**Vector Database:**

**This is about how RAG-based LLMs use vector databases to give better responses to a user query.**

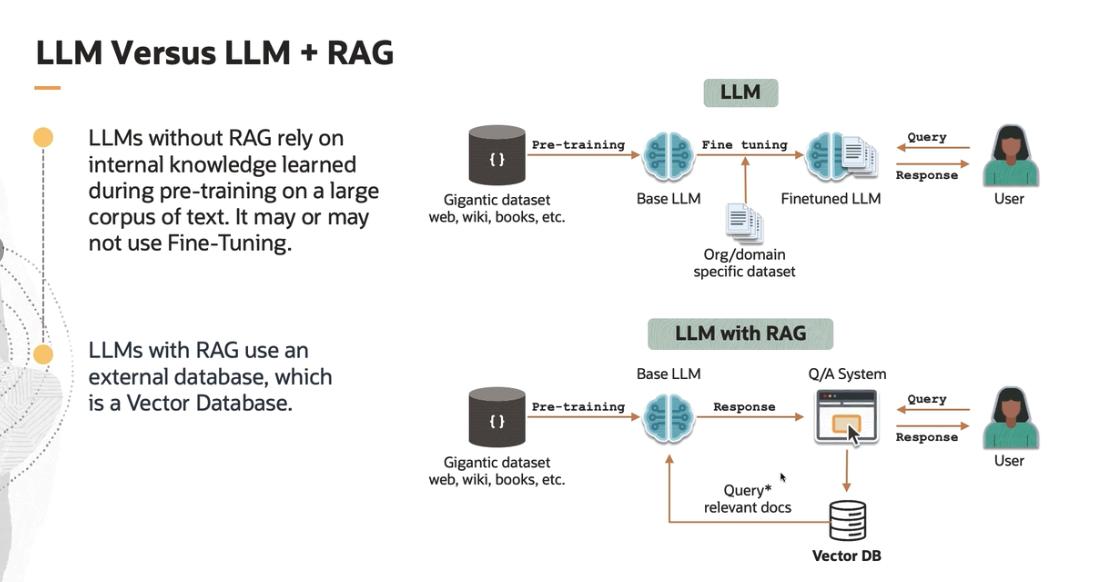
This is possible because LLMs with RAG augment- It is process by using an external database, which is a vector database.

- **LLMs without RAG** typically do not use vector database in their standard operation. Instead, they rely on internal knowledge learned during the pre-training on a large corpus of text.

- Once these models are trained, they don't automatically know about the latest happenings or the information locked away in private documents that weren't part of their training materials.

- The model's parameters encode information and patterns from the data it was trained on, which it then uses to generate responses to queries based on its understanding acquired during this pre-training. This could be with or without even fine-tuning.

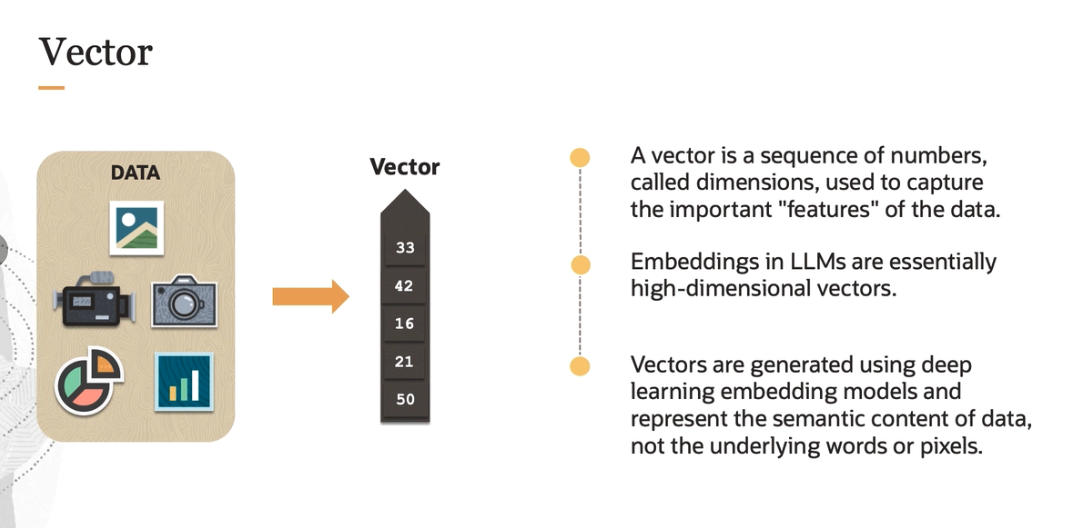
**- LLMs with RAG** augment this process by using an external database, which is a vector database. Vector databases are a core component of RAG-based LLMs. They are a type of database optimized for storing and querying vectors instead of a traditional rule-based.



