HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÙI KHẮC NGỌC

PHÁT TRIỂN CÔNG CỤ ỨNG DỤNG AI TRONG VIỆC CHUYỂN ĐỔI NGÔN NGỮ TỰ NHIÊN THÀNH CÂU TRUY VẤN SQL NHẰM HỖ TRỢ TRA CỨU NHANH THÔNG TIN

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

HÀ NỘI - 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÙI KHẮC NGỌC

PHÁT TRIỂN CÔNG CỤ ỨNG DỤNG AI TRONG VIỆC CHUYỂN ĐỔI NGÔN NGỮ TỰ NHIÊN THÀNH CÂU TRUY VẤN SQL NHẰM HỖ TRỢ TRA CỨU NHANH THÔNG TIN

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. NGÔ QUỐC DŨNG

HÀ NỘI - 2025

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong đề án là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện đề án này đã được cảm ơn và các thông tin trích dẫn trong đề án đã được chỉ rõ nguồn gốc.

Hà Nội, ngày ... tháng ... năm

...

Bùi Khắc Ngọc

LỜI CẢM ƠN

Trong suốt quá trình học tập, nghiên cứu để thực hiện đề án, ngoài nỗ lực của bản thân, tôi cũng đã nhận được sự hướng dẫn nhiệt tình của quý thầy cô, cùng với sự động viên và ủng hộ của gia đình, bạn bè và đồng nghiệp. Với lòng kính trọng và biết ơn sâu sắc, tôi xin gửi lời cảm ơn chân thành tới: Ban Giám Đốc, Phòng đào tạo sau đại học và quý thầy cô đã tạo mọi điều kiện thuận lợi giúp tôi hoàn thành đề án.

Tôi xin chân thành cảm ơn Thầy PGS. TS. Ngô Quốc Dũng đã hết lòng giúp đỡ, hướng dẫn, động viên, tạo điều kiện cho tôi trong suốt quá trình thực hiện và hoàn thành đề án.

Tôi rất mong nhận được sự góp ý của quý thầy cô cùng bạn bè đồng nghiệp để kiến thức của tôi ngày một hoàn thiện hơn.

Xin chân thành cảm ơn!

Hà Nội, ngày 10 tháng 12 năm 2025

Học viên thực hiện đề án

Bùi Khắc Ngọc

MỤC LỤC

LỜI CAM ĐOAN.....i
LỜI CẢM ƠN.....ii
MỤC LỤC.....iii
BẢNG TỪ VIẾT TẮT.....viii
DANH MỤC HÌNH ẢNH.....x
DANH MỤC BẢNG.....xi
MỞ ĐẦU.....1
CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN.....7
1.1. Tổng quan về cơ sở dữ liệu.....7
1.2. Tổng quan về đại học số và nhu cầu truy vấn dữ liệu.....8
1.3. Tổng quan về trí tuệ nhân tạo.....8
1.3.1. Khái niệm về trí tuệ nhân tạo.....8
1.3.2. Xử lý ngôn ngữ tự nhiên.....9
1.3.3. Mô hình ngôn ngữ lớn.....9
1.4. Bài toán chuyển đổi ngôn ngữ tiếng Việt thành SQL.....10
1.4.1. Định nghĩa.....10
1.4.2. Các giai đoạn phát triển.....11
1.4.3. Các thách thức trong bài toán.....11
1.4.4. Các công cụ hiện có.....12
1.5. Đặc thù của chuyển đổi ngôn ngữ tiếng Việt thành câu truy vấn SQL13
1.6. Kết luận chương.....14
CHƯƠNG 2: PHÂN TÍCH, ĐÁNH GIÁ VÀ THIẾT KẾ GIẢI PHÁP.....15
2.1. Phân tích yêu cầu hệ thống.....15
2.1.1. Phân tích yêu cầu chức năng.....15
2.1.2. Phân tích yêu cầu phi chức năng.....17
2.1.3. Xác định các ràng buộc và giới hạn.....18
2.2. Nghiên cứu và đánh giá các phương pháp chuẩn hóa schema.....18
2.2.1. Khái niệm về chuẩn hóa schema.....18

2.2.2. Các phương pháp chuẩn hóa schema.....19
2.2.3. Đánh giá và lựa chọn phương pháp phù hợp.....20
2.3. Nghiên cứu xây dựng bộ từ điển dữ liệu.....21
2.3.1. Vai trò của từ điển dữ liệu trong Text-to-SQL.....21
2.3.2. Phương pháp xây dựng từ điển dữ liệu.....22
2.3.3. Cấu trúc và nội dung từ điển dữ liệu.....24
2.4. Nghiên cứu thiết kế Semantic Layer cho hệ thống.....26
2.4.1. Vai trò của Semantic Layer trong Text-to-SQL.....26
2.4.2. Phân tích các thành phần của Semantic Layer.....27
2.4.3. Thiết kế Semantic Layer cho đại học số.....27
2.5. Thiết kế cấp cao (High-level design).....28
2.5.1. Lớp Giao diện và Tương tác.....29
2.5.2. Lớp Ứng dụng và Điều phối.....29
2.5.3. Lớp Trí tuệ và Dữ liệu Cốt lõi.....29
2.5.4. Luồng Xử lý Từ Đầu Đến Cuối.....30
2.6. Thiết kế kiến trúc hệ thống.....32
2.6.1. Kiến trúc tổng thể của hệ thống.....32
2.6.2. Các thành phần cốt lõi của kiến trúc.....33
2.6.3. Mô hình kiến trúc vi dịch vụ.....35
2.6.4. Luồng xử lý từ đầu đến cuối.....35
2.6.5. Vai trò của cơ sở dữ liệu vector.....36
2.6.6. Tích hợp mô hình ngôn ngữ lớn.....37
2.6.7. Cơ chế xử lý lỗi, tự sửa lỗi và bảo mật hệ thống.....38
2.6.8. Chỉ số đánh giá hệ thống.....39
2.7. Khảo sát và đánh giá các công cụ hiện có.....41
2.7.1. Phân loại các công cụ.....42
2.6.2. Tiêu chí đánh giá.....42
2.7.3. Kết quả khảo sát các công cụ.....43
2.8. Kết luận chương 2.....47
CHƯƠNG 3: XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG.....48
3.1. Môi trường và công nghệ triển khai.....48

3.2. Xây dựng các tài nguyên nền tảng.....52
3.2.1. Xây dựng và chuẩn hóa schema cơ sở dữ liệu.....52
3.2.2. Xây dựng bộ từ điển dữ liệu.....53
3.2.3. Xây dựng bộ câu hỏi mẫu.....55
3.3. Triển khai các module xử lý chính.....55
3.3.1. Triển khai module tiền xử lý câu hỏi.....55
3.3.2. Triển khai module ánh xạ schema.....56
3.3.3. Triển khai module tương tác với mô hình ngôn ngữ lớn.....59
3.3.4. Triển khai module kiểm tra và thực thi SQL.....60
3.4. Xây dựng giao diện người dùng.....61
3.4.1. Triển khai giao diện nhập câu hỏi.....62
3.4.2. Triển khai giao diện hiển thị kết quả.....63
3.4.3. Triển khai các chức năng quản lý hỗ trợ.....66
3.5. Thử nghiệm và đánh giá hệ thống.....68
3.5.1. Kế hoạch và phương pháp thử nghiệm.....68
3.5.2. Đánh giá độ chính xác của hệ thống.....69
3.5.3. Đánh giá hiệu suất của hệ thống.....69
3.5.4. So sánh hiệu suất các mô hình ngôn ngữ lớn.....71
3.6. Cải tiến và tối ưu hóa.....72
3.6.1. Phân tích điểm yếu của hệ thống.....72
3.6.2. Các giải pháp cải tiến.....73
3.7. Triển khai và hướng dẫn sử dụng.....73
3.7.1. Hướng dẫn cài đặt và cấu hình.....74
3.7.2. Hướng dẫn sử dụng cho người dùng cuối.....75
3.7.3. Hướng dẫn vận hành và bảo trì.....76
3.8. Kết luận chương 3.....76
KẾT LUẬN VÀ KIẾN NGHỊ.....78
TÀI LIỆU THAM KHẢO.....80

BẢNG TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
API	Application Programming Interface	Giao diện lập trình ứng dụng
BERT	Bidirectional Encoder Representations from Transformers	Mô hình mã hóa hai chiều từ Transformers
CSDL	Cơ Sở Dữ Liệu	Database
DB	Database	Cơ sở dữ liệu
DBMS	Database Management System	Hệ quản trị cơ sở dữ liệu
DDL	Data Definition Language	Ngôn ngữ định nghĩa dữ liệu
ERD	Entity Relationship Diagram	Biểu đồ thực thể mối quan hệ
GPA	Grade Point Average	Điểm trung bình tích lũy
GPT	Generative Pre-trained Transformer	Mô hình chuyển đổi tiền huấn luyện sinh
GRU	Gated Recurrent Unit	Đơn vị hồi quy có cổng
HEMIS	Higher Education Management Information System	Hệ thống thông tin quản lý giáo dục đại học
IDE	Integrated Development Environment	Môi trường phát triển tích hợp
JSON	JavaScript Object Notation	Ký hiệu đối tượng JavaScript
JWT	JSON Web Token	Mã thông báo web JSON
KPI	Key Performance Indicator	Chỉ số đo lường hiệu suất

Từ viết tắt	Tiếng Anh	Tiếng Việt
		chính
LLM	Large Language Model	Mô hình ngôn ngữ lớn
LMS	Learning Management System	Hệ thống quản lý học tập
LSTM	Long Short-Term Memory	Bộ nhớ dài ngắn hạn
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
RAG	Retrieval-Augmented Generation	Sinh tăng cường truy xuất
RBAC	Role-Based Access Control	Kiểm soát truy cập dựa trên vai trò
REST	Representational State Transfer	Chuyển giao trạng thái đại diện
RLS	Row-Level Security	Bảo mật cấp hàng
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
UI	User Interface	Giao diện người dùng
UX	User Experience	Trải nghiệm người dùng

DANH MỤC HÌNH ẢNH

Hình 2.1: Text-to-sql pipeline 17
Hình 2.2: Thiết kết cấp cao của hệ thống Text-to-SQL 31
Hình 2.3: Kiến trúc tổng thể của hệ thống hệ thống 33
Hình 3.1: Lược đồ ERD cơ sở dữ liệu quản lý đào tạo 48
Hình 3.2: Giao diện câu hỏi SQL 51
Hình 3.3: Pipeline truy xuất thông tin cho quá trình xử lý truy vấn 52
Hình 3.4: Mô hình hoạt động của module ánh xạ schema 53
Hình 3.5: Quy trình chuyển đổi từ ngôn ngữ tiếng Việt đến câu truy vấn SQL 55
Hình 3.6: Kiến trúc Tổng quan BasicPipeline WrenAI 56
Hình 3.7: Luồng gửi câu hỏi 58
Hình 3.8: Giao diện nhập câu hỏi 58
Hình 3.9: Giao diện câu trả lời 59
Hình 3.10: Giao diện câu truy vấn SQL 60
Hình 3.11: Giao diện biểu đồ trực quan 61
Hình 3.12: Giao diện quản lý lịch sử truy vấn 62
Hình 3.13: Giao diện quản lý schema 63
Hình 3.14: Ví dụ về kết quả bước Text-to-SQL với truy vấn "Hiển thị phổ điểm tổng kết của các sinh viên trong kỳ vừa qua" 64
Hình 3.15: Hình ảnh báo cáo của artillery với câu hỏi “Tìm điểm thành phần của các học phần sinh viên có mã B23DCCC144” với truy vấn của 105 người. 65
Hình 3.16: Cấu hình công cụ Wren AI trong file docker compose 68
Hình 3.17: Giao diện cài đặt cơ sở dữ liệu nguồn 69
Hình 3.18: Giao diện truy vấn và lịch sử truy vấn của người dùng 70

DANH MỤC BẢNG

Bảng 3.1: Tổng hợp môi trường và công nghệ triển khai 46
Bảng 3.2: Đánh giá khả năng chịu tải của hệ thống 73
Bảng 3.3: Bảng so sánh tổng hợp các mô hình ngôn ngữ lớn 74

MỞ ĐẦU

1. Lựa chọn đề tài

Hiện nay với sự thúc đẩy chuyển đổi số [1], nhiều cơ sở giáo dục ở Việt Nam đang có nhiều ứng dụng công nghệ vào việc quản lý giáo dục. Chính vì thế họ đã lưu trữ được khối lượng dữ liệu cực lớn liên quan đến các hoạt động quản lý.

Hiện tại phản ánh rằng mặc dù các dữ liệu đã được số hóa đầy đủ, nhưng việc quản lý dữ liệu để ra quyết định đúng và nhanh vẫn còn nhiều hạn chế [2]. Những trở ngại chủ yếu bao gồm:

- Rào cản về năng lực kỹ thuật: Phần lớn người dùng có nhu cầu khai thác dữ liệu thường xuyên nhưng lại thiếu kiến thức để làm việc với cơ sở dữ liệu [3].
- Áp lực công việc lớn đối với đơn vị công nghệ thông tin: Bộ phận này thường xuyên phải tiếp nhận và xử lý một khối lượng yêu cầu lớn, ảnh hưởng nghiêm trọng đến thời gian phản hồi và chất lượng kết quả công việc [4].
- Hạn chế trong tính chủ động ra quyết định: Việc người dùng không thể truy xuất nhanh thông tin có thể làm tăng thời gian ra quyết định khi dữ liệu luôn luôn được cập nhật.
- Xu hướng của các ứng dụng không cần lập trình: Nhiều công cụ Low-code / No-code kết hợp AI trở nên phổ biến hơn.

Vài năm gần đây đã có sự phát triển của ngôn ngữ lớn cùng lĩnh xử lý ngôn ngữ tự nhiên có nhiều cải tiến mới. Bước đột phá trong bài toán chuyển đổi ngôn ngữ tiếng Việt thành câu lệnh truy vấn cơ sở dữ liệu trên các tập huấn luyện bằng tiếng Việt. Các hướng tiếp cận mới này cho phép chuyển đổi trực tiếp ý định của người dùng bằng ngôn ngữ thông thường thành các câu truy vấn có cấu trúc hợp lệ, qua đó nâng cao đáng kể tính chính xác, tốc độ phản hồi và trải nghiệm tương tác của người dùng khi truy vấn dữ liệu.

2. Tổng quan về vấn đề nghiên cứu

Chuyển đổi số đã trở thành định hướng chiến lược quốc gia của nước ta nhằm nâng cấp vị thế kinh tế và dẫn đầu khu vực về công nghệ số. Tuy nhiên, hạn chế trong việc khai thác các dữ liệu đã lưu trữ. Đa số cán bộ quản lý và giảng viên – người dùng chính cần ra khai thác các thông tin đã lưu trữ lại không thể truy vấn trực tiếp do: (1) yêu cầu kiến thức chuyên sâu về cơ sở dữ liệu phức tạp (hàng trăm bảng, hàng nghìn cột với quan hệ chéo đa tầng); (2) thời gian phản hồi chậm do phụ thuộc các khâu trung gian; (3) nguy cơ sai sót cao khi người dùng không chuyên tự viết truy vấn; (4) khó khăn trong bảo trì khi cơ sở dữ liệu thay đổi.

Để khắc phục, hướng tiếp cận hiện đại là phát triển hệ thống Text-to-SQL cho phép người dùng cuối đặt câu hỏi thông thường và hệ thống tự động sinh câu lệnh SQL chính xác, loại bỏ hoàn toàn rào cản kỹ thuật. Mặc dù bài toán Text-to-SQL đã được nghiên cứu từ lâu, chỉ đến khi các mô hình LLM và mô hình mã nguồn mở xuất hiện, độ chính xác mới đạt mức chấp nhận được trong thực tế. Trong khi phần lớn nghiên cứu hiện hữu đạt hiệu suất cao trên tiếng Anh và các cơ sở dữ liệu chuẩn hóa, hiệu suất giảm mạnh khi chuyển sang tiếng Việt và với tập dữ liệu đặc thù.

Do đó, một lỗ hổng lớn giữa nghiên cứu và nhu cầu triển khai thực tiễn. Việc tích hợp thêm lớn ngữ nghĩa và chuẩn hóa dữ liệu sát với nghiệp vụ đã làm tăng độ chính xác làm các báo cáo đáng tin cậy hơn. Do đó, đề tài không chỉ tập trung vào tích hợp AI mà còn chuẩn hóa bộ dữ liệu và tạo bộ từ điển sát với nghiệp vụ.

3. Mục đích nghiên cứu

Đề tài đặt ra mục tiêu là tạo ra bộ công cụ ứng dụng trí tuệ nhân tạo để chuyển đổi ngôn ngữ tiếng Việt thành câu truy vấn SQL, hỗ trợ cán bộ bản lý và chuyên viên tra cứu nhanh các thông tin trong hệ thống quản lý. Công cụ sẽ loại bỏ rào cản kỹ thuật, cho phép người dùng có ít kinh nghiệm về truy vấn cơ sở dữ liệu có thể thao tác đọc dữ liệu bằng ngôn ngữ tiếng Việt, từ đó ra quyết định nhanh và chính xác hơn.

Để đạt được mục tiêu tổng thể này, đề tài xác định các mục tiêu cụ thể theo phương pháp tiếp cận có hệ thống:

- Xây dựng kiến trúc hệ thống: Đề xuất một kiến trúc hệ thống hoàn chỉnh, dựa trên nguyên lý Retrieval-Augmented Generation (RAG) và Semantic Layer.
- Chuẩn hóa dữ liệu: Nghiên cứu tập trung vào chuẩn hóa dữ liệu và tạo bộ từ điển dữ liệu sát với nghiệp vụ bằng tiếng Việt.
- Tích hợp mô hình ngôn ngữ lớn: Ứng dụng các công cụ để tối ưu các câu prompt trước khi đưa vào mô hình LLM để xử lý làm giảm tài nguyên và tăng độ chính xác.
- Đảm bảo tính thực tiễn và bảo mật: Nghiên cứu chỉ cho phép thực hiện câu truy vấn đọc dữ liệu, không thay đổi dữ liệu.

Hoàn thành các mục tiêu trên hứa hẹn mang lại những đóng góp đáng kể cả trong nghiên cứu học thuật lẫn ứng dụng thực tiễn. Ngoài việc mở rộng hiểu biết về lĩnh vực .

4. Đối tượng và phạm vi nghiên cứu

*** Đối tượng nghiên cứu:**

Đề tài tập trung vào việc xây dựng một công cụ hướng đến các người dùng là: Ban lãnh đạo của trường đại học, Ban giám đốc của học viện, cùng với các chuyên viên giảng viên của các”đơn vị trực thuộc”đơn vị giáo dục đại học.

Đề tài xác định mục tiêu hướng đến là xây dựng công cụ chuyển đổi tiếng Việt thành câu truy vấn SQL trong lĩnh vực giáo dục đại học. Công cụ ứng dụng LLM để hiểu và xử lý câu hỏi tiếng Việt, các kỹ thuật ánh xạ câu hỏi với cơ sở dữ liệu và giải pháp đảm bảo tính đúng đắn và hữu ích khi chạy trong môi trường triển khai thực tế.

*** Phạm vi nghiên cứu:**

Phạm vi nghiên cứu được xác định trên bốn phương diện chính trong đề tài. Thứ nhất, tập trung vào các cơ sở dữ liệu quan hệ. Thứ hai, về chức năng và loại truy

vấn, hệ thống chỉ hỗ trợ các thao tác đọc dữ liệu, không thể ghi, chỉnh sửa hay xóa nhằm bảo đảm toàn vẹn dữ liệu. Thứ ba, về nội dung và lĩnh vực ứng dụng, đề tài tập trung vào dữ liệu đặc thù của quản lý dữ liệu đại học, bao gồm thông tin sinh viên, ngành học, chương trình đào tạo, học phần, điểm tích lũy và văn bằng. Cuối cùng, về phạm vi ngôn ngữ, nghiên cứu hướng tới xử lý các câu truy vấn được diễn đạt bằng tiếng Việt.

5. Phương pháp nghiên cứu

Nghiên cứu được triển khai dựa trên ba trụ cột chính: nền tảng lý thuyết, mô hình hóa thực nghiệm và kiểm chứng khả năng triển khai thực tiễn. Các hoạt động nghiên cứu tuân theo một chuỗi logic từ việc thiết lập khung lý thuyết, qua giai đoạn kiến tạo hệ thống, đến khâu thẩm định và xác minh hiệu quả.

5.1. Nghiên cứu và tìm hiểu lý thuyết

Tìm hiểu các hệ thống và công cụ chuyển đổi ngôn ngữ tiếng Việt thành câu truy vấn SQL hiện có. Đề tài khảo sát các công cụ Text-to-SQL cho tiếng Việt, bao gồm các mô hình đa ngôn ngữ và những giải pháp được chuyển đổi từ tiếng Anh hay ngôn ngữ khác, phân tích kiến trúc cùng ưu nhược điểm của từng hướng tiếp cận để xác định giải pháp tương thích với kiến trúc đại học số Việt Nam.

Đề tài triển khai theo ba hướng nghiên cứu chính. Thứ nhất, về mặt kiến trúc dữ liệu, nghiên cứu khảo sát quy trình chuẩn hóa schema cùng phương pháp biểu diễn ngữ nghĩa và liên kết giữa các thực thể dữ liệu. Thứ hai, đề tài phân tích chuyên sâu các đặc trưng tiếng Việt trong bối cảnh Text-to-SQL, bao gồm cấu trúc cú pháp, hệ thống thuật ngữ giáo dục và các mẫu hình diễn đạt đặc thù, qua đó kiến tạo phương pháp luận xử lý nhằm tối đa hóa độ chính xác. Thứ ba, về công nghệ triển khai, nghiên cứu các kiến trúc mô hình xử lý ngôn ngữ Việt, tập trung vào năng lực trích xuất thông tin, nhận diện thực thể và ánh xạ cấu trúc, cũng như khả năng tích hợp API của mô hình ngôn ngữ lớn để sinh truy vấn SQL tự động.

5.2. Xây dựng các tài nguyên nền tảng

Đề tài tiến hành thiết lập ánh xạ có hệ thống giữa tập thuật ngữ tiếng Việt và các thành phần metadata của cơ sở dữ liệu khai thác trong lĩnh vực đại học số. Từ điển này thực hiện hai chức năng chính: Thống nhất cách diễn giải các khái niệm chuyên ngành và cung cấp nguồn tri thức nền tảng cho lớp ngữ nghĩa và mô hình LLM.

Đề tài thực hiện chuẩn hóa và mô tả chi tiết schema cơ sở dữ liệu hệ thống đại học số, bao gồm tài liệu hóa toàn diện các yếu tố: ngữ nghĩa nghiệp vụ, kiểu dữ liệu, quan hệ liên bảng. Thông tin được tổ chức theo định dạng chuẩn hóa nhằm đảm bảo tính nhất quán và tạo đầu vào tối ưu cho các mô hình xử lý ngôn ngữ.

