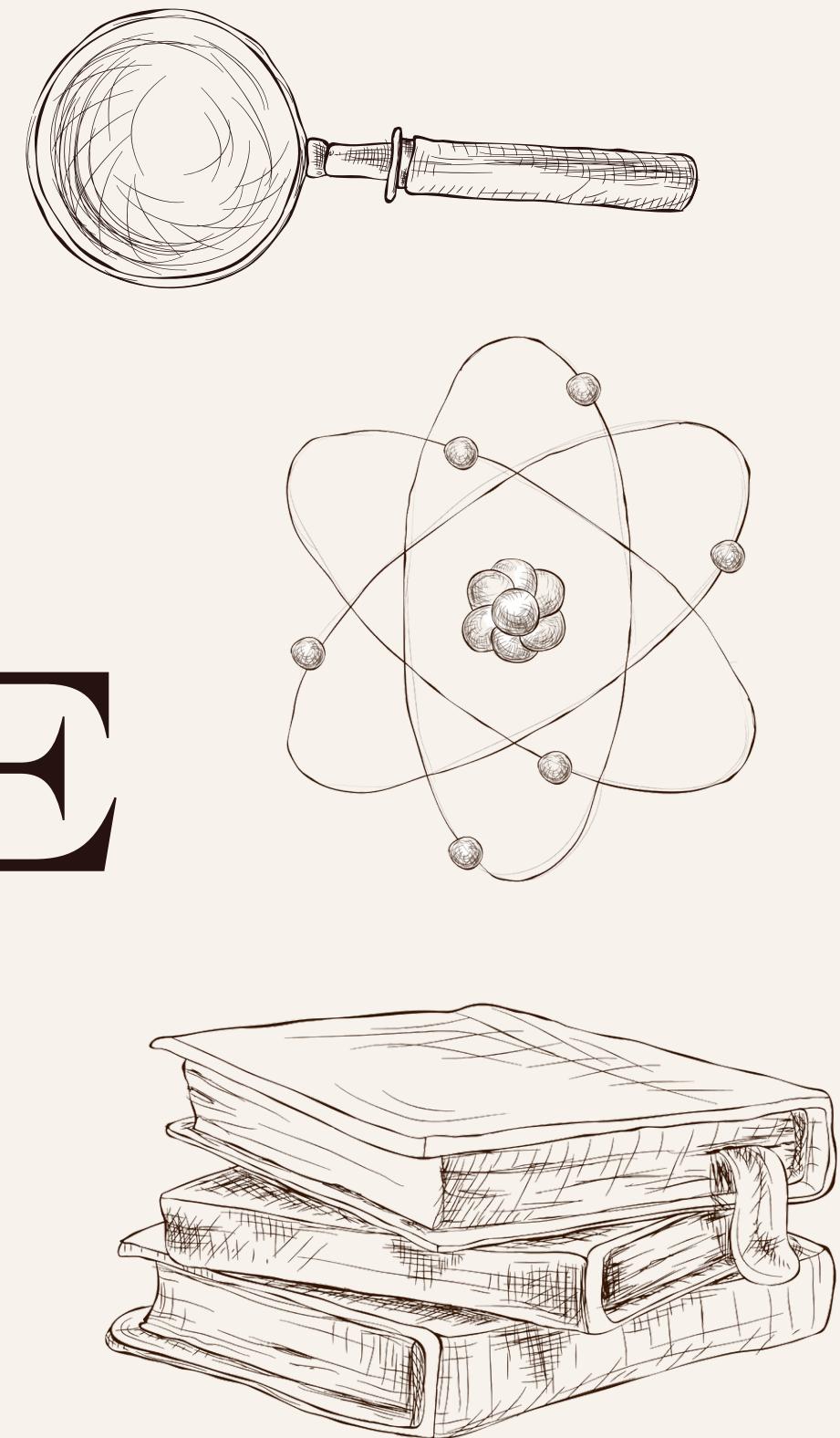
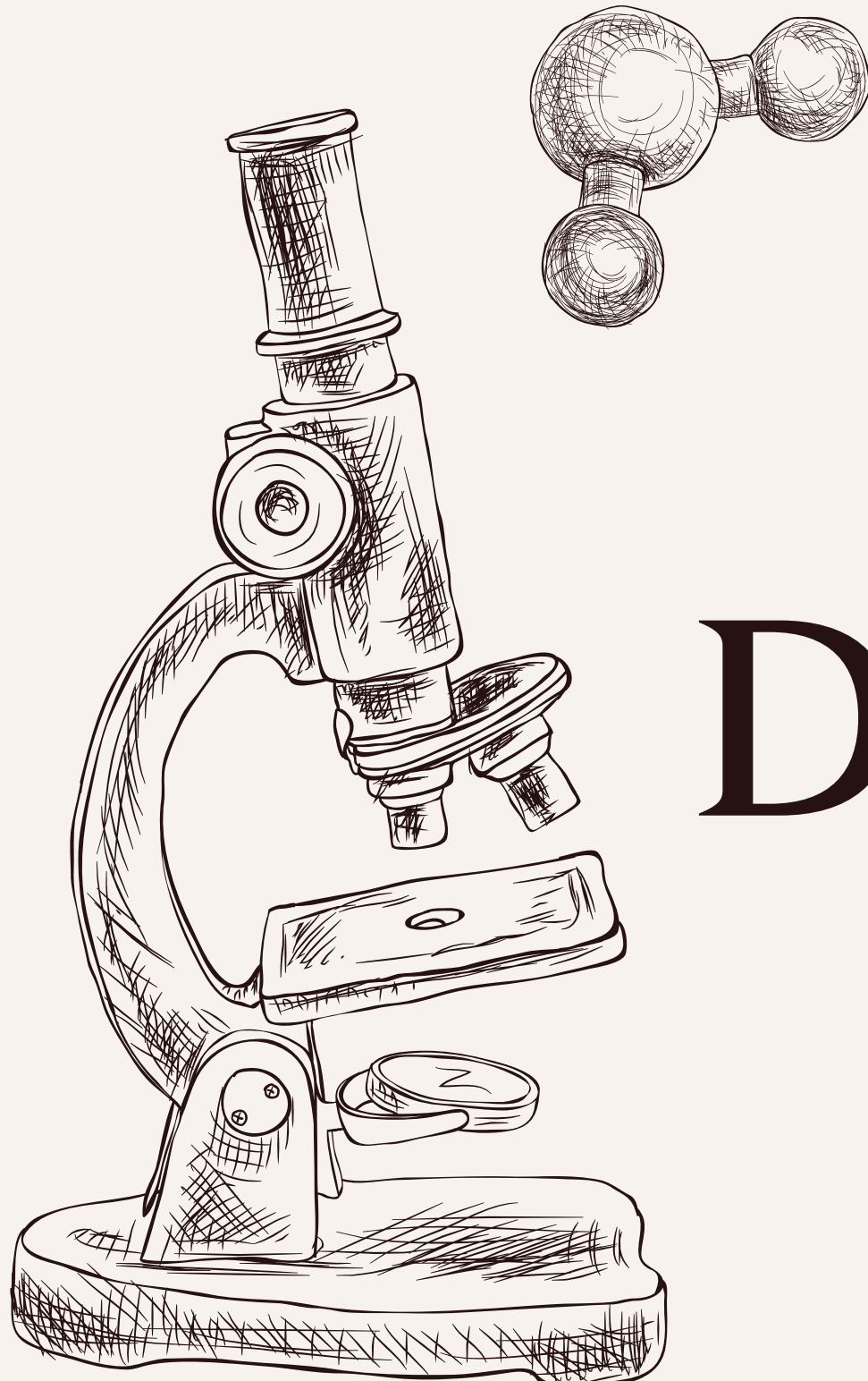


VECTOR DATABASE

Presented by Nguyễn Tiến Thắng



Các cơ sở dữ liệu trước đây:

- + SQL: Dữ liệu có cấu trúc, sử dụng khóa, bảng để mô tả thuộc tính.
 - +NoSQL: Dữ liệu bán cấu trúc và phi cấu trúc. Ở dạng kho tài liệu.
- => Khó mở rộng dữ liệu, hạn chế tìm kiếm nếu không biết cấu trúc, thông tin được lưu.

Ví dụ: Ta muốn tìm bộ phim có nội dung giống phim "Ba chàng ngự lâm". Thể loại: Phim hài.

