 5.7GB

Approximate nearest neighbor (ANN)

- Find similar vectors without searching all of them
- Faster than exact nearest neighbor
- “Recall” – % of expected results



Recall: 80%

PostgreSQL as Vector DB

- Best open source Relational database
- Scalability
- Performance
- Availability and Reliability
- Strong Data Consistency
- Enterprise grade Security
- Highly Extensible
- Parallel processing
- Operationally Efficient

What is pgvector?

An open source extension that is available PostgreSQL v12 onwards and adds support for **storage, indexing, searching, metadata** with choice of **distance**

vector data type

Co-locate with embeddings

Exact nearest neighbor (K-NN) search
Approximate nearest neighbor (ANN) search using index

Distance operators **(<->, <=>, <#>)**

Supports IVFFlat/HNSW indexing

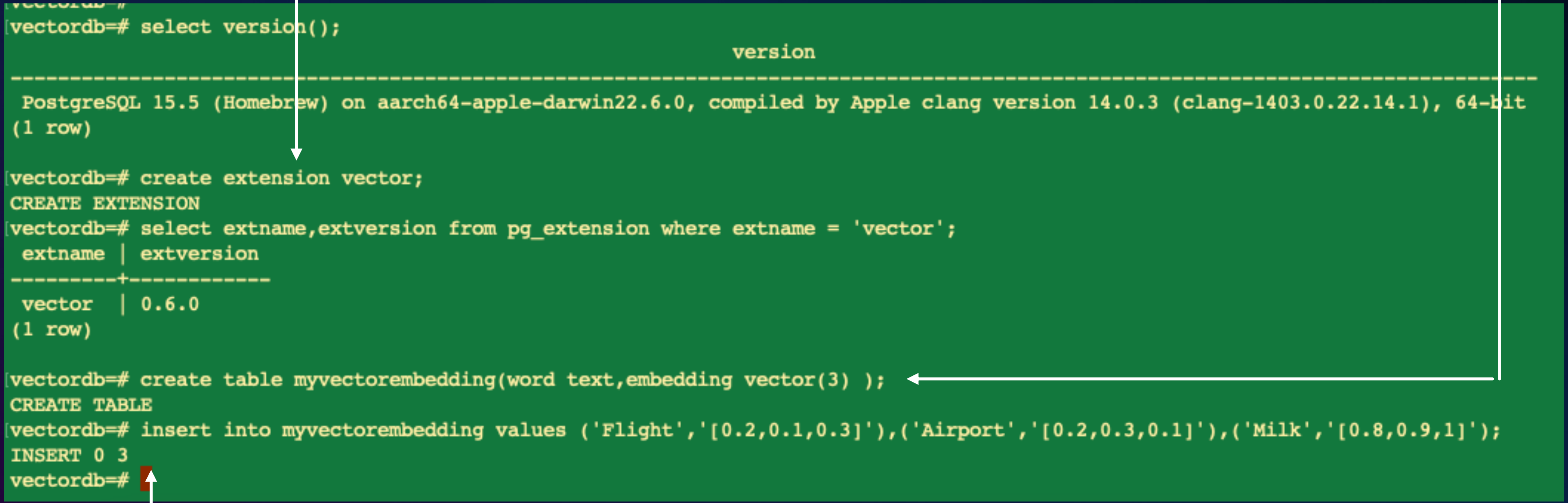
KNN :- K-Nearest Neighbors Algorithm
ANN :- Approximate Nearest Neighbor

github.com/pgvector/pgvector

PostgreSQL – Vector Data Type

Vector Extension created on PostgreSQL Db. Supported on PostgreSQL v12+. Latest pgvector extension is v0.6.0

