Vector Databases and Embedding Techniques

Generative AI and Prompt Engineering 2025

Ram N Sangwan

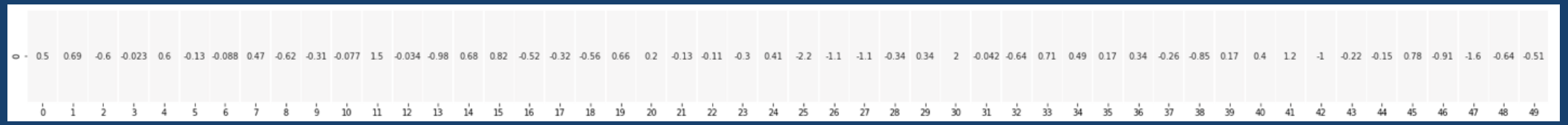
- Vector Databases
- Working with Embedding
- Embedding Models
- Using Oracle 23ai
- Capabilities and Benefits

Word Embeddings

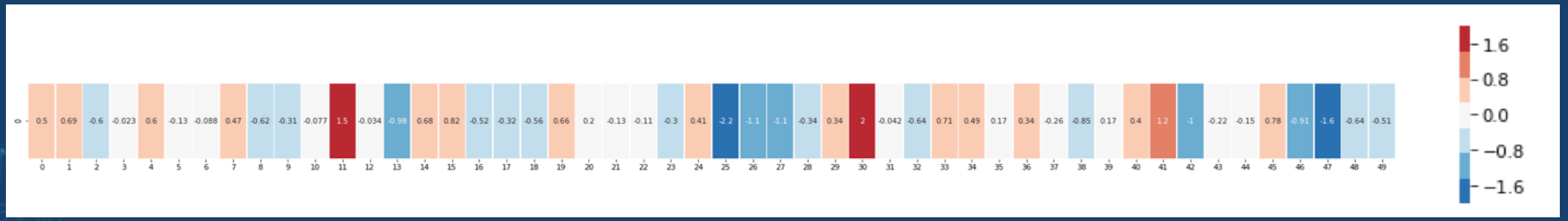
- This is a word embedding for the word “king” (GloVe vector trained on Wikipedia):

```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -
0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756
, -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102
, 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]
```

- It’s a list of 50 numbers.
- We can’t tell much by looking at the values. But let’s visualize it a bit so we can compare it other word vectors.
- Let’s put all these numbers in one row:

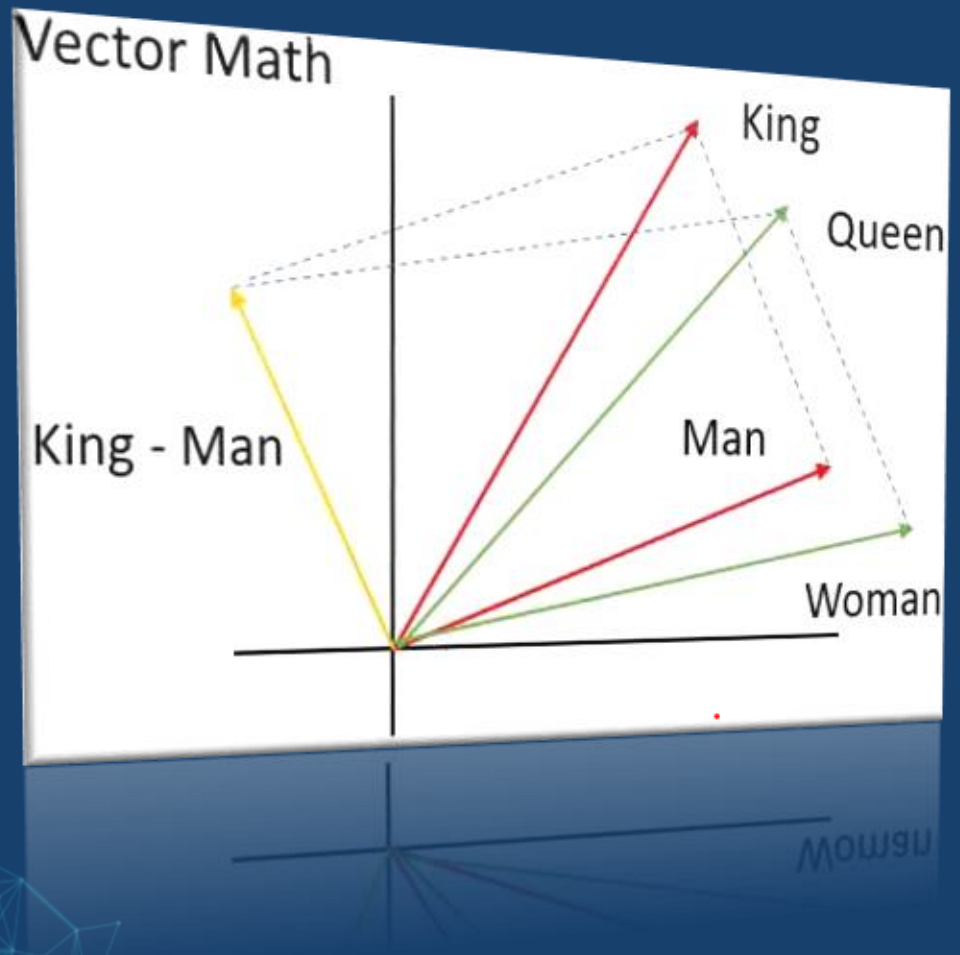


- Let’s color code the cells based on their values (red if they’re close to 2, white if they’re close to 0, blue if they’re close to -2):

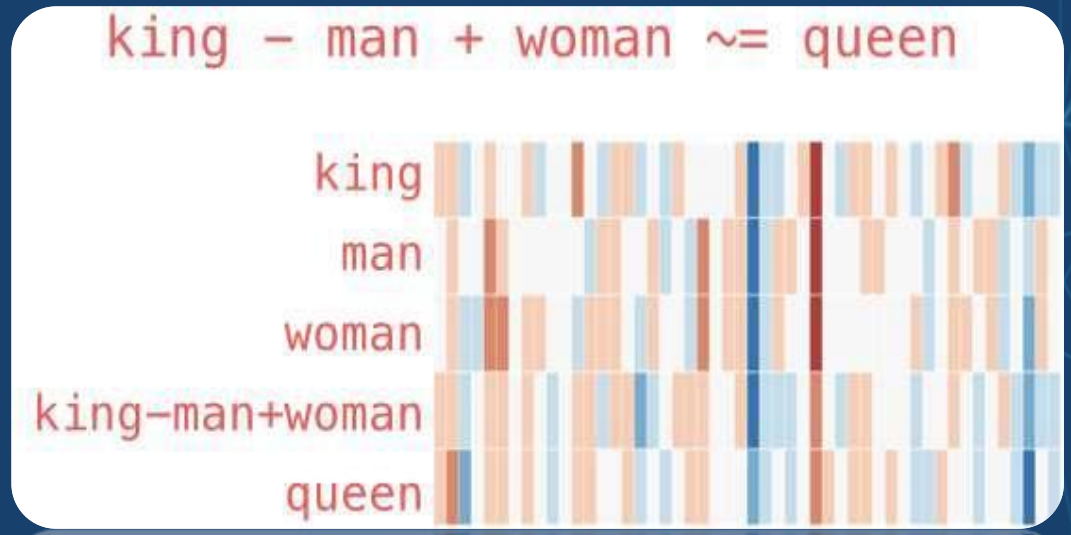


Vectors/Embeddings - King & Queen Example

Conceptual example



How it is implemented



Word Embeddings

Word Embeddings capture properties of the word.