**Vector:**

Data is represented as vectors in a multi-dimensional space. Each vector representing a complex data item like a video, image, text, or a set of features extracted by your machine learning model. In the context of machine learning and particularly with large ML models, vectors are often used to represent embeddings, which are essentially high dimensional vectors.

- Vector embeddings are generated by deep learning models to capture semantic meaning of words, sentences, documents, images, or other data types. And they can be used to compute similarity between items efficiently.

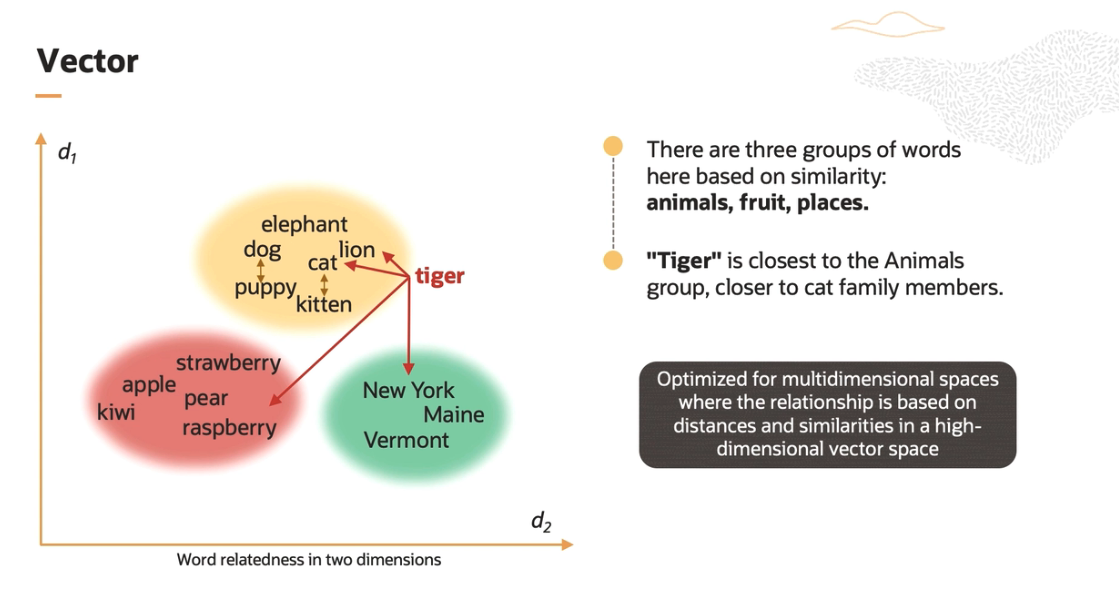


**How were these vectors represented?**

The structure of vector database is fundamentally different from traditional relational databases. They are optimized for multi-dimensional spaces, where the relationship between data points is not linear or tabular, but is instead based on distances and similarities in a high-dimensional vector space.

- Vector databases are particularly adept at handling operations that involve searching the meaning and nearest neighbor queries in high-dimensional spaces. Many vector databases uses a distributed architecture to handle the storage and computational demands of large scale high-dimensional data. This setup allows for horizontal scaling, improving performance and storage capacity.

- Imagine, there are three groups of words based on similarity-- animals, fruit, and places. If a user inputs another word "Tiger," it is closest to animals and even closer to the cat family.



**How were these distance calculated?**

The **dot product** and **cosine distance** are commonly used in the field of NLP to evaluate how similar or different two text embeddings are.

- **dot product** is the measure of the magnitude of the projection of one vector onto the other and gives you magnitude and direction.