Creating table of vector data type for column Embedding



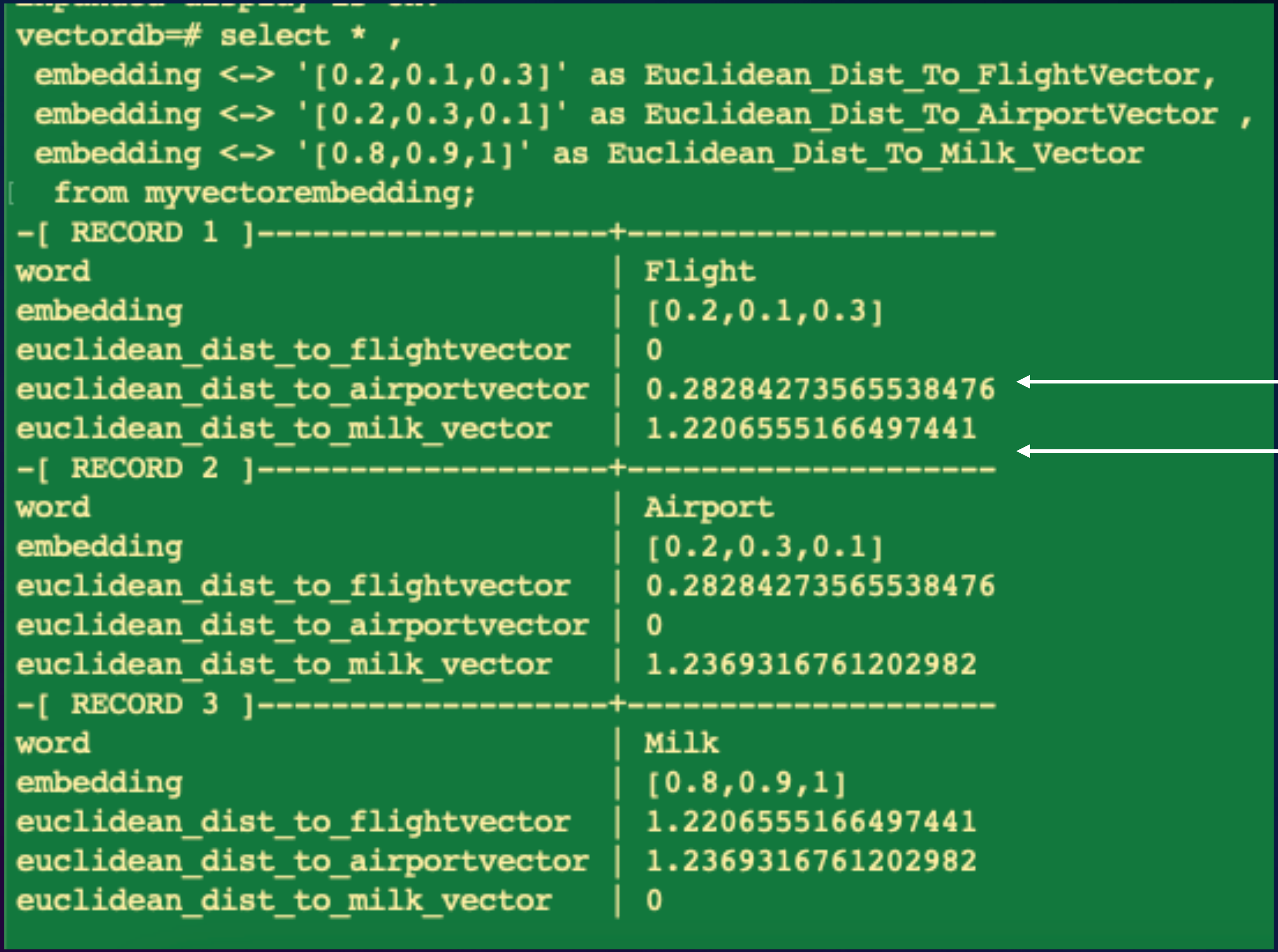
```
[vectordb=# select version();
                                                              version
-------------------------------------------------------------------------------------------------------------------------------------
 PostgreSQL 15.5 (Homebrew) on aarch64-apple-darwin22.6.0, compiled by Apple clang version 14.0.3 (clang-1403.0.22.14.1), 64-bit
(1 row)

[vectordb=# create extension vector;
CREATE EXTENSION
[vectordb=# select extname,extversion from pg_extension where extname = 'vector';
 extname | extversion
---------+------------
 vector  | 0.6.0
(1 row)

[vectordb=# create table myvectorembedding(word text,embedding vector(3) );
CREATE TABLE
[vectordb=# insert into myvectorembedding values ('Flight','[0.2,0.1,0.3]'),('Airport','[0.2,0.3,0.1]'),('Milk','[0.8,0.9,1]');
INSERT 0 3
vectordb=#
```