The example here shows two properties:

- Age (vertical axis)
- Size (horizontal axis)

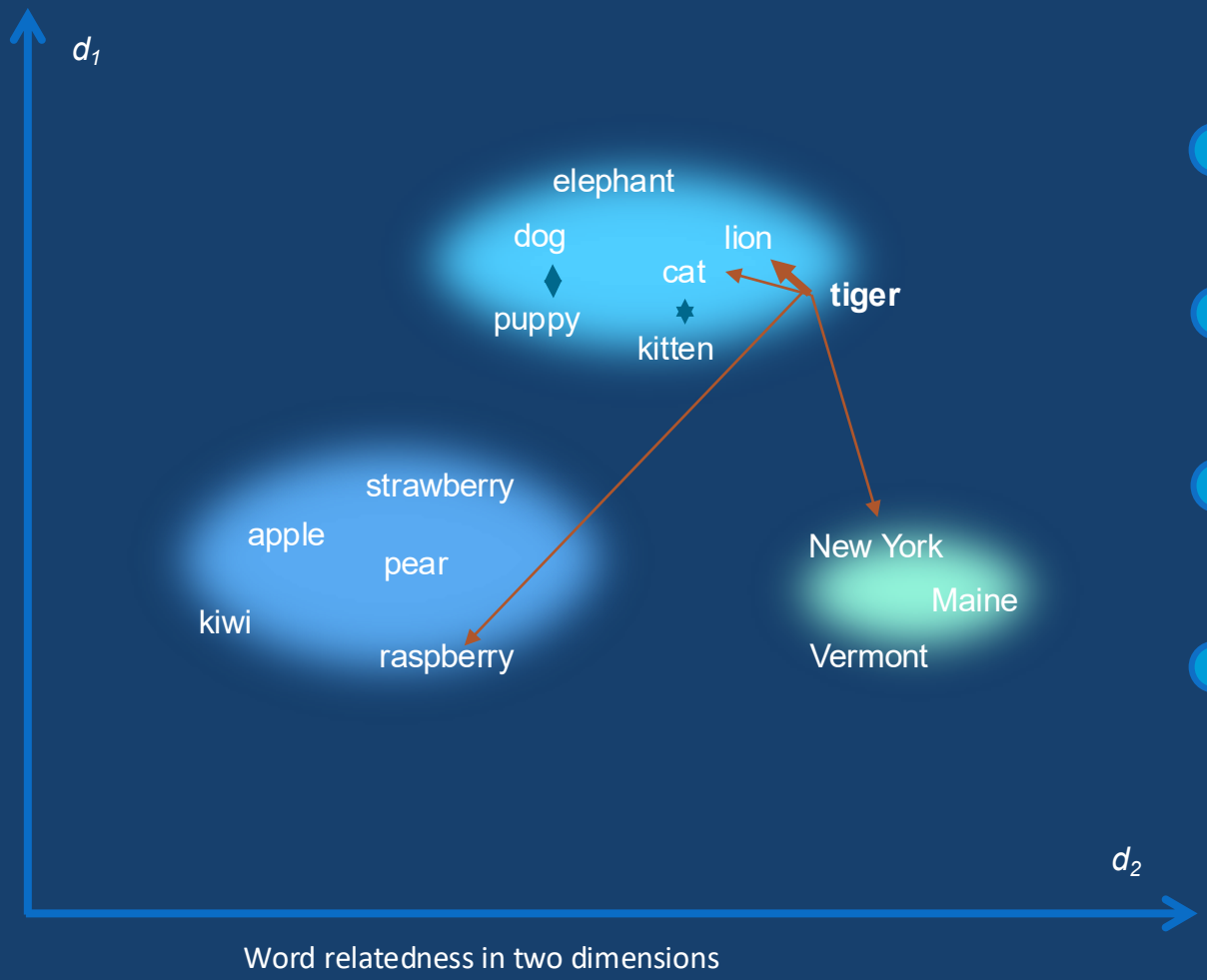
Actual Embeddings represent more properties (coordinates) than just two.

These rows of coordinates are called vectors and represented as numbers

Word	Embeddings					
Puppy	0.02806	0.03906	0.0386	…	…	…
Kitten	0.0420	0.03006	0.5286	…	…	…
Cat	-0.024	0.0568	0.4280	0.91606	…	…
Dog	-0.0829	-0.4280	0.9280	0.8245	…	…

Age | Size | Other Properties

Semantic Similarity

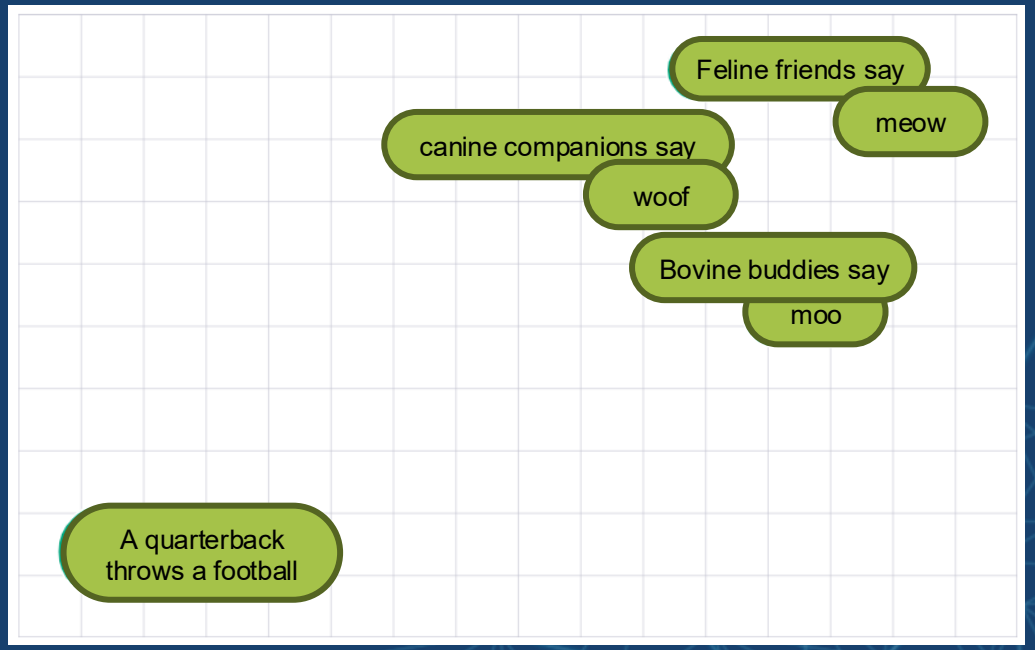


- Cosine and Dot Product Similarity can be used to compute **numerical similarity**.
- Embeddings that are **numerically similar** are also **semantically similar**.
- E.g., embedding vector of “Puppy” will be more similar to “Dog” than that of “Lion.”
- There are three groups of words here based on similarity: Animals, Fruits, and Places.
- "Tiger" is closest to the Animals group, closer to cat family members.