5.3. Phát triển và tích hợp hệ thống

Đề tài ứng dụng các mô hình xử lý ngôn ngữ tiếng Việt, bao gồm chuẩn hóa văn bản, phân tích ý định người dùng và trích xuất thực thể, nhằm xác định chính xác phạm vi dữ liệu cần truy vấn từ câu hỏi tiếng Việt. Trên cơ sở đó, một mô-đun ánh xạ schema được phát triển để nhận diện và liên kết các thành phần ngôn ngữ trong câu hỏi với thông tin các trong cơ sở dữ liệu.

Nghiên cứu tích hợp mô hình ngôn ngữ lớn thông qua API bằng cách xây dựng cấu trúc prompt tối ưu có kết hợp thông tin schema và bộ từ điển, xử lý phản hồi từ mô hình để trích xuất câu truy vấn SQL, đồng thời kiểm tra và xác thực cú pháp trước khi thực thi. Bên cạnh đó, phải đảm bảo an toàn cho hệ thống chỉ thực hiện các truy vấn đọc, ngăn chặn các câu lệnh có khả năng thay đổi dữ liệu và kiểm soát quyền truy cập. Cuối cùng, một giao diện web thân thiện hỗ trợ người dùng nhập câu hỏi tiếng Việt tự nhiên, xem kết quả truy vấn dưới dạng bảng và cung cấp phản hồi về chất lượng hệ thống.

5.4. Thực nghiệm và đánh giá

Quy trình kiểm thử được triển khai trên bộ dữ liệu câu hỏi mẫu đã xây dựng, thực thi các câu truy vấn SQL do hệ thống sinh ra và kiểm tra kết quả thu được với mẫu chuẩn nhằm đánh giá mức độ chính xác. Bên cạnh đó, hiệu suất hệ thống được đánh giá toàn diện thông qua độ chính xác của kết quả sinh ra so với truy vấn chuẩn,

thời gian phản hồi từ khi người dùng nhập câu hỏi đến khi trả về kết quả, tỷ lệ trả lời đúng với các nhóm câu hỏi có mức độ phức tạp khác nhau, và kết quả tốt khi thực hiện khảo sát với người dùng thao tác trên hệ thống.

Nghiên cứu cũng phân loại và phân tích các lỗi phát sinh, tìm nguyên nhân và kiến nghị giải pháp cải tiến như tối ưu hóa prompt hoặc điều chỉnh cơ chế xử lý, với quy trình cải tiến cho đến khi đạt được các chỉ số chất lượng mong muốn. Hệ thống sau đó được triển khai thử nghiệm cho một nhóm người dùng thực tế nhằm theo dõi quá trình vận hành và thu thập phản hồi phục vụ việc hoàn thiện trước khi mở rộng triển khai. Phương pháp nghiên cứu tuân theo quy trình có hệ thống gồm năm giai đoạn: nghiên cứu lý thuyết, xây dựng tài nguyên nền tảng, phát triển hệ thống, thực nghiệm-đánh giá, và hoàn thiện để triển khai thực tiễn tại các cơ sở giáo dục đại học.

6. Cấu trúc của đề án

Ngoài phần Mở đầu , Kết luận vàPhụ lục, nội dung chính của đề án gồm có ba chương như sau:

Chương 1: Tổng quan về bài toán

Chương 2: Phân tích, đánh giá và thiết kế giải pháp

Chương 3: Xây dựng và phát triển hệ thống

CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN

1.1. Tổng quan về cơ sở dữ liệu

Một cơ sở dữ liệu được hình thành từ tập hợp các thông tin được tổ chức có cấu trúc, lưu trữ dưới dạng số và truy cập thông qua các hệ thống máy tính. Cơ sở dữ liệu đóng vai trò nền tảng trong tổ chức hiện đại, đảm bảo việc lưu trữ, quản lý và khai thác thông tin phục vụ vận hành và quá trình ra quyết định.

Đối với nhiều bài toán thực tiễn, ngày nay mô hình cơ sở dữ liệu có quan hệ đang được ứng dụng nhiều nhất. Bảng được cấu trúc theo mô hình hàng- cột, trong đó hàng biểu diễn bản ghi và cột biểu diễn thuộc tính. Để đảm bảo tính nhất quán các thao tác với dữ liệu phải thực hiện qua truy vấn SQL.

Một mô hình lưu trữ dữ liệu khác là mô hình cơ sở dữ liệu NoSQL được triển khai cho các tập dữ liệu phi cấu trúc hoặc có tính động cao, nhằm đáp ứng nhu cầu về khả năng mở rộng (scalability) và hiệu năng xử lý. Kiến trúc NoSQL được phát triển theo nhiều dạng thức khác nhau, tương thích với các yêu cầu lưu trữ và đặc tính dữ liệu riêng biệt.

Ngoài ra, để tối ưu cho việc xử lý dữ liệu lớn các mô hình cơ sở dữ liệu được tạo ra chuyên môn hóa cho các kho dữ liệu và phân tích kinh doanh. Các nền tảng như Teradata, Amazon Redshift hay Google BigQuery được xây dựng nhằm đáp ứng những tác vụ tính toán truy xuất với quy mô dữ liệu rất lớn.

Mặc dù có sự phát triển của nhiều mô hình khác nhau, cơ sở dữ liệu quan hệ vẫn được sử dụng nhiều trong việc quản lý dữ liệu có cấu trúc tại các tổ chức giáo dục, tài chính, y tế và hành chính công. Nền tảng lý luận của mô hình xuất phát từ lý thuyết quan hệ và đại số quan hệ, đảm bảo tính hình thức hóa và tính nhất quán logic trong cấu trúc dữ liệu.

Với tư cách là chuẩn ISO/ANSI, SQL được công nhận làm ngôn ngữ chính thống để làm việc với hệ cơ sở dữ liệu quan hệ. Ngôn ngữ này cung cấp các câu lệnh để xây dựng cấu trúc và các thao tác với dữ liệu, đồng thời kiểm soát quyền truy cập cũng như quản lý các giao dịch.

1.2. Tổng quan về đại học số và nhu cầu truy vấn dữ liệu

Đại học số (Digital University) được hiểu là mô hình triển khai công nghệ thông tin đồng bộ và toàn diện vào mọi hoạt động quản lý và hỗ trợ học thuật trong môi trường giáo dục đại học. Theo Nghị quyết 71 về phát triển giáo dục [2], việc xây dựng và triển khai đại học số sẽ nâng cao chất lượng giảng dạy, cải thiện hiệu quả vận hành và năng lực điều hành trong các đào tạo đại học.

Kiến trúc của một đại học số hiện đại gồm nhiều thành phần có tính liên kết mật thiết bao gồm các phần mềm quản lý và ứng dụng người học. Hệ thống lưu trữ rất nhiều thông tin liên như tuyển sinh, đào tạo, nghiên cứu khoa học, tổ chức nhân sự, tài chính.

Hiện nay, quá trình khai thác dữ liệu trong đại học số chủ yếu thông qua các phương thức: (i) truy cập các báo cáo dựng sẵn trên giao diện web – vốn thiếu tính linh hoạt; (ii) yêu cầu bộ phận công nghệ thông tin cung cấp khi cần truy vấn đặc thù; (iii) xuất dữ liệu thủ công dưới dạng Excel để phân tích; hoặc (iv) sử dụng các công cụ Business Intelligence, tuy nhiên vẫn đòi hỏi kiến thức kỹ thuật và khả năng hiểu mô hình dữ liệu quan hệ. Thực tiễn cho thấy người dùng là cán bộ, giảng viên thường gặp nhiều khó khăn trong việc truy vấn dữ liệu do thiếu kiến thức về SQL cũng như cấu trúc cơ sở dữ liệu. Các trở ngại phổ biến bao gồm: (i) không nắm được mô hình và quan hệ giữa các bảng; (ii) không có khả năng viết câu lệnh SQL; và (iii) không hiểu cách kết nối các bảng có quan hệ khóa ngoại.

Từ thực trạng này, nhu cầu về một công cụ cho phép người dùng tra cứu dữ liệu bằng tiếng Việt tự nhiên trở nên cấp thiết. Một hệ thống như vậy cần có khả năng: diễn giải chính xác mục đích của câu hỏi; ánh xạ câu hỏi sang mô hình dữ liệu; sinh câu truy vấn SQL chính xác và an toàn; trình bày kết quả theo dạng dễ hiểu; đảm bảo tốc độ phản hồi; và cung cấp giao diện người dùng cần rõ ràng và trực quan.

1.3. Tổng quan về trí tuệ nhân tạo

1.3.1. Khái niệm về trí tuệ nhân tạo

Nhận diện hình ảnh, xử lý ngôn ngữ, ra quyết định và học từ kinh nghiệm - những năng lực nhận thức đặc trưng của con người - đang được tái tạo trong các hệ thống máy móc thông qua lĩnh vực nghiên cứu trí tuệ nhân tạo (AI), một ngành then chốt của khoa học máy tính hiện đại.

Trong AI, có nhiều hướng tiếp cận chính. Thứ nhất là AI dựa trên các luật đã có sẵn. Thứ hai là học máy, nơi hệ thống học từ dữ liệu. Thứ ba là học sâu khai thác cấu trúc mạng nơ-ron nhiều lớp để tự động trích xuất và học các biểu diễn trừu tượng từ dữ liệu. Thứ tư là học tăng cường, nơi hệ thống học thông qua thử nghiệm và sai sót trong môi trường tương tác.

1.3.2. Xử lý ngôn ngữ tự nhiên

Việc cho phép máy tính hiểu và xử lý ngôn ngữ con người là mục tiêu cốt lõi của Xử lý ngôn ngữ tự nhiên (NLP) - một hướng nghiên cứu then chốt trong trí tuệ nhân tạo. Đối với tiếng Việt, lĩnh vực này phải đối diện với các thách thức từ đặc tính ngôn ngữ học phức tạp: tính nhập nhằng, sự phụ thuộc cao vào ngữ cảnh, đa dạng cấu trúc diễn đạt và nhu cầu về tri thức nền, khiến việc xây dựng hệ thống xử lý chính xác trở thành bài toán khó đòi hỏi phương pháp luận thích hợp.

NLP ứng dụng tập hợp các kỹ thuật nền tảng để chuyển đổi văn bản thành biểu diễn có thể xử lý được bởi máy tính, thông qua việc nhận diện từ ngữ, phân tích vai trò cú pháp của các thành phần câu và định danh thực thể. Các phương pháp này không chỉ giải quyết thách thức xử lý ngôn ngữ mà còn trở thành nền tảng công nghệ cho dịch máy, chatbot, trợ lý ảo, tóm tắt tự động, trích xuất thông tin và các hệ thống hỏi-đáp thông minh.

1.3.3. Mô hình ngôn ngữ lớn

Mô hình ngôn ngữ lớn (Large Language Models, viết tắt LLM) là các mô hình học sâu được huấn luyện trên khối lượng văn bản khổng lồ, có thể hiểu và tạo ra văn bản tự nhiên với chất lượng cao. LLM có các đặc điểm nổi bật: quy mô lớn; khả năng học từ ngữ cảnh để hiểu ý nghĩa mà không cần huấn luyện lại; tính đa năng trong việc thực hiện"nhiều nhiệm vụ khác nhau chỉ với prompt phù hợp; few-shot learning để

học từ vài ví dụ trong prompt; zero-shot learning để thực hiện nhiệm vụ mới mà không cần ví dụ.

Có nhiều mô hình ngôn ngữ lớn như BERT (Bidirectional Encoder Representations from Transformers) của Google với thế mạnh về hiểu ngôn ngữ; GPT (Generative Pre-trained Transformer) của OpenAI; T5 (Text-to-Text Transfer Transformer) của Google để xử lý mọi nhiệm vụ NLP như chuyển đổi text-to-text; LLaMA của Meta là mô hình mã nguồn mở; Claude của Anthropic; PaLM của Google là mô hình quy mô rất lớn.

LLM có nhiều khả năng ấn tượng: hiểu ngôn ngữ tiếng Việt để phân tích câu phức tạp và nắm bắt ý định; sinh văn bản mạch lạc và tự nhiên; thực hiện suy luận logic đơn giản; chuyển đổi giữa các định dạng khác nhau; học nhanh từ prompt để thích ứng với nhiệm vụ mới chỉ qua hướng dẫn.

Tuy nhiên, LLM cũng có các hạn chế: đưa ra thông tin không chính xác nhưng trông có vẻ đúng; thiếu kiến thức cập nhật vì chỉ biết thông tin đến thời điểm huấn luyện; chi phí tính toán cao và đòi hỏi tài nguyên lớn để chạy; khó kiểm soát vì đôi khi đưa ra kết quả không mong muốn; phụ thuộc prompt vì chất lượng phụ thuộc nhiều vào cách thiết kế prompt.

1.4. Bài toán chuyển đổi ngôn ngữ tiếng Việt thành SQL

1.4.1. Định nghĩa

Text-to-SQL là bài toán chuyển đổi các yêu cầu được diễn đạt bằng ngôn ngữ tiếng Việt thành câu truy vấn SQL có thể thực thi trực tiếp trên cơ sở dữ liệu. Mục tiêu của bài toán là thiết lập cơ chế liên kết giữa cách con người nêu câu hỏi và cách hệ thống quản trị cơ sở dữ liệu truy xuất thông tin. Trong mô hình này, đầu vào bao gồm câu hỏi tiếng Việt của người dùng; schema cơ sở dữ liệu mô tả các bảng, cột, kiểu dữ liệu, khóa và mối quan hệ; cùng với các thông tin tùy chọn như ví dụ mẫu, giá trị dữ liệu hoặc mô tả nghiệp vụ. Đầu ra là câu truy vấn SQL tương ứng phản ánh đúng yêu cầu thông tin.

Để thực hiện chuyển đổi thành công, hệ thống cần đồng thời đáp ứng nhiều yêu cầu quan trọng. Trước hết, hệ thống phải hiểu chính xác ý định của câu hỏi nhằm xác định nội dung người dùng muốn truy xuất. Tiếp theo, quá trình ánh xạ với schema phải được thực hiện để liên kết các thực thể được nhắc đến trong câu hỏi với bảng và cột tương ứng trong cơ sở dữ liệu. Hệ thống cũng phải xác định cấu trúc của câu truy vấn và diễn giải các điều kiện trong ngôn ngữ tiếng Việt thành biểu thức SQL phù hợp. Cuối cùng, câu lệnh SQL được sinh ra cần đảm bảo tính đúng đắn về cú pháp, khả năng thực thi và độ chính xác của kết quả trả về.

1.4.2. Các giai đoạn phát triển

Từ những năm 2000, các hệ thống Text-to-SQL chủ yếu dựa trên luật và template. Các hệ thống sử dụng các luật được viết thủ công để phân tích câu hỏi và ánh xạ với cấu trúc SQL. Mặc dù dễ kiểm soát và giải thích, cho kết quả chính xác với các câu hỏi trong phạm vi luật được định nghĩa, nhưng thường tốn công viết luật, khó mở rộng, không xử lý được câu hỏi đa dạng và khó bảo trì khi schema thay đổi.

Từ năm 2015, sự tiến bộ của học sâu đã thúc đẩy việc ứng dụng rộng rãi các mô hình mạng nơ-ron. Các kiến trúc Seq2Seq với LSTM/GRU và cơ chế Attention được sử dụng để chuyển đổi câu hỏi thành SQL. Năm 2018, dataset Spider được công bố bởi Yu et al. [5], đánh dấu bước ngoặt quan trọng với 10,181 câu hỏi trên 200 cơ sở dữ liệu. Chúng học từ dữ liệu thay vì luật, xử lý được câu hỏi đa dạng hơn và có khả năng tổng quát hóa. Tuy nhiên, chúng vẫn gặp hạn chế với câu hỏi phức tạp và cơ sở dữ liệu mới.

Từ năm 2020 đến nay, với các mô hình ngôn ngữ lớn, Text-to-SQL đã có bước tiến vượt bậc. Gao et al. (2024) [6] báo cáo rằng GPT-4 đạt 85.3% độ chính xác trên Spider dataset, vượt xa các phương pháp trước đó. LLM có ưu điểm: hiểu ngôn ngữ tiếng Việt tốt và thực hiện được câu hỏi phức tạp, đa dạng; few-shot learning để học nhanh từ vài ví dụ; zero-shot để xử lý cơ sở dữ liệu mới mà không cần huấn luyện lại; không cần dataset lớn nên giảm công sức thu thập dữ liệu.

Tuy nhiên, LLM cũng có thách thức: hallucination khi đôi khi sinh SQL không chính xác; chi phí khi sử dụng API thương mại; độ trễ vì thời gian xử lý có thể cao; bảo mật khi cần kiểm soát SQL được sinh ra.

1.4.3. Các thách thức trong bài toán

Theo Lei et al. (2020) [7], cơ chế liên kết schema đóng vai trò then chốt cho chất lượng của các hệ thống Text-to-SQL.Schema linking là quá trình thiết lập sự tương ứng giữa các thực thể xuất hiện trong câu hỏi và các thành phần tương ứng trong schema của cơ sở dữ liệu. Thách thức bao gồm: sự không khớp về từ vựng; tính mơ hồ khi một từ có thể ánh xạ với nhiều cột; cần ngữ cảnh để hiểu nghiệp vụ và ánh xạ đúng; schema lớn và phức tạp với nhiều bảng, nhiều cột và mối quan hệ phức tạp.

Hệ thống cần hiểu được các cách diễn đạt khác nhau này và sinh ra cùng một câu SQL. Ngoài ra, câu hỏi có thể chứa lỗi chính tả, ngữ pháp, hoặc câu hỏi không rõ ràng, thiếu thông tin.

Nascimento et al. (2025) [9] nghiên cứu Text-to-SQL cho cơ sở dữ liệu thực tế, chỉ ra rằng khoảng cách giữa dataset nghiên cứu và ứng dụng thực tế vẫn còn lớn do độ phức tạp và yêu cầu nghiệp vụ.

Các thách thức trong xử lý truy vấn phức tạp được phân loại theo ba nhóm chính. Thứ nhất, thách thức về tích hợp dữ liệu đa nguồn khi hệ thống cần xác định các nguồn dữ liệu liên quan và thiết lập mối liên kết chính xác giữa chúng dựa trên quan hệ. Thứ hai, thách thức về xử lý cấu trúc phân tầng khi nhiều câu hỏi đòi hỏi truy vấn phải có cấu trúc lồng nhau với các mức độ phụ thuộc khác nhau, yêu cầu hệ thống phải hiểu và tổ chức logic truy vấn theo đúng thứ tự thực thi. Thứ ba, thách thức về biểu diễn logic nghiệp vụ tiếng Việt sang câu truy vấn SQL phải làm rõ về mặt ngữ nghĩa nghiệp vụ và mô hình dữ liệu.

1.4.4. Các công cụ hiện có

Hiện nay có nhiều công cụ xử lý bài toán Text-to-SQL được phát triển, cả học thuật và thương mại. DBCopilot của Wang et al. (2025) [10] là hệ thống phải đáp

ứng nhiều cơ sở dữ liệu. ASKSQL (2025) [11] giúp giảm chi phí sử dụng mô hình ngôn ngữ tiếng Việt. SPS-SQL (2025) [12] tăng cường khả năng Text-to-SQL trên các mô hình nhỏ. WrenAI [15] là giải pháp mã nguồn mở sử dụng kiến trúc semantic engine và RAG (Retrieval Augmented Generation), đặc biệt có cơ chế tự kiểm tra và sửa lỗi SQL, hỗ trợ nhiều cơ sở dữ liệu.

Hầu hết kiến trúc của các công cụ Text-to-SQL gồm: module tiền xử lý để chuẩn hóa câu hỏi và phân tích cú pháp; module schema linking để ánh xạ câu hỏi với schema; module sinh SQL sử dụng LLM hoặc mô hình chuyên biệt; module hậu xử lý để kiểm tra cú pháp, tối ưu và kiểm tra an toàn; module thực thi để chạy SQL và trả về kết quả.

Các giải pháp dựa trên LLM có ưu điểm về khả năng hiểu ngôn ngữ và tính tổng quát, nhưng có chi phí cao và khó kiểm soát. Các giải pháp fine-tuning mô hình nhỏ có chi phí thấp hơn nhưng cần dataset lớn và khả năng tổng quát hạn chế. Các hệ thống thương mại có giao diện đẹp và tích hợp tốt nhưng thường đóng và tốn phí. Cần cân nhắc giữa độ chính xác, chi phí và tính linh hoạt.

1.5. Đặc thù của chuyển đổi ngôn ngữ tiếng Việt thành câu truy vấn SQL

Tiếng Việt thuộc nhóm ngôn ngữ đơn lập với nhiều đặc điểm khác biệt cơ bản so với tiếng Anh - ngôn ngữ chủ đạo trong nghiên cứu Text-to-SQL. Về mặt cấu trúc, tiếng Việt có hệ thống từ ghép phức tạp gây khó khăn cho việc phân tách và xử lý ngôn ngữ. Về ngữ pháp, cách diễn đạt, thứ tự từ và biểu đạt quan hệ logic trong tiếng Việt khác biệt đáng kể, đồng thời có tính đa nghĩa cao tùy theo ngữ cảnh. Đặc biệt, trong chuyên môn, tiếng Việt có hệ thống thuật ngữ riêng phản ánh đặc thù văn hóa và thể chế không thể dịch trực tiếp từ tiếng Anh.

Những đặc điểm này tạo ra các thách thức đặc thù khi phát triển hệ thống Text-to-SQL tiếng Việt. Thứ nhất, thiếu nghiêm trọng các công cụ NLP và mô hình ngôn ngữ chuyên biệt cho tiếng Việt so với tiếng Anh. Thứ hai, thiếu dataset Text-to-SQL chất lượng cao, với các dataset hiện có chủ yếu được dịch từ tiếng Anh nên không phản ánh cách diễn đạt tự nhiên của người Việt. Thứ ba, trong thực tế, cơ sở dữ liệu

tại Việt Nam thường sử dụng schema tiếng Anh trong khi câu hỏi người dùng là tiếng Việt, đòi hỏi xử lý phức tạp. Nghiên cứu của Dou et al. (2022) [13] cho thấy hiệu suất Text-to-SQL giảm trung bình 6.1% khi xử lý tiếng Việt so với tiếng Anh.

Về mặt tài nguyên, các dataset công khai cho Text-to-SQL tiếng Việt với khoảng 10,000 cặp câu hỏi-SQL, được dịch thủ công từ Spider dataset. Tiếp theo, Multi Spider của Dou et al. (2022) [13] và Multi Spider 2.0 (2025) [14] mở rộng sang đa ngôn ngữ có bao gồm tiếng Việt. Tuy nhiên, các dataset này vẫn tồn tại hạn chế: được xây dựng qua dịch thuật nên không phản ánh đúng cách diễn đạt tự nhiên của người Việt; thiếu thuật ngữ chuyên ngành đặc thù Việt Nam; và schema đã được chuyển tiếng Việt trong khi thực tế triển khai thường giữ nguyên tiếng Anh. Do đó, để phát triển hệ thống Text-to-SQL hiệu quả cho các lĩnh vực cụ thể như đại học số tại Việt Nam, cần xây dựng tài nguyên chuyên biệt bao gồm từ điển thuật ngữ, schema được chuẩn hóa và mô tả chi tiết, cùng bộ câu hỏi mẫu phản ánh ngữ cảnh nghiệp vụ thực tế.