Với SQL: có thể tìm sự tương đồng thông qua thể loại bằng câu lệnh sql:

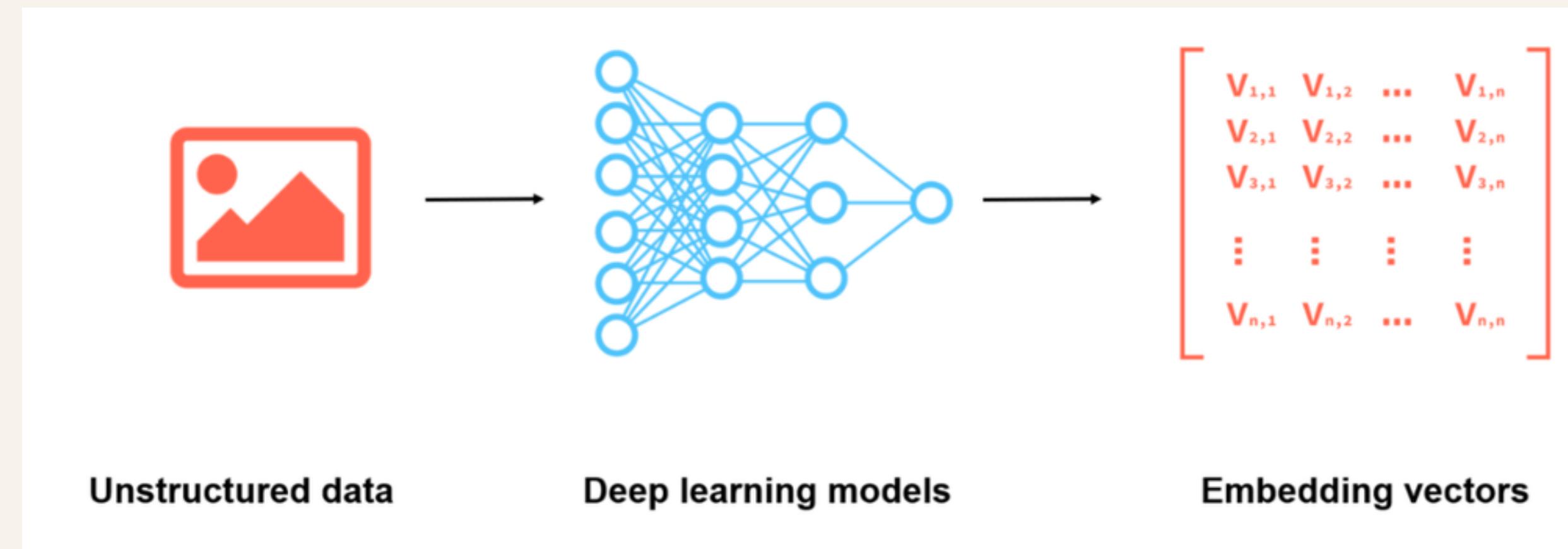
Select * from phim where theloai="Phim hài"

Kết quả trả về là phim siêu ngốc gấp nhau dù 4 bộ phim đều có thể loại gần giống nhau.

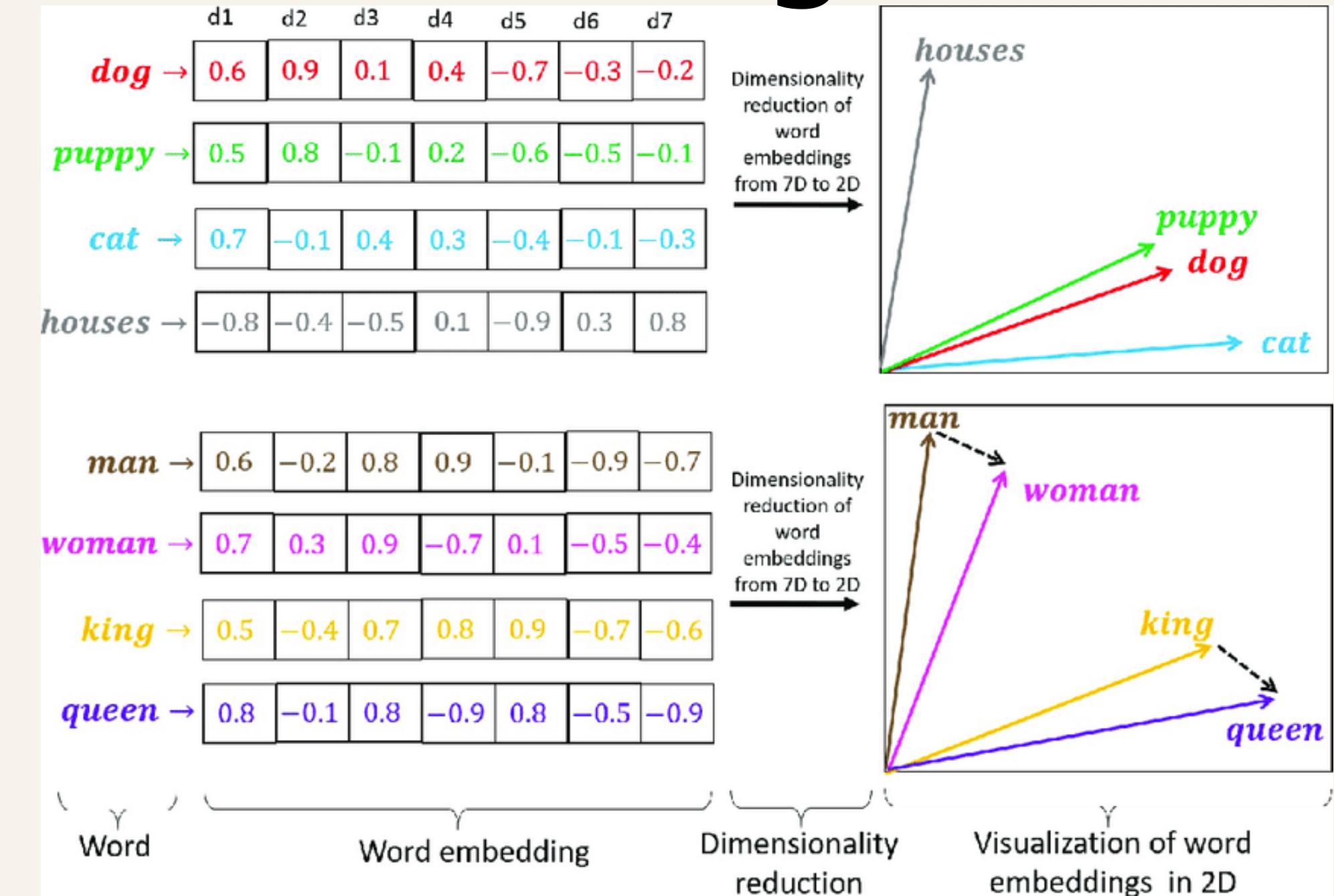
id	ten	the loai	dao dien
001	Ở nhà một mình	Hài hước	Chris Columbus
002	Mr. Bean	Hài kịch	Mel Smith
003	Siêu ngốc gặp nhau	Phim hài	Bobby Farrelly, Peter Farrelly
004	Ba chàng ngốc	Hài hước	Rajkumar Hirani

1) Vector embedding

- Để các mô hình AI có thể sử dụng các dữ liệu phi cấu trúc: văn bản, video, audio,
Việc phải làm đó là chuyển đổi các dữ liệu này về cùng một kiểu.
- Vector đang được sử dụng là kiểu cấu trúc để biểu diễn các dữ liệu phi cấu trúc đó.
- Các dữ liệu sẽ được chuyển thành vector thông qua một quá trình nhúng (embedding).