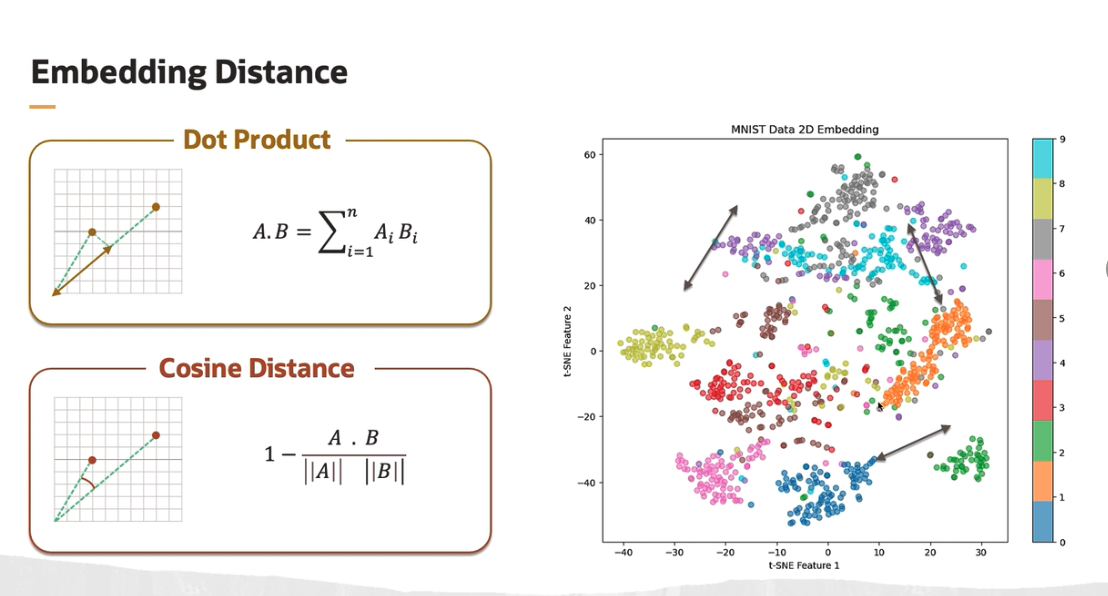
- **Cosine distance** is the measure of difference in the directionality between vectors so that only the angle between the vectors matters, not their magnitude.

**Eg**: Imagine you and your friend are pushing a car. The dot product helps us understand how much of your pushing effort influences the movement of the car in the same direction that your friend is pushing and vice versa.

A high positive value means both are pushing in a very similar direction, suggesting that their embeddings are similar. A low value that is close to 0 means they are pushing in more orthogonal or unrelated directions. So their embeddings are not that similar. A negative value means they are pushing in opposite directions, indicating their embeddings are dissimilar.

- Cosine distance refines this by focusing only on the direction, not on the strength of the push.

- A cosine distance of 0 means that you both are pushing in the same direction, and the embeddings are similar. A cosine distance of 1 means your efforts are, again, orthogonal, so the embeddings are unrelated.



The image here is a scatter plot of the MNIST handwritten data set reduced to two dimension using t-SNE, which is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional data sets.

The plot shows cluster of points in different colors, each representing one of the digits from 0 to 9 from the data set. You can clearly see that how this technique was able to separate and group embeddings in case of images. So, for example, if you see this particular color zero, these are all the images representing the digit 0. Similarly, for one, and so on.

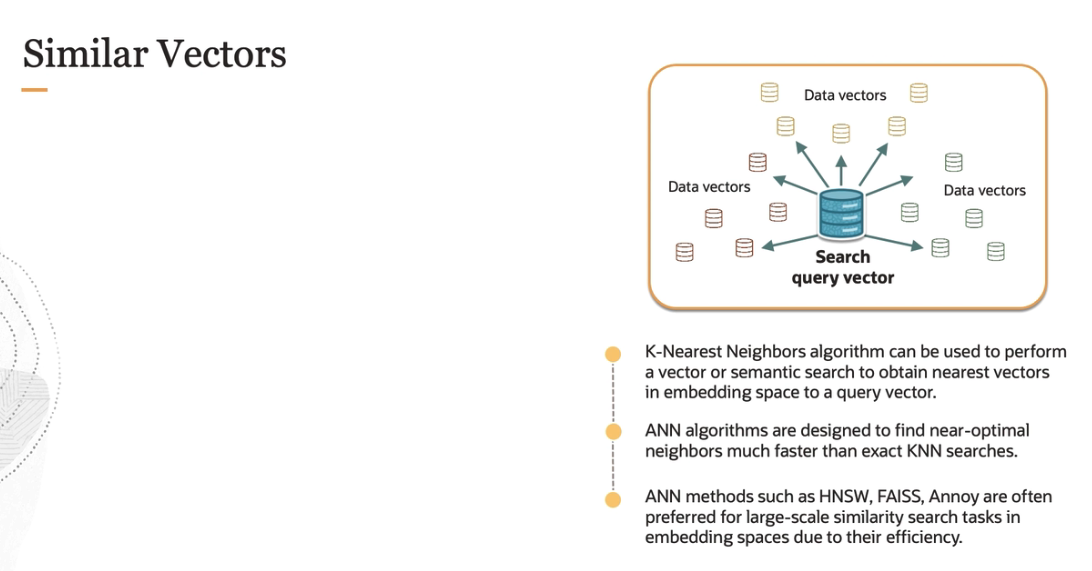
**How the nearest vectors are obtained with a given query or a word?**

k nearest neighbors algorithm can be used to accurately obtain the nearest vectors in embedding space to a query vector to perform a vector or semantic search.