1.6. Kết luận chương

Chương này đã thực hiện công tác hệ thống hóa các vấn đề nền tảng liên quan đến bài toán chuyển đổi ngôn ngữ tiếng Việt sang truy vấn SQL trong bối cảnh chuyển đổi số giáo dục đại học. Phân tích hiện trạng chỉ ra rằng mặc dù cơ sở dữ liệu quan hệ và ngôn ngữ SQL đóng vai trò trụ cột trong hệ thống quản lý thông tin, đa số người dùng phi chuyên vẫn gặp khó khăn đáng kể khi thực hiện các thao tác truy xuất dữ liệu, dẫn đến nhu cầu cấp thiết về một công cụ hỗ trợ thông minh có khả năng thu hẹp khoảng cách giữa ngôn ngữ tự nhiên và ngôn ngữ truy vấn hình thức. Sự phát triển nhanh chóng của trí tuệ nhân tạo đã mở ra những hướng tiếp cận đầy triển vọng cho bài toán Text-to-SQL. Tuy nhiên, các thách thức kỹ thuật như schema linking (liên kết giữa ngôn ngữ tự nhiên và cấu trúc cơ sở dữ liệu), sự khác biệt về đặc tính ngôn ngữ học giữa tiếng Việt và các ngôn ngữ phương Tây, cùng với hạn chế về nguồn dữ liệu huấn luyện chất lượng cao cho tiếng Việt vẫn là những rào cản đáng kể cần vượt qua.

Qua tổng quan các công trình nghiên cứu trong và ngoài nước cũng như phân tích và ứng dụng thực tiễn tại các cơ sở giáo dục đại học số, chương này đã xác định được khoảng trống nghiên cứu quan trọng: sự thiếu vắng một hệ thống Text-to-SQL tiếng Việt được thiết kế chuyên biệt cho nghiệp vụ và yêu cầu đặc thù của lĩnh vực giáo dục đại học. Từ nhận định này, đề tài đã xác lập hướng tiếp cận phù hợp dựa trên nền tảng mô hình ngôn ngữ lớn, kết hợp với các tài nguyên được tùy chỉnh và phát triển riêng bao gồm schema cơ sở dữ liệu chuẩn hóa, từ điển dữ liệu chuyên ngành và tập câu hỏi mẫu phản ánh đặc thù nghiệp vụ giáo dục đại học. Những"nội dung"được trình bày trong chương này không chỉ làm rõ bối cảnh và thách thức của bài toán mà còn tạo nền tảng vững chắc cho chương tiếp theo, nơi sẽ tập trung vào việc phân tích chi tiết các yêu cầu chức năng và phi chức năng, cũng như thiết kế kiến trúc tổng thể của giải pháp hệ thống.

CHƯƠNG 2: PHÂN TÍCH, ĐÁNH GIÁ VÀ THIẾT KẾ GIẢI PHÁP

2.1. Phân tích yêu cầu hệ thống

2.1.1. Phân tích yêu cầu chức năng

2.1.1.1. Yêu cầu về nhập và xử lý câu hỏi tiếng Việt

Hệ thống cần đảm bảo khả năng tiếp nhận và xử lý câu hỏi tiếng Việt từ người dùng thông qua một tập hợp các yêu cầu chức năng cốt lõi. Trước hết, hệ thống phải hỗ trợ nhiều hình thức diễn đạt khác nhau trong tiếng Việt, bao gồm câu hỏi trực tiếp, gián tiếp và các cấu trúc mang tính mệnh lệnh. Tính linh hoạt này là cần thiết bởi tiếng Việt có mức độ đa dạng cao trong ngữ pháp và cách đặt câu hỏi. Bên cạnh đó, hệ thống phải có khả năng xử lý các lỗi đầu vào thường gặp như sai chính tả, viết tắt không chuẩn hoặc thiếu dấu câu; đồng thời cung cấp cơ chế phát hiện và hiệu chỉnh tự động nhằm đảm bảo chất lượng dữ liệu đầu vào.

Ngoài ra, hệ thống cần thực hiện nhận diện chính xác ý định của người dùng để xác định loại tác vụ được yêu cầu, chẳng hạn như truy xuất thông tin, thống kê, so sánh hoặc phân tích dữ liệu. Quá trình này là bước quan trọng để sinh ra câu truy vấn SQL phù hợp. Song song đó, hệ thống phải trích xuất được các thông tin quan trọng từ câu hỏi, bao gồm các thực thể, thuộc tính, điều kiện và các toán tử logic, làm cơ sở cho quá trình ánh xạ với schema và xây dựng truy vấn một cách chính xác.

2.1.1.2. Yêu cầu về chuyển đổi và sinh câu truy vấn

Hệ thống được thiết kế để đáp ứng một tập hợp các chức năng quan trọng nhằm đảm bảo quá trình chuyển đổi từ ngôn ngữ tiếng Việt sang truy vấn SQL được thực hiện chính xác và hiệu quả. Trước hết, hệ thống phải thực hiện ánh xạ câu hỏi người dùng với schema của cơ sở dữ liệu, bao gồm việc liên kết các thực thể được đề cập trong câu hỏi với các bảng và thuộc tính tương ứng. Nghiên cứu của Lei et al. (2020) [7] nhấn mạnh rằng mức độ chính xác của hệ thống Text-to-SQL phụ thuộc đáng kể vào chất lượng của bước ánh xạ này. Quá trình xử lý cần bao gồm cả ánh xạ

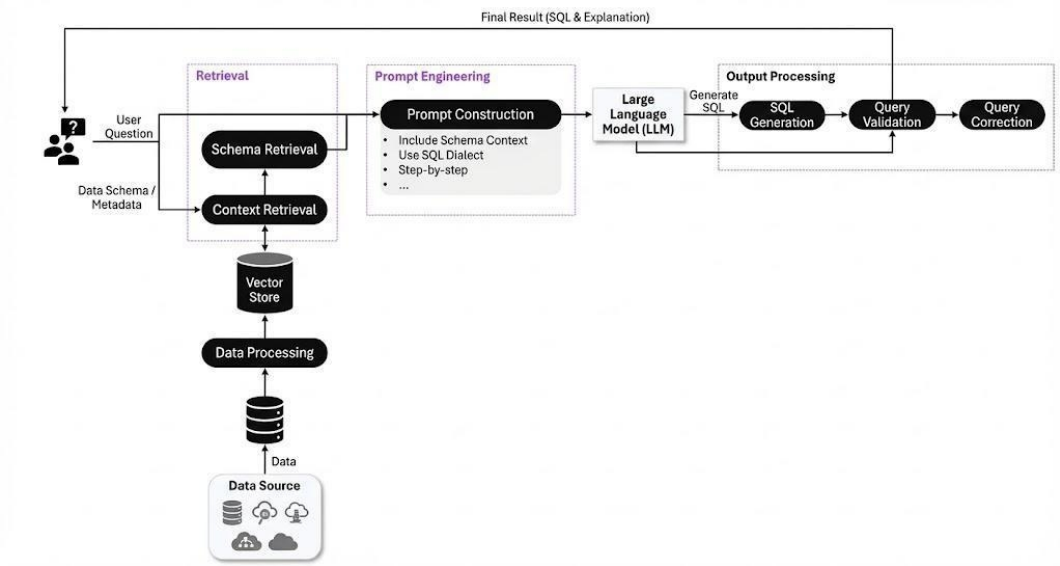
trực tiếp dựa trên sự tương đồng từ khóa và ánh xạ gián tiếp thông qua từ điển thuật ngữ và lớp ngữ nghĩa.

Dựa trên thông tin thu được từ bước phân tích và ánh xạ schema, hệ thống phải sinh ra câu truy vấn SQL có cú pháp hợp lệ và phản ánh chính xác ý định của người dùng. Bên cạnh các truy vấn đơn giản, hệ thống cần hỗ trợ xử lý các truy vấn phức tạp như kết nối nhiều bảng, truy vấn lồng nhau, phép tổng hợp, nhóm dữ liệu, lọc kết quả, sắp xếp và giới hạn số lượng bản ghi. Cuối cùng, để đảm bảo hiệu năng, câu SQL được tạo ra cần được tối ưu hóa bằng cách loại bỏ các phép toán dư thừa, tận dụng chỉ mục và áp dụng cơ chế giới hạn kết quả khi phù hợp. Các yêu cầu này nhằm đảm bảo hệ thống hoạt động ổn định, chính xác và đáp ứng được nhu cầu khai thác dữ liệu thực tế.

2.1.1.3. Thực thi và trả về kết quả

Sau khi sinh câu truy vấn SQL, hệ thống phải thực hiện quá trình thực thi và trình bày kết quả cho người dùng theo các yêu cầu kỹ thuật đã được xác định. Trước hết, việc thực thi cần đảm bảo tính an toàn, trong đó câu SQL phải được kiểm tra nhằm ngăn chặn nguy cơ SQL injection và các hình thức tấn công liên quan; đồng thời, hệ thống chỉ cho phép thực hiện các truy vấn ở chế độ đọc (READ-ONLY) để hạn chế rủi ro ảnh hưởng tới dữ liệu. Trong trường hợp truy vấn gặp lỗi cú pháp hoặc lỗi logic, hệ thống cần có khả năng phát hiện và ghi nhận thông tin chi tiết vào log, đồng thời có thể áp dụng cơ chế thử lại với truy vấn đã được điều chỉnh, hoặc hiển thị thông báo lỗi rõ ràng cho người dùng.

Kết quả truy vấn sau khi được trả về cần được định dạng theo hướng dễ hiểu và thân thiện, bao gồm việc trình bày dữ liệu dưới dạng bảng với tiêu đề cột bằng tiếng Việt, chuẩn hóa định dạng số và ngày tháng theo quy ước trong nước, và làm nổi bật các thông tin quan trọng. Bên cạnh dữ liệu thô, hệ thống có thể cung cấp phần giải thích ngắn gọn bằng ngôn ngữ tiếng Việt để hỗ trợ người dùng hiểu rõ hơn về nội dung kết quả. Cuối cùng, hệ thống cần hỗ trợ chức năng xuất dữ liệu sang văn bản truyền thống nhằm phục vụ các mục đích phân tích, lưu trữ hoặc báo cáo.



Hình 2.1: Text-to-sql pipeline

2.1.2. Phân tích yêu cầu phi chức năng

Hệ thống cần đảm bảo thời gian phản hồi dưới 2–3 giây đối với truy vấn đơn giản và 5–7 giây đối với truy vấn phức tạp. Khi thời gian xử lý vượt quá 10 giây, cần cung cấp cơ chế thông báo tiến độ để giảm cảm giác chờ đợi. Bên cạnh đó, hệ thống phải hỗ trợ xử lý đồng thời nhiều truy vấn và tối ưu mức sử dụng tài nguyên (CPU, RAM, băng thông, số lượng API call đến LLM) nhằm duy trì chi phí vận hành hợp lý.

Ngoài hiệu năng, hiệu năng và độ chính xác là chỉ số trọng yếu đánh giá chất lượng mô hình. Hệ thống phải đạt tối thiểu 80% câu truy vấn SQL có cú pháp hợp lệ và khả thi; với các truy vấn phức tạp, ngưỡng chấp nhận có thể ở mức 70%. Tuy nhiên, quan trọng hơn cú pháp là độ chính xác kết quả, với mục tiêu đạt 75–80% Execution Accuracy. Trong các trường hợp mô hình không chắc chắn, hệ thống cần có cơ chế làm rõ thông tin thông qua hỏi lại người dùng hoặc cung cấp các cách diễn giải thay thế.

Bảo mật là yêu cầu trọng yếu khi cho phép truy vấn cơ sở dữ liệu thông qua ngôn ngữ tiếng Việt. Hệ thống cần áp dụng biện pháp ngăn ngừa SQL Injection, bao gồm kiểm tra mẫu truy vấn, giới hạn câu lệnh cho phép, và chặn các từ khóa nguy hiểm như DROP, TRUNCATE, DELETE. Ngoài ra, hệ thống phải thực thi kiểm soát

quyền truy cập, tự động bổ sung các điều kiện lọc dựa trên vai trò người dùng theo chính sách phân quyền của đơn vị. Các dữ liệu nhạy cảm (ví dụ: số định danh cá nhân, địa chỉ, điểm thi chưa công bố) cần được ẩn hoặc yêu cầu quyền truy cập đặc biệt. Việc ghi nhận audit log đầy đủ (người dùng, thời điểm, câu hỏi, câu SQL được sinh, dữ liệu được truy xuất) là cần thiết để phục vụ giám sát và truy vết sự cố.

2.1.3. Xác định các ràng buộc và giới hạn

Hệ thống cần xác định rõ các ràng buộc và giới hạn nhằm bảo đảm tính ổn định và khả năng vận hành khi triển khai trong môi trường thực tế. Trước hết, phạm vi truy vấn được giới hạn ở các thao tác đọc dữ liệu; các thao tác ghi, cập nhật hoặc can thiệp cấu trúc cơ sở dữ liệu không được hỗ trợ. Bên cạnh đó, hiệu quả xử lý phụ thuộc đáng kể vào mức độ rõ ràng của đầu vào: hệ thống được tối ưu hóa cho các câu hỏi có cấu trúc và ngữ nghĩa minh bạch, trong khi các truy vấn mơ hồ hoặc yêu cầu suy luận sâu có thể dẫn đến suy giảm độ chính xác.

Phạm vi dữ liệu truy vấn cũng được kiểm soát chặt chẽ. Hệ thống chỉ làm việc với các bảng đã được cấu hình và có metadata đầy đủ trong semantic layer; các bảng mới hoặc không thuộc phạm vi cấu hình sẽ không được truy cập tự động. Ngoài ra, hệ thống có sự phụ thuộc vào dịch vụ mô hình ngôn ngữ lớn (LLM) thông qua API. Do đó, các sự cố kỹ thuật hoặc thay đổi chính sách vận hành từ phía nhà cung cấp có thể ảnh hưởng trực tiếp đến hiệu quả và chi phí vận hành của hệ thống.

2.2. Nghiên cứu và đánh giá các phương pháp chuẩn hóa schema

2.2.1. Khái niệm về chuẩn hóa schema

Schema cơ sở dữ liệu là mô tả hình thức về cấu trúc dữ liệu, bao gồm các bảng, thuộc tính, kiểu dữ liệu, khóa chính, khóa ngoại và các ràng buộc toàn vẹn. Trong bài toán Text-to-SQL, schema đóng vai trò nền tảng định hướng mô hình trong việc hiểu cấu trúc dữ liệu và xác định mối liên hệ giữa các thực thể. Qua đó, hệ thống có thể ánh xạ các khái niệm xuất hiện trong câu hỏi ngôn ngữ tiếng Việt tới các thành phần tương ứng trong cơ sở dữ liệu một cách chính xác và nhất quán.

Chuẩn hóa schema trong Text-to-SQL khác với chuẩn hóa trong thiết kế CSDL truyền thống. Nó không liên quan đến các dạng chuẩn (1NF, 2NF, 3NF) mà là quá trình biến đổi thông tin schema từ dạng kỹ thuật thuần túy (technical schema) sang dạng có ngữ nghĩa phong phú (semantic-rich schema) mà mô hình ngôn ngữ có thể sử dụng hiệu quả.

Chuẩn hóa schema bao gồm các hoạt động chính:

Truy vấn metadata: Thu thập thông tin cấu trúc từ hệ quản trị cơ sở dữ liệu: danh sách bảng, cột, kiểu dữ liệu, khóa, và ràng buộc.

Làm giàu thông tin ngữ nghĩa: Bổ sung thông tin chi tiết bằng ngôn ngữ tiếng Việt cho từng thành phần. Mô tả cần rõ ràng, ngắn gọn và phản ánh đúng ý nghĩa nghiệp vụ.

Chuẩn hóa tên gọi: Chuyển đổi tên bảng và cột từ dạng kỹ thuật sang dạng dễ hiểu hơn. Xử lý các trường hợp tên viết tắt, tên không rõ nghĩa, hoặc tên bằng tiếng Anh trong khi người dùng hỏi bằng tiếng Việt.

Xác định mối quan hệ: Ngoài ràng buộc khóa ngoại, cần xác định các liên kết ngữ nghĩa giữa các bảng và những quan hệ không được định nghĩa rõ ràng trong schema nhưng tồn tại trong logic nghiệp vụ.

Bổ sung ví dụ giá trị: Cung cấp một số giá trị mẫu điển hình cho mỗi cột để giúp mô hình hiểu rõ hơn về dữ liệu. Điều này quan trọng với các cột mã hóa hoặc có domain giá trị đặc thù.

Định nghĩa business rules: Ghi nhận các quy tắc nghiệp vụ liên quan đến dữ liệu.

2.2.2. Các phương pháp chuẩn hóa schema

Có nhiều phương pháp khác nhau để chuẩn hóa schema. Khảo sát ba nhóm phương pháp chính được sử dụng trong các hệ thống Text-to-SQL. Chúng sử dụng các định dạng chuẩn để mô tả metadata cơ sở dữ liệu, giúp tạo ra một representation thống nhất và dễ xử lý. Các định dạng phổ biến bao gồm:

JSON Schema là phương pháp mô tả cấu trúc cơ sở dữ liệu dưới dạng JSON, trong đó mỗi bảng và mỗi cột được biểu diễn bằng các đối tượng chứa các thuộc tính

như name, type, description và constraints. Cách tiếp cận này có ưu điểm về tính linh hoạt, khả năng chuyển đổi thuận tiện và hỗ trợ kiểm tra tính hợp lệ một cách tự động. Tuy vậy, JSON Schema còn hạn chế trong việc biểu diễn các mối quan hệ phức tạp giữa các bảng cũng như các quy tắc nghiệp vụ nâng cao.

YAML-based Schema: Tương tự JSON nhưng sử dụng YAML để dễ đọc và viết hơn cho con người. YAML thường được sử dụng trong các công cụ như dbt (data build tool) để định nghĩa models và relationships. Định dạng này cho phép thêm comments và có cấu trúc phân cấp rõ ràng, phù hợp với việc quản lý schema phức tạp.

DDL (Data Definition Language) mở rộng: Sử dụng các câu lệnh CREATE TABLE chuẩn của SQL nhưng được bổ sung comments và annotations để mô tả ngữ nghĩa. Phương pháp này có ưu điểm là gần gũi với developers quen thuộc với SQL, nhưng hạn chế trong việc biểu diễn metadata phức tạp.

OpenAPI/Swagger cho Database: Mở rộng ý tưởng của Open API specification để mô tả database schema như API. Phương pháp này đặc biệt hữu ích khi cơ sở dữ liệu được expose qua REST API hoặc GraphQL.

2.2.3. Đánh giá và lựa chọn phương pháp phù hợp

Sau khi khảo sát các phương pháp chuẩn hóa schema, chúng tôi tiến hành đánh giá và lựa chọn phương pháp phù hợp với hệ thống đại học số dựa trên các tiêu chí cụ thể.

Dựa trên phân tích ở trên và đặc thù của hệ thống, chúng tôi đề xuất sử dụng phương pháp kết hợp bao gồm ba thành phần chính:

1. JSON Schema làm nền tảng cấu trúc:

Sử dụng JSON để lưu trữ metadata cơ bản của schema (tables, columns, types, keys, constraints). JSON dễ phân giải, dễ tích hợp với các công cụ hiện đại, và có thể validate tự động. Cấu trúc JSON sẽ tuân theo schema chuẩn, đảm bảo tính nhất quán.

2. Natural Language Descriptions làm giàu ngữ nghĩa:

Mỗi bảng và cột đều có mô tả chi tiết bằng tiếng Việt. Mô tả bao gồm: ý nghĩa nghiệp vụ, các quy tắc liên quan, và các thông tin như giá trị tham chiếu, các mô tả liên quan đến khóa ngoại các liên kết khi cấu hình. Điều này giúp LLM hiểu rõ ngữ cảnh và xử lý tốt câu hỏi tiếng Việt.

3. Linearization được tối ưu cho LLM:

Khi cần đưa schema vào context của LLM, sử dụng phương pháp linearization với thứ tự được tối ưu: (a) Ưu tiên các bảng có khả năng liên quan cao với câu hỏi (thông qua schema linking); (b) Sắp xếp theo mức độ quan trọng và tần suất sử dụng; (c) Sử dụng format ngắn gọn nhưng đầy đủ thông tin để tiết kiệm token.

Phương pháp này tận dụng được ưu điểm của nhiều approach khác nhau: JSON Schema cung cấp kiến trúc và tiêu chuẩn; Natural Language Descriptions cung cấp semantic richness; Linearization tối ưu cho LLM processing. Đồng thời, phương pháp này tương thích tốt với kiến trúc Semantic Layer của WrenAI, giúp việc tích hợp trở nên dễ dàng.

Trong Chương 3, chúng tôi sẽ trình bày chi tiết về cách triển khai phương pháp này, bao gồm cấu trúc JSON Schema cụ thể, quy trình tạo descriptions, và thuật toán linearization cho từng use case.

2.3. Nghiên cứu xây dựng bộ từ điển dữ liệu

Từ điển dữ liệu là một tài liệu mô tả cấu trúc, nội dung và các định nghĩa cho bộ dữ liệu. Từ điển dữ liệu là công cụ then chốt đối với khả năng tái lập (reproducibility) vì nó giúp người khác hiểu được dữ liệu của bạn.

Từ điển dữ liệu (Data Dictionary) đóng vai trò quan trọng trong việc kết nối ngôn ngữ tiếng Việt với cấu trúc cơ sở dữ liệu. Trong bối cảnh chuyển đổi ngôn ngữ tiếng Việt thành câu truy vấn SQL cho đại học số, từ điển dữ liệu không chỉ đơn thuần là danh sách thuật ngữ mà còn là cầu nối giữa cách diễn đạt của người Việt với schema database thường được đặt tên theo các quy tắc đặt tên trong lập trình.

2.3.1. Vai trò của từ điển dữ liệu trong Text-to-SQL

Từ điển dữ liệu trong hệ thống Text-to-SQL có vai trò khác biệt so với từ điển thông thường. Nó không chỉ định nghĩa ý nghĩa của các thuật ngữ mà còn thiết lập mối liên kết cụ thể giữa ngôn ngữ tiếng Việt và cơ sở dữ liệu. Việc ánh xạ chính xác giữa thông tin trong câu hỏi với thuộc tính trong cơ sở dữ liệu là một trong những yếu tố quyết định hiệu suất của hệ thống.

Hỗ trợ schema linking: Từ điển dữ liệu cung cấp thông tin để hệ thống có thể nhận diện khi người dùng đề cập đến một bảng hoặc cột trong cơ sở dữ liệu.

Xử lý đa ngôn ngữ và đa cách diễn đạt: Trong môi trường Việt Nam, một khái niệm có thể được gọi bằng nhiều tên khác nhau. Từ điển cần lưu trữ tất cả các biến thể này và ánh xạ chúng về cùng một thực thể trong cơ sở dữ liệu. Điều này quan trọng với tiếng Việt vì có nhiều cách diễn đạt cùng một ý.

