Báo cáo: Tìm hiểu về Vector Database (VectorDB), Retrieval-Augmented Generation (RAG) và các biến thể của RAG

1. Giới thiệu chung

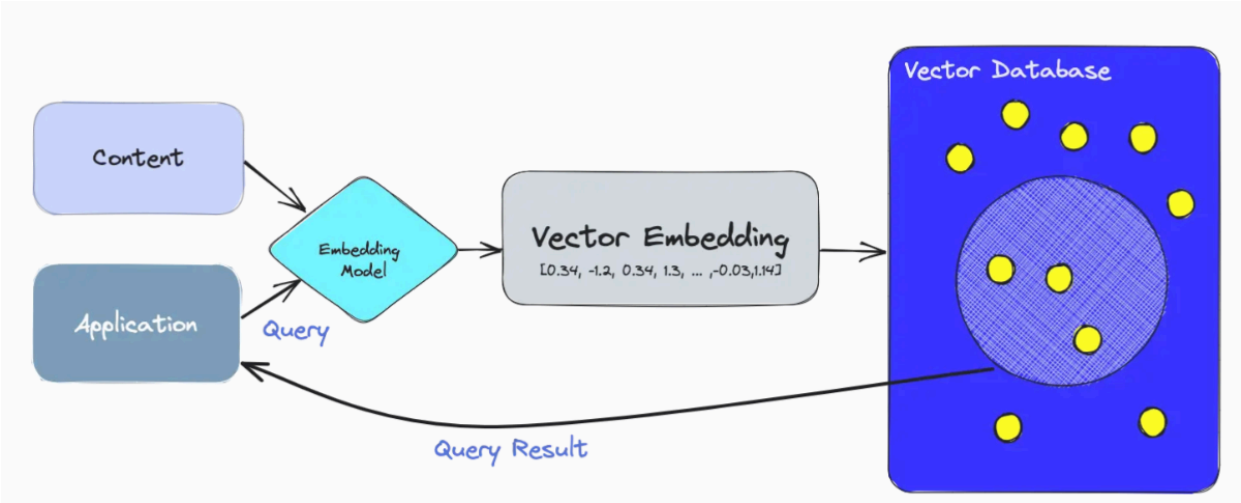
Trong kỷ nguyên dữ liệu bùng nổ, **Vector Database (VectorDB)** và **Retrieval-Augmented Generation (RAG)** đang nổi lên như hai trụ cột công nghệ then chốt, góp phần giải quyết thách thức cốt lõi của trí tuệ nhân tạo (AI): làm thế nào để lưu trữ, truy xuất và khai thác tri thức quy mô lớn một cách nhanh chóng, chính xác và tiết kiệm chi phí.

Trong khi VectorDB cung cấp hạ tầng lưu trữ và truy vấn hiệu quả cho *vector embeddings*—biểu diễn số chiều cao của dữ liệu phi cấu trúc (văn bản, âm thanh, hình ảnh…)—thì RAG bổ sung lớp suy luận sáng tạo (generation) được hỗ trợ bởi cơ chế tra cứu (retrieval) thời gian thực từ VectorDB hoặc các kho tri thức khác. Sự kết hợp này giúp hệ thống AI:

- **Tăng tính chính xác** (grounded-answering) nhờ dẫn chứng từ nguồn dữ liệu gốc.
- **Giảm chi phí đào tạo** vì mô hình ngôn ngữ lớn (LLM) không phải ghi nhớ mọi kiến thức tĩnh.
- **Mở rộng tri thức động** khi dữ liệu gốc thay đổi liên tục.

---

2. Tìm hiểu về Vector Database



2.1 Khái niệm cơ bản và nguyên lý hoạt động

VectorDB là hệ quản trị cơ sở dữ liệu tối ưu cho việc lưu trữ và truy vấn *vector embeddings*—các mảng số thực có hàng trăm đến hàng nghìn chiều mô tả ngữ nghĩa của đối tượng. Khác với cơ sở dữ liệu truyền thống dựa trên chỉ mục B-tree, VectorDB sử dụng **chỉ mục truy vấn gần đúng (Approximate Nearest Neighbor – ANN)** như HNSW, IVF, PQ… để rút ngắn thời gian tìm kiếm *k* vector gần nhất (k-NN) trong không gian cao chiều.

Quy trình truy vấn thường gồm ba bước:

1. **Vector hoá** (Embedding): Văn bản/hình ảnh đầu vào được mô hình encoder biến thành vector.