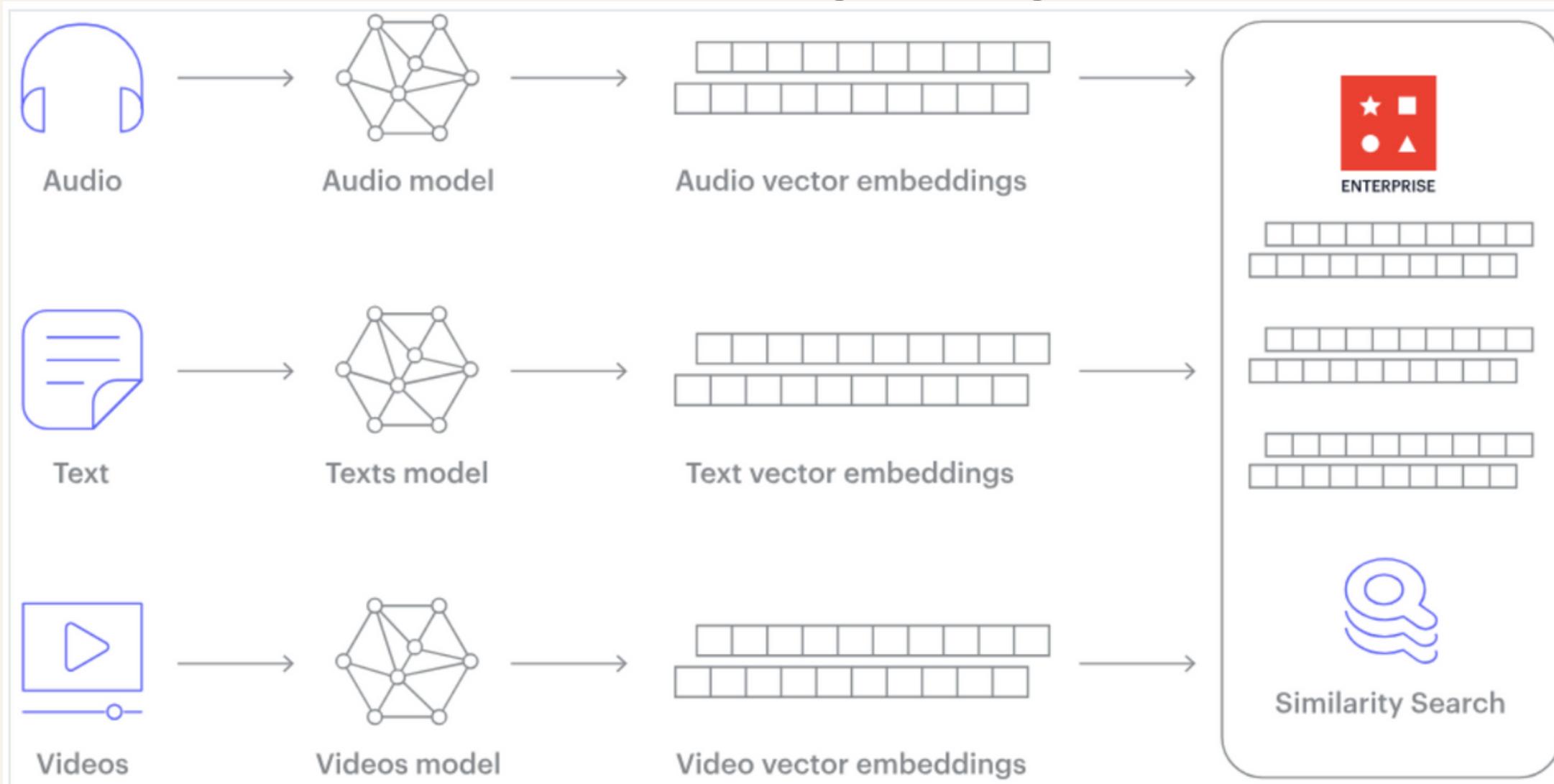
1) Vector embedding



Các vector sẽ biểu diễn đặc trưng cho các dữ liệu nên khoảng cách giữa của các vector biểu diễn cho các dữ liệu giống nhau sẽ gần nhau trong không gian.

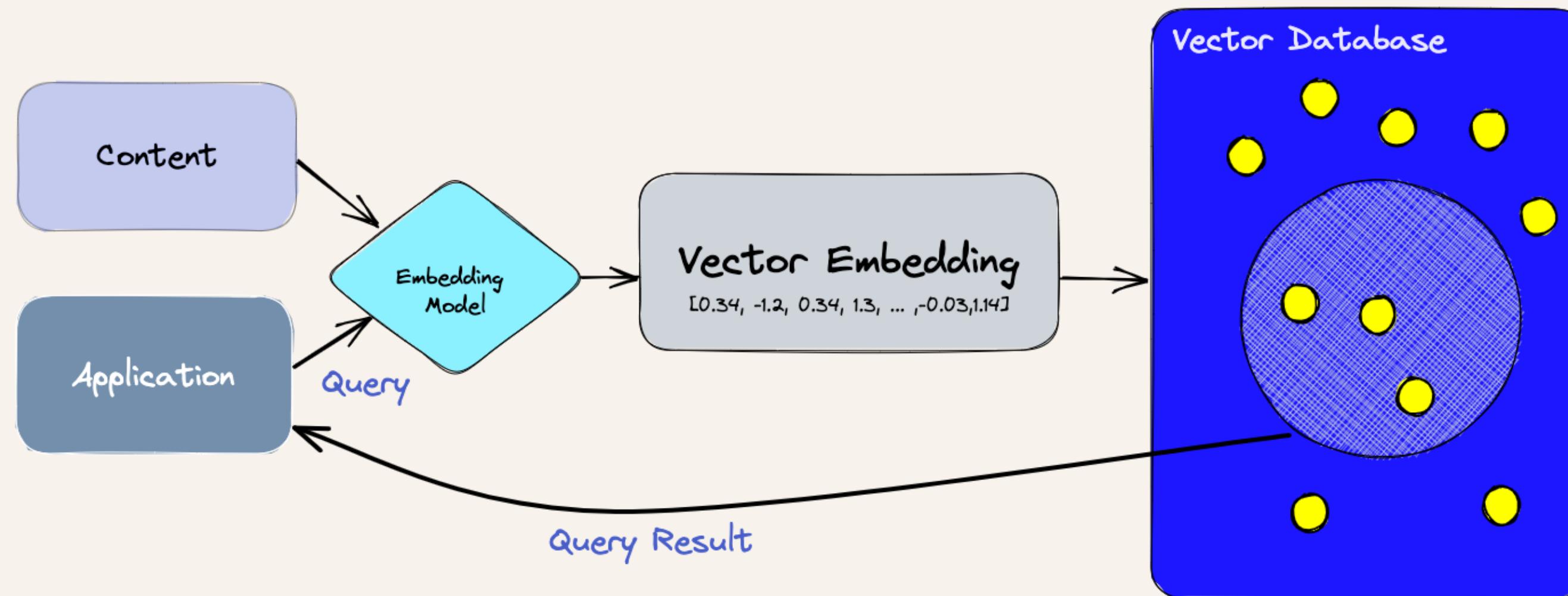
1) Vector embedding

- Với tính chất đó, đầu tiên chúng ta sẽ chuyển đổi tất cả dữ liệu thông qua embedding model về dạng embedding vector
- Sau đó thực hiện một thuật toán khoảng cách (có thể là Euclidean, Cosine,...) để chọn ra các vector có khoảng cách gần nhất với dữ liệu tìm kiếm.



2) Vector database.

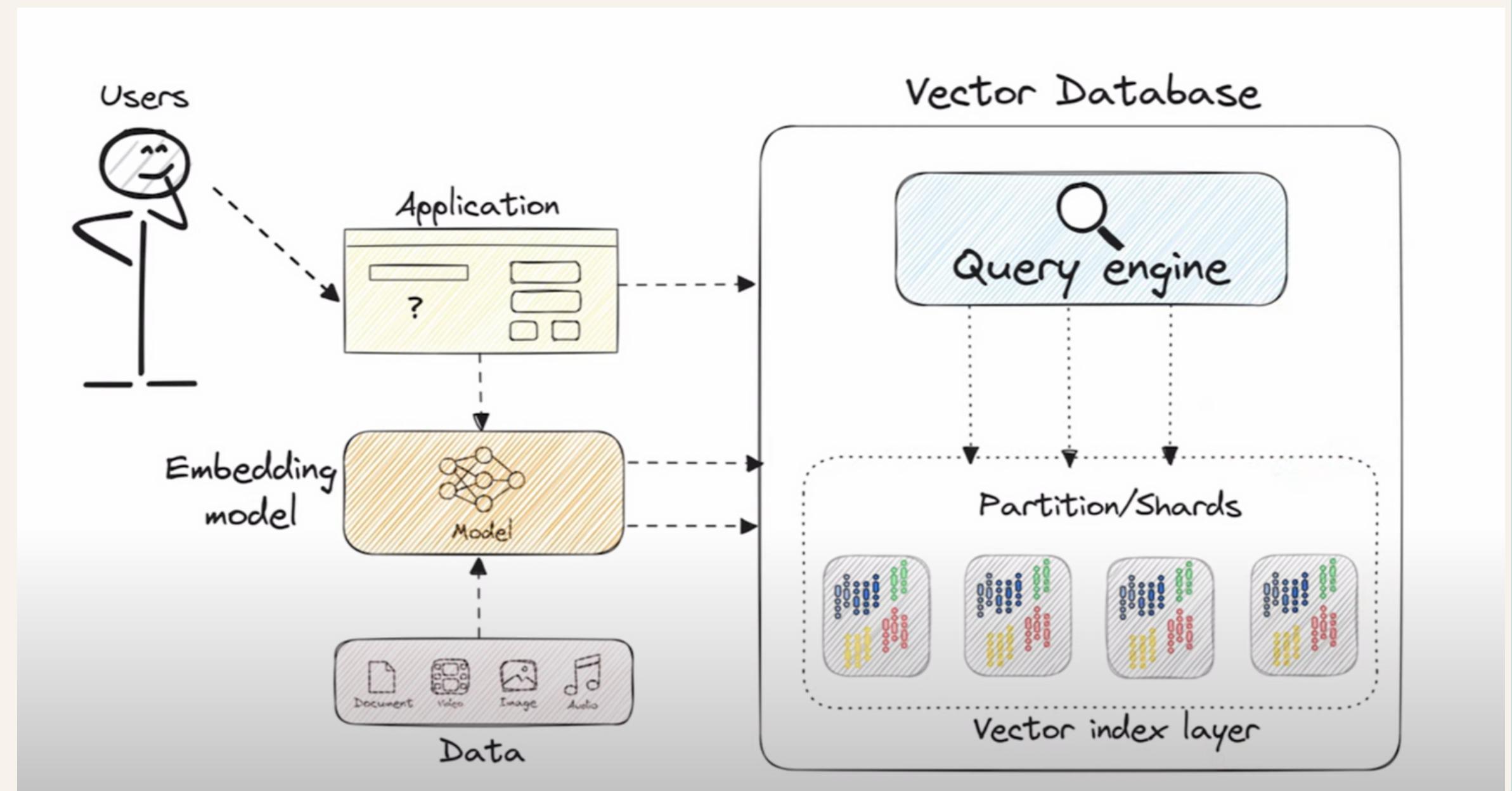
Vector database chứa các vector được chuyển đổi từ dữ liệu ảnh, video, text và quy định cách tính khoảng cách giữa 2 vector.