Cung cấp ngữ cảnh nghiệp vụ: Từ điển lưu trữ không chỉ tên gọi mà còn định nghĩa, ví dụ, và các quy tắc liên quan đến từng thuật ngữ.

Hỗ trợ xử lý câu hỏi mơ hồ: Khi người dùng sử dụng thuật ngữ không rõ ràng hoặc có thể hiểu theo nhiều cách, từ điển giúp hệ thống đưa ra các khả năng diễn giải và hỏi lại người dùng.

Tăng độ chính xác của LLM: Khi đưa vào context của mô hình ngôn ngữ lớn, thông tin từ từ điển giúp LLM hiểu chính xác hơn về thông tin và sinh ra câu SQL đúng đắn. Thay vì để LLM phải đoán ý nghĩa của các thuật ngữ, từ điển cung cấp định nghĩa rõ ràng.

Hỗ trợ mở rộng và bảo trì: Khi thêm bảng/cột mới hoặc thay đổi thuật ngữ trong tổ chức, việc cập nhật từ điển đơn giản hơn so với phải huấn luyện lại mô hình hoặc thay đổi mã nguồn. Từ điển đóng vai trò như một cơ sở tri thức có thể quản lý độc lập.

2.3.2. Phương pháp xây dựng từ điển dữ liệu

Xây dựng từ điển dữ liệu cho hệ thống Text-to-SQL trong lĩnh vực đại học số là một quy trình có hệ thống, bao gồm ba giai đoạn chính: thu thập thuật ngữ, phân loại và chuẩn hóa, và thiết lập ánh xạ với cơ sở dữ liệu.

Giai đoạn thu thập là nền tảng cho chất lượng của toàn bộ từ điển. Cần tiếp cận từ nhiều nguồn khác nhau để đảm bảo tính đầy đủ và phản ánh đúng cách sử dụng thực tế:

Từ tài liệu chính thức: Thu thập từ các văn bản pháp lý, quy chế, quy định của Bộ Giáo dục và Đào tạo, các thông tư hướng dẫn đào tạo, và quy chế của nhà trường. Các tài liệu này cung cấp thuật ngữ chuẩn và định nghĩa chính thức.

Từ hệ thống thông tin hiện có: Phân tích schema database hiện tại của trường để xác định các thực thể và thuộc tính đã được sử dụng. Xem xét tên bảng, tên cột, chú thích trong database, và tài liệu kỹ thuật để hiểu rõ cấu trúc dữ liệu hiện tại. Đây là nguồn quan trọng vì cần ánh xạ ngược từ database về ngôn ngữ tiếng Việt.

Từ người dùng thực tế: Phỏng vấn và khảo sát cán bộ đào tạo, giảng viên, và sinh viên để thu thập các thuật ngữ họ thường sử dụng khi cần tra cứu thông tin. Ghi nhận cả cách nói chính thức và cách nói thông thường.

Từ các câu hỏi mẫu: Thu thập các câu hỏi thường gặp từ phòng đào tạo, hệ thống hỏi đáp, email, và các kênh tương tác với sinh viên. Phân tích các câu hỏi này để trích xuất các thực thể,thuộc tính, và cách diễn đạt phổ biến.

Từ các hệ thống tương tự: Nghiên cứu từ điển thuật ngữ của các trường đại học khác hoặc các hệ thống quản lý sinh viên, quản lý đào tạo, văn bằng chứng chỉ, LMS để học hỏi và tham khảo.

Sau khi thu thập, cần tổ chức và chuẩn hóa các thuật ngữ để tạo thành một hệ thống nhất quán:

Phân loại theo nhóm thực thể: Chia thuật ngữ thành các nhóm theo loại thực thể nghiệp vụ như: thực thể chính, quan hệ, hành động.

Xác định thuật ngữ chuẩn: Với mỗi khái niệm, chọn một thuật ngữ làm chuẩn dựa trên: tính chính thống dựa trên các văn bản hướng dẫn, tính phổ biến, và tính rõ ràng.

Xây dựng định nghĩa chuẩn: Viết định nghĩa ngắn gọn, rõ ràng cho mỗi thuật ngữ. Định nghĩa nên giải thích ý nghĩa nghiệp vụ, có ví dụ mô tả rõ ràng, có ghi chú nếu cần thiết.

Bước cuối cùng và quan trọng nhất là tạo ánh xạ từ thuật ngữ nghiệp vụ đến các thành phần cụ thể trong schema trong cơ sở dữ liệu:

- Ánh xạ entity-to-table: Liên kết mỗi entity nghiệp vụ với một hoặc nhiều bảng trong database.
- Ánh xạ attribute-to-column: Liên kết thuộc tính nghiệp vụ với cột cụ thể. Cần ghi nhận.
- Ánh xạ relationship: Ghi nhận cách biểu diễn mối quan hệ nghiệp vụ trong cơ sở dữ liệu.
- Ánh xạ value domains: Với các cột có tập giá trị hữu hạn.
- Ghi chú các điều kiện đặc biệt: Lưu lại các lưu ý quan trọng cho việc query.

2.3.3. Cấu trúc và nội dung từ điển dữ liệu

2.3.3.1. Cấu trúc lưu trữ từ điển

Cấu trúc của từ điển dữ liệu có thể được thiết kế theo nhiều dạng khác nhau, tùy thuộc vào mục đích ứng dụng và nền tảng công nghệ được sử dụng trong hệ thống:

Cấu trúc JSON/YAML: Đây là format phổ biến và dễ sử dụng. Mỗi entry trong từ điển là một object với các thuộc tính như: thuật ngữ chuẩn, định nghĩa, phân loại,ánh xạ với database, và thông tin bổ sung. JSON/YAML dễ đọc, dễ parse, và tương thích tốt với nhiều các ngôn ngữ lập trình.

Cơ sở dữ liệu quan hệ: Sử dụng các bảng trong database để lưu từ điển. Thiết kế bao gồm: bảng 'terms' (thuật ngữ chính), 'term variants' (các biến thể), 'term mapping' (ánh xạ với schema), 'term_relationships' (quan hệ giữa các thuật ngữ). Cách này phù hợp khi từ điển lớn, phức tạp, và cần query nhanh.

Graph database: Biểu diễn từ điển dưới dạng đồ thị với nodes là các thuật ngữ và edges là quan hệ giữa chúng (synonym, hypernym, hyponym, related_to). Neo4j hoặc các graph databases khác cho phép query phức tạp về sự tương quan giữa các khái niệm. Phù hợp khi cần khai thác semantic relationships sâu.

Vector database: Kết hợp với embeddings để hỗ trợ semantic search. Mỗi thuật ngữ được embed thành vector, cho phép tìm kiếm các thuật ngữ tương tự dựa trên similarity. Đặc biệt hữu ích khi xử lý các câu hỏi không khớp chính xác với thuật ngữ trong từ điển.

Hybrid approach: Kết hợp nhiều phương pháp: JSON files cho việc quản lý và version control, PostgreSQL cho storage và query nhanh, vector database cho semantic search. Đây là cách tiếp cận được đề xuất vì tận dụng được ưu điểm của từng loại.

2.3.3.2. Các loại ánh xạ trong từ điển

Trong một hệ thống từ điển dữ liệu chuẩn không chỉ là liên kết bảng mà là cầu nối giữa ngôn ngữ tiếng Việt và cấu trúc dữ liệu lưu trữ trên máy tính

Các loại ánh xạ dữ liệu:

- Ánh xạ ngữ nghĩa: Là mối quan hệ giữa thuật ngữ nghiệp vụ và phần tử dữ liệu.
- Ánh xạ miền giá trị: Ánh xạ giữa mã lưu trữ và giá trị hiển thị.
- Ánh xạ cấu trúc - quan hệ: Mô tả mối quan hệ giữa hai hoặc nhiều bảng trong quan hệ dữ liệu.
- Ánh xạ nguồn - đích: Chỉ ra dữ liệu ở bảng hiện tại được lấy từ nguồn nào.

2.3.3.3. Cơ chế cập nhật và mở rộng

Cơ chế cập nhật dữ từ điển sinh ra để giải quyết vấn đề khi cấu trúc dữ liệu thực tế thay đổi nhưng tài liệu mô tả chưa kịp thay đổi theo.

Các loại cơ chế cập nhật từ điển dữ liệu sau:

- Cập nhật thụ động: Chuyên viên thực hiện thay đổi cấu trúc bảng thì họ phải tự cập nhật vào tài liệu để cập nhật thủ công. Điều này giúp mô tả chi tiết ngữ nghĩa mà máy không hiểu được. Tuy nhiên, việc phụ thuộc vào con người dễ xảy ra việc từ điển không cập nhật kịp thời sẽ chạy sai yêu cầu.

- Cập nhật chủ động: Sử dụng các công cụ tự động rà quét thay đổi từ cơ sở dữ liệu để cập nhật vào từ điển dữ liệu. Điều này giúp đảm bảo độ chính xác 100% về mặt cấu trúc kỹ thuật. Tuy nhiên, mô tả của các thuộc tính sẽ không sát với ý nghĩa nghiệp vụ.
- Cập nhật kết hợp: Kết hợp giữa tự động của các công cụ và con người. Điều này giúp đảm bảo độ chính xác về mặt cấu trúc kỹ thuật và mô tả ngữ nghĩa sát với ý nghĩa nghiệp vụ.

Cơ chế mở rộng từ điển là cơ chế bổ sung thêm bảng mới và làm giàu ngữ nghĩa để phục vụ các mục đích cao hơn.

Các loại cơ chế mở rộng từ điển dữ liệu sau:

- Mở rộng tập từ đồng nghĩa: Giúp làm giàu ý nghĩa nghiệp vụ của một trường thông tin kỹ thuật lưu trong cơ sở dữ liệu, trong xử lý ngôn ngữ tiếng Việt cơ chế mở rộng này cho phép gán nhiều nhãn vào một trường thông tin.
- Mở rộng qua vòng lặp phản hồi: Là cơ chế quan trọng nhất để từ điển dữ liệu thông minh hơn theo thời gian dựa trên hành vi thực tế đã diễn ra trước đó.
- Mở rộng metadata thống kê: Lưu cấu trúc tính, từ điển mở rộng sẽ lưu thêm đặc điểm dữ liệu. Điều đó giúp tối ưu hóa truy vấn và giúp các mô hình trí tuệ nhân tạo hiểu ngữ cảnh dữ liệu sâu hơn.

2.4. Nghiên cứu thiết kế Semantic Layer cho hệ thống

2.4.1. Vai trò của Semantic Layer trong Text-to-SQL

Semantic Layer là một lớp trừu tượng nằm giữa dữ liệu thô (raw data) trong cơ sở dữ liệu vật lý và người dùng cuối. Chức năng chính của lớp này là dịch các cấu trúc dữ liệu kỹ thuật phức tạp thành các khái niệm nghiệp vụ dễ hiểu.

Trong bài toán Text-to-SQL, Semantic Layer đóng vai trò là cung cấp ngữ cảnh rõ ràng về ý nghĩa của dữ liệu giúp các mô hình ngôn ngữ lớn sinh ra câu lệnh SQL chính xác hơn.

Khi đặt câu hỏi, hệ thống không gửi toàn bộ schema cho LLM. Thay vào đó, nó phân tích câu hỏi để xác định các thực thể liên quan, rồi truy vấn Semantic Layer để lấy đúng các model và metric cần thiết. Semantic Layer chỉ lưu trữ nghĩa và cấu trúc dữ liệu - nó không tự chọn. Phần metadata đã chọn lọc sau đó mới được gửi vào LLM để tạo SQL. Cách làm này giảm nhiễu, giúp giảm thiểu chi phí và tăng độ chính xác của LLM hơn

Trong lĩnh vực đại học cấu trúc dữ liệu rất đặc thù và phức tạp. Semantic Layer giải quyết các vấn đề sau như: chuẩn hóa thuật ngữ, tính nhất quán của chỉ số, đảm bảo tính bảo mật thông tin nhạy cảm.

2.4.2. Phân tích các thành phần của Semantic Layer

Một Semantic Layer hoàn chỉnh trong các framework hiện đại (như WrenAI, Cube, dbt) thường bao gồm 4 thành phần cốt lõi.

- Business Models và Entities: Đây là sự ánh xạ từ thông tin các bảng sang các đối tượng nghiệp vụ. Loại bỏ các cột không cần thiết, đổi tên cột từ kỹ thuật sang ngôn ngữ tiếng Việt.
- Relationships và Connections: Mô tả cách các Business Models liên kết với nhau. Đây là chỉ dẫn quan trọng nhất để LLM biết cách thực hiện phép tính toán quan hệ. Relationships giúp LLM hiểu cách join các bảng khi cần. Thay vì phải đoán xem join thế nào, LLM sử dụng relationship đã được định nghĩa sẵn với ý nghĩa rõ ràng.
- Metrics và Calculations: Metrics là các chỉ số được tính toán, tiêu biểu cho các phép đo nghiệp vụ. Đây là thành phần quan trọng giúp người dùng có thể hỏi về các chỉ số phức tạp mà không cần biết công thức tính. Hệ thống thiết lập trước các công thức tính toán nhằm đảm bảo tính nhất quán, không phụ thuộc vào việc mô hình tự tạo ra công thức.
- Context và Descriptions: Context và Descriptions là các thông tin bổ sung giúp LLM hiểu sâu hơn về domain và cách sử dụng các thành

phần trong Semantic Layer. Context càng phong phú thì LLM càng xử lý tốt các câu hỏi phức tạp, cần yêu cầu suy luận nhiều.

2.4.3. Thiết kế Semantic Layer cho đại học số

Semantic Layer (tầng ngữ nghĩa) đã được chứng minh là yếu tố then chốt trong việc cải thiện độ chính xác của công cụ Text-to-SQL. Nghiên cứu của Curran và Mariani (2024) tại AtScale cho thấy việc tích hợp semantic layer đã nâng độ chính xác từ 20% lên 92.5% trên dataset TPC-DS với 40 câu hỏi phức tạp [8]. Đặc biệt, hệ thống đạt 100% độ chính xác trên các câu hỏi đơn giản khi có semantic layer, so với chỉ 60% khi không có. Kết quả này khẳng định vai trò không thể thiếu của semantic layer trong việc ánh xạ ngôn ngữ tiếng Việt sang cấu trúc dữ liệu.

Semantic layer đóng vai trò như một "kho lưu trữ tập trung cho logic nghiệp vụ, các chỉ số KPI và mối quan hệ giữa các bảng", giúp các nhóm khác nhau trong tổ chức sử dụng các định nghĩa KPI nhất quán [8]. Thay vì cung cấp trực tiếp DDL (Data Definition Language) của bảng cho LLM, metadata của semantic layer được đưa vào prompt. Điều quan trọng là chỉ metadata được gửi đến LLM, không có dữ liệu thực tế từ các bảng trong data warehouse, đảm bảo tính bảo mật. Hệ thống AtScale sử dụng Query Engine để thực thi các truy vấn được sinh ra thay vì truy vấn trực tiếp vào data warehouse, cho phép kiểm soát và tối ưu hóa tốt hơn [8].

Áp dụng vào bối cảnh đại học số, semantic layer cần được thiết kế để mã hóa các đặc thù nghiệp vụ phức tạp. Semantic layer cho phép "định nghĩa chính xác các thuật ngữ nghiệp vụ, đảm bảo rằng khi người dùng truy vấn dữ liệu bằng các thuật ngữ quen thuộc, hệ thống biết chính xác dữ liệu nào cần truy xuất và cách xử lý" [9]. Trong đại học số, điều này có nghĩa là các khái niệm cần được định nghĩa một lần duy nhất trong MDL với đầy đủ logic tính toán, điều kiện và mối quan hệ. Khi người dùng hỏi bằng nhiều cách diễn đạt khác nhau, semantic layer đảm bảo SQL được sinh ra luôn tuân theo cùng một logic nghiệp vụ đã được định nghĩa.

Tóm lại, semantic layer dựa trên MDL là giải pháp đã được chứng minh hiệu quả bởi các nghiên cứu và triển khai thực tế của AtScale [8], WrenAI và Denodo. Việc áp dụng MDL cho đại học số không chỉ giúp cải thiện độ chính xác Text-to-

SQL mà còn đảm bảo tính nhất quán, khả năng bảo trì và mở rộng hệ thống. Semantic layer đóng vai trò như "lớp dịch" giữa ngôn ngữ tiếng Việt của người dùng và cấu trúc kỹ thuật của cơ sở dữ liệu, cho phép hệ thống hiểu đúng ngữ cảnh nghiệp vụ và sinh ra SQL chính xác, đáng tin cậy.

2.5. Thiết kế cấp cao (High-level design)

Thiết kế Cấp cao của hệ thống chuyển đổi ngôn ngữ tiếng Việt thành SQL được xây dựng dựa trên kiến trúc Retrieval-Augmented Generation (RAG) chuyên biệt cho Text-to-SQL. Kiến trúc này sử dụng WrenAI Engine làm trung tâm điều phối, tách biệt rõ ràng giữa lớp ngôn ngữ tiếng Việt và lớp dữ liệu vật lý. Thiết kế HLD được tổ chức thành ba lớp logic chính nhằm đảm bảo tính mô-đun hóa, khả năng mở rộng và dễ bảo trì : Lớp Giao diện và Tương tác, Lớp Ứng dụng và Điều phối, và Lớp Trí tuệ và Dữ liệu Cốt lõi.

2.5.1. Lớp Giao diện và Tương tác

Lớp này cung cấp kênh giao tiếp chính giữa người dùng cuối và hệ thống xử lý, tập trung vào trải nghiệm người dùng.

- Web Portal/Chatbot UI: Cung cấp giao diện trực quan cho phép người dùng nhập câu hỏi bằng ngôn ngữ tiếng Việt tự nhiên.
- Hiển thị Kết quả: Trình bày kết quả truy vấn dưới dạng dễ hiểu, bao gồm bảng dữ liệu, câu lệnh SQL được sinh ra, và khả năng hiển thị biểu đồ trực quan hóa dữ liệu.
- Quản lý Tương tác: Hỗ trợ chức năng lịch sử truy vấn và khả năng điều chỉnh câu hỏi hoặc câu truy vấn SQL khi cần.

2.5.2. Lớp Ứng dụng và Điều phối

Lớp này điều phối toàn bộ luồng xử lý truy vấn, từ tiếp nhận câu hỏi đến xử lý và trả về kết quả, hoạt động như một dịch vụ không lưu trạng thái.

- API Gateway: Điểm tiếp nhận mọi yêu cầu từ Lớp Giao diện.

- Module Tiền xử lý Ngôn ngữ Việt: Thực hiện chuẩn hóa câu hỏi và phân tích ý định, trích xuất thực thể (Slot Filling).
- Dịch vụ Điều phối RAG: Quản lý toàn bộ quy trình Truy xuất tăng cường sinh (RAG), chịu trách nhiệm truy xuất các thành phần cấu trúc liên quan và lắp ráp chúng thành ngữ cảnh cho LLM.
- Module Xử lý Lỗi và Tự Sửa lỗi: Áp dụng cơ chế kiểm tra và sửa lỗi tự động cho câu truy vấn SQL được sinh ra, sử dụng vòng lặp phản hồi từ lỗi thực thi.

2.5.3. Lớp Trí tuệ và Dữ liệu Cốt lõi

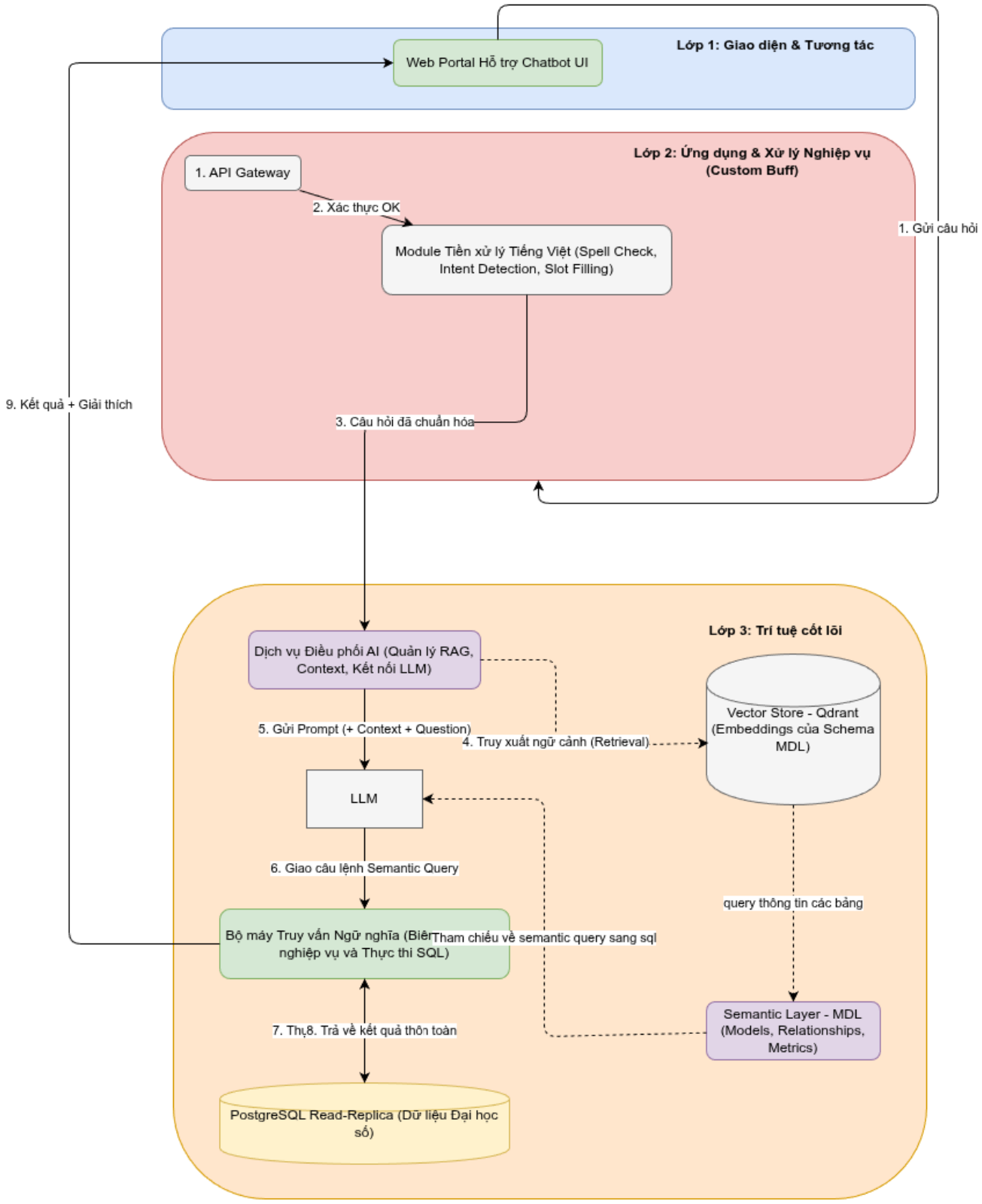
Lớp này cung cấp tri thức nền tảng và năng lực sinh truy vấn, là thành phần quyết định độ chính xác của hệ thống.