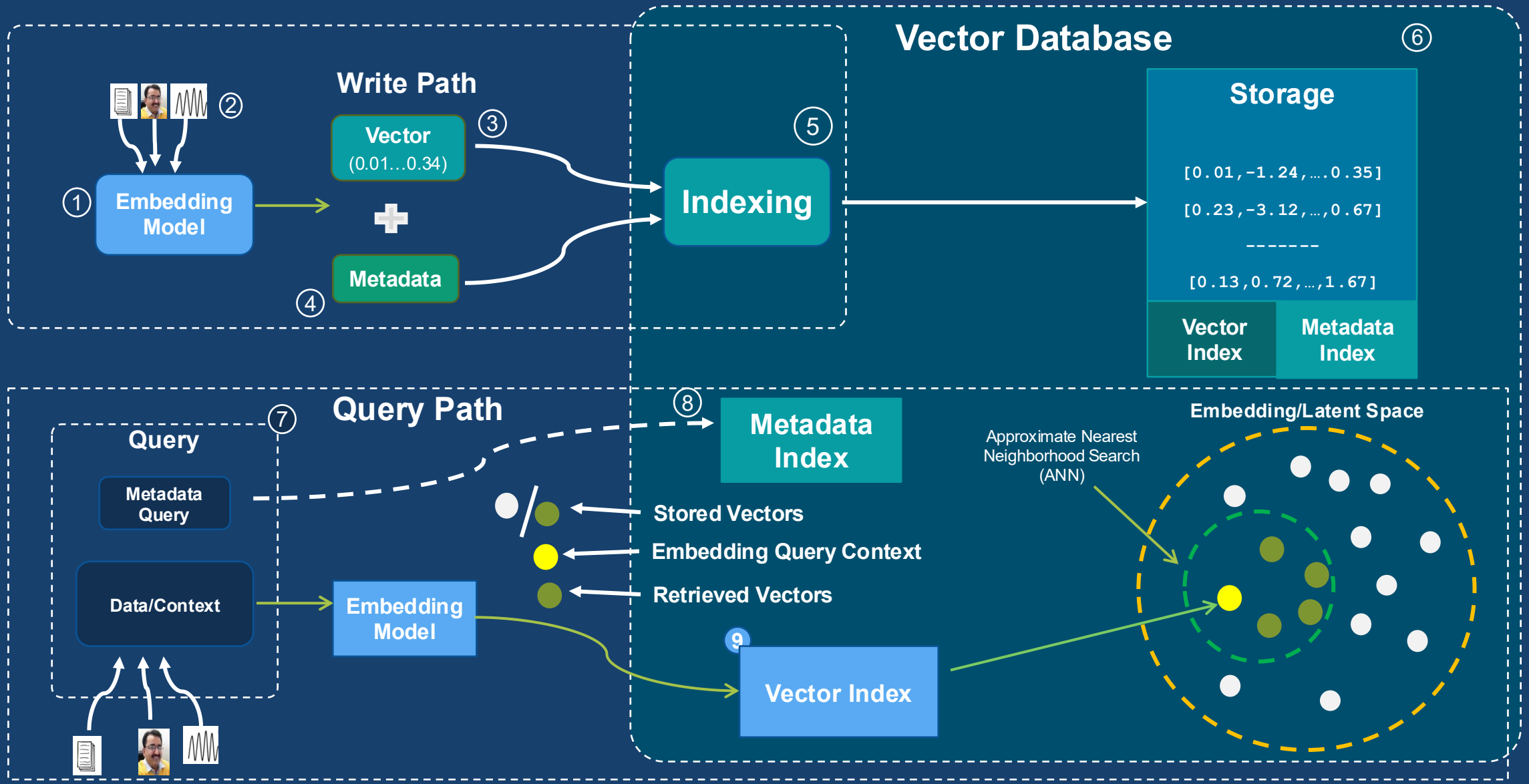
Sentence Embeddings

- A sentence embedding associates every sentence with a vector of numbers.
- Similar sentences are assigned to similar vectors, different sentences are assigned to different vectors.
- The embedding vector of “canine companions say” will be more similar to the embedding vector of “woof” than that of “meow.”