2) Vector database.

Cách thức hoạt động của vector database:

- Chuyển đổi dữ liệu thành vector và lưu trữ.
- Đánh index cho các dữ liệu để có thể tìm kiếm và lấy dữ liệu ra.
- Chuyển đổi yêu cầu người dùng sang vector yêu cầu.
- Tìm vector trong vector database có khoảng cách nhỏ nhất với vector yêu cầu.
- Trả về kết quả.



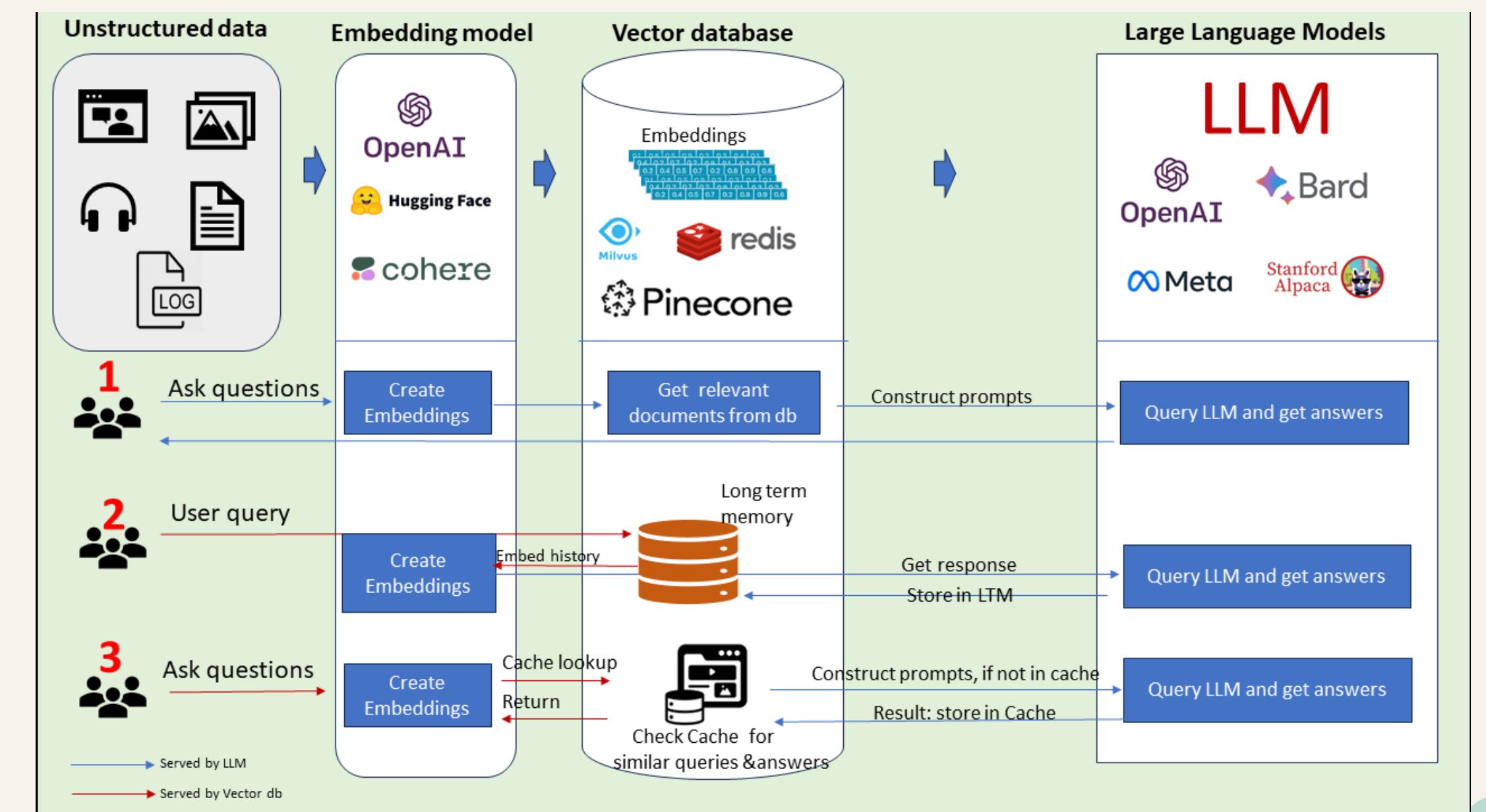
3) Ứng dụng của Vector database

- Có thể dễ dàng tìm kiếm sự tương đồng, tìm kiếm tương tự. Ứng dụng cho việc gợi ý mua hàng, gợi ý xem phim.
- Và ứng dụng đặc biệt khiến vector database thu hút được sự chú ý và đầu tư lớn đó là khả năng tăng cường dữ liệu cho các mô hình ngôn ngữ lớn.

- Pinecone (\$138M funding)
- Zillis (\$113M funding)
- Weaviate (\$67.7M funding)
- Chrome (\$20.3M funding)
- Qdrant (\$9.8 M funding)

3) Ứng dụng của Vector database

Ví dụ ChatGPT chỉ có dữ liệu đến năm 2021, nó sẽ không thể trả lời các dữ liệu ở thời điểm mới, từ 2022 đến nay. Và vector database sẽ giúp chatgpt khắc phục được dữ liệu này, thông qua việc thêm những dữ liệu mới vào mô hình.



4) Các thuật toán search vector.

Khi khối lượng dữ liệu lớn lên từng ngày, thách thức dành cho các mô hình AI đó là làm sao để có thể tìm kiếm vector trong kho dữ liệu khổng lồ một cách nhanh nhất.

 Pinecone	Proprietary composite index
 milvus / zilliz	Flat, Annoy, IVF, HNSW/RHNSW (Flat/PQ), DiskANN
 Weaviate	Customized HNSW, HNSW (PQ), DiskANN (in progress...)
 drant	Customized HNSW
 chroma	HNSW
 LanceDB	IVF (PQ), DiskANN (in progress...)
 vespa	HNSW + BM25 hybrid
 Vald	NGT
 elasticsearch	Flat (brute force), HNSW
 redis	Flat (brute force), HNSW
 pgvector	IVF (Flat), IVF (PQ) in progress...

4) Các thuật toán search vector.

- **Linear search**

- + Tính khoảng cách từ vector yêu cầu đến tất cả các vector trong database.
- + Vì lí do dữ liệu, Linear search không hoạt động tốt khi dữ liệu trở nên lớn, mà nó chỉ hoạt động tốt trong không gian dữ liệu nhỏ, vừa.