- Semantic Layer (Tầng Ngữ nghĩa): Là lớp trừu tượng hóa, mã hóa logic nghiệp vụ, định nghĩa các chỉ số (Metrics), mối quan hệ (Relationships), và Models. Vai trò quan trọng nhất là ánh xạ thuật ngữ nghiệp vụ tiếng Việt sang cấu trúc kỹ thuật của cơ sở dữ liệu.
- Vector Store (Qdrant): Cơ sở dữ liệu vector lưu trữ embeddings của schema, ví dụ câu hỏi-SQL, và metadata. Được sử dụng cho tìm kiếm ngữ nghĩa (Semantic Search) tìm ngữ cảnh.
- Mô hình Ngôn ngữ Lớn (LLM): Thành phần trung tâm thực hiện”chuyển đổi từ ngôn ngữ tiếng Việt sang câu truy vấn SQL dựa trên ngữ cảnh được cung cấp từ Semantic Layer.
- Bộ máy Truy vấn Ngữ nghĩa (Query Execution Engine): Chịu trách nhiệm kiểm tra tính hợp lệ và an toàn của SQL (ngăn chặn thao tác nguy hiểm như DELETE, DROP) , sau đó thực thi câu truy vấn ở chế độ chỉ đọc trên cơ sở dữ liệu nguồn.
- Cơ sở dữ liệu Nguồn: Bao gồm PostgreSQL Read-Replica, chứa dữ liệu thực tế của hệ thống đại học số.

2.5.4. Luồng Xử lý Từ Đầu Đến Cuối

Luồng xử lý một truy vấn ngôn ngữ tiếng Việt (NL) trong kiến trúc RAG diễn ra qua sáu giai đoạn chính:

- Hiểu Câu hỏi: Phân tích câu hỏi NL để xác định ý định, các thực thể và ràng buộc.
- Truy xuất Cấu trúc: Sử dụng tìm kiếm ngữ nghĩa (từ Vector Store) để truy xuất các thành phần cấu trúc (bảng, cột, quan hệ, metrics) liên quan từ Semantic Layer.
- Lắp ráp Ngữ cảnh: Kết hợp cấu trúc đã truy xuất, ví dụ mẫu và quy tắc nghiệp vụ thành câu nhắc toàn diện (Prompt Construction).
- Sinh SQL: LLM sinh câu truy vấn SQL dựa trên ngữ cảnh đã lắp ráp.
- Kiểm tra và Sửa lỗi: Kiểm tra cú pháp, ngữ nghĩa, và an toàn. Áp dụng cơ chế tự sửa lỗi nếu cần.
- Thực thi và Trả Kết quả: Thực thi SQL an toàn trên cơ sở dữ liệu và định dạng kết quả trả về cho người dùng.



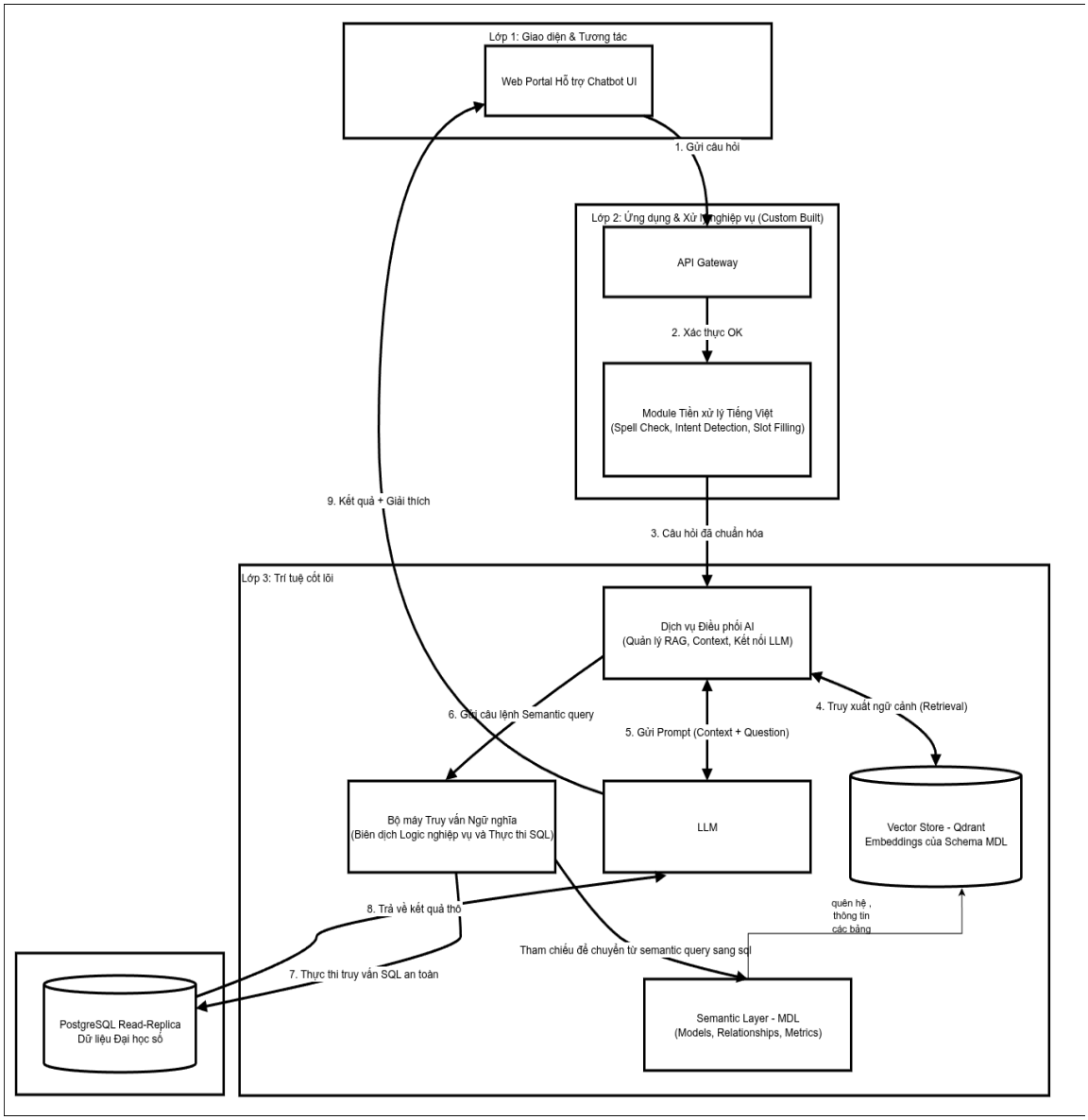
Hình 2.2: Thiết kết cấp cao của hệ thống Text-to-SQL

2.6. Thiết kế kiến trúc hệ thống

2.6.1. Kiến trúc tổng thể của hệ thống

Hệ thống được thiết kế dựa trên kiến trúc Retrieval-Augmented Generation (RAG) chuyên biệt cho Text-to-SQL. Khác với các mô hình Chatbot thông thường chỉ dựa vào dữ liệu huấn luyện, kiến trúc này sử dụng WrenAI Engine làm trung tâm điều phối, tách biệt hoàn toàn giữa lớp ngôn ngữ tiếng Việt và lớp dữ liệu vật lý.

Các nghiên cứu gần đây về bài toán chuyển đổi ngôn ngữ tiếng Việt sang SQL dựa trên mô hình ngôn ngữ lớn cho thấy kiến trúc truy xuất tăng cường sinh (Retrieval-Augmented Generation – RAG) đang trở thành hướng tiếp cận nổi trội. Kiến trúc này khắc phục các hạn chế của những mô hình thuần túy dựa trên tri thức đã được huấn luyện bằng cách bổ sung cơ chế truy xuất động thông tin từ nguồn dữ liệu bên ngoài. Theo nguyên lý hoạt động, thay vì chỉ dựa vào tri thức nội tại của mô hình ngôn ngữ lớn, hệ thống sẽ truy xuất các thông tin liên quan từ kho tri thức để xây dựng ngữ cảnh, sau đó cung cấp ngữ cảnh này cho mô hình nhằm sinh ra câu truy vấn SQL có độ chính xác và mức độ phù hợp cao hơn.



Hình 2.3: Kiến trúc tổng thể của hệ thống hệ thống

Các báo cáo của Google Cloud (2024) và AtScale (2024) đều nhấn mạnh rằng việc cung cấp ngữ cảnh đầy đủ cho mô hình ngôn ngữ lớn đóng vai trò then chốt trong việc nâng cao độ chính xác. Cách tiếp cận này cho phép hệ thống xử lý hiệu quả các cơ sở dữ liệu quy mô lớn, với hàng trăm bảng, mà không cần tiến hành tinh chỉnh mô hình theo từng miền ứng dụng.

2.6.2. Các thành phần cốt lõi của kiến trúc

Dựa trên phân tích từ các công cụ Text-to-SQL hiện đại, một công cụ bao gồm sáu thành phần chính:

Tầng hiểu ngôn ngữ tiếng Việt: Tầng này phân tích làm rõ yêu cầu của người dùng từ câu hỏi. Các chức năng chính bao gồm phân loại ý định truy vấn, nhận diện các thực thể như tên bảng và giá trị cụ thể, phân tích cấu trúc ngữ nghĩa để xác định điều kiện lọc và phép toán. Với mô hình ngôn ngữ lớn, các tác vụ này có thể được thực hiện bằng học vài mẫu hoặc học không mẫu, giảm đáng kể nhu cầu dữ liệu huấn luyện.

Tầng hiểu cấu trúc cơ sở dữ liệu: Đây là thành phần chịu trách nhiệm ánh xạ giữa ngôn ngữ tiếng Việt và cấu trúc cơ sở dữ liệu. Quá trình liên kết cấu trúc ánh xạ các từ trong câu hỏi với các thành phần như bảng, cột, giá trị. Các nghiên cứu cho thấy việc thiết lập liên kết cấu trúc một cách chính xác là yếu tố quan trọng quyết định chất lượng của câu truy vấn SQL được sinh ra. Các phương pháp thường được sử dụng bao gồm so khớp trực tiếp, so khớp mờ dựa trên khoảng cách chỉnh sửa, và tìm kiếm tương đồng theo vector nhằm phát hiện quan hệ ngữ nghĩa giữa các thực thể.

Tầng ngữ nghĩa nghiệp vụ: Tầng ngữ nghĩa là một lớp trừu tượng nằm giữa câu hỏi ngôn ngữ tiếng Việt và cấu trúc cơ sở dữ liệu vật lý. Theo nghiên cứu của AtScale (2024) và Denodo (2024), tầng ngữ nghĩa có thể cải thiện độ chính xác từ 20% lên 92.5% khi được triển khai đúng cách.

Vai trò của tầng ngữ nghĩa bao gồm: mã hóa logic nghiệp vụ và các quy tắc vào định dạng có cấu trúc; thống nhất các thuật ngữ nghiệp vụ, ánh xạ từ thuật ngữ kinh doanh sang tên kỹ thuật trong cơ sở dữ liệu; quản lý mối quan hệ rõ ràng giữa các thực thể, giúp mô hình hiểu cách kết nối các bảng đúng cách; định nghĩa các chỉ số tính toán một lần và tái sử dụng, đảm bảo tính nhất quán. Tầng ngữ nghĩa giải quyết vấn đề sai lệch chỉ số và mơ hồ trong thuật ngữ nghiệp vụ.

Tầng truy xuất và lắp ráp ngữ cảnh: Thành phần này thực hiện quy trình truy xuất tăng cường, chịu trách nhiệm truy xuất thông tin liên quan từ kho tri thức và lắp ráp thành ngữ cảnh cho mô hình ngôn ngữ lớn. Quy trình bao gồm: chuyển câu hỏi và các thành phần cấu trúc thành vector; sử dụng cơ sở dữ liệu vector để tìm các thành

phần liên quan nhất; xếp hạng lại để cải thiện chất lượng kết quả; ghép nối thông tin đã truy xuất, ví dụ mẫu và hướng dẫn thành một câu nhắc toàn diện.

Tầng sinh SQL: Tầng này sử dụng mô hình ngôn ngữ lớn để sinh câu truy vấn SQL dựa trên ngữ cảnh đã được chuẩn bị. Các phương pháp chính bao gồm: học trong ngữ cảnh bằng cách cung cấp ví dụ mẫu trong câu nhắc; suy luận chuỗi tư duy yêu cầu mô hình giải thích quá trình từng bước trước khi sinh SQL; tinh chỉnh lặp sử dụng vòng lặp tự sửa lỗi, nếu SQL sinh ra có lỗi thì thông báo lỗi được gửi lại cho mô hình để tinh chỉnh.

Tầng kiểm tra và thực thi: Tầng này đảm bảo SQL được sinh ra là an toàn và chính xác trước khi thực thi. Các chức năng bao gồm: kiểm tra cú pháp SQL; xác minh các bảng và cột được tham chiếu tồn tại; đảm bảo SQL không chứa các thao tác nguy hiểm như xóa hoặc sửa dữ liệu; thử thực thi để xác minh tính khả thi; áp dụng các chính sách bảo mật dựa trên quyền người dùng.

2.6.3. Mô hình kiến trúc vi dịch vụ

Các hệ thống sẵn sàng đưa vào sản xuất thường áp dụng kiến trúc vi dịch vụ để đảm bảo khả năng mở rộng, dễ bảo trì và tính mô-đun. Kiến trúc điển hình bao gồm ba tầng chính:

Tầng giao diện: Cung cấp giao diện người dùng cho phép nhập câu hỏi, xem kết quả dưới dạng bảng hoặc biểu đồ, quản lý các truy vấn đã lưu và lịch sử.

Tầng ứng dụng: Điều phối toàn bộ quy trình từ hiểu câu hỏi, truy xuất, sinh SQL, đến kiểm tra. Tầng này thường được triển khai như một dịch vụ không lưu trạng thái có thể mở rộng theo chiều ngang.

Tầng dữ liệu: Bao gồm cơ sở dữ liệu vector cho tìm kiếm ngữ nghĩa, kho lưu trữ siêu dữ liệu cho thông tin cấu trúc, và các kết nối đến các cơ sở dữ liệu đích.

2.6.4. Luồng xử lý từ đầu đến cuối

Dựa trên phân tích kiến trúc của các hệ thống hiện đại, luồng xử lý một câu hỏi điển hình diễn ra qua sáu giai đoạn:

Giai đoạn 1 - Hiểu câu hỏi: Phân tích câu hỏi để xác định ý định, các thực thể và các ràng buộc.

Giai đoạn 2 - Truy xuất cấu trúc: Sử dụng tìm kiếm ngữ nghĩa để truy xuất các thành phần cấu trúc liên quan từ kho tri thức.

Giai đoạn 3 - Lắp ráp ngữ cảnh: Kết hợp cấu trúc đã truy xuất, ví dụ mẫu, quy tắc nghiệp vụ thành câu nhắc toàn diện.

Giai đoạn 4 - Sinh SQL: Mô hình ngôn ngữ lớn sinh câu truy vấn SQL dựa trên ngữ cảnh đã lắp ráp.

Giai đoạn 5 - Kiểm tra và sửa lỗi: Kiểm tra cú pháp và ngữ nghĩa, áp dụng tự sửa lỗi nếu có vấn đề.

Giai đoạn 6 - Thực thi và trả kết quả: Thực thi SQL trên cơ sở dữ liệu, định dạng và gửi kết quả cho người dùng.

Nghiên cứu chỉ ra rằng việc thêm một giai đoạn riêng cho việc chọn bảng khi xử lý với cơ sở dữ liệu lớn (hàng trăm bảng) là rất quan trọng cho các hệ thống thực tế.

2.6.5. Vai trò của cơ sở dữ liệu vector

Cơ sở dữ liệu vector (vector database) là thành phần then chốt trong kiến trúc RAG (Retrieval-Augmented Generation), đóng vai trò lưu trữ và tìm kiếm thông tin dựa trên độ tương đồng ngữ nghĩa thay vì khớp từ khóa chính xác như cơ sở dữ liệu truyền thống. Trong hệ thống chuyển đổi ngôn ngữ tiếng Việt thành SQL, vector database lưu trữ các embedding (vector nhúng) của schema, ví dụ câu hỏi-SQL, tài liệu và metadata dưới dạng vector số học trong không gian nhiều chiều, cho phép tìm kiếm nhanh các thông tin liên quan nhất với câu hỏi người dùng.

Quy trình hoạt động của vector database trong hệ thống bao gồm ba giai đoạn chính. Giai đoạn thứ nhất là embedding và indexing. Khi xây dựng hệ thống, tất cả các thông tin quan trọng như schema của bảng (tên bảng, tên cột, kiểu dữ liệu, mô tả), các câu hỏi mẫu kèm SQL tương ứng, documentation của semantic layer, và các ví dụ về logic nghiệp vụ đặc thù được chuyển đổi thành vector bằng mô hình

embedding. Các vector này sau đó được lưu trữ trong vector database và được index để tối ưu tốc độ truy vấn.

Giai đoạn thứ hai là semantic search (tìm kiếm ngữ nghĩa). Khi người dùng nhập câu hỏi, hệ thống chuyển câu hỏi thành vector bằng cùng mô hình embedding đã dùng ở giai đoạn 1. Sau đó, vector database thực hiện similarity search để tìm các vector gần nhất với vector câu hỏi. Độ gần được đo bằng cosine similarity hoặc euclidean distance. Vector database thường trả về top-k kết quả có điểm tương đồng cao nhất (k thường từ 3-10).

Giai đoạn thứ ba là context assembly (lắp ráp ngữ cảnh). Các kết quả từ vector database được tổng hợp thành context để đưa vào LLM. Context này bao gồm: (1) Schema của các bảng liên quan ; (2) Các ví dụ câu hỏi-SQL tương tự - giúp LLM học được pattern và cú pháp SQL đúng; (3) Documentation của semantic layer - giải thích các metrics, relationships đã định nghĩa; (4) Business rules - các quy tắc đặc thù như cách tính GPA, điều kiện học phần. Context được lắp ráp một cách có cấu trúc để LLM dễ hiểu và sử dụng.

Các công nghệ vector database phổ biến hiện nay bao gồm: Pinecone - dịch vụ cloud managed, dễ sử dụng nhưng có chi phí; Weaviate - mã nguồn mở, hỗ trợ nhiều mô hình embedding, có thể self-host; Qdrant - hiệu năng cao, viết bằng Rust, phù hợp production; Chroma - lightweight, dễ tích hợp, phù hợp prototype; Milvus - scalable, hỗ trợ billions vectors, phù hợp enterprise. Trong WrenAI, hệ thống tích hợp sẵn vector database và tự động xử lý việc embedding, indexing và retrieval.

2.6.6. Tích hợp mô hình ngôn ngữ lớn

Việc lựa chọn LLM phụ thuộc vào các tiêu chí: khả năng sinh code SQL, hỗ trợ ngôn ngữ địa phương, kích thước context window, độ trễ phản hồi, chi phí sử dụng và khả năng tùy chỉnh. Các LLM thế hệ mới với khả năng hiểu ngữ cảnh tốt và context window lớn (hàng trăm nghìn tokens) cho phép cung cấp nhiều thông tin về schema, relationships và business logic trong một lần gọi API.

Quy trình tích hợp LLM bao gồm ba giai đoạn. Giai đoạn đầu tiên là thiết lập kết nối API, bao gồm cấu hình authentication, thiết lập các tham số điều khiển như

temperature (độ ngẫu nhiên trong sinh text), giới hạn độ dài, top_p (sampling strategy) và tránh lặp lại. Giai đoạn thứ hai là xây dựng cơ chế xử lý lỗi với retry logic, timeout handling và fallback mechanism. Giai đoạn thứ ba là parsing và validation output, bao gồm extract SQL từ response, validate cú pháp và kiểm tra an toàn trước khi thực thi.

2.6.7. Cơ chế xử lý lỗi, tự sửa lỗi và bảo mật hệ thống

Hệ thống production đòi hỏi cơ chế xử lý lỗi tự động và bảo mật nhiều tầng để đảm bảo độ tin cậy và an toàn dữ liệu. Hai khía cạnh này cần được thiết kế tích hợp từ đầu trong kiến trúc hệ thống.

Quy trình xử lý lỗi được triển khai theo ba giai đoạn. Giai đoạn validation trước khi thực thi bao gồm: syntax validation sử dụng SQL parser để phát hiện lỗi cú pháp, security validation kiểm tra các lệnh nguy hiểm, schema validation xác minh tên bảng và cột có tồn tại, và logic validation kiểm tra tính hợp lý của điều kiện và JOIN. SQL không pass validation sẽ bị từ chối và trigger regeneration từ LLM.

Giai đoạn error detection trong quá trình thực thi capture các loại lỗi từ database engine: syntax errors, runtime errors (division by zero, type mismatch, invalid date format), logic errors (query chạy được nhưng kết quả không đúng), và timeout errors (query quá chậm). Mỗi loại lỗi được phân loại và xử lý theo chiến lược khác nhau, với error message chi tiết được log để phân tích và cải thiện hệ thống.

Giai đoạn self-correction với LLM sử dụng feedback loop để tự động sửa lỗi. Error message từ database được đưa vào prompt mới cùng với SQL bị lỗi và instruction để fix. Hệ thống cho phép số lần retry giới hạn để tránh vòng lặp vô hạn. Kỹ thuật progressive refinement đơn giản hóa query từng bước khi gặp lỗi: loại bỏ query phức tạp, giảm số bảng JOIN, đơn giản hóa điều kiện WHERE, và nếu vẫn thất bại thì phân tách thành các query nhỏ hơn để xử lý từng phần.

Bảo mật được triển khai theo mô hình defense in depth với sáu tầng bảo vệ. Tầng authentication và authorization sử dụng các giao thức chuẩn (JWT, OAuth 2.0) để xác thực người dùng và phân quyền dựa trên vai trò (RBAC). Mỗi role có tập quyền truy cập riêng biệt đối với dữ liệu và chức năng hệ thống.

Tầng query-level security tự động inject điều kiện lọc dựa trên context và quyền của người dùng. Security layer chèn các điều kiện an toàn vào mọi SQL query trước khi thực thi, đảm bảo người dùng chỉ truy cập dữ liệu được phép. Row-level security (RLS) và column-level security (CLS) được enforced thông qua các policy được định nghĩa trong semantic layer.

Tầng SQL injection prevention sử dụng parameterized queries và prepared statements để ngăn chặn SQL injection. Mọi input từ người dùng đều được sanitize và escape các ký tự đặc biệt. SQL parser phát hiện và block các pattern injection phổ biến. LLM được instruct rõ ràng không bao giờ nhúng trực tiếp user input vào SQL mà phải sử dụng parameters.

Tầng read-only enforcement cấu hình database connection chỉ có quyền SELECT. User account không được cấp quyền INSERT, UPDATE, DELETE, DROP hay bất kỳ operation nào có thể thay đổi dữ liệu. Ngay cả khi LLM sinh ra SQL nguy hiểm, database engine sẽ từ chối thực thi do không có quyền.

Tầng sensitive data protection đánh dấu các cột nhạy cảm và tự động loại khỏi schema cung cấp cho LLM. Data masking được áp dụng cho các trường cần thiết phải hiển thị nhưng chứa thông tin nhạy cảm. Privacy-preserving techniques như differential privacy có thể được áp dụng cho các truy vấn thống kê aggregate.