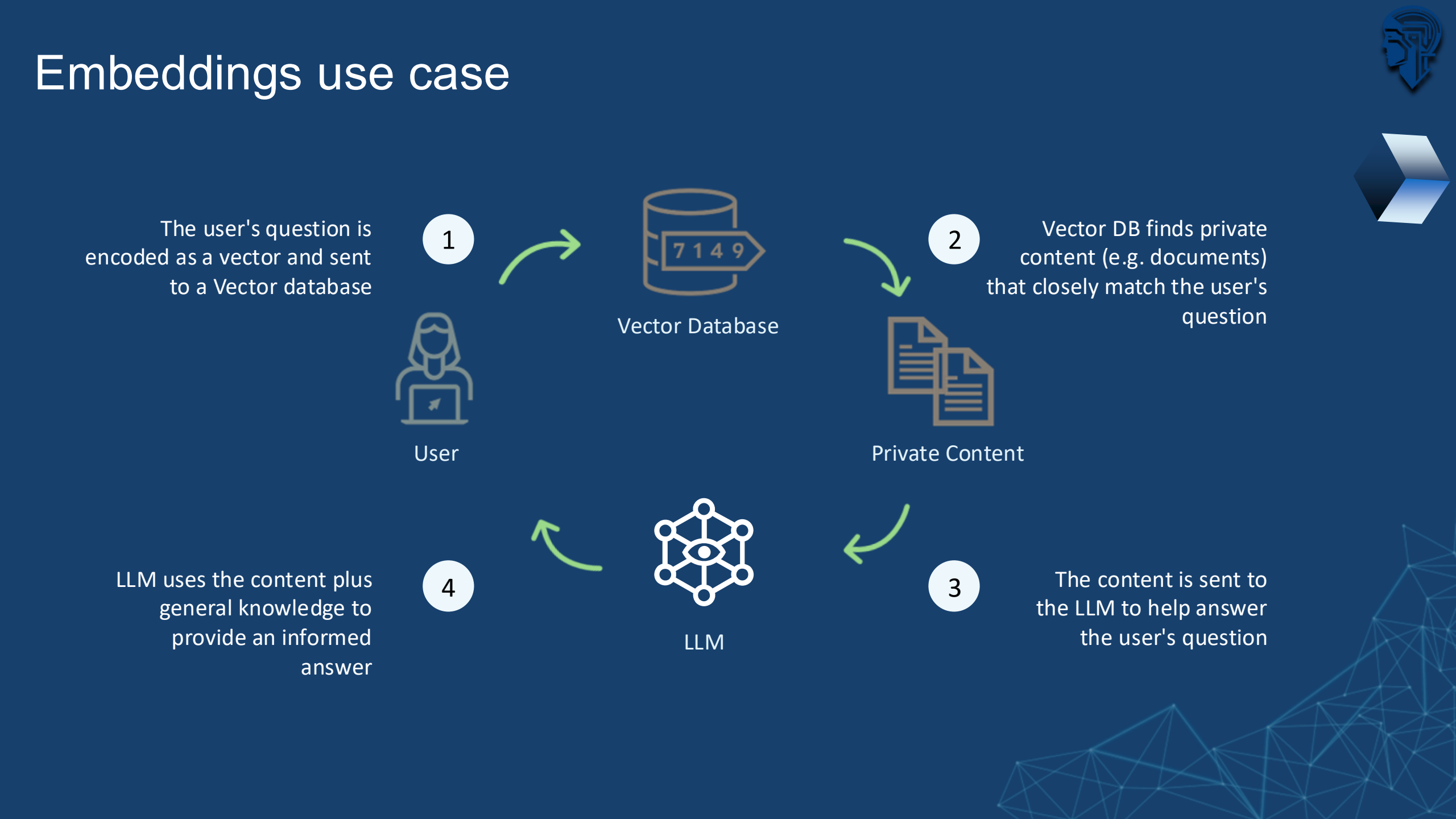
Sentences	Embeddings					
Feline friends say	0.02806	0.03906	…	…	…	…
Canine companion says	0.0420	0.03006	…	…	…	…
Bovine buddies say	-0.024	0.0568	…	…	…	…
A quarterback throws a football	-0.0829	-0.4280	…	….	…	…



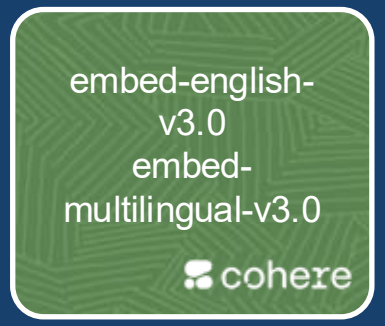
How Embeddings are Generated and Used



Embeddings use case



Embedding Models in Generative AI



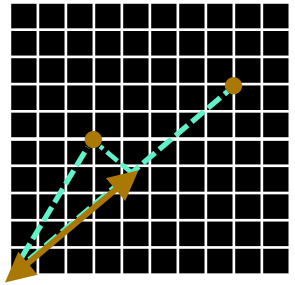
- English and Multilingual
- Model creates a 1024-dimensional vector for each embedding
- Max 512 tokens per embedding

- Smaller, faster version; English and Multilingual
- Model creates a 384-dimensional vector for each embedding.
- Max 512 tokens per embedding

- Previous generation models, English
- Model creates a 1024 dimensional vector for each embedding
- Max 512 tokens per embedding

Embedding Distance

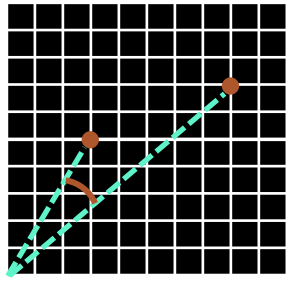
Dot Product



$$A.B = \sum_{i=1}^{n} A_i B_i$$

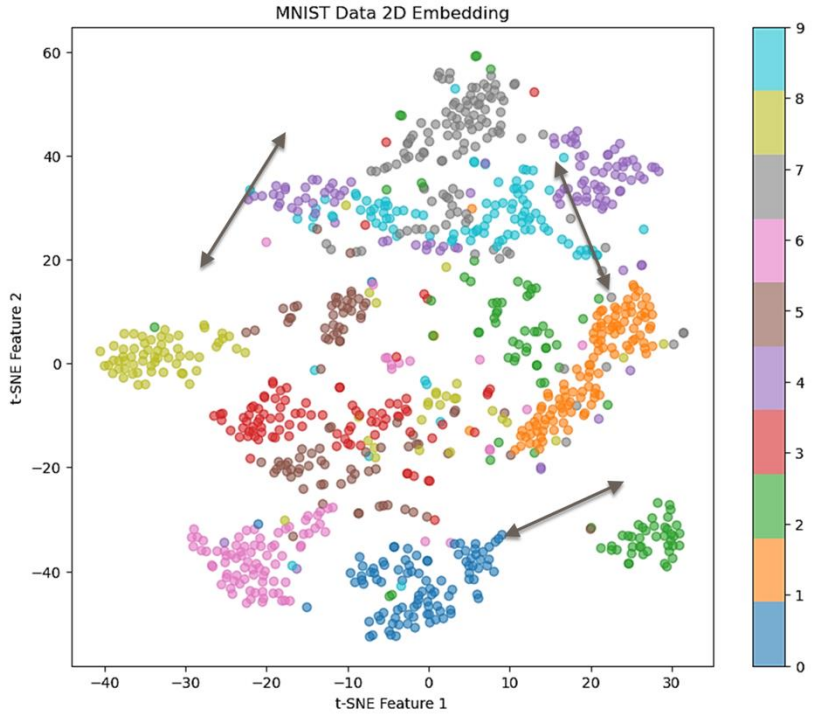
Dot Product is the Measure of the magnitude of the projection of one vector on to the other and gives you magnitude and direction.

Cosine Distance



$$1 - \frac{A \ . \ B}{||A|| \ \ ||B||}$$

Cosine Distance is the Measure of the difference in directionality between vectors so that only the angle between the vectors matters, not their magnitude.