- **Space partitioning (Phân vùng không gian)**

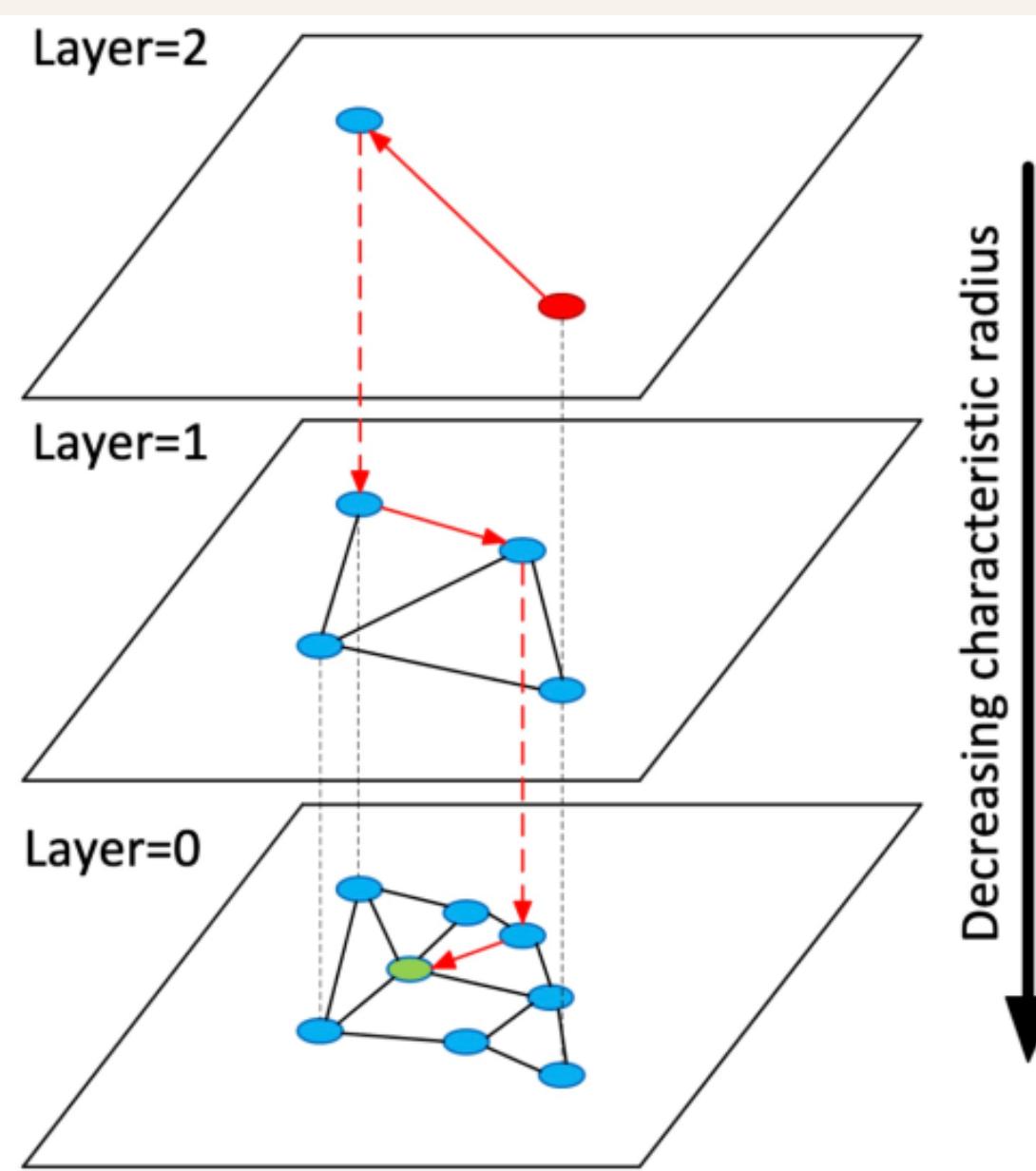
- + Space partitioning là một nhóm thuật toán sử dụng cùng một khái niệm.
- + K-dimensional trees (kd-trees) có lẽ là cây nổi tiếng nhất trong họ này và hoạt động bằng cách liên tục chia đôi không gian tìm kiếm (chia vector thành các nhóm “trái” và “phải”) theo cách tương tự như cây tìm kiếm nhị phân .

- **Quantization (Lượng tử hóa)**

- + Quantization là một kỹ thuật nhằm giảm tổng kích thước của cơ sở dữ liệu bằng cách giảm độ chính xác của vector.
- + Ví dụ: Một kỹ thuật Quantization là sử dụng việc nhân vector với đại lượng vô hướng để chuyển các phần tử của vector sang số nguyên gần nhất chúng.

4) Các thuật toán search vector.

- Hierarchical Navigable Small Worlds (HNSW)



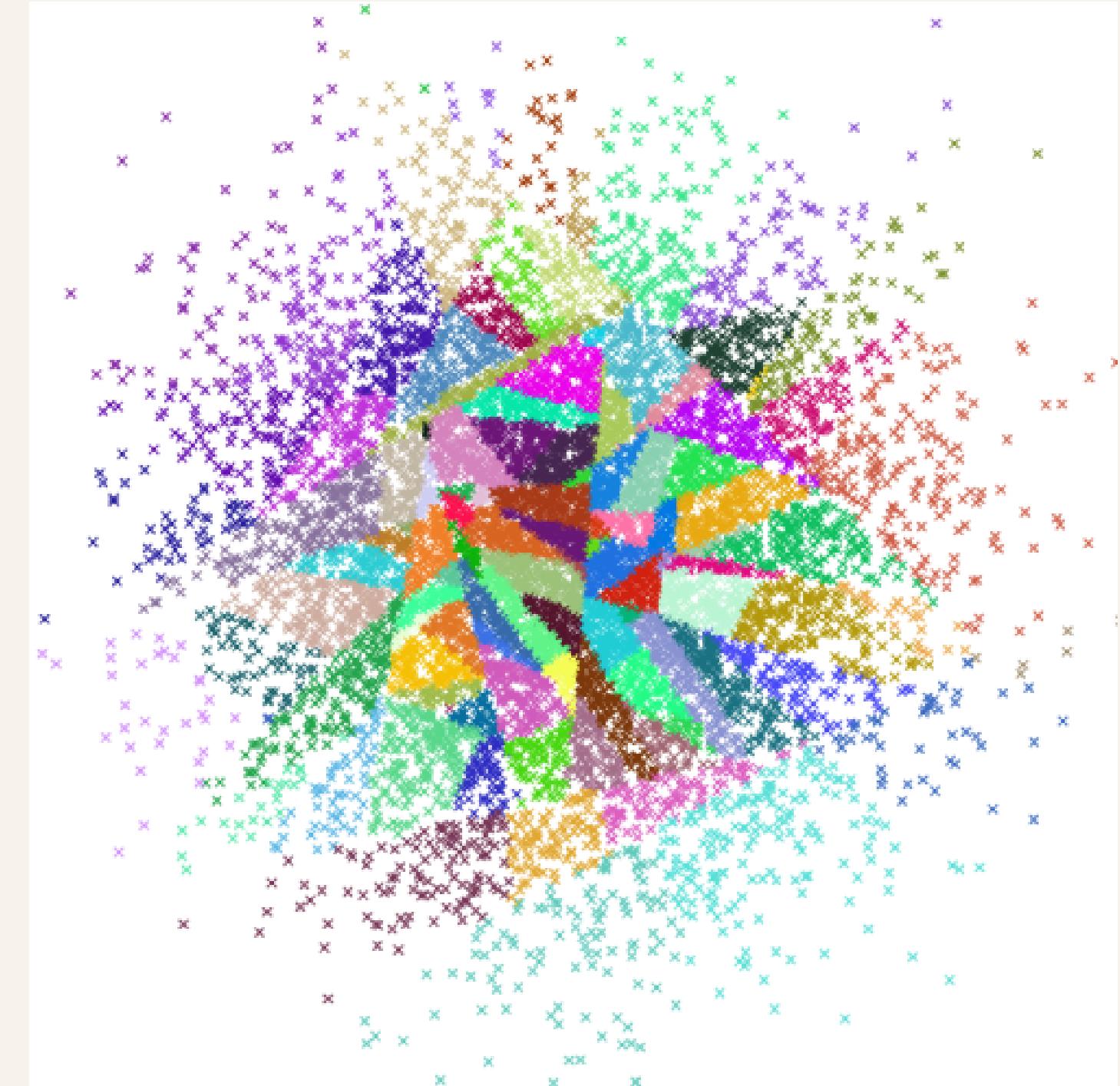
- HNSW là một thuật toán truy xuất và lập index dựa trên đồ thị.
- HNSW tạo biểu đồ nhiều lớp từ dữ liệu gốc.
- Các lớp trên chỉ chứa "kết nối dài" trong khi các lớp dưới chỉ chứa "kết nối ngắn" giữa các vector trong cơ sở dữ liệu. Các kết nối biểu đồ riêng lẻ được tạo theo danh sách bỏ qua.
- Với kiến trúc này, việc tìm kiếm trở nên khá đơn giản – chúng ta nhanh chóng duyệt qua biểu đồ trên cùng (biểu đồ có kết nối giữa các vectơ dài nhất) để tìm vectơ gần nhất với vectơ truy vấn của chúng ta.

4) Các thuật toán search vector.

- **Approximate Nearest Neighbors Oh Yeah (ANNOY)**

ANNOY hoạt động bằng cách trước tiên chọn ngẫu nhiên hai vectơ trong cơ sở dữ liệu và chia đôi không gian tìm kiếm dọc theo siêu phẳng ngăn cách hai vectơ đó.

Việc này được thực hiện lặp đi lặp lại cho đến khi có ít hơn một số tham số được xác định trước số vector quy định.