Tầng audit logging ghi lại đầy đủ mọi hoạt động: timestamp, user identifier, câu hỏi gốc, SQL được sinh, execution time, số bản ghi trả về, và status (success/failure). Audit logs cho phép phát hiện hành vi bất thường, forensic analysis khi có sự cố, compliance reporting, và continuous monitoring để cải thiện hệ thống. Rate limiting và query timeout được áp dụng để chống DoS và đảm bảo tài nguyên hệ thống được sử dụng công bằng.

2.6.8. Chỉ số đánh giá hệ thống

Đánh giá hiệu quả hệ thống Text-to-SQL đòi hỏi một framework toàn diện với các chỉ số định lượng rõ ràng. Các chỉ số được phân thành bốn nhóm chính: độ chính xác SQL, hiệu năng hệ thống, độ tin cậy, và trải nghiệm người dùng. Mỗi chỉ số có công thức tính toán cụ thể và ngưỡng mục tiêu phù hợp với môi trường production.

2.6.8.1. Độ chính xác SQL

Công thức 2.1: Exact Match Accuracy (EM)

$$EX = \frac{N_{exact}}{N_{total}} * 100\%$$

Trong đó:

- N_{exact}: Số câu SQL khớp hoàn toàn về cú pháp và cấu trúc với ground truth
- N_{total}: Tổng số câu hỏi trong test set

Exact Match Accuracy (EM) đo lường tỷ lệ SQL được sinh ra khớp hoàn toàn với SQL ground truth. Hai SQL được coi là exact match khi chúng giống hệt nhau sau khi chuẩn hóa (normalization): loại bỏ khoảng trắng thừa, chuẩn hóa chữ hoa/thường của keywords, chuẩn hóa tên alias. EM là chỉ số nghiêm ngặt nhưng có hạn chế: không công nhận các SQL tương đương về mặt logic. Trong benchmark Spider (Yu et al., 2018), EM được sử dụng làm baseline metric với threshold 75%.

Công thức 2.2: Execution Accuracy (EX)

$$EX = \frac{N_{correct}}{N_{total}} * 100\%$$

Trong đó:

- $N_{correct}$: Số câu SQL cho kết quả đúng khi thực thi trên database
- N_{total}: Tổng số câu hỏi trong test set
- Kết quả đúng được xác định bởi:
    - Set matching: Output rows trùng khớp với expected results
    - Bỏ qua thứ tự rows nếu không có ORDER BY trong SQL
    - Tolerance cho số thực (floating point comparison)

Execution Accuracy (EX) là chỉ số quan trọng nhất trong thực tế vì đo lường chất lượng kết quả thực sự. Một SQL được coi là correct nếu khi thực thi trên database, nó trả về tập kết quả giống với expected result. Phương pháp so sánh: (1)

Set matching - so sánh hai tập hợp rows bất kể thứ tự, (2) Schema matching - đảm bảo các cột trong output khớp với expected schema, (3) Value matching - so sánh giá trị từng cell với tolerance cho số thực. EX cho phép nhiều cách viết SQL khác nhau miễn là đều cho kết quả đúng, do đó thường cao hơn EM 10-15%. Trong benchmark Spider, EX là official metric chính.

2.6.8.2. Hiệu năng hệ thống

Công thức 2.4: Response Time Breakdown

$$RT_{total} = T_{parse} + T_{retrieval} + T_{llm} + T_{validate} + T_{execute} + T_{format}$$

Trong đó:

- T_{parse}: Thời gian phân tích câu hỏi và intent classification
- $T_{retrieval}$: Thời gian RAG retrieval từ vector database (schema, examples)
- T_{llm}: Thời gian LLM inference để sinh SQL
- $T_{validate}$: Thời gian validate syntax, security và logic
- $T_{execute}$: Thời gian thực thi SQL trên database
- T_{format}: Thời gian format và render kết quả cho người dùng

Percentile Latency: P50, P95, P99

- P50 (median): 50% requests hoàn thành trong thời gian ≤ P50
- P95: 95% requests hoàn thành trong thời gian ≤ P95
- P99: 99% requests hoàn thành trong thời gian ≤ P99

Response Time được phân tích thành các thành phần để xác định bottleneck và tối ưu hóa. T_llm thường chiếm 60-70% tổng thời gian do LLM inference latency cao. Optimization strategies: (1) LLM caching cho queries phổ biến, (2) Prompt compression giảm input tokens, (3) Streaming response để hiển thị kết quả sớm. T_execute phụ thuộc vào độ phức tạp query và database indexing. Percentile metrics (P50, P95, P99) quan trọng hơn average vì phản ánh tail latency - trải nghiệm của người dùng chậm nhất. Production SLA thường đặt P95 < 5s và P99 < 10s.

2.7. Khảo sát và đánh giá các công cụ hiện có

Sau khi khảo sát kiến trúc tổng quan của hệ thống chuyển đổi ngôn ngữ tiếng Việt thành SQL, phần này tiến hành khảo sát và đánh giá các công cụ hiện có trên thị trường. Việc khảo sát này nhằm mục đích xác định công cụ phù hợp nhất cho việc xây dựng hệ thống tra cứu thông tin trong lĩnh vực đại học số.

2.7.1. Phân loại các công cụ

Các công cụ chuyển đổi ngôn ngữ tiếng Việt thành SQL có thể được phân loại thành hai nhóm chính dựa trên cách tiếp cận và tính năng:

Nhóm 1 - Công cụ chuyển đổi cơ bản: Các công cụ này tập trung vào việc chuyển đổi đơn giản từ câu hỏi ngôn ngữ tiếng Việt thành câu truy vấn SQL. Người dùng nhập mô tả câu hỏi và thông tin cấu trúc cơ sở dữ liệu, nhận về câu SQL được tạo ra, sau đó tự thực thi trên hệ thống của mình. Nhóm này phù hợp với các trường hợp sử dụng đơn giản, ngân sách hạn chế, và người dùng có kiến thức kỹ thuật cơ bản về SQL.

Nhóm 2 - Nền tảng trí tuệ kinh doanh tích hợp: Các nền tảng này cung cấp trải nghiệm tương tác giống chatbot, cho phép người dùng hội thoại với cơ sở dữ liệu. Công cụ tự động hiểu cấu trúc dữ liệu, truy xuất và giải thích kết quả cho người dùng. Nhiều công cụ còn hỗ trợ phân tích dữ liệu nâng cao, trực quan hóa và tạo dashboard. Nhóm này phù hợp với doanh nghiệp cần giải pháp toàn diện, quản trị dữ liệu chặt chẽ, và hỗ trợ nhiều người dùng phi kỹ thuật.

2.6.2. Tiêu chí đánh giá

Để đánh giá và lựa chọn công cụ phù hợp, đề tài xác định các tiêu chí quan trọng sau:

- Hỗ trợ tầng ngữ nghĩa nghiệp vụ: Khả năng mã hóa logic nghiệp vụ, định nghĩa chỉ số và mối quan hệ giữa các thực thể.
- Kiến trúc hệ thống: Mức độ phù hợp với kiến trúc đã thiết kế trong phần 2.5, bao gồm các thành phần cốt lõi.
- Khả năng mở rộng: Hỗ trợ nhiều nguồn dữ liệu, khả năng tích hợp API, triển khai phân tán.

- Tính mã nguồn mở: Khả năng tùy chỉnh, minh bạch về thuật toán, không phụ thuộc vào nhà cung cấp.
- Bảo mật và quản trị: Kiểm soát truy cập theo vai trò, kiểm soát cấp hàng/cột, nhật ký kiểm tra.
- Hỗ trợ ngôn ngữ Việt: Khả năng xử lý câu hỏi tiếng Việt, hỗ trợ cấu trúc dữ liệu có thuật ngữ tiếng Việt.
- Chi phí triển khai: Chi phí giấy phép, chi phí sử dụng API mô hình ngôn ngữ lớn, chi phí vận hành.
- Cộng đồng và tài liệu: Sự hỗ trợ từ cộng đồng, tài liệu hướng dẫn, độ phổ biến.

2.7.3. Kết quả khảo sát các công cụ

2.7.3.1. WrenAI

Wren AI là một nền tảng mã nguồn mở chuyên biệt cho Text-to-SQL, được phát triển bởi Canner với mục tiêu cung cấp giải pháp toàn diện cho việc chuyển đổi ngôn ngữ tự nhiên thành câu truy vấn SQL trong môi trường doanh nghiệp [15]. Khác với các công cụ Text-to-SQL truyền thống chỉ tập trung vào việc sinh ra câu SQL đơn thuần, Wren AI được xây dựng theo triết lý "Semantic-First", ưu tiên việc hiểu ngữ nghĩa nghiệp vụ trước khi thực hiện chuyển đổi kỹ thuật.

Điểm đặc trưng của Wren AI nằm ở khả năng tích hợp ba yếu tố then chốt: (1) Semantic Layer - lớp ngữ nghĩa được định nghĩa thông qua Modeling Definition Language (MDL), (2) RAG Pipeline - cơ chế truy xuất tăng cường thông tin, và (3) Multi-LLM Support - hỗ trợ đa mô hình ngôn ngữ lớn. Kiến trúc này cho phép hệ thống không chỉ sinh ra SQL chính xác về mặt cú pháp mà còn đảm bảo tính đúng đắn về mặt nghiệp vụ.

Nghiên cứu thực nghiệm của Curran và Mariani (2024) tại AtScale đã chứng minh vai trò then chốt của semantic layer trong Text-to-SQL [16]. Kết quả cho thấy việc tích hợp semantic layer có thể nâng độ chính xác từ 20% lên 92.5% trên dataset TPC-DS với 40 câu hỏi phức tạp, và đạt 100% độ chính xác trên các câu hỏi đơn giản

khi có semantic layer, so với chỉ 60% khi không có [16]. Kết quả này minh chứng cho tầm quan trọng của việc mã hóa logic nghiệp vụ một cách có hệ thống, điều mà Wren AI thực hiện thông qua cơ chế MDL.

Kiến trúc của Wren AI được tổ chức theo mô hình ba tầng, mỗi tầng đảm nhận vai trò cụ thể trong quy trình xử lý từ câu hỏi ngôn ngữ tự nhiên đến câu truy vấn SQL.

Presentation Layer (Tầng Giao diện): Tầng này bao gồm Wren UI - giao diện web được xây dựng bằng Next.js, cung cấp trải nghiệm tương tác dạng chatbot cho người dùng cuối. Giao diện được thiết kế theo nguyên tắc Progressive Disclosure, chỉ hiển thị thông tin phức tạp khi người dùng cần thiết. Người dùng có thể nhập câu hỏi bằng ngôn ngữ tự nhiên, xem kết quả dưới dạng bảng hoặc biểu đồ, và quản lý lịch sử truy vấn. Đặc biệt, giao diện hỗ trợ chế độ SQL Preview cho phép người dùng xem và chỉnh sửa câu truy vấn trước khi thực thi.

Application Layer (Tầng Ứng dụng): Wren AI Service đóng vai trò orchestrator điều phối toàn bộ quy trình xử lý. Dịch vụ này được triển khai bằng Python với framework FastAPI, đảm bảo hiệu năng cao và khả năng mở rộng tốt. Các chức năng chính bao gồm: Question Understanding để phân tích câu hỏi và xác định ý định; Schema Retrieval sử dụng semantic search trong vector database để tìm các thành phần schema liên quan; Context Assembly để lắp ráp thông tin schema, ví dụ mẫu và business rules thành prompt toàn diện; LLM Orchestration để gọi API của mô hình ngôn ngữ lớn với cơ chế retry và fallback; và SQL Validation để kiểm tra cú pháp và tính an toàn của SQL được sinh ra.

Tầng này tích hợp Self-Correction Loop - một cơ chế quan trọng giúp hệ thống tự động sửa lỗi. Khi SQL được sinh ra gặp lỗi khi thực thi, thông báo lỗi được đưa vào prompt mới cùng với câu SQL lỗi, yêu cầu LLM phân tích và sửa chữa. Cơ chế này cho phép tối đa ba lần thử lại, giúp cải thiện đáng kể tỷ lệ thành công của hệ thống.

Data & Intelligence Layer (Tầng Dữ liệu và Trí tuệ): Tầng này bao gồm các thành phần cốt lõi cung cấp tri thức và năng lực xử lý. Wren Engine là semantic query

engine được xây dựng trên Apache DataFusion, chịu trách nhiệm chuyển đổi các định nghĩa MDL thành các thao tác truy vấn thực tế [15]. Engine này không chỉ thực thi SQL mà còn áp dụng các business rules, metric calculations và security policies đã được định nghĩa trong semantic layer. Wren Engine hỗ trợ nhiều nguồn dữ liệu thông qua connector architecture, bao gồm PostgreSQL, MySQL, BigQuery, Snowflake và nhiều hệ quản trị cơ sở dữ liệu khác.

Qdrant Vector Database lưu trữ embeddings của schema metadata, ví dụ câu hỏi-SQL mẫu và documentation. Việc sử dụng vector search cho phép hệ thống tìm kiếm thông tin dựa trên độ tương đồng ngữ nghĩa thay vì chỉ khớp từ khóa chính xác. Điều này đặc biệt quan trọng khi xử lý tiếng Việt với nhiều cách diễn đạt khác nhau cho cùng một ý nghĩa. PostgreSQL được sử dụng làm metadata store, lưu trữ các định nghĩa MDL, configuration, user queries history và audit logs.

Semantic Layer đóng vai trò như một "lớp dịch" giữa ngôn ngữ tự nhiên của người dùng và cấu trúc kỹ thuật của cơ sở dữ liệu [16]. Trong Wren AI, semantic layer được định nghĩa thông qua MDL - một ngôn ngữ khai báo (declarative) được thiết kế đặc biệt cho Text-to-SQL. Nghiên cứu của Curran và Mariani (2024) chỉ ra rằng semantic layer cho phép "định nghĩa chính xác các thuật ngữ nghiệp vụ, đảm bảo rằng khi người dùng truy vấn dữ liệu bằng các thuật ngữ quen thuộc, hệ thống biết chính xác dữ liệu nào cần truy xuất và cách xử lý" [16].

MDL bao gồm bốn thành phần chính. Models đại diện cho các thực thể nghiệp vụ, có thể mapping với một hoặc nhiều bảng vật lý. Mỗi model bao gồm các columns với metadata chi tiết như tên nghiệp vụ bằng tiếng Việt, kiểu dữ liệu, mô tả, ví dụ giá trị và các ràng buộc. Relationships định nghĩa cách các models liên kết với nhau, không chỉ mô tả join condition mà còn giải thích ý nghĩa nghiệp vụ của mối quan hệ. Metrics là các chỉ số tính toán được định nghĩa một lần và tái sử dụng nhiều lần, đảm bảo tính nhất quán trong cách tính toán các KPIs quan trọng. Views là các derived models được tạo từ việc kết hợp và biến đổi các models khác, cho phép đóng gói các logic truy vấn phức tạp thành các thực thể đơn giản hơn.

Cơ chế này đảm bảo rằng khi một thuật ngữ nghiệp vụ được định nghĩa trong MDL, mọi truy vấn liên quan đến thuật ngữ đó đều sử dụng cùng một logic tính toán. Điều này đặc biệt quan trọng trong môi trường đại học số, nơi các chỉ số như GPA, tỷ lệ sinh viên đạt chuẩn đầu ra, hay điểm trung bình học phần có công thức tính toán đặc thù và cần được áp dụng nhất quán.

Wren AI được thiết kế để hỗ trợ nhiều mô hình ngôn ngữ lớn khác nhau, cho phép người dùng lựa chọn model phù hợp với yêu cầu về độ chính xác, tốc độ và chi phí. Hiện tại, Wren AI hỗ trợ OpenAI GPT-4 và GPT-3.5, Google Gemini Pro, Anthropic Claude, và các local models thông qua Ollama hoặc vLLM.

Cơ chế chuyển đổi giữa các models được thực hiện thông qua một abstraction layer, cho phép thay đổi LLM chỉ bằng cách cập nhật configuration mà không cần sửa code. Thiết kế này đặc biệt hữu ích khi cần fallback sang model khác khi model chính gặp sự cố, muốn A/B testing để so sánh hiệu quả của các models, hoặc cần tối ưu chi phí bằng cách sử dụng model nhẹ cho câu hỏi đơn giản và model mạnh cho câu hỏi phức tạp.

2.7.3.3. Chat2DB

Chat2DB là SQL client đa năng tích hợp khả năng AI ngay từ đầu, hỗ trợ nhiều loại cơ sở dữ liệu.

Đặc điểm kỹ thuật:

- Giao diện chuyên nghiệp cho quản trị cơ sở dữ liệu.
- Hỗ trợ MySQL, PostgreSQL, SQL Server, TiDB và các cache phổ biến.
- Không có tầng ngữ nghĩa chính thức.

Phù hợp cho kỹ sư dữ liệu, quản trị viên cơ sở dữ liệu cần công cụ hỗ trợ AI trong công việc hàng ngày.

2.7.3.4. Text2SQL.AI

Text2SQL.AI là dịch vụ thương mại chuyên về chuyển đổi cơ bản từ văn bản sang SQL.

Đặc điểm kỹ thuật:

- Giao diện đơn giản, dễ sử dụng với giá cả hợp lý.
- Hỗ trợ nhiều loại cơ sở dữ liệu, bao gồm cả sinh formula và regex.
- Dịch vụ đóng, không có tầng ngữ nghĩa.

Phù hợp cho các tác vụ đơn giản. Không phù hợp cho hệ thống phức tạp cần logic nghiệp vụ đặc thù hoặc yêu cầu tùy chỉnh cao.

2.8. Kết luận chương 2

Chương 2 đã hoàn thành mục tiêu nghiên cứu và thiết kế nền tảng cho hệ thống chuyển đổi ngôn ngữ tiếng Việt thành SQL. Thông qua việc nghiên cứu ba khía cạnh quan trọng (chuẩn hóa schema, từ điển dữ liệu, bộ câu hỏi), thiết kế kiến trúc tổng thể dựa trên các nghiên cứu khoa học mới nhất, và đánh giá có hệ thống các công cụ hiện có, chương đã tạo ra một kế hoạch triển khai rõ ràng và khoa học.

Đặc biệt, việc xác định WrenAI là công cụ phù hợp nhất không chỉ dựa trên các tiêu chí kỹ thuật mà còn xuất phát từ sự phù hợp với kiến trúc đã thiết kế. WrenAI với tầng ngữ nghĩa MDL, kiến trúc RAG hoàn chỉnh, và tính mã nguồn mở đáp ứng đầy đủ các yêu cầu của một hệ thống production-ready cho lĩnh vực đại học số.

Kết quả của Chương 2 tạo tiền đề vững chắc cho Chương 3 trong việc xây dựng hệ thống thực tế. Với nền tảng lý thuyết rõ ràng, kiến trúc được thiết kế kỹ lưỡng và công cụ đã được lựa chọn có cơ sở khoa học, quá trình triển khai trong Chương 3 sẽ diễn ra một cách có hệ thống và hiệu quả. Cuối cùng, hệ thống khi hoàn thành sẽ cung cấp một công cụ mạnh mẽ giúp cán bộ, giảng viên và sinh viên có thể truy xuất thông tin từ cơ sở dữ liệu đại học số một cách dễ dàng và chính xác bằng ngôn ngữ tiếng Việt tiếng Việt, góp phần nâng cao hiệu quả quản lý và ra quyết định dựa trên dữ liệu.

CHƯƠNG 3: XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG

3.1. Môi trường và công nghệ triển khai

Hệ thống được thiết kế theo kiến trúc microservices với các thành phần độc lập, dễ dàng mở rộng và bảo trì. WrenAI cung cấp nền tảng vững chắc với các tính năng: semantic layer để hiểu ngữ cảnh nghiệp vụ, RAG (Retrieval-Augmented Generation) để cải thiện độ chính xác SQL, và hỗ trợ đa mô hình LLM.

Sau khi lựa chọn Wren AI làm nền tảng cho hệ thống (như đã phân tích trong Mục 2.7.3.1), phần này trình bày chi tiết quy trình cài đặt, cấu hình và tích hợp Wren AI vào môi trường đại học số. Quá trình triển khai được thực hiện theo phương pháp có hệ thống, đảm bảo tính ổn định và khả năng bảo trì trong dài hạn.

Wren AI được thiết kế để triển khai dễ dàng thông qua Docker containerization, đảm bảo tính nhất quán giữa các môi trường development, testing và production. Quy trình cài đặt được thực hiện theo các bước chuẩn hóa.

Yêu cầu hệ thống: Máy chủ triển khai cần đáp ứng cấu hình tối thiểu gồm 4 CPU cores, 8GB RAM, 50GB dung lượng ổ cứng và hệ điều hành Ubuntu 24.04 LTS. Để đảm bảo hiệu năng tối ưu, cấu hình khuyến nghị là 8 CPU cores, 16GB RAM và 100GB SSD. Các dependencies chính bao gồm Docker Engine phiên bản 24.0 trở lên, Docker Compose phiên bản 2.0 trở lên, và kết nối Internet ổn định để pull images và gọi LLM APIs.

Hệ thống được xây dựng dựa trên nền tảng WrenAI - một công cụ Text-to-SQL mã nguồn mở. Bảng 3.1 tổng hợp các công nghệ và công cụ chính được sử dụng trong quá trình phát triển và triển khai hệ thống.

Bảng 3.1: Tổng hợp môi trường và công nghệ triển khai

Phân loại	Công cụ	Phiên bản	Mục đích sử dụng
Môi trường phát triển	Docker + Docker Compose	Latest	Container hóa ứng dụng, đảm bảo tính nhất quán môi trường

Phân loại	Công cụ	Phiên bản	Mục đích sử dụng
	Ubuntu Linux	24.04 LTS	Hệ điều hành triển khai
	Visual Studio Code	Latest	IDE phát triển
Backend & Xử lý dữ liệu	Python	3.11+	Ngôn ngữ lập trình chính, xử lý AI Service
	FastAPI	0.104+	Web framework, xây dựng RESTful API
	PostgreSQL		Lưu trữ metadata, schema, từ điển
	Qdrant	1.7+	Vector database, semantic search
	Wren Engine	0.8+	Semantic engine, mapping business terms
Tích hợp LLM	OpenAI API (GPT-4)	GPT-4-turbo	Mô hình ngôn ngữ lớn chính cho text-to-SQL
	LangChain	0.1+	Prompt management và LLM orchestration
	text-embedding-ada-002	v2	Model embedding cho semantic search
Công cụ hỗ trợ	Git & GitHub	-	Quản lý mã nguồn
	DBeaver	Latest	Quản lý cơ sở dữ liệu
	Postman	Latest	Kiểm thử API

MDL là thành phần cốt lõi của semantic layer, định nghĩa cách hệ thống hiểu và xử lý dữ liệu. Việc xây dựng MDL chất lượng cao là yếu tố quyết định độ chính xác của hệ thống Text-to-SQL.