2. **Tạo chỉ mục**: Vector được nạp vào cấu trúc ANN, được phân cụm hoặc phân tầng để tối ưu truy vấn.

3. **Tìm kiếm gần nhất**: Với vector truy vấn, VectorDB trả về các mục gần nhất dựa trên khoảng cách (cosine, Euclidean hoặc dot-product).

2.2 Đặc điểm và lợi ích chính

- **Hiệu năng cao**: Thời gian truy vấn thường <10 ms cho hàng triệu bản ghi nhờ kỹ thuật ANN và phân mảnh (sharding).

- **Khả năng mở rộng tuyến tính**: Có thể nạp hàng tỷ vector nhờ kiến trúc phân tán.

- **Hỗ trợ metadata filtering**: Kết hợp điều kiện (tag, timestamp…) với truy vấn k-NN.

- **Cập nhật động**: Thêm/xóa vector trong thời gian thực, quan trọng cho hệ thống RAG yêu cầu dữ liệu luôn mới.

2.3 Các trường hợp ứng dụng phổ biến

Ứng dụng	Mô tả ngắn
Tìm kiếm ngữ nghĩa	Tra cứu tài liệu tương tự nội dung câu hỏi, cải thiện chất lượng tìm kiếm so với Keyword Search.
Hệ khuyến nghị	Tìm người dùng/mặt hàng "gần" nhau về sở thích, hành vi.
Phát hiện gian lận	So khớp đặc trưng giao dịch với lịch sử để phát hiện bất thường.
Bảo mật sinh trắc	So khớp vector đặc trưng khuôn mặt, vân tay…
Phân loại chủ đề	Truy vấn nearest-centroid để gán nhãn.

2.4 So sánh với cơ sở dữ liệu truyền thống

Tiêu chí	CSDL quan hệ / NoSQL	VectorDB
Dữ liệu chính	Bảng, tài liệu JSON, key-value	Vector số thực N chiều
Kiểu truy vấn tối ưu	CRUD, JOIN, Index Scan	k-NN, Range Search theo độ tương đồng
Chỉ mục	B-tree, Hash, Inverted Index	HNSW, IVF-PQ, ScaNN…
Hiệu năng cho vector search	Cao (O(n))	Rất cao (O(log n) xấp xỉ)
Điển hình triển khai	MySQL, PostgreSQL, MongoDB	Milvus, Pinecone, FAISS, Weaviate

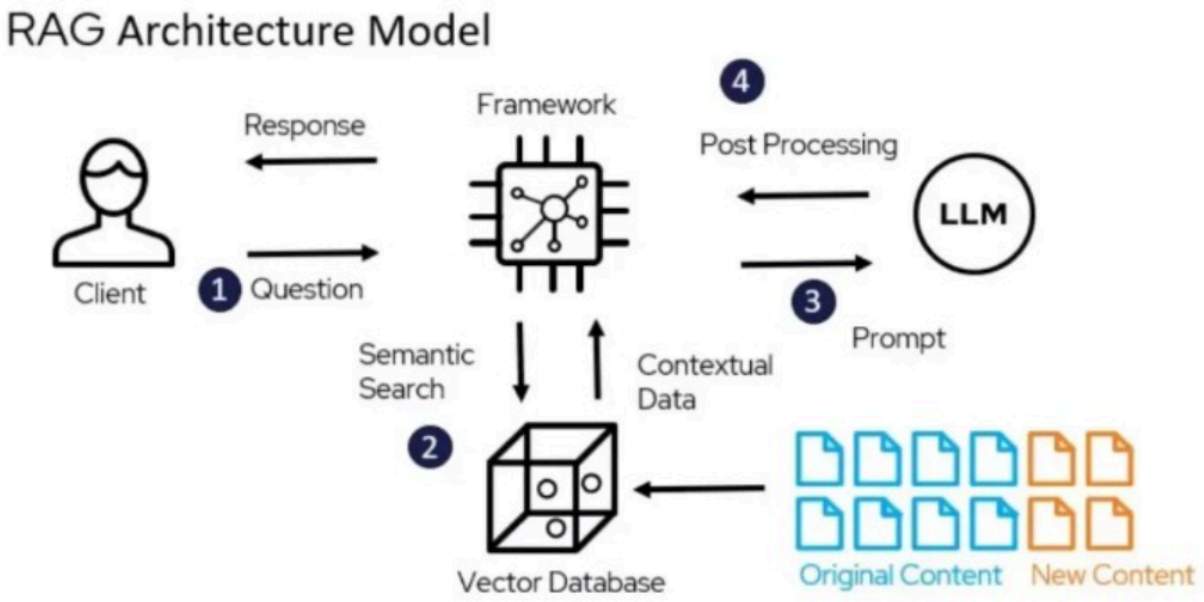
2.5 Các nền tảng và công cụ phổ biến

- **FAISS (Facebook AI Similarity Search)**: Thư viện C++/Python mã nguồn mở, hỗ trợ GPU, thích hợp dựng proof-of-concept.