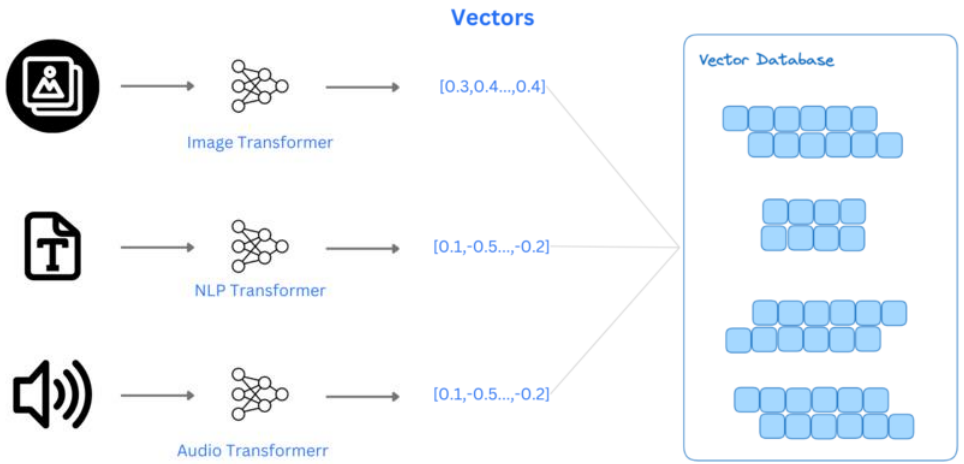


- This is a scatter plot of the MNIST handwritten dataset reduced to two dimensions using t-SNE, which is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.
- The plot shows clusters of points in different colors, each representing one of the digits from 0 to 9 from the dataset.
- You can see that how this was able to separate and group embeddings in case of images.

Vector Databases

What are Vector Databases?

- A **vector database** is a specialized type of **database** that **indexes** and **stores vector embeddings** for **fast retrieval** and **similarity search**.



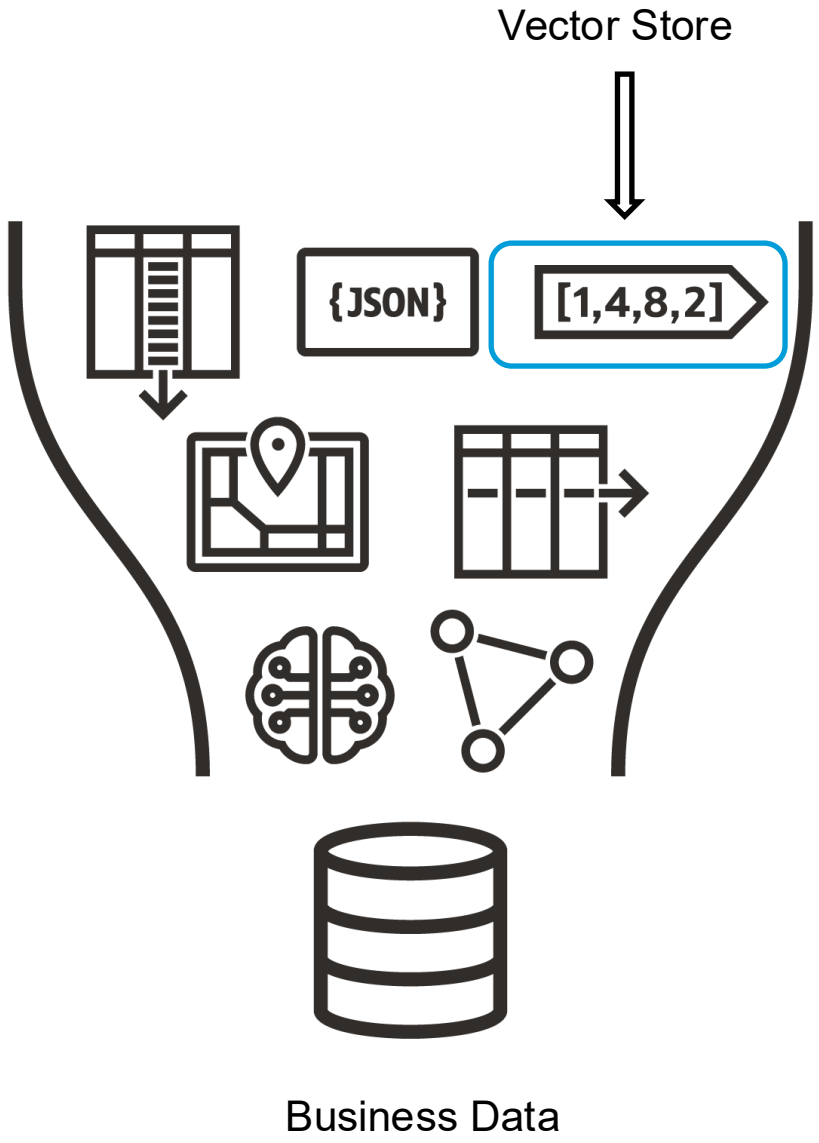
Vector Databases

- Oracle AI Vector Search, FAISS, Pinecone, Chroma and Weaviate are some prominent vector databases.
- Oracle already holds **most of the world's operational enterprise data.**
- **So, it will be an order of magnitude easier** to **add vector embeddings to already stored enterprise data in Oracle databases** than replicating enterprise data to a separate vector database.

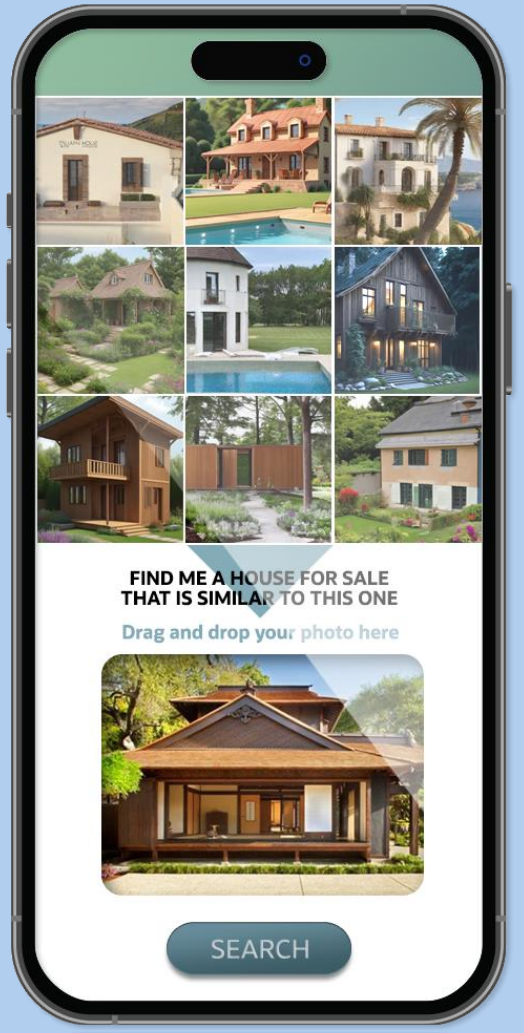
Oracle 23 ai

AI Vector Search

- Relational search excels in handling structured data, such as customer records, sales transactions, or inventory databases.
- AI Vector Search simplifies and optimizes searching for documents, images, patterns, and data that have similar semantics.
- When combined, AI vector search and relational search can provide a more comprehensive view of business data.