Định nghĩa Models: Models trong MDL ánh xạ các thực thể nghiệp vụ với bảng trong database. Mỗi model được định nghĩa với các thành phần: tên nghiệp vụ bằng tiếng Việt không dấu; chứa câu SQL định nghĩa nguồn dữ liệu, có thể là SELECT đơn giản hoặc JOIN phức tạp; columns liệt kê các cột với metadata chi tiết; primaryKey xác định khóa chính; và description mô tả ý nghĩa nghiệp vụ bằng tiếng Việt.

Định nghĩa Relationships: Relationships mô tả cách các models liên kết với nhau, thay thế cho JOIN thủ công. Mỗi relationship bao gồm: name đặt tên mối quan hệ; models liệt kê các models tham gia; joinType xác định loại join (LEFT, INNER, RIGHT); condition mô tả điều kiện join; và description giải thích ý nghĩa nghiệp vụ.

Định nghĩa Metrics: Metrics là các chỉ số tính toán được định nghĩa một lần và tái sử dụng nhiều lần. Việc định nghĩa metrics trong MDL đảm bảo tính nhất quán trong cách tính toán các KPIs quan trọng.

Định nghĩa Views: Views đóng gói logic truy vấn phức tạp thành các thực thể đơn giản hơn.

RAG pipeline trong Wren AI sử dụng vector database để lưu trữ và truy xuất thông tin liên quan, cải thiện đáng kể độ chính xác.

Cấu hình Qdrant Vector Database: Qdrant được sử dụng để lưu trữ embeddings của schema metadata, examples và documentation. Quá trình setup bao gồm việc tạo collection có dimension 1536 và distance metric là Cosine. Collection lưu trữ các cặp câu hỏi-SQL mẫu với cùng cấu hình. Collection chứa documentation và business rules.

Tạo Embeddings cho Schema: Khi MDL được cập nhật, Wren AI tự động tạo embeddings cho mỗi model, column và relationship. Quá trình này sử dụng OpenAI

text-embedding-ada-002 hoặc các embedding models khác tùy cấu hình. Mỗi schema element được chuyển thành một document với format: tên tiếng Việt, tên kỹ thuật, mô tả chi tiết, kiểu dữ liệu, và ví dụ giá trị. Document này được embed và lưu vào Qdrant với metadata bao gồm model_name, column_name, data_type và relationships.

Xây dựng bộ Examples: Few-shot examples đóng vai trò quan trọng trong việc hướng dẫn LLM sinh SQL đúng pattern [6]. Mỗi example bao gồm: câu hỏi tiếng Việt phản ánh cách người dùng thực tế đặt câu hỏi; câu SQL tương ứng đã được verify; difficulty level (easy, medium, hard); và tags mô tả các đặc điểm (single_table, join, aggregation, subquery).

Semantic Search trong RAG: Khi người dùng đặt câu hỏi, hệ thống thực hiện các bước sau. Question embedding: Câu hỏi được chuyển thành vector bằng cùng embedding model. Schema retrieval: Tìm top-k models và columns có embedding gần nhất với câu hỏi (k=5-10). Example retrieval: Tìm top-n examples tương tự (n=3-5). Ranking: Sắp xếp lại kết quả dựa trên relevance score và business rules. Context assembly: Kết hợp schema + examples + rules thành prompt.

3.1.1.4. Tích hợp và cấu hình LLM

Wren AI hỗ trợ nhiều LLM providers, cho phép lựa chọn model phù hợp với yêu cầu về accuracy, latency và cost.

Wren AI hỗ trợ nhiều LLM providers, cho phép lựa chọn model phù hợp với yêu cầu về accuracy, latency và cost.

Cấu hình OpenAI GPT: Để sử dụng GPT-4 hoặc GPT-3.5, cần cấu hình API key trong biến môi trường.

Prompt Template Optimization: Wren AI sử dụng một prompt template được tối ưu hóa cho Text-to-SQL. Template này bao gồm system instruction định nghĩa vai trò của LLM; schema context với các models, columns và relationships liên quan; few-shot examples của các câu hỏi-SQL tương tự; business rules và constraints; và user question.

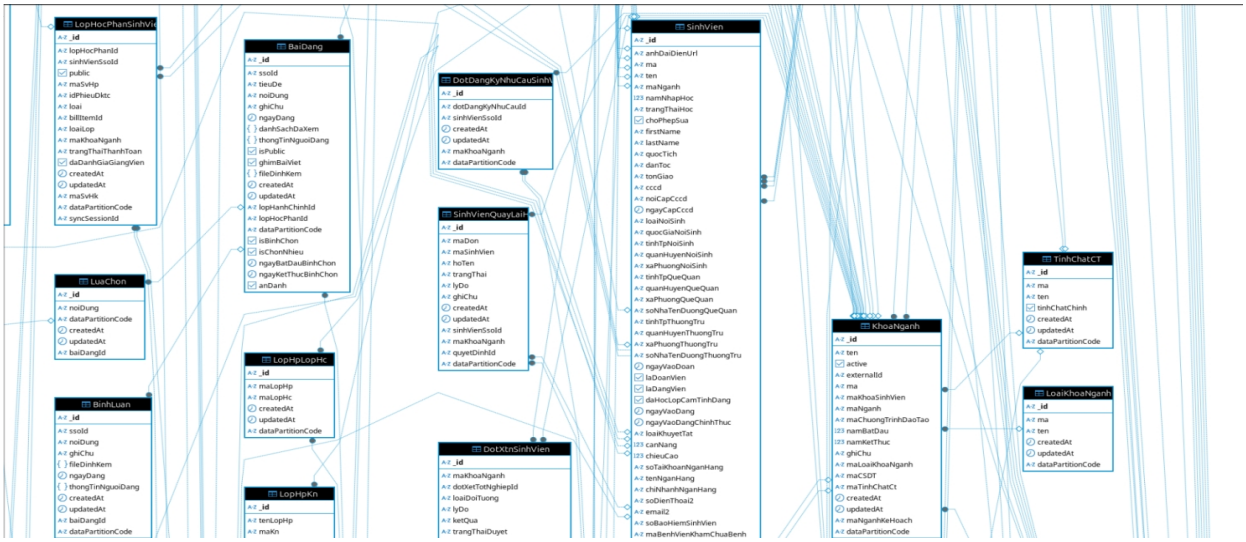
Retry và Error Handling: Wren AI triển khai retry mechanism với exponential backoff khi gọi LLM API. Cấu hình mặc định là max_retries=3, initial_delay=1s, backoff_factor=2. Khi API call thất bại, hệ thống log error và retry sau 1s, 2s, 4s. Nếu sau 3 lần vẫn fail, trả error message cho người dùng.

3.2. Xây dựng các tài nguyên nền tảng

Để hệ thống Text-to-SQL hoạt động hiệu quả, việc xây dựng các tài nguyên nền tảng đóng vai trò then chốt. Các tài nguyên này bao gồm: schema cơ sở dữ liệu được chuẩn hóa và tài liệu hóa đầy đủ, bộ từ điển ánh xạ thuật ngữ tiếng Việt với cấu trúc database, và bộ câu hỏi mẫu phục vụ few-shot learning. Phần này trình bày chi tiết quy trình xây dựng từng loại tài nguyên, từ phân tích và mô tả schema, xây dựng metadata, đến việc tạo các ví dụ mẫu cho mô hình AI.

3.2.1. Xây dựng và chuẩn hóa schema cơ sở dữ liệu

Hệ thống sử dụng cơ sở dữ liệu PostgreSQL 15 để quản lý toàn bộ dữ liệu đào tạo của Học viện Công nghệ Bưu chính Viễn thông. CSDL được thiết kế theo mô hình quan hệ chuẩn hóa, tuân thủ các quy tắc để đảm bảo tính toàn vẹn và tránh dư thừa dữ liệu. Cấu trúc cơ sở dữ liệu bao gồm hơn 200 bảng chính hơn 3000 thuộc tính của các bảng, được tổ chức thành các nhóm chức năng: quản lý tổ chức (khóa sinh viên, ngành, chương trình đào tạo), quản lý sinh viên, quản lý học vụ (học phần, lớp học phần, đăng ký học phần, điểm).



Hình 3.1: Lược đồ ERD cơ sở dữ liệu quản lý đào tạo

CSDL hiện đang lưu trữ dữ liệu của gần 5.000 sinh viên đang theo học, 70 ngành đào tạo, khoảng 1000 học phần trên 42 chương trình đào tạo. Dữ liệu ghi nhận thông tin điểm tích lũy qua từng kỳ và cả khóa của sinh viên.

Để hỗ trợ mô hình AI hiểu rõ ngữ cảnh nghiệp vụ, mỗi bảng và cột trong database được bổ sung metadata. Metadata không chỉ giải thích ý nghĩa kỹ thuật của từng trường mà còn cung cấp thông tin nghiệp vụ, ví dụ về giá trị, và các ràng buộc logic. Các metadata này sau đó được trích xuất và đưa vào context của LLM để cải thiện độ chính xác trong việc sinh SQL.

Để mô hình LLM có thể hiểu và sử dụng schema hiệu quả, cần chuyển đổi cấu trúc database sang định dạng phù hợp với context window của model. WrenAI sử dụng cơ chế Modeling Definition Language (MDL) để tổ chức schema thành các Models, Views, Relationships, và Metrics. Thông tin này được serialize thành JSON và lưu trong PostgreSQL, đồng thời được embedding và lưu vào Qdrant vector database để phục vụ semantic search.

Kết quả của quá trình này là một semantic layer hoàn chỉnh, cho phép LLM không chỉ biết tên bảng và cột mà còn hiểu được ý nghĩa nghiệp vụ, mối quan hệ dữ liệu, và context sử dụng. Khi người dùng đặt câu hỏi, hệ thống sẽ sử dụng semantic search trong Qdrant để tìm các tables/columns liên quan nhất, sau đó đưa thông tin này vào prompt gửi tới GPT-4 để sinh SQL chính xác.

3.2.2. Xây dựng bộ từ điển dữ liệu

Bộ từ điển dữ liệu (Data Dictionary) đóng vai trò cầu nối quan trọng giữa ngôn ngữ tiếng Việt tiếng Việt mà người dùng sử dụng và cấu trúc database với tên bảng, cột thường được đặt bằng tiếng Anh hoặc không dấu. Từ điển dữ liệu cung cấp ánh xạ này một cách có hệ thống, bao gồm cả các từ đồng nghĩa và cách diễn đạt khác nhau mà người dùng có thể sử dụng.

Bộ từ điển được lưu trữ trong một bảng riêng trong PostgreSQL, cho phép tra cứu nhanh và dễ dàng cập nhật. Cấu trúc bảng được thiết kế để lưu trữ đầy đủ thông tin ánh xạ giữa thuật ngữ tiếng Việt với các thành phần database, đồng thời hỗ trợ tìm kiếm và quản lý từ đồng nghĩa.

Cấu trúc của từ điển xây dựng gồm các thành phần:

- Tên bảng: tên bảng được lưu trữ trong cơ sở dữ liệu
- Tên thuộc tính: tên column được lưu trữ trong cơ sở dữ liệu
- Kiểu dữ liệu: Kiểu dữ liệu của column trong cơ sở dữ liệu. Các giá trị của kiểu dữ liệu phụ thuộc vào cơ sở dữ liệu cung cấp.
- Mô tả: Mô tả tên bảng, tên column tương ứng dưới dạng ngôn ngữ tiếng Việt
- Ghi chú tham chiếu: Nếu giá trị của column được định nghĩa bởi người dùng sẽ mô tả vùng giá trị của dữ liệu.
- Metadata: Chứa thông tin bổ sung của column như: độ dài dữ liệu, có cho phép giá trị null, khóa chính, khóa ngoại, enum,...

Quy trình xây dựng bộ từ điển dữ liệu thực hiện qua 3 bước chính:

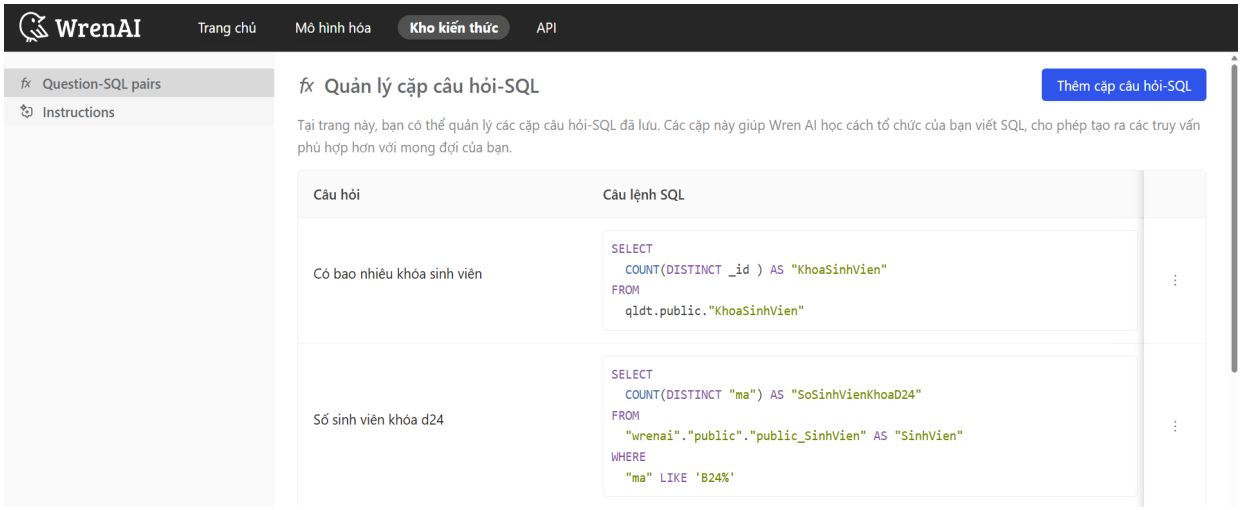
- Bước 1: Thu thập thuật ngữ từ schema: Sử dụng script Python để tự động trích xuất tên bảng và tên cột từ information_schema của PostgreSQL.
- Bước 2: Bổ sung từ tài liệu nghiệp vụ: Tham khảo các tài liệu như quy chế đào tạo, văn bản hành chính, biểu mẫu của nhà trường để thu thập các thuật ngữ chuyên ngành. Các thuật ngữ này được thêm vào từ điển với mô tả rõ ràng.
- Bước 3: Validate và kiểm tra xung đột: Sử dụng query SQL để tìm các thuật ngữ trùng lặp hoặc đa nghĩa. Xử lý bằng cách thêm context vào description hoặc dựa vào các từ kèm theo trong câu hỏi để phân biệt.

Từ điển được tích hợp vào WrenAI thông qua hai cơ chế: (1) Exact matching - tìm kiếm trực tiếp trong bảng data_dictionary sử dụng full-text search với GIN index, và (2) Semantic matching - embeddings của các terms được lưu vào Qdrant để hỗ trợ tìm kiếm ngữ nghĩa. Khi người dùng đặt câu hỏi, hệ thống sẽ kết hợp cả hai phương pháp để tìm ra các bảng và cột liên quan nhất, sau đó đưa thông tin này vào context của LLM prompt.

3.2.3. Xây dựng bộ câu hỏi mẫu

Bộ câu hỏi mẫu là tập hợp các cặp (câu hỏi tiếng Việt, SQL query tương ứng) được chuẩn bị sẵn và đưa vào context của LLM khi xử lý câu hỏi người dùng. Kỹ thuật này cho phép mô hình học cách chuyển đổi ngôn ngữ tiếng Việt thành SQL thông qua các ví dụ cụ thể. Trong hệ thống WrenAI, bộ câu hỏi mẫu được sử dụng để cải thiện độ chính xác của SQL generation, đặc biệt là với các trường hợp phức tạp hoặc đặc thù của ngữ cảnh đào tạo Việt Nam.

Bộ câu hỏi mẫu phục vụ mục đích cung cấp ví dụ cho LLM, và tạo bộ dữ liệu kiểm thử. Quy trình xây dựng bộ câu hỏi mẫu bao gồm phân tích domain để xác định các pattern truy vấn phổ biến, tạo cặp câu hỏi - SQL chuẩn với nhiều cách diễn đạt và đa dạng hóa. Để xây dựng một bộ câu hỏi mẫu, ta có thể định nghĩa sẵn các câu hỏi mẫu và câu SQL chuẩn. Khi người dùng hỏi và nhận câu trả lời sai, chúng ta có thể chỉnh sửa lại câu truy vấn cho đúng và cập nhật vào kho mẫu để làm giàu tài nguyên mẫu.



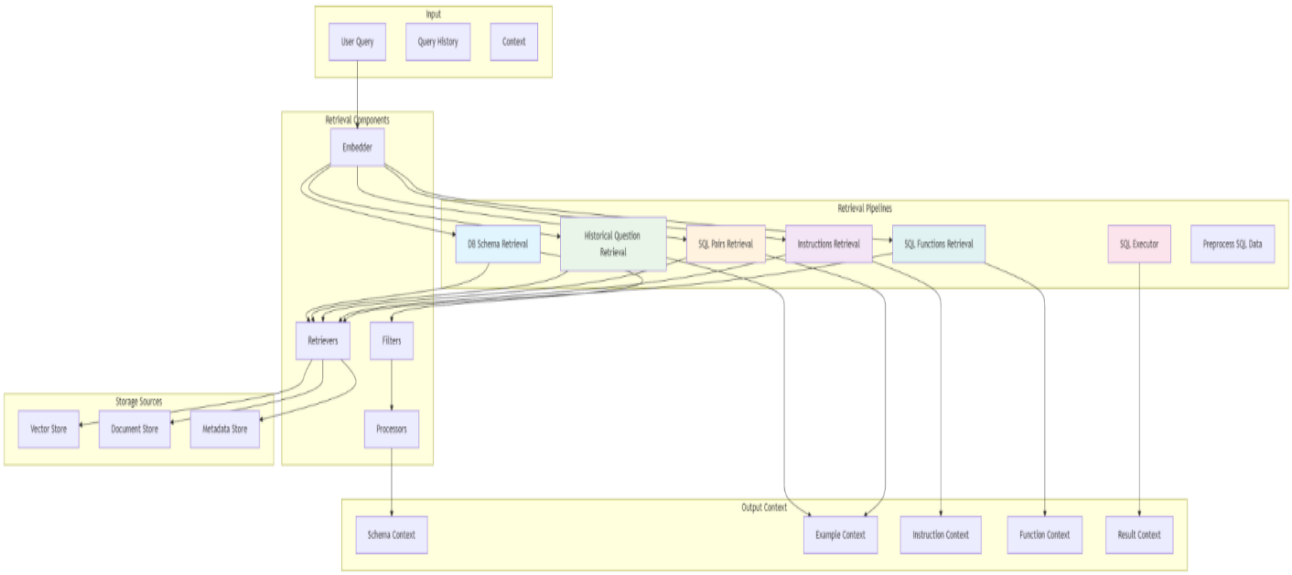
Hình 3.2: Giao diện câu hỏi SQL

3.3. Triển khai các module xử lý chính

Hệ thống được tổ chức thành các module xử lý độc lập theo kiến trúc pipeline của WrenAI. Mỗi module thực hiện chức năng cụ thể và có thể được cấu hình, kiểm thử riêng biệt, đồng thời tích hợp chặt chẽ thông qua interface chuẩn hóa.

3.3.1. Triển khai module tiền xử lý câu hỏi

Module tiền xử lý chuẩn hóa input và trích xuất thông tin quan trọng. Tiền xử lý tốt có thể cải thiện độ chính xác khi ánh xạ các thuộc tính schema. Khi người dùng nhập câu hỏi tự nhiên qua giao diện Wren UI, câu hỏi được gửi tới Wren AI Service. Tại đây sẽ thực hiện các bước xử lý: truy vấn vector DB để lấy context sau đó chuẩn bị prompt kèm theo câu hỏi để đưa vào LLM. Nhờ đó, WrenAI có thể ghi nhận các từ đồng nghĩa, alias, giúp chuyển ngôn ngữ tiếng Việt liên tục thành thông tin schema ra ràng.



Hình 3.3: Pipeline truy xuất thông tin cho quá trình xử lý truy vấn

3.3.2. Triển khai module ánh xạ schema

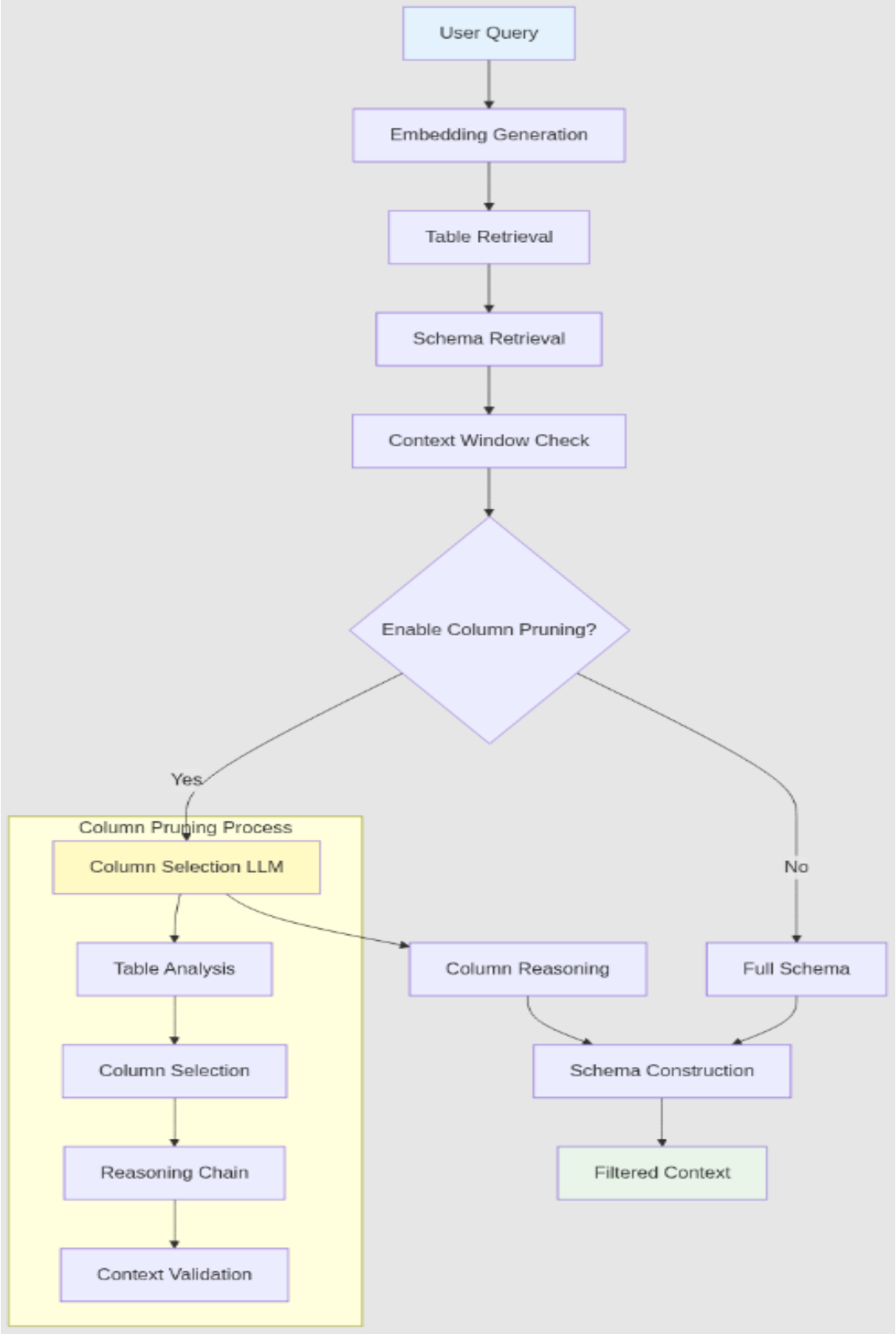
Một trong các điểm cốt lõi của WrenAI là việc sử dụng một semantic layer, định nghĩa metadata/schema để làm cầu nối giữa ngôn ngữ tiếng Việt với cấu trúc cơ sở dữ liệu vật lý. WrenAI dùng định dạng Metadata Definition Language (MDL) để định nghĩa schema, cột, quan hệ, metric/business logic.