Example Embedding Data Inserted into PostgreSQL table

PostgreSQL – Vector Data Type



```
vectordb=# select * ,
 embedding <-> '[0.2,0.1,0.3]' as Euclidean_Dist_To_FlightVector,
 embedding <-> '[0.2,0.3,0.1]' as Euclidean_Dist_To_AirportVector ,
 embedding <-> '[0.8,0.9,1]' as Euclidean_Dist_To_Milk_Vector
  from myvectorembedding;
-[ RECORD 1 ]------------------+--------------------
word                           | Flight
embedding                      | [0.2,0.1,0.3]
euclidean_dist_to_flightvector | 0
euclidean_dist_to_airportvector | 0.28284273565538476
euclidean_dist_to_milk_vector  | 1.2206555166497441
-[ RECORD 2 ]------------------+--------------------
word                           | Airport
embedding                      | [0.2,0.3,0.1]
euclidean_dist_to_flightvector | 0.28284273565538476
euclidean_dist_to_airportvector | 0
euclidean_dist_to_milk_vector  | 1.2369316761202982
-[ RECORD 3 ]------------------+--------------------
word                           | Milk
embedding                      | [0.8,0.9,1]
euclidean_dist_to_flightvector | 1.2206555166497441
euclidean_dist_to_airportvector | 1.2369316761202982
euclidean_dist_to_milk_vector  | 0
```

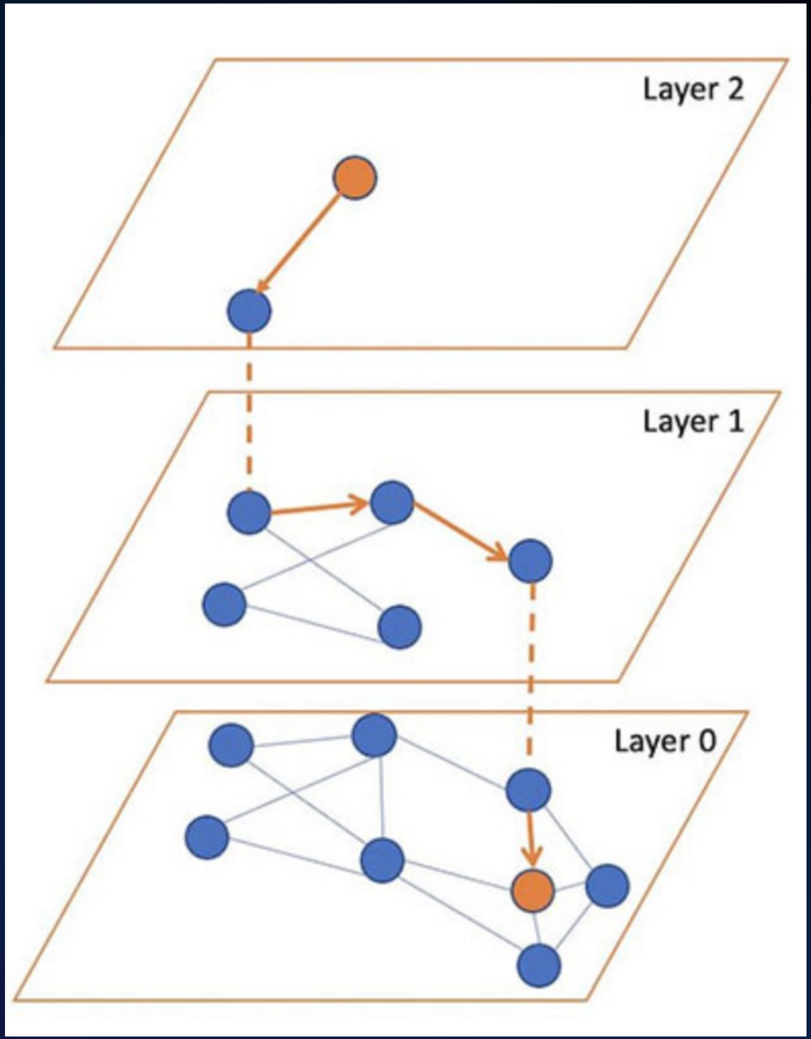
Vector Distance of word Flight is less to word Airport (0.248) . Represents they are similar.