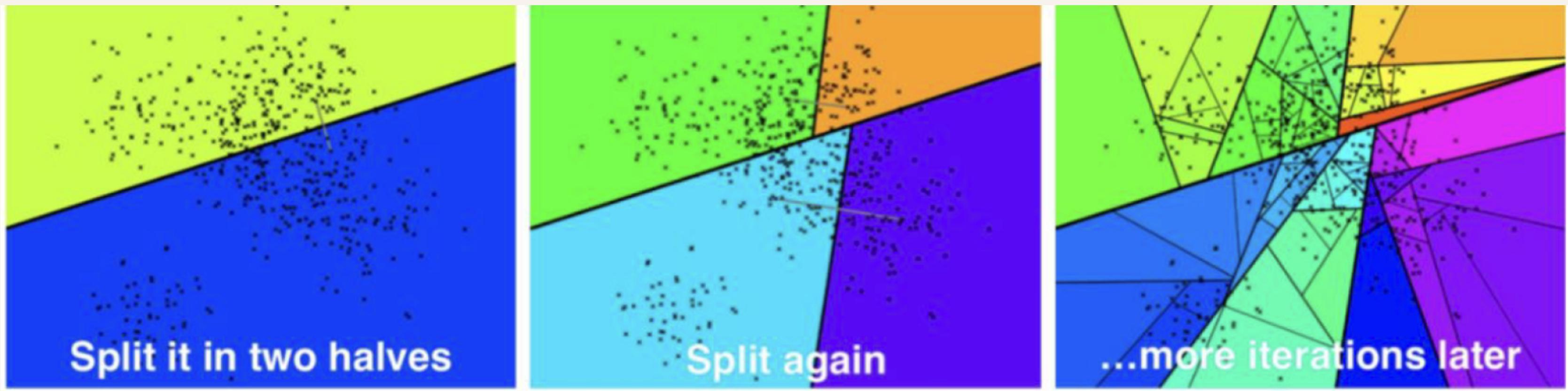


4) Các thuật toán search vector.

- Approximate Nearest Neighbors Oh Yeah (ANNOY)

ANNOY hoạt động bằng cách trước tiên chọn ngẫu nhiên hai vectơ trong cơ sở dữ liệu và chia đôi không gian tìm kiếm dọc theo siêu phẳng ngăn cách hai vectơ đó.

Việc này được thực hiện lặp đi lặp lại cho đến khi có ít hơn một số tham số được xác định trước số vector quy định.



4) Các vector database hàng đầu

- **Milvus**



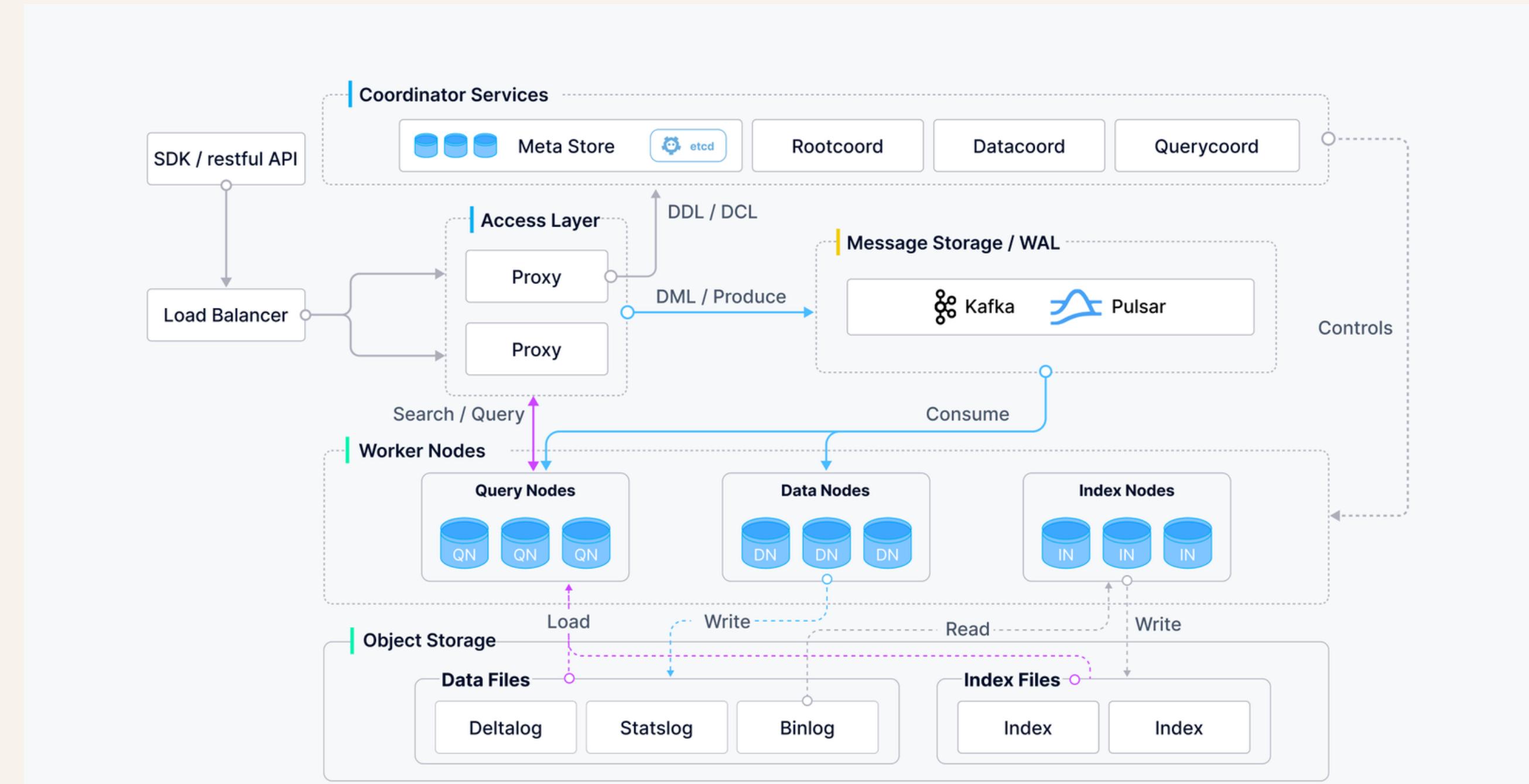
Được xây dựng dựa trên các thư viện tìm kiếm vectơ phổ biến bao gồm Faiss, Annoy, HNSW, v.v.,

Các tính năng chính:

- Tìm kiếm mili giây trên tập dữ liệu vectơ nghìn tỷ:
- Quản lý dữ liệu phi cấu trúc đơn giản.
- Đáng tin cậy:
- Khả năng mở rộng và mềm dẻo cao:
- Hỗ trợ tìm kiếm các kiểu ngoài dạng vector:
- Sử dụng cấu trúc Lambda hợp nhất:

4) Các vector database hàng đầu

- Milvus



4) Các vector database hàng đầu

• Chroma

Sử dụng thuật toán HNSW.

-Chroma cung cấp các công cụ để:

- + Lưu trữ các embedding và bối cảnh của chúng

- + Nhúng tài liệu và truy vấn

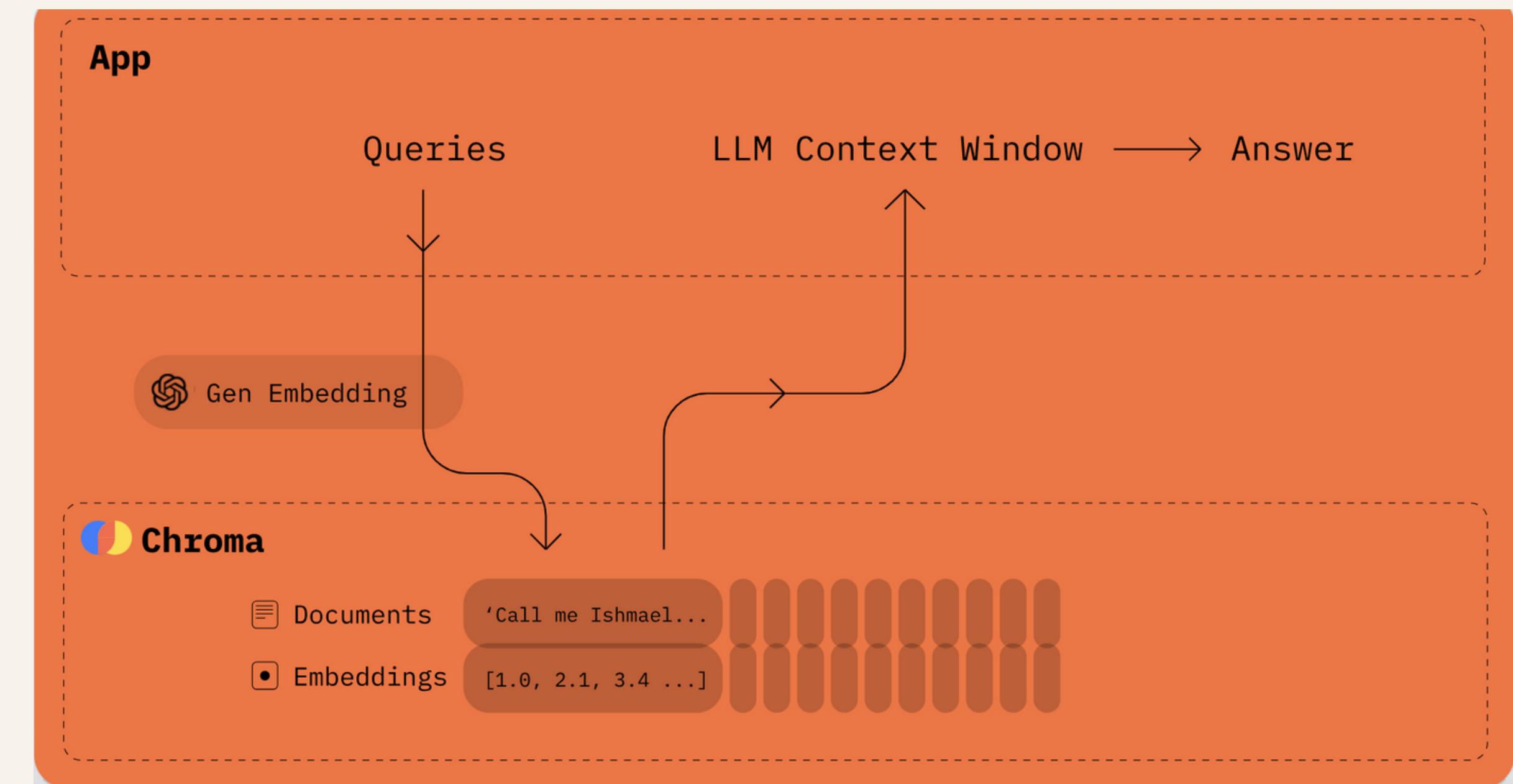
- + Tìm kiếm nhúng

-Chroma ưu tiên:

- + Sự đơn giản và năng suất của nhà phát triển

- + Phân tích những kết quả tìm kiếm hàng đầu.

- + Tốc độ tìm kiếm nhanh.



4) Các vector database hàng đầu

• Weaviate

- Sử dụng kết hợp nhiều thuật toán: Customized HNSW, HNSW (HQ), DiskANN,...

-Weaviate có thể được sử dụng độc lập (hay còn gọi là bring your vectors – tức là chúng ta thực hiện embedding rồi đưa vector vào Weaviate lưu trữ) hoặc với nhiều mô-đun khác nhau có thể thực hiện vector hóa cho và mở rộng các khả năng cốt lõi.

-Weaviate có GraphQL-API để truy cập dữ liệu một cách dễ dàng.

-Weaviate rất nhanh.

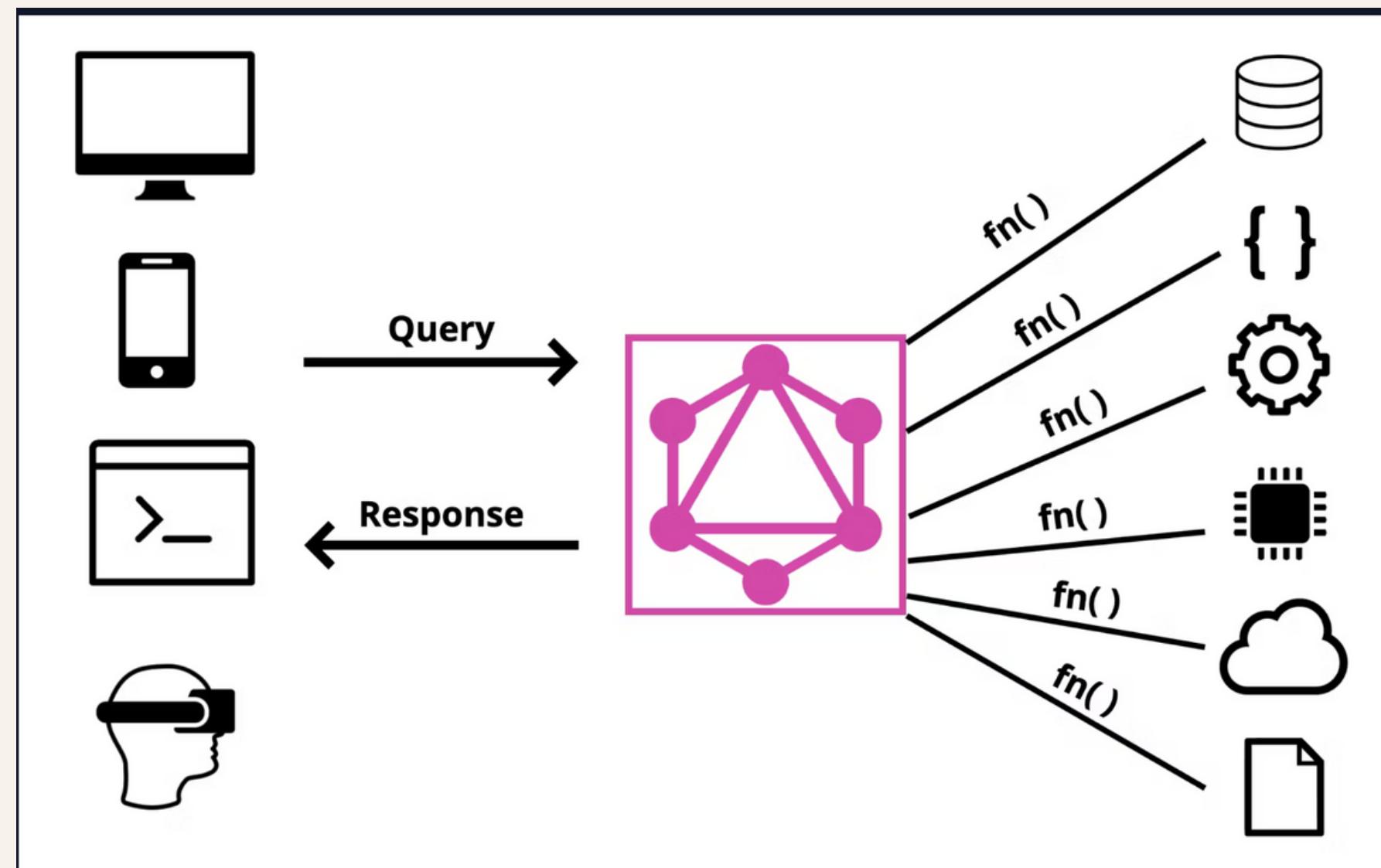
4) Các vector database hàng đầu

• Weaviate

GraphQL API

Đặc điểm:

- + Không lấy quá nhiều hay quá ít tài nguyên.
- + Ít điểm cuối cần quản lý hơn.
- + Tích hợp linh hoạt với cơ sở dữ liệu, API REST, cloud và tệp JSON