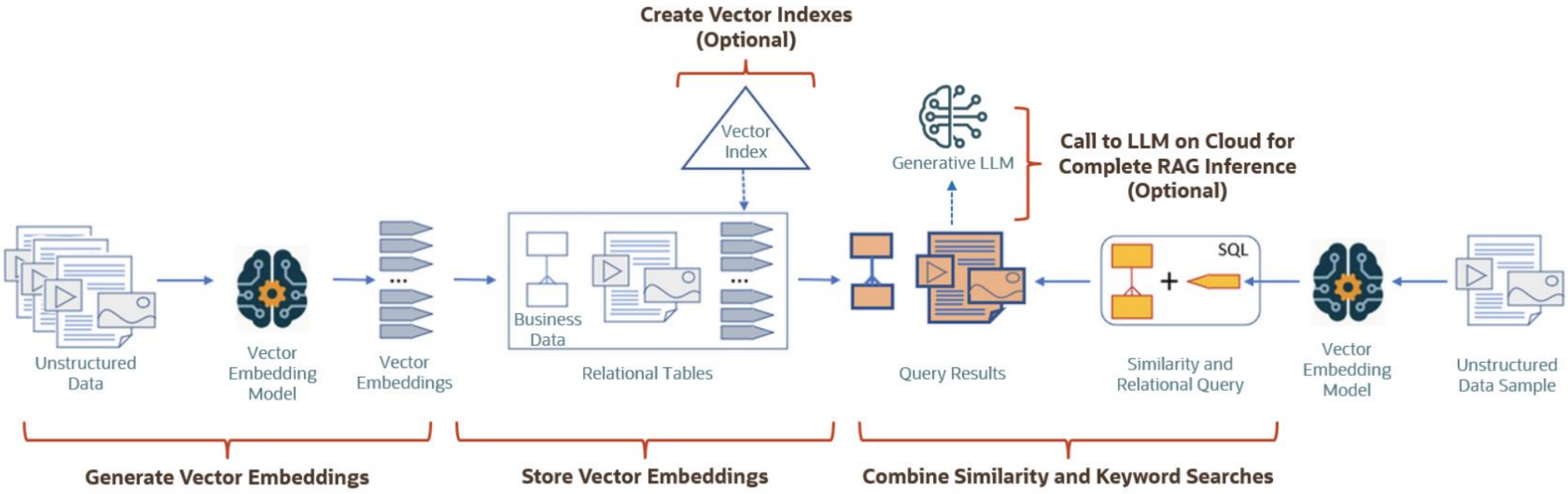


Use Case



- Imagine an app that helps customers find houses for sale that are similar to a picture the customer uploads.
- Finding a good match requires combining semantic picture search with searches on business data including:
  - **Customer data** such as the customer's preferred city and budget.
  - **Product data** such as houses available for sale in each city and their price

Oracle AI Vector Search Workflow



Vector Datatype

New built-in VECTOR datatype

```
CREATE TABLE my_images (
      id          NUMBER,
      data_image BLOB,
      image_vec   VECTOR(768, FLOAT32));
```

Optional dimension count

Optional dimension format

Dimension format can be:
INT8, FLOAT32, and FLOAT64

Simpler VECTOR specification

```
CREATE TABLE my_images (
      id          NUMBER,
      data_image BLOB,
      image_vec   VECTOR);
```

Why is this useful?

You can embed your data with newer ML embedding models as AI technology evolves, but your schema can stay the same, and applications don't need to be rewritten

Vector Distance Function

The key operation is vector distance computation to gauge similarity

```
VECTOR_DISTANCE(VECTOR1, VECTOR2, <optional distance metric>)
```

Different embedding models can use different metrics, but the basic concept remains the same:

- The Distance between two vectors is smaller for entities that are more similar
- Distance functions supported in Oracle Database 23ai are:

COSINE (default), EUCLIDEAN, EUCLIDEAN_SQUARED, HAMMING, MANHATTAN, DOT

Database-Native Vector Embedding Generation

VECTOR_CHUNKS
VECTOR_EMBEDDING

Generate vector from query image

```
VECTOR_EMBEDDING(resnet_50 USING query_image)
```

Image dataset already embedded into vectors and stored in Oracle Database 23ai

```
VECTOR_EMBEDDING(resnet_50 USING data_image)
```

Query Image

Load image as BLOB

Query Image Embedding Generation

Vector embedding

23ai

Vector Search for similar matches

Top 2 results

Vector Index Syntax

- Basic index creation

```
CREATE VECTOR INDEX photo_idx ON customer(photo_vec)
ORGANIZATION [INMEMORY NEIGHBOR GRAPH | NEIGHBOR PARTITIONS]
DISTANCE COSINE | EUCLIDEAN | MANHATTAN | ... WITH TARGET ACCURACY 90
```

ORGANIZATION for an index is based if it will fit in-memory:

- If the index data will fit in-memory, use INMEMORY NEIGHBOR GRAPH
- Else use NEIGHBOR PARTITIONS

TARGET ACCURACY clause is added to indicate the default accuracy the index should provide for similarity search queries

Similarity Searches in Oracle 23ai

- Exact Similarity Search
  - Looks for the relative order of vectors compared to a query vector.
  - Result set of your top closest vectors, not the distance between them.
- Approximate Similarity Search
  - Better for faster search speeds with large vector spaces.
  - Approximate searches use vector indexes, which trade off accuracy for performance.
- Multi-Vector Similarity Search
  - Used for multi-document search, where documents are split into chunks.
  - Consists of retrieving top-K vector matches using grouping criteria known as partitions based on the documents' characteristics.

Vector Search SQL

- Vector Search is used to find Top K closest matches to a given query item

Find the top 10 houses that are similar to this picture.

```
SELECT   …
FROM     house_for_sale
ORDER BY vector_distance(for_sale_house_vector, :input_vector)
FETCH FIRST 10 ROWS ONLY;
```

You can now find data that is semantically similar to an input.