Vector Distance of word Flight is more to word Milk (1.22) Represents they are less similar.

HNSW

- Hierarchical Navigable Small Worlds – An graph based approximate KNNS algorithm
- Navigable Small-World + Skip List
- Uses greedy traversal on proximity graphs to approximate K-Nearest Neighbors

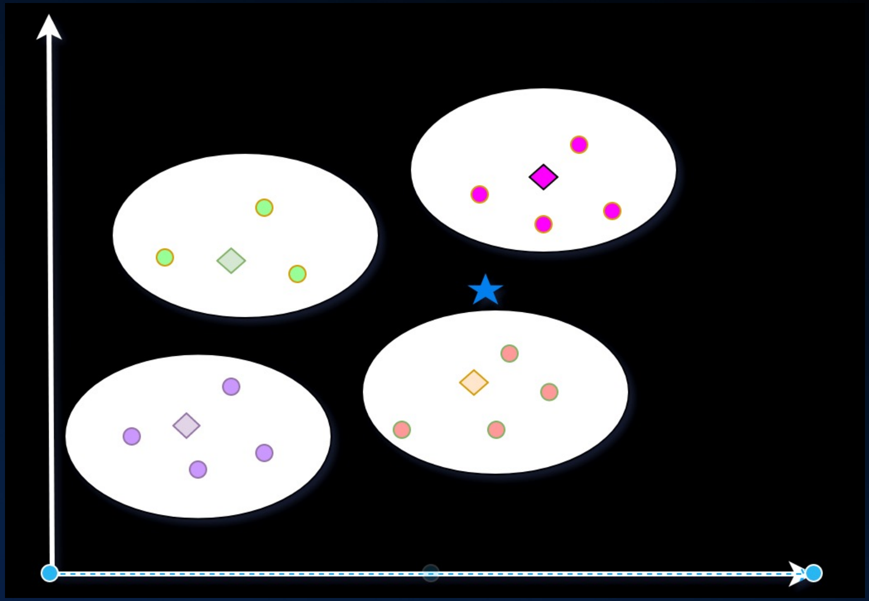
- Important Build Parameters
  - **m** – The maximum number of links a vector can have (default - 16)
  - **ef_construction** – The size of the queue of the candidate Nearest Neighbor nodes (default - 64)
- Important Query Parameters
  - **hnsw .ef_search** – The size of the queue of the candidate nodes to visit during traversal