- The distances between all the vectors and the query vectors or a query vector are calculated. Then those distances are sorted. And finally, the top k best matching objects with the best distance are returned. So this is a known classical machine learning method, but it's not feasible for real life applications due to use computation costs, which might be incurred due to a large vector database.

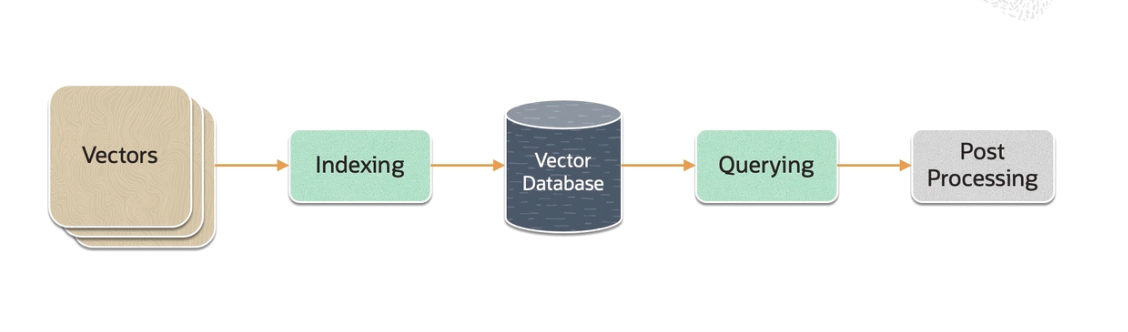
- Also, this will be too slow. And for those reasons, approximate nearest neighbor or ANN algorithms are used, which are designed to find nearest optimal neighbors much faster than exact KNN searches.

- They trade off a bit of accuracy for significant gains in speed and reduced memory consumptions. Again, the choice of algorithm depends on several factors, including the size of the data set, the dimensionality of the vectors, and the desired balance between accuracy and speed.



- In practice, ANN methods are often preferred for large scale similarity search tasks in embedding spaces due to their efficiency. For real world application, it's common to start with an ANN library like files from Meta, Annoy from Spotify, or HNSW, as these have been widely used and benchmarked across different scenarios.

**Vector Database workflow**:



The first is **vectors**. This represent the data set of high-dimensional vectors that need to be stored and queried. These vectors could be embedding of text, images, audio, or other data types generated by models like neural networks. They encapsulate rich, context-aware information about the data.

The second part in workflow is **indexing**. So the vector database indexes these vectors using an algorithm such as HNSW. This step maps the vector to a data structure that will enable faster search.

Before vectors can be queried effectively, they need to be indexed. Indexing organizes the vectors in a way that allows for efficient retrieval. This is often done using data structures designed for high-dimensional spaces such as tree structures or using approximate methods to facilitate faster searches at the cost of some accuracy.

The third is **vector database**. Again, this is the storage system specialized for vector data. After indexing, vectors are stored here. A vector database is optimized to handle the high-dimensional nature of the data, enabling efficient storage, search, and retrieval operations.

Then comes the **querying**. The vector database compares the indexed query vector to the indexed vectors in the data set to find the nearest neighbors. When a query vector is submitted to the system, the vector database performs a search to find the most similar vectors. Similarity is typically measured using distance metrics like we saw earlier-- cosine similarity or cosine distance.

And then comes the **post-processing**. So in some cases, the vector database retrieves the final nearest neighbors from the data set and post-processes them to return the final results. This step can include reranking the nearest neighbors using a different similarity measure.

Some Vector databases are:



**Important features, which makes these vector databases so desirable with LLMs.**

**1.** first one is **accuracy**. Vector databases store data in a way that preserves semantic relationships, which is crucial for the accuracy of language models. Traditional databases may not effectively capture these nuanced relationships.