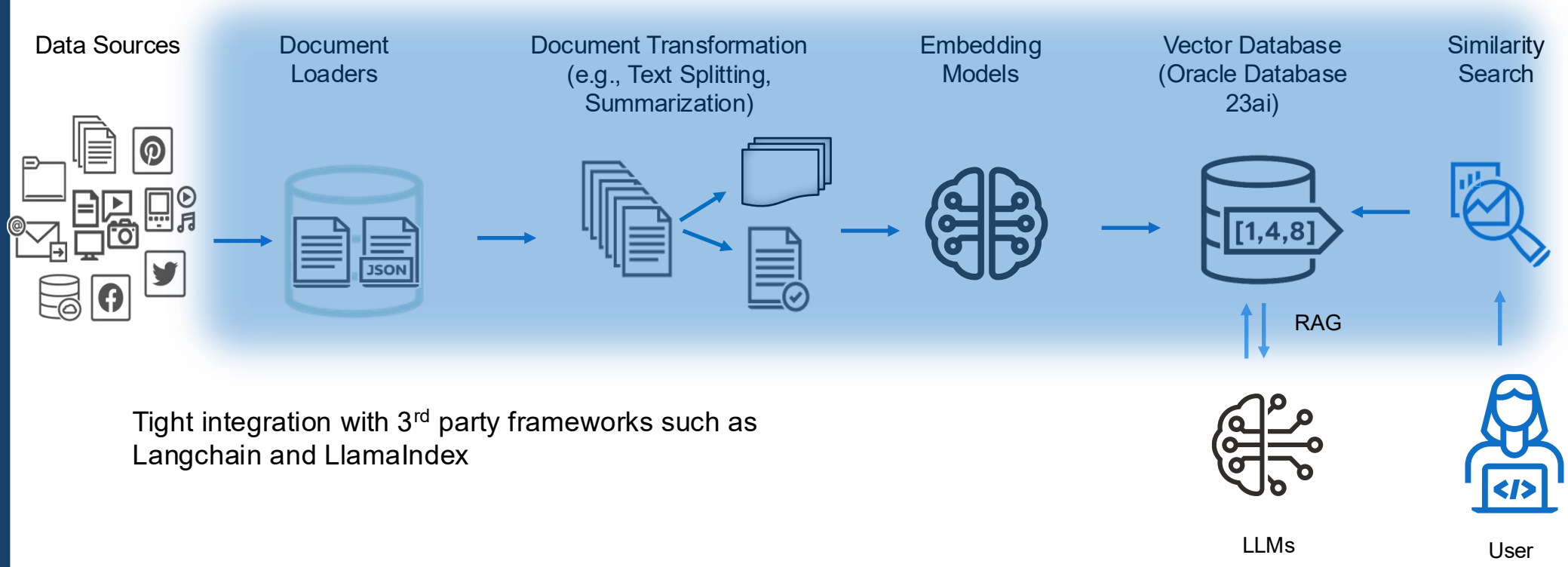
Find houses that are similar to this picture and match the customer's preferred city and budget

```
SELECT …
FROM   house_for_sale
WHERE  price <= (SELECT budget      FROM customer …)
AND    city  in (SELECT search_city FROM customer …)
ORDER BY vector_distance(for_sale_house_vector, :input_vector)
FETCH FIRST 10 ROWS ONLY;
```

Run queries that combine AI Vector Search with business data about customers and products.

AI Vector Search powers Gen AI pipelines

AI Vector Search in Oracle Database 23ai



Tight integration with 3rd party frameworks such as Langchain and LlamaIndex

Getting Started with Oracle 23 ai

- Prepare a Docker Environment

```
sudo dnf install podman podman-docker
```

- Open port 1521 in the virtual network.

```
sudo firewall-cmd --list-all
sudo firewall-cmd --zone=public --add-port=1521/tcp --permanent
sudo firewall-cmd --reload
```

Launch Database 23ai

- Download and start the container using Podman.

```
podman run -p 1521:1521 --name database-23ai \
container-registry.oracle.com/database/free:latest
```

- When you see this message, the database has successfully started.

```
…
#########################
DATABASE IS READY TO USE!
#########################
…
```

- Set the administrator user's password.

```
docker exec -it database-23ai ./setPassword.sh oracle123
```

```
(base) info@MacBook-Pro data % docker exec -it database-23ai ./setPassword.sh oracle123
The Oracle base remains unchanged with value /opt/oracle

SQL*Plus: Release 23.0.0.0.0 - Production on Wed Oct 30 03:22:55 2024
Version 23.5.0.24.07

Copyright (c) 1982, 2024, Oracle.  All rights reserved.


Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.5.0.24.07

SQL>
User altered.

SQL>
User altered.

SQL>
Session altered.

SQL>
User altered.
```

Create a User

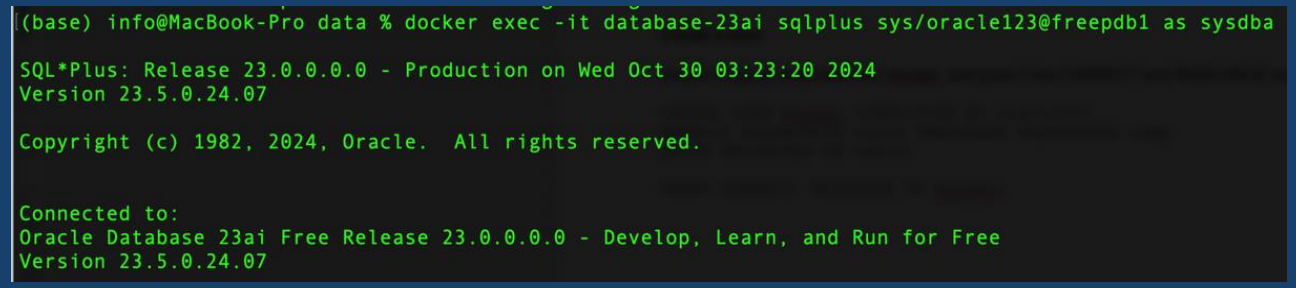
- Log in to the database as the administrator user (= sys).

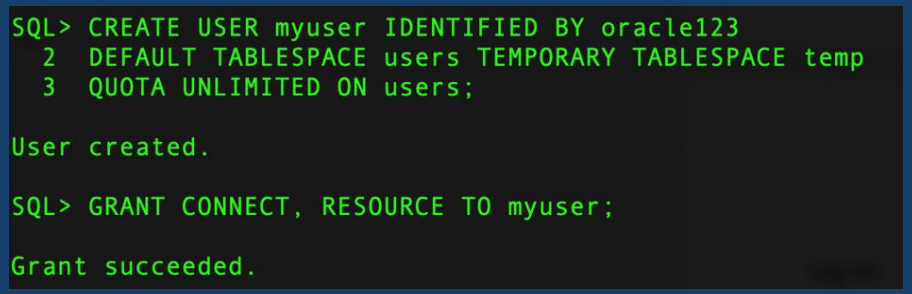
```
podman exec -it database-23ai sqlplus sys/oracle123@freepdb1 as sysdba
```

- Create a regular database user myuser, and grant it the CONNECT and RESOURCE roles.

```
CREATE USER myuser IDENTIFIED BY oracle123
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON users;

GRANT CONNECT, RESOURCE TO myuser;
```