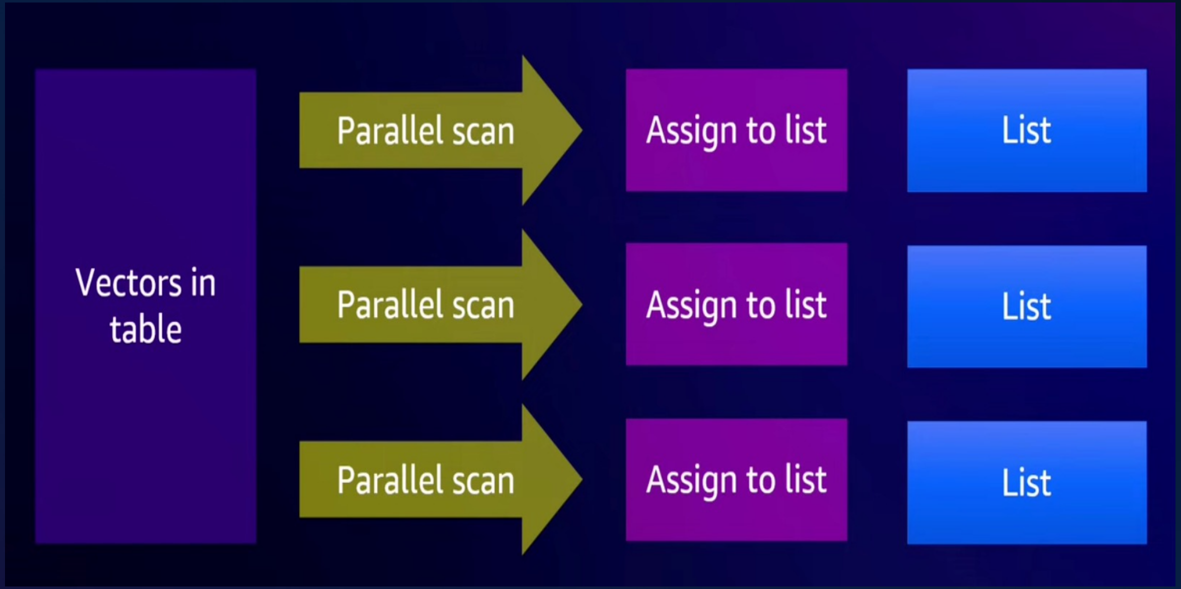


IVFFlat

- Inverted File Flat – An Cluster based approximate KNN algorithm
- Cluster centroids are determined using unsupervised clustering
- Use parallelism to accelerate build times

- Important Build Parameters
  - **lists** – The number of clusters/lists will be created
    - rows / 1000 if rows < 1M
    - sqrt(rows)
- Important Query Parameters
  - **ivfflat.probes –** Number of clusters/lists to be visited to find Nearest Neighbor vector
    - sqrt(lists)





IVFFlat Index Vs. HNSW Index

Aspects	IVFFlat	HNSW
Method	K-means	Graph
Build Time	Faster	Slower */ (Faster with **pgvector v 0.6**)
Index Size	Lower	Higher
Recall Vs. Performance	Lower	Higher
Memory consumption	Needs less Memory to build	Need more Memory to build
Data requirement	Create the index *after* the table has some data	Index can be created without any data in the table since there isn't a training step like IVFFlat.

Design Patterns

- Use the same distance function for index creation that was used to train your embedding model
- Perform prefiltering whenever possible (e.g. schema_name = 'HR')
- Multicolumn indexes not supported for Vector data
  - *ERROR: access method "hnsw" does not support multicolumn indexes*
- Partial index
  - *CREATE INDEX "adapters_hnsw_idx" ON public.amazon_pqa USING hnsw (question_embedding vector_cosine_ops) WITH (m=16, ef_construction=64) WHERE category = 'adapters';*
- Partition table
  - *CREATE TABLE public.amzn_pqa_adapters PARTITION OF public.amzn_pqa FOR VALUES IN ("adapters");*
  - *CREATE INDEX ON public.amazon_pqa_adapters USING hnsw (question_embedding vector_cosine_ops);*