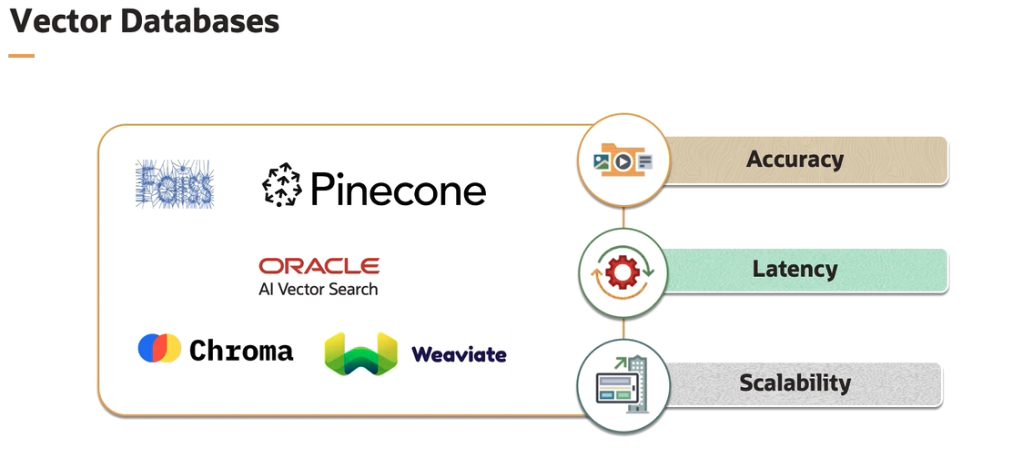
Accurate language modeling requires understanding context, which is embedded in the high-dimensional data vectors. Vector databases are better at managing this context leading to a more precise language generation.

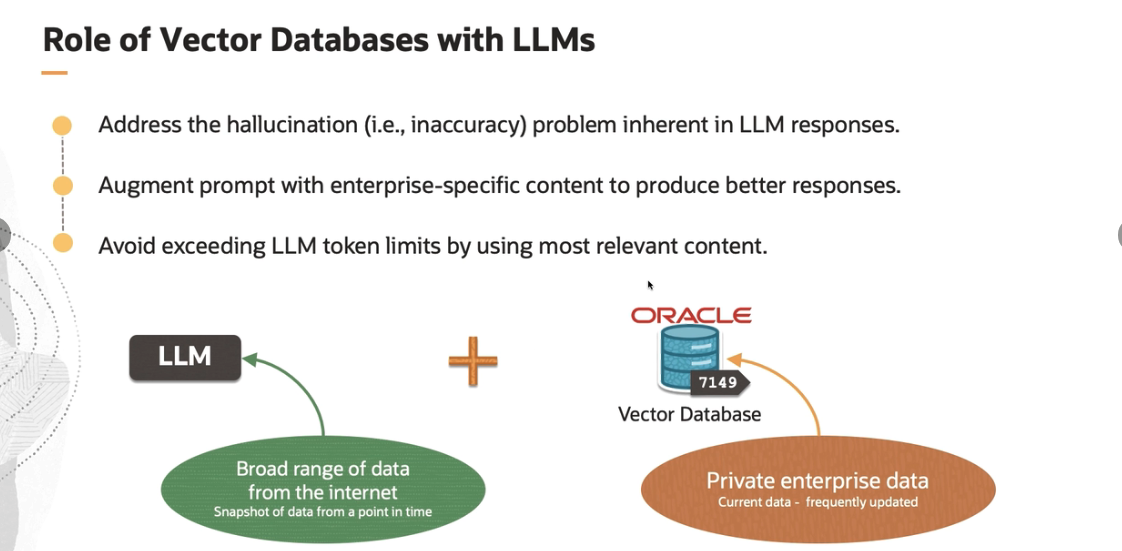
**2.** The second is **latency**. Vector databases are optimized for quick retrieval of high-dimensional data, a key requirement for reducing latency in LLMs. Traditional databases might not offer the same level of efficiency in data retrieval. The indexing mechanisms in vector databases are designed for the efficient searching of high-dimensional data. And this efficiency is crucial for reducing the response time of LLMs in real time applications.

Vector databases are also optimized for parallel processing, which is essential for handling multiple queries simultaneously, which is very important in real time interaction like chat bots and virtual assistants.

3. The third one is **scalability**. LLMs handle enormous data sets. Traditional databases struggle with the sheer volume of the data and its high-dimensional nature. Vector databases, however, are specifically designed for this scale, which efficiently manage large volumes of complex data. Also, vector databases are adept at managing complex structures, something traditional relational databases are not inherently designed for.

So in the real world applications, vector databases address the problem of hallucination. So hallucination refers to instances where the model generates text that is factually incorrect or nonsensical, even though it might be coherent and contextually relevant in style and tone.





**What business prospects these vector databases bring?**

It's cheaper than fine-tuning since you are not updating the model layers. Again, it helps with real time knowledge. And it can cache previous prompts and responses to give more contextual responses. So overall, vector databases brings better business outcome with reduced cost.