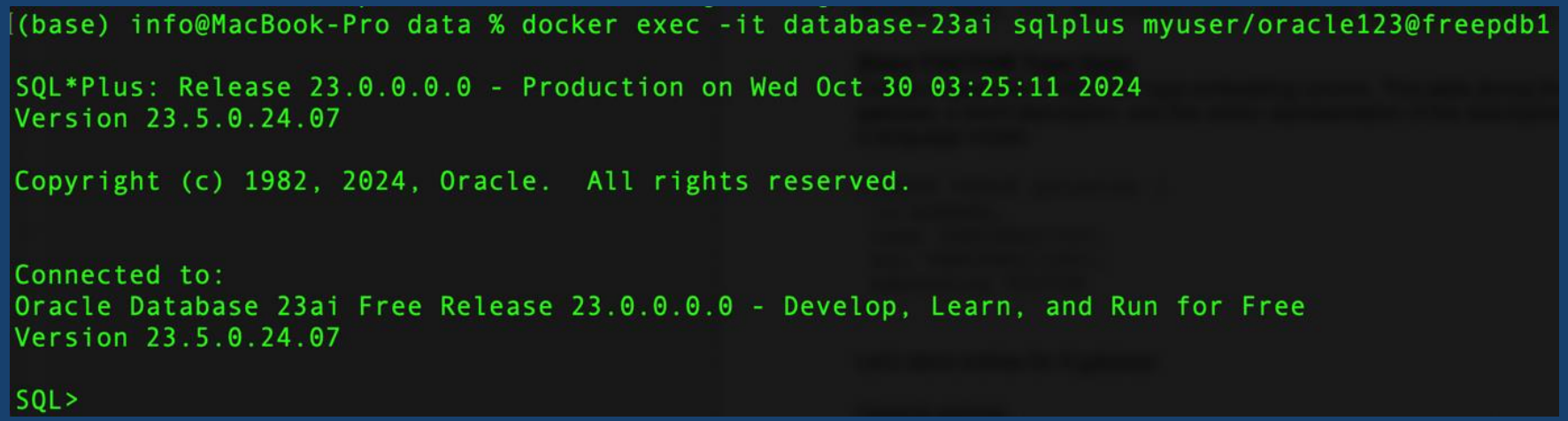


Log out and Verify with New User.

```
exit
```

- Verify you can connect with the new user.

```
podman exec -it database-23ai sqlplus myuser/oracle123@freepdb1
```



Store VECTOR Type Data

- Create a table with a VECTOR type embedding column.
- This table stores the names of various galaxies, a short description, and the vector representation of the description retrieved through a language model.

```
CREATE TABLE galaxies (
 id NUMBER, name VARCHAR2(50), doc VARCHAR2(500),
 embedding VECTOR );
```

Store Some Data

- Let’s store entries for 9 galaxies. (5 dimensions. Can be higher dimensions):

```
M31: [0.26833928, 0.012467232, -0.48890606, 0.61341953, 0.5590402]
M33: [0.43291375, -0.06379729, -0.40366283, 0.77222455, 0.2219036]
M58: [-0.07266286, 0.0802545, -0.34327424, 0.8917586, 0.27424216]
```

- The vectors are passed as strings to the INSERT statement as follows.

```
INSERT INTO galaxies VALUES (1, 'M31', 'Messier 31 is a barred spiral galaxy in the Andromeda constellation which has a lot of barred spiral galaxies.', '[0.26833928, 0.012467232, -0.48890606, 0.61341953, 0.5590402]');
INSERT INTO galaxies VALUES (2, 'M33', 'Messier 33 is a spiral galaxy in the Triangulum constellation.', '[0.43291375, -0.06379729, -0.40366283, 0.77222455, 0.2219036]');
INSERT INTO galaxies VALUES (3, 'M58', 'Messier 58 is an intermediate barred spiral galaxy in the Virgo constellation.', '[-0.07266286, 0.0802545, -0.34327424, 0.8917586, 0.27424216]');
INSERT INTO galaxies VALUES (4, 'M63', 'Messier 63 is a spiral galaxy in the Canes Venatici constellation.', '[0.16099164, -0.28416803, -0.32071748, 0.7423811, 0.4892247]’);
INSERT INTO galaxies VALUES (5, 'M77', 'Messier 77 is a barred spiral galaxy in the Cetus constellation.', '[0.43336692, -0.20248333, -0.38971466, 0.6521541, 0.44046697]');
INSERT INTO galaxies VALUES (6, 'M91', 'Messier 91 is a barred spiral galaxy in the Coma Berenices constellation.', '[0.41154075, 0.14537011, -0.42996794, 0.33680823, 0.7149753]');
INSERT INTO galaxies VALUES (7, 'M49', 'Messier 49 is a giant elliptical galaxy in the Virgo constellation.', '[0.45378348, 0.37351337, -0.33156702, 0.7252909, 0.13632572]');
INSERT INTO galaxies VALUES (8, 'M60', 'Messier 60 is an elliptical galaxy in the Virgo constellation.', '[0.124404885, 0.1522709, -0.27538168, 0.9206997, 0.19445813]');
INSERT INTO galaxies VALUES (9, 'NGC1073', 'NGC 1073 is a barred spiral galaxy in Cetus constellation.', '[-0.11507724, -0.3145153, -0.42180404, 0.50861514, 0.67173606]');
```

```
COMMIT;
```

Nearest Neighbour Vector Search

- Based on the descriptions, calculate the distance between galaxy ID = 1 and other galaxies, ordering them by closeness (i.e., similarity in description).
- These three lines are for formatting the results if you're using `sqlplus`.

```
col galaxy_1 for a10
col galaxy_2 for a10
col distance for 0.00000000000000000
```

The VECTOR_DISTANCE function in 23ai.

- This function returns the distance between two vectors as a number:

```
SELECT g1.name AS galaxy_1, g2.name AS galaxy_2,
VECTOR_DISTANCE(g2.embedding, g1.embedding) AS distance
FROM galaxies g1, galaxies g2
WHERE g1.id = 1
ORDER BY distance ASC;
```

```
SQL> SELECT g1.name AS galaxy_1, g2.name AS galaxy_2,
VECTOR_DISTANCE(g2.embedding, g1.embedding) AS distance
FROM galaxies g1, galaxies g2
WHERE g1.id = 1
ORDER BY distance ASC;

GALAXY_1   GALAXY_2               DISTANCE
---------- ---------- --------------------
M31        M31        -0.00000011920928955
M31        M77         0.04941838979721069
M31        M91         0.07123649120330811
M31        M63         0.07465428113937378
M31        M33         0.08952373266220093
M31        NGC1073     0.14105635881423950
M31        M58         0.15033429861068726
M31        M60         0.15659791231155396
M31        M49         0.19035100936889648

9 rows selected.
```

Specify Distance Calculation Method

- Since the distance calculation method (= metric) was not specified, the default cosine distance was calculated.
- You can also specify it with COSINE:

```
SELECT g1.name AS galaxy_1, g2.name AS galaxy_2,
VECTOR_DISTANCE(g2.embedding, g1.embedding, COSINE) AS distance
FROM galaxies g1, galaxies g2
WHERE g1.id = 1 ORDER BY distance ASC;
```

```
GALAXY_1   GALAXY_2               DISTANCE
---------- ---------- --------------------
M31        M31        -0.00000011920928955
M31        M77         0.04941838979721069
M31        M91         0.0712364912033081
M31        M63         0.07465428113937378
M31        M33         0.08952373266220093
M31        NGC1073     0.14105635881423950
M31        M58         0.15033429861068726
M31        M60         0.15659791231155396
M31        M49         0.19035100936889648

9 rows selected.
```

- Besides cosine distance, you can use DOT, EUCLIDEAN, EUCLIDEAN_SQUARED, HAMMING, and MANHATTAN.

– Thank You