Storage : Key Consideration

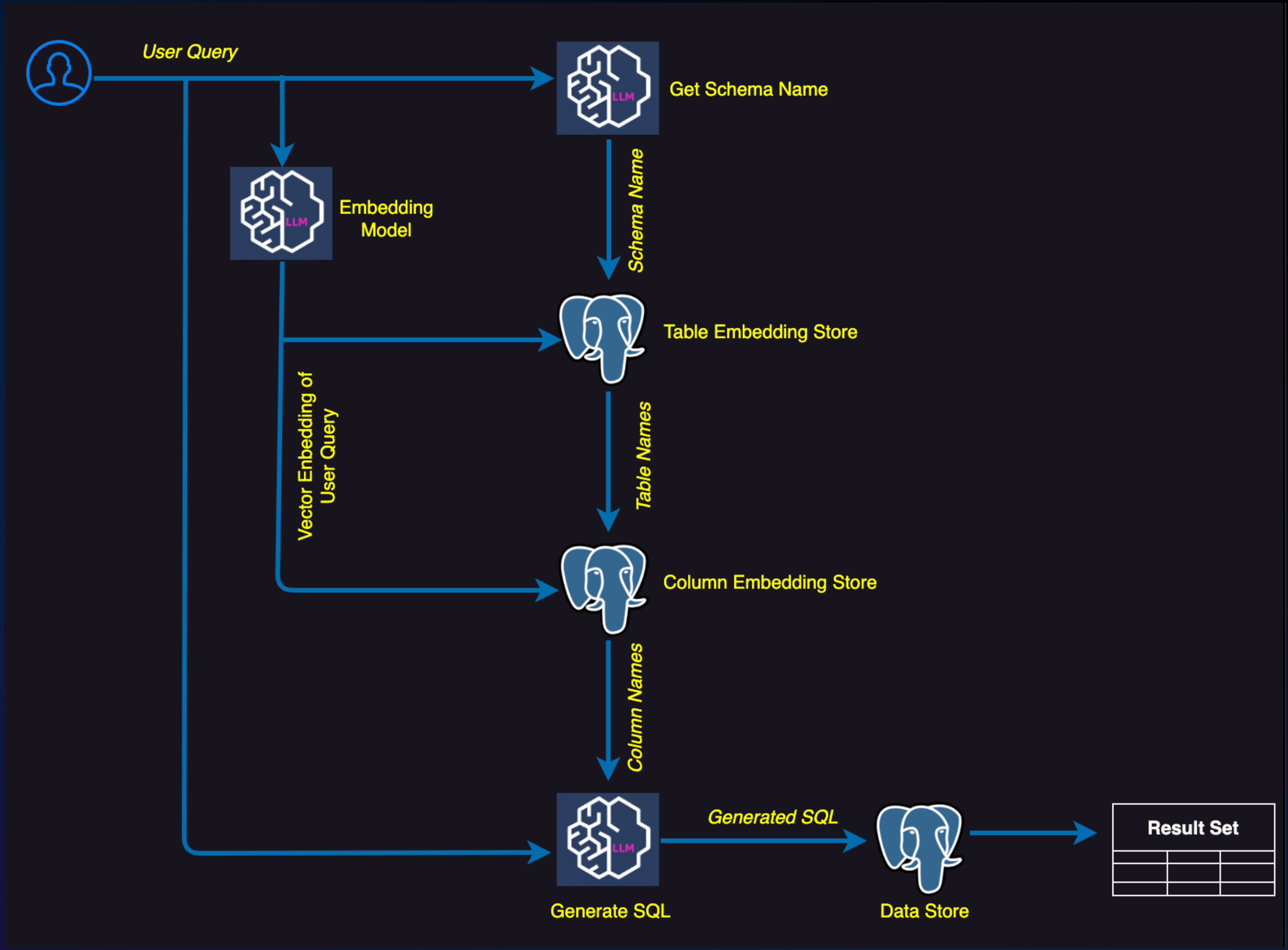
- Store maximum 16k Dimension Vector
- Each datapoint 4 byte floats
- Storage – PostgreSQL TOASTs values over 2KB (TOAST_TUPLE_THRESHOLD).
- By default TOAST management strategies EXTENDED (before 0.6), use PLAIN if vector fit in a single page
- From pgvector 0.6 storage changed from EXTENDED to EXTERNAL
- Maximum dimension that can be indexed is 2000

* TOAST - The Oversized-Attribute Storage Technique

Conclusion

- Primary design decision: **query performance** and **recall**
- Determine where to invest: **storage**, **compute**, **indexing strategy**
- Plan for today and tomorrow: pgvector is rapidly innovating

Demo


User Query
Get Schema Name
Embedding Model
Schema Name
Table Embedding Store
Vector Embedding of User Query
Table Names
Column Embedding Store
Column Names
Generated SQL
Generate SQL
Data Store
Result Set

Q & A

Thank you!

Sachin Khanna
ekhannas@amazon.com

Jitender Kumar
kumarlmu@amazon.com