- **Milvus**: Hệ VectorDB phân tán, hỗ trợ HNSW, IVF-PQ, GPU acceleration, replication, shard-ing.
- **Pinecone**: Dịch vụ VectorDB SaaS quản lý hoàn toàn, có tính năng *namespace*, *metadata filter*, giao diện REST & gRPC.
- **Weaviate**: VectorDB + GraphQL, có mô-đun tích hợp sẵn mô hình embedding (Transformers, OpenAI).
- **Chroma**: Vector store Python lightweight, phổ biến trong prototyping RAG.

---

3. Tìm hiểu về Retrieval-Augmented Generation (RAG)



3.1 Định nghĩa và nguyên lý hoạt động

RAG là kiến trúc kết hợp giữa **truy xuất thông tin (retrieval)** và **mô hình sinh ngôn ngữ (generation)**. Thay vì yêu cầu LLM "nhớ" toàn bộ tri thức, RAG thực hiện:

1. **Retrieval**: Tìm *top-k* đoạn văn liên quan từ VectorDB bằng embedding của truy vấn.
2. **Augmentation**: Ghép (prepend/append) các đoạn hồi đáp vào *prompt* của LLM.

3. **Generation**: LLM tạo câu trả lời dựa trên *context* đã mở rộng.

Sơ đồ pipeline:

> *User Query → Embed → VectorDB (ANN Search) → Retrieved Docs → Prompt Builder → LLM → Response*

3.2 Quy trình triển khai & xử lý dữ liệu

- **Chuẩn hoá dữ liệu**: Tách tài liệu theo ngữ nghĩa (chunking 500-1000 token), gán metadata.
- **Embedding**: Sử dụng mô hình Sentence-BERT, OpenAI text-embedding-3-large, Cohere v3…
- **Build Index**: Nạp vào VectorDB, duy trì background indexer khi dữ liệu thay đổi.
- **Query Flow**: Nhận câu hỏi, thực hiện *k-NN* (thường k=3-10), có thể thêm bước *re-ranking*.
- **Prompt Engineering**: Chèn hướng dẫn hệ thống, chain-of-thought, phân vùng context để tránh vượt token limit.
- **Post-processing**: Trích dẫn nguồn, kiểm tra tính nhất quán, tóm tắt.

3.3 Vai trò của VectorDB trong RAG

VectorDB cung cấp:

- **Low-latency Retrieval**: <100 ms cho k-NN, bảo đảm trải nghiệm realtime.
- **Scalability**: Lưu hàng tỷ đoạn văn—điều không khả thi nếu embed sẵn toàn bộ vào prompt.
- **Filtering**: Kết hợp tìm kiếm ngữ nghĩa với điều kiện có cấu trúc (ngày tháng, tác giả…).

3.4 Ưu điểm & hạn chế của RAG

Ưu điểm	Hạn chế
Giảm hallucination nhờ tham chiếu nguồn.	Chất lượng phụ thuộc độ phủ dữ liệu & mô hình embed.

Dễ cập nhật tri thức chỉ cần re-index, không phải fine-tune LLM.	Risk *prompt injection* nếu đoạn văn chứa nội dung độc hại.
Tiết kiệm chi phí vì không retrain LLM lớn.	Phức tạp về kiến trúc (VectorDB, pipeline).
Khả năng truy vết (traceability) khi trích dẫn.	Giới hạn context window khiến model bỏ sót thông tin dài.

---

4. Các biến thể của Retrieval-Augmented Generation

4.1 RAG tiêu chuẩn (Standard RAG)

Pipeline cơ bản gồm **Single-pass Retrieval** + **Single-pass Generation**. Phù hợp ứng dụng yêu cầu latency thấp, dữ liệu <1 triệu đoạn.