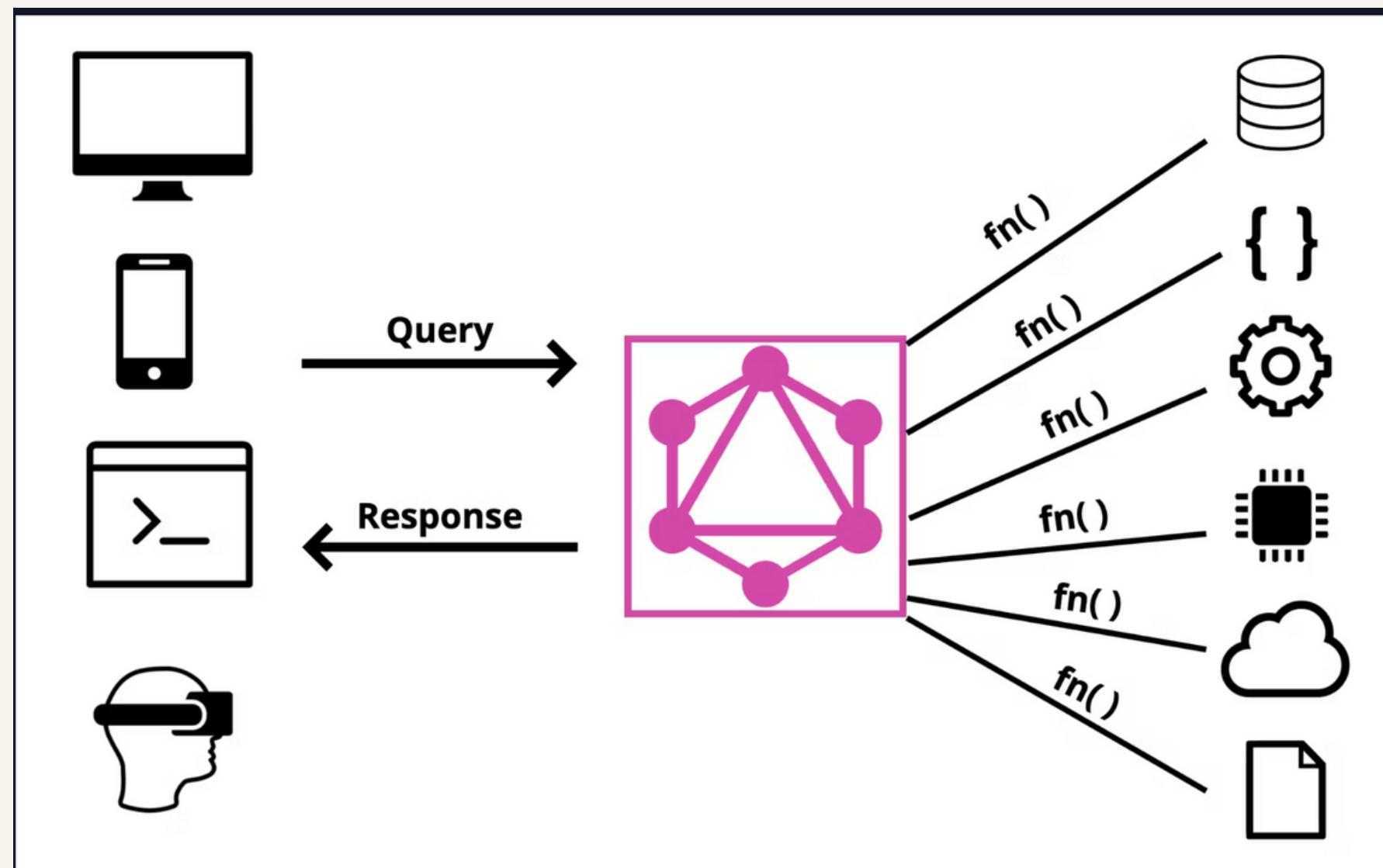


4) Các vector database hàng đầu

• Weaviate

Một hoạt động GraphQL điển hình diễn ra như sau:

- Dữ liệu được yêu cầu từ máy chủ GraphQL thông qua một truy vấn.
- Một hàm được gọi trên máy chủ GraphQL để lấy dữ liệu từ nguồn thích hợp.
- Máy chủ GraphQL trả về phản hồi cho máy khách.



5) Cách để chia dữ liệu theo từng sinh viên.

a) Cách thức Siri lấy dữ liệu cá nhân để cải thiện dịch vụ:

Các nguồn dữ liệu cá nhân mà Siri thu thập:

- Những ứng dụng mà người dùng cho phép Siri lấy dữ liệu có sẵn trên hệ thống apple (Ví dụ như sức khỏe, lời nhắc,...).
- Dữ liệu được lưu trong Icloud của người dùng.
- Các yêu cầu mà người dùng gửi đến Siri.

Cách lưu trữ dữ liệu:

- Dữ liệu Siri, bao gồm các yêu cầu Siri của bạn, được liên kết với một mã định danh ngẫu nhiên do thiết bị tạo ra.
- Mã định danh ngẫu nhiên này không được liên kết với ID Apple, địa chỉ email của bạn hoặc dữ liệu khác mà Apple có thể có được từ việc bạn sử dụng các dịch vụ khác của Apple.
- Các dữ liệu sẽ bị xóa sau 6 tháng.

5) Cách để chia dữ liệu theo từng sinh viên.

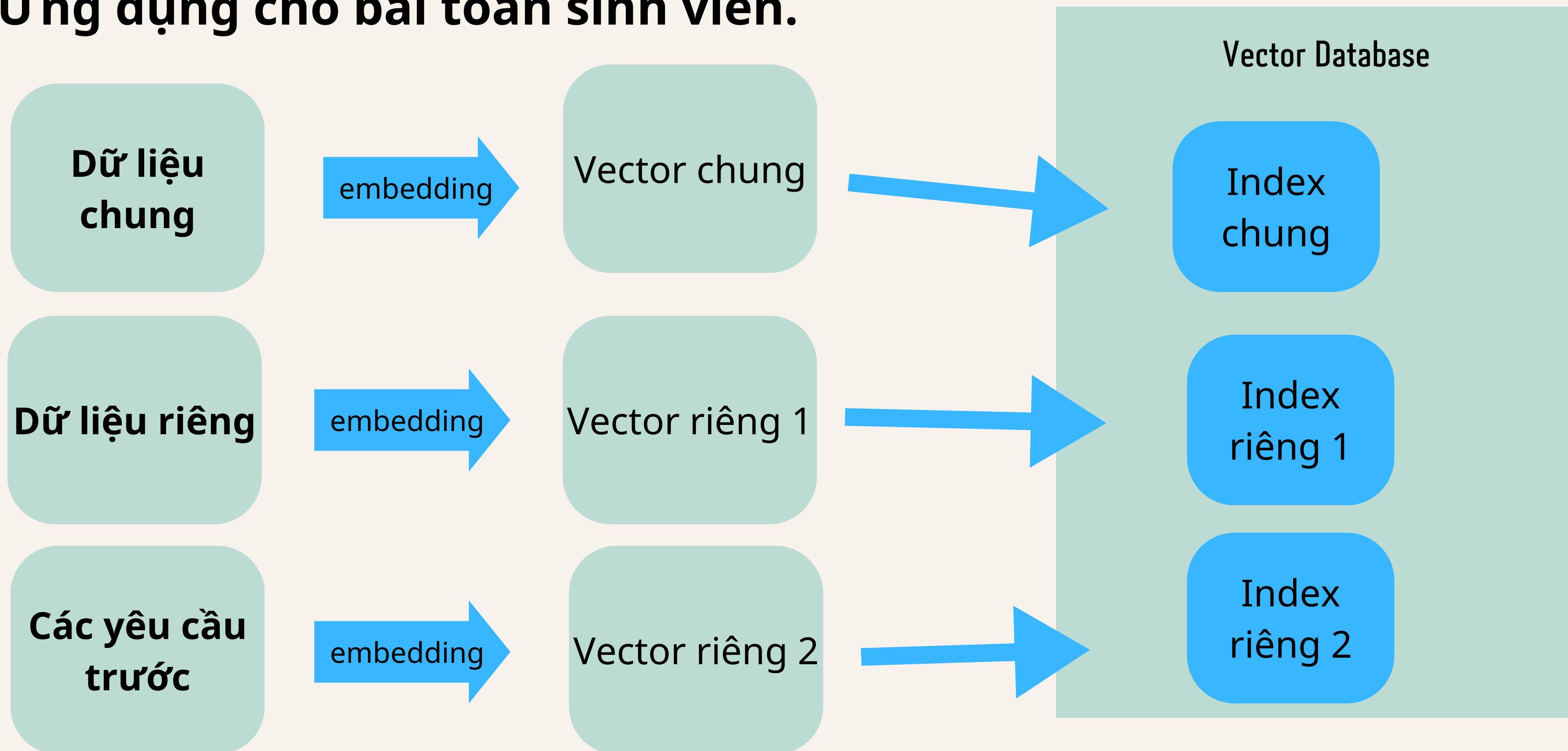
b) Ứng dụng cho bài toán sinh viên.

Nguồn dữ liệu mà bài toán sử dụng:

- Các thông tin chung mà sinh viên cần biết (quy chế trường, điều kiện học bổng,...).
- Các thông tin riêng của mỗi sinh viên: Dữ liệu trên slink, qldt.

5) Cách để chia dữ liệu theo từng sinh viên.

b) Ứng dụng cho bài toán sinh viên.



5) Cách để chia dữ liệu theo từng sinh viên.

b) Ứng dụng cho bài toán sinh viên.

Cách tổ chức dữ liệu:

- Mỗi sinh viên được cấp 1 lượng bộ nhớ riêng nhỏ, để chứa các index riêng 1 và index riêng 2 mà không lưu trữ các vector tương ứng của nó.
- Bộ nhớ chính sẽ chứa toàn bộ các vector embedding và index của nó.
- Những vector embedding có index chung và index riêng 1 sẽ không bị xóa. Những vector embedding có index riêng 2 sẽ bị xóa theo định kì.

5) Cách để chia dữ liệu theo từng sinh viên.

b) Ứng dụng cho bài toán sinh viên.

Cách search vector:

- Từ câu hỏi của sinh viên thực hiện embedding để tạo vector yêu cầu.
- Vector yêu cầu của sinh viên đó sẽ được xét khoảng cách với các vector có index chung và những index được lưu trong bộ nhớ riêng của sinh viên.

5) Cách để chia dữ liệu theo từng sinh viên.

b) Ứng dụng cho bài toán sinh viên.

Cách tăng cường dữ liệu:

- Thông qua những câu hỏi của học sinh, thực hiện lấy định kì dữ liệu của hội thoại của chatbot và học sinh.
- Thực hiện embedding dữ liệu và tạo index rồi lưu index vào bộ nhớ riêng của học sinh, lưu embedding vector và index của nó vào bộ nhớ chính.
- Tương tự mỗi kì học lấy những dữ liệu mới của qldt, slink để tạo thêm các index riêng.

**Thank
you!**