4.2 RAG kết hợp Re-ranking

Thêm bước **re-ranking** (Cross Encoder/MonoT5/ColBERT) để xếp hạng lại *top-k* (ví dụ 100) kết quả ANN bằng mô hình mạnh nhưng tốn tài nguyên. Cải thiện tính chính xác tới 5-15 % nhưng tăng độ trễ.

4.3 Dense Passage Retrieval (DPR)

Đề xuất bởi Facebook AI (2020), dùng hai encoder (Question/BPassage) học biểu diễn độc lập, được huấn luyện song song với Negative Sampling. DPR ban đầu cho Open-Domain QA nhưng nay trở thành baseline retriever cho RAG.

4.4 Fusion-in-Decoder (FiD)

Khác với RAG chuẩn (concatenate context → LLM), FiD encode từng đoạn riêng rồi **Fusion** trong *decoder* của Transformer. Cho phép model cân nhắc hàng chục đoạn (n = 40-100) mà không nổ token limit. Hiệu quả tốt cho QA phức tạp (nhiều nguồn).

4.5 Hybrid Search RAG (Dense + Sparse)

Kết hợp chỉ mục **BM25/Inverted Index** (từ khoá) và **Vector Search** (ngữ nghĩa). Hai chiến lược:

1. **Parallel**: Truy vấn song song, rồi hợp nhất (merge-dedup).
2. **Cascade**: BM25 shortlist → Vector rerank.

Hybrid cải thiện recall khi từ khóa đặc thù hoặc thuật ngữ hiếm.

4.6 So sánh chi tiết các biến thể

Biến thể	Độ chính xác	Độ trễ	Yêu cầu tài nguyên	Phù hợp
Standard RAG	Trung bình	Thấp	Thấp	Chatbot, FAQ
RAG + Re-rank	Cao	Trung bình	Vừa	Tư vấn y khoa, pháp lý
DPR	Cao	Thấp	Huấn luyện custom	Kho QA chuyên ngành
FiD	Rất cao	Cao	GPU mạnh	QA đa tài liệu, multi-hop
Hybrid	Cao (recall)	Trung bình	Index kép	Search domain-heavy

---

5. Ứng dụng thực tế và nghiên cứu điển hình

- **BloombergGPT + RAG** (2024): Kết hợp VectorDB chứa 400 tỷ token tin tài chính, giúp chatbot trả lời dự báo thị trường với tỷ lệ chính xác tăng 18 % so baseline.
- **GitHub Copilot Workspace**: Sử dụng RAG trên VectorDB chứa mã nguồn riêng của tổ chức, giúp sinh code tuân thủ style nội bộ.
- **You.com**: Công cụ tìm kiếm cá nhân hoá dùng hybrid RAG kết hợp BM25 và vector, cung cấp câu trả lời trích dẫn đa nguồn.
- **OpenAI ChatGPT Plugins**: Triển khai dạng RAG: plugin “Browsing” truy xuất web → kết quả đưa vào prompt GPT-4o.

Phân tích hiệu quả:

- **Giảm thời gian nghiên cứu** tới 60 % tại Morgan Stanley khi dùng RAG trợ lý tài chính nội bộ.
- **Tăng tỉ lệ giải đáp đúng** trên bộ HotpotQA từ 45 → 58 F1 khi dùng FiD.

---

6. Thách thức và hướng phát triển tương lai

6.1 Thách thức kỹ thuật & triển khai

- **Scaling vs. Cost**: Lưu trữ hàng tỷ vector tiêu hao bộ nhớ & chi phí cloud.
- **Freshness**: Dữ liệu thay đổi liên tục yêu cầu re-index online, dễ gây *index skew*.
- **Security & Privacy**: Vector có thể leak thông tin nhạy cảm qua *membership inference*.
- **Evaluation**: Thiếu thước đo tiêu chuẩn cho RAG; khó đánh giá HALU vs. Coverage.

6.2 Xu hướng nghiên cứu mới

1. **Self-Refine RAG**: LLM tự đánh giá & truy vấn bổ sung nếu phát hiện thiếu căn cứ.
2. **Multimodal RAG**: VectorDB lưu trữ hình ảnh, audio để RAG trả lời đa phương tiện.
3. **Personalized RAG**: Kết hợp graph + vector để tạo truy vấn contextual theo người dùng.
4. **In-Context Learning Indexing**: Nạp tri thức vừa truy vết vừa fine-tune light adapter, giảm latency.
5. **Federated / On-device RAG**: Chạy VectorDB cục bộ (Chromadb WASM) đảm bảo quyền riêng tư.