MDL manifest (JSON) định nghĩa các models (tương ứng bảng hoặc view abstraction), các columns, kiểu dữ liệu, và relationships giữa các model (join logic, foreign-key).

Nếu cần, có thể định nghĩa các trường tính toán (calculated fields / metrics / business logic), alias / synonyms / business-term mapping, giúp người dùng dùng thuật ngữ nghiệp vụ (business term) mà không cần biết tên bảng/cột kỹ thuật.

Khi hệ thống nhận một truy vấn NL, semantic engine thực hiện tức là convert schema kết hợp metadata thành vector embedding, sau đó tìm những bảng/cột/metadata phù hợp với nội dung câu hỏi. Điều này giúp giảm trường hợp LLM tạo bảng/cột không tồn tại và tăng độ chính xác của mapping.

Nhờ vậy, hệ thống có thể hỗ trợ mapping business-term → bảng/cột, resolve join và relationship phù hợp, ngay cả khi người dùng không biết chi tiết schema kỹ thuật.



Hình 3.4: Mô hình hoạt động của module ánh xạ schema

Như vậy, module ánh xạ schema đóng vai trò then chốt đảm bảo rằng SQL được sinh sau này sẽ đúng với schema thật, đúng với mối quan hệ giữa bảng, đúng business logic, và ổn định khi schema/metadata thay đổi.

3.3.3. Triển khai module tương tác với mô hình ngôn ngữ lớn

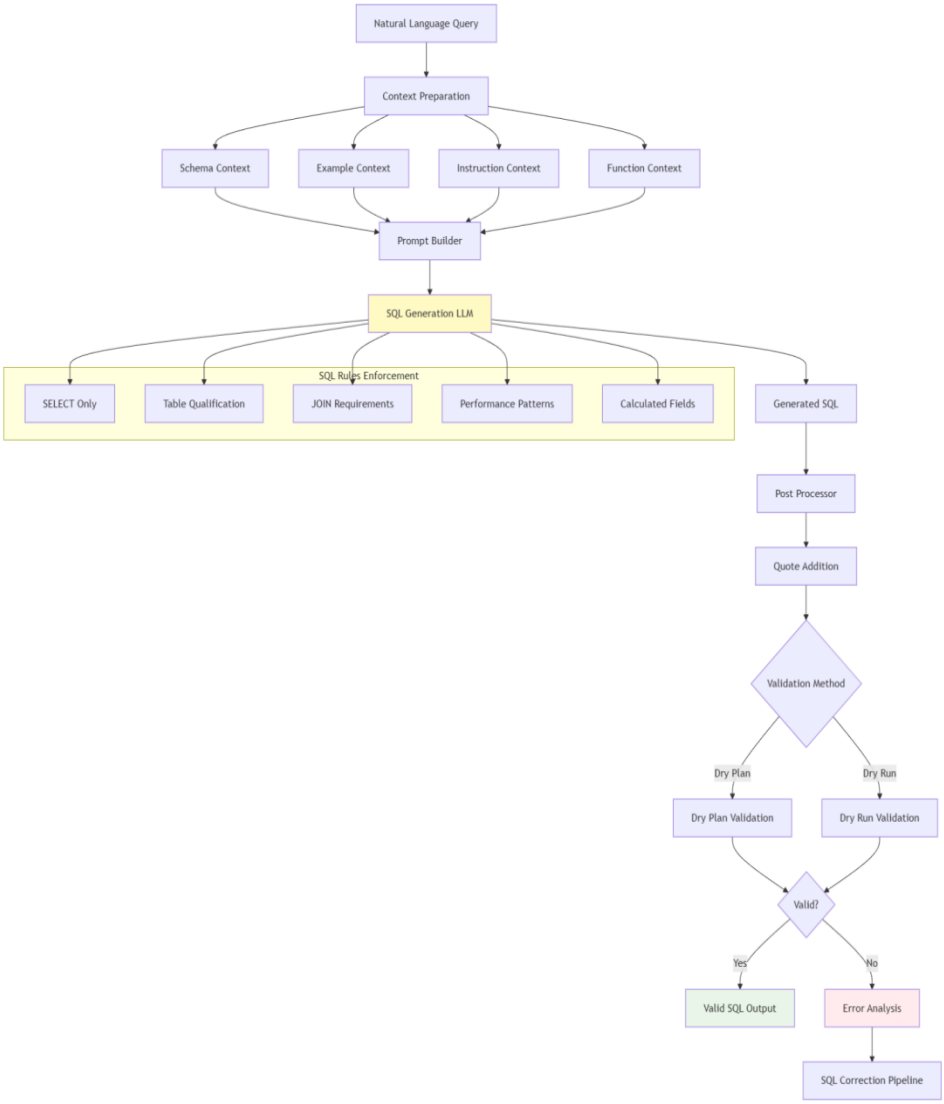
Sau khi pre-processing và ánh xạ schema / metadata, hệ thống chuyển sang bước sinh truy vấn SQL. Với WrenAI, phần này được thực hiện bởi Wren AI Service, sử dụng một hoặc nhiều LLM.

Quy trình cơ bản:

Wren AI Service kết hợp: metadata/schema (từ semantic layer), context/schema embeddings, business-term mapping + prompt template, plus câu hỏi đã pre-processing → tạo prompt gửi tới LLM.

LLM (có thể là OpenAI GPT-series, Google Gemini, hoặc các LLM được hỗ trợ khác) sinh ra một truy vấn SQL (và/hoặc định dạng biểu đồ, summary nếu user yêu cầu), dựa trên context + prompt. WrenAI hỗ trợ multi-LLM, linh hoạt tuỳ môi trường deploy.

Kết quả output từ LLM được chuyển lại cho Wren AI Service để xử lý tiếp: parse output, validate format (SQL hợp lệ, đúng cú pháp, không chứa lệnh nguy hiểm), chuẩn hoá nếu cần (nếu LLM sử dụng alias khác, hoặc nếu cần mapping lại tên bảng/cột theo schema thực).



Hình 3.5: Quy trình chuyển đổi từ ngôn ngữ tiếng Việt đến câu truy vấn SQL

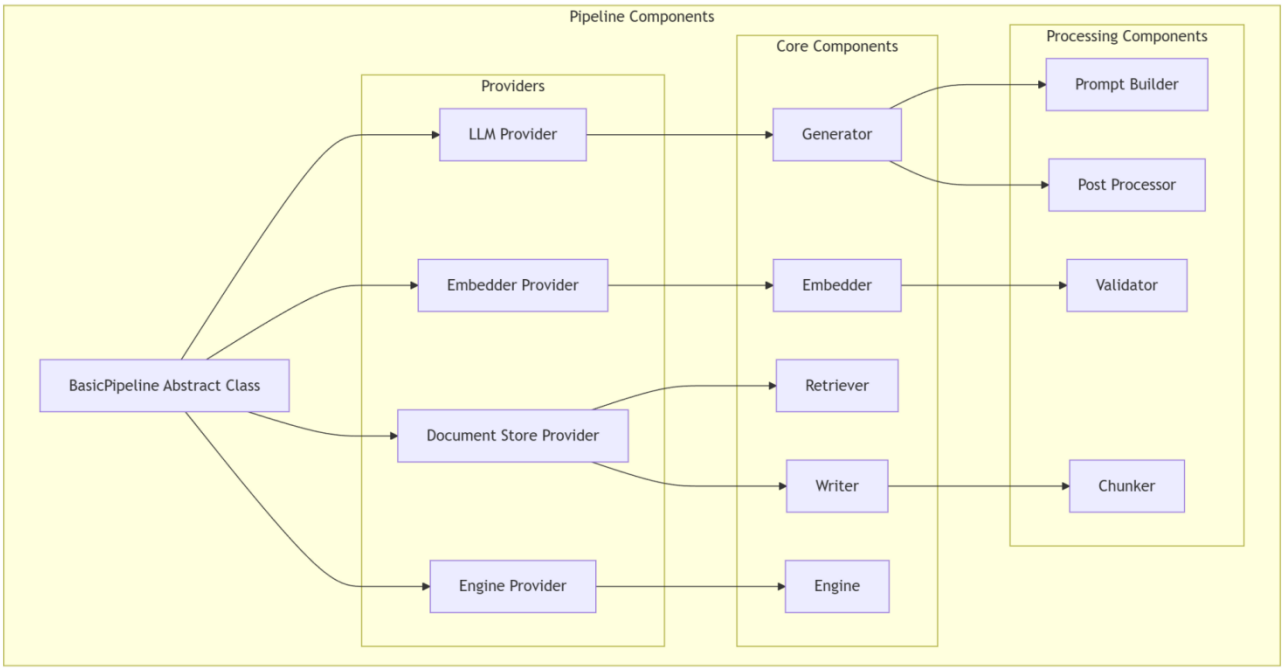
Nhờ có semantic layer kết hợp prompt và context metadata, WrenAI giảm thiểu lỗi phổ biến nếu dùng LLM thuần.

3.3.4. Triển khai module kiểm tra và thực thi SQL

Việc sinh câu truy vấn SQL từ LLM là không đủ cần kiểm tra cú pháp, bảo đảm hợp lệ, sau đó thực thi ở database đích, và trả kết quả cho user. Trong WrenAI, phần này được đảm nhiệm bởi Wren Engine core execution / query engine / semantic engine.

Các bước xử lý bao gồm:

- Wren Engine nhận SQL dịch sang dialect SQL phù hợp database thực tế.
- Có thể thực hiện kiểm tra cú pháp, schema existence, logic quan hệ, tránh thực thi những câu truy vấn SQL không hợp lệ.
- Nếu hợp lệ thực thi SQL trong database thu kết quả; nếu user yêu cầu thêm chuyển kết quả thành bảng, biểu đồ, summary, report, tùy UI / visualization module của WrenAI.
- Nếu có lỗi runtime quay lại xử lý: thông báo lỗi, logging, có thể yêu cầu user hiệu chỉnh câu hỏi, hoặc fallback.
- Như vậy, module kiểm tra & thực thi đảm bảo rằng hệ thống không chỉ viết SQL đúng cú pháp, mà chạy đúng trả đúng dữ liệu, an toàn, tuân thủ schema, business logic, bảo mật.



Hình 3.6: Kiến trúc Tổng quan BasicPipeline WrenAI

3.4. Xây dựng giao diện người dùng

Giao diện người dùng (UI) trong hệ thống này đóng vai trò cầu nối giữa người sử dụng và lớp xử lý dữ liệu/AI phía sau. UI không đơn thuần chỉ là phương tiện nhập

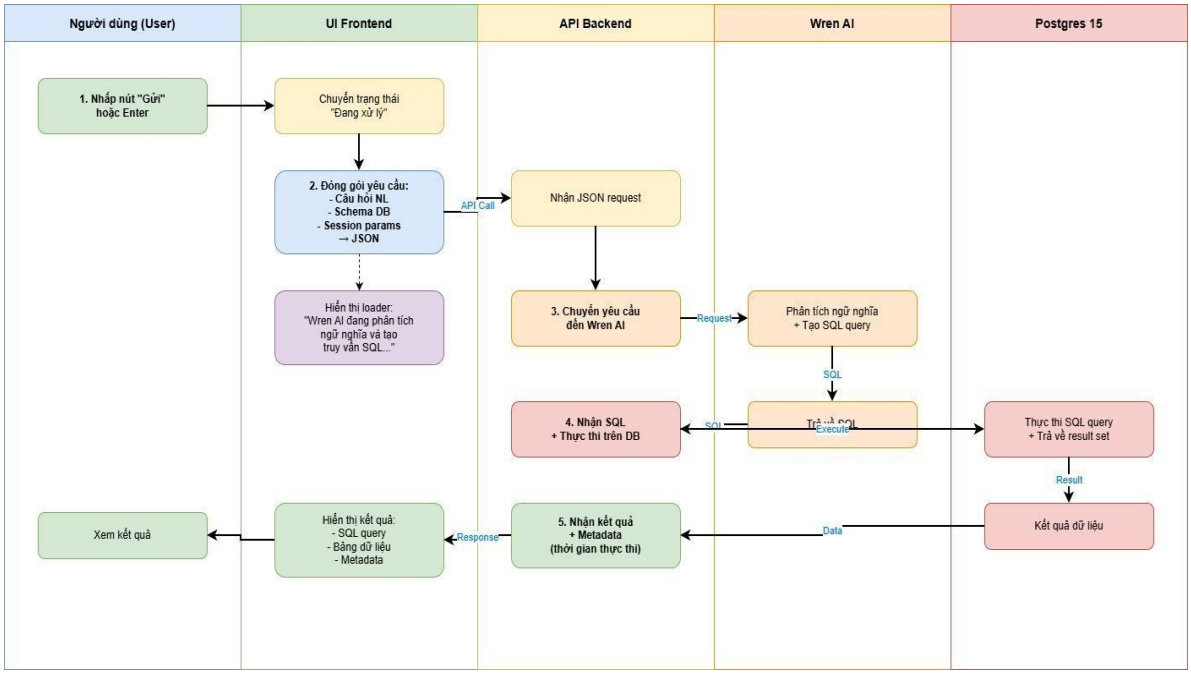
dữ liệu, mà là không gian tương tác giúp người dùng diễn đạt câu hỏi theo ngôn ngữ tiếng Việt, nhận lại kết quả dữ liệu hoặc biểu đồ, đồng thời quản lý metadata/schema, lịch sử truy vấn, và các thiết lập liên quan. Giao diện đảm bảo rằng trải nghiệm người dùng thân thiện, dễ hiểu, và dữ liệu được truy vấn đúng, an toàn. Việc xây dựng UI cần tuân theo các nguyên tắc UX/UI phù hợp với phân tích dữ liệu: rõ ràng, trực quan, giúp người dùng dễ hiểu, hỗ trợ hiển thị dữ liệu lớn và cung cấp tính năng quản lý để dễ bảo trì.

3.4.1. Triển khai giao diện nhập câu hỏi

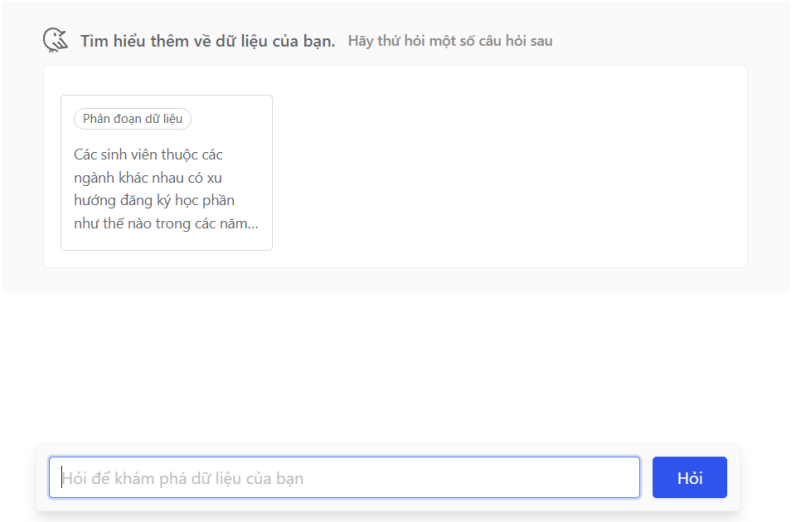
Giao diện nhập câu hỏi là điểm tương tác đầu tiên và quan trọng nhất của người dùng với hệ thống Text-to-SQL. Thiết kế của nó cần tập trung vào sự đơn giản, hiệu quả và khả năng định hướng người dùng đến việc đặt câu hỏi có cấu trúc tốt.

Bố cục của màn hình nhập câu hỏi được thiết kế tập trung sự chú ý của người dùng vào khu vực nhập liệu. Cấu trúc tổng thể được chia thành ba khu vực chính:

- Khu vực nhập liệu chính: Nằm ở trung tâm, nổi bật nhất trên màn hình. Ô nhập liệu này là nơi người dùng viết câu hỏi bằng ngôn ngữ tiếng Việt.
- Khu vực hỗ trợ ngữ cảnh và gợi ý: Nằm ngay dưới hoặc bên cạnh ô nhập liệu. Khu vực này bao gồm các chức năng hỗ trợ như gợi ý nhanh, hiển thị schema
- Khu vực điều khiển: Bao gồm nút gửi yêu cầu và các tùy chọn cấu hình cơ bản.



Hình 3.7: Luồng gửi câu hỏi



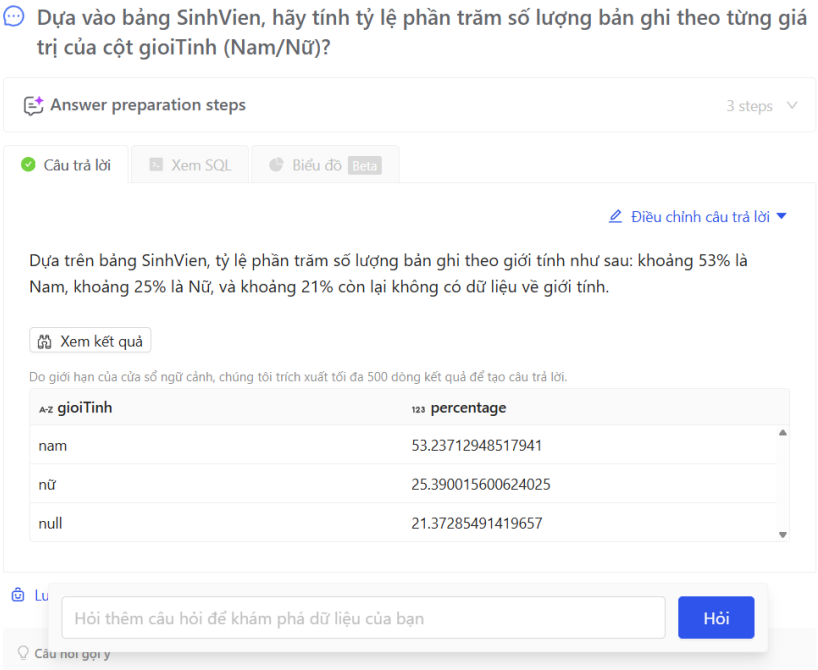
Hình 3.8: Giao diện nhập câu hỏi

3.4.2. Triển khai giao diện hiển thị kết quả

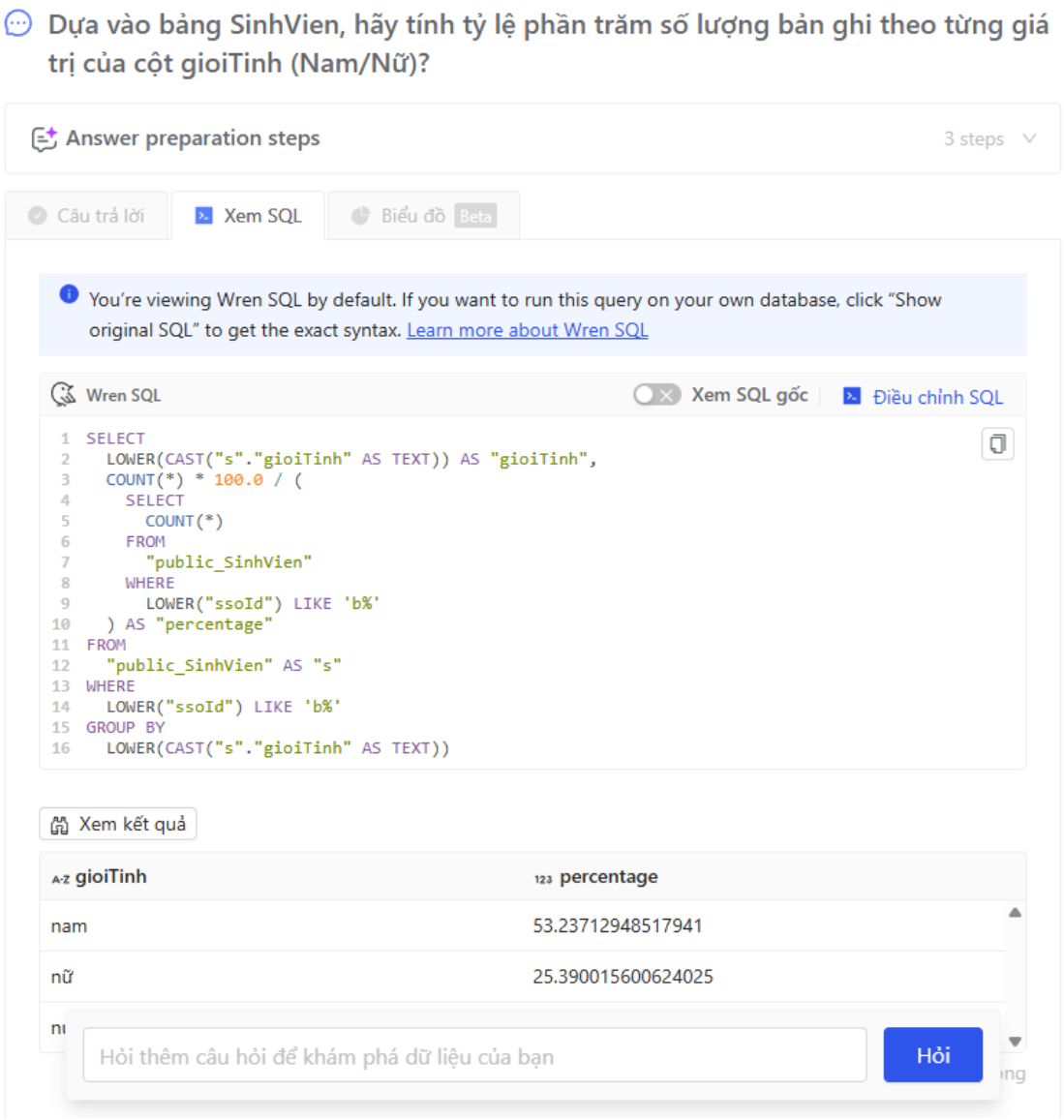
Giao diện hiển thị kết quả là nơi tổng hợp và trực quan hóa toàn bộ quy trình Text-to-SQL. Thiết kế ở mục này phải đảm bảo tính minh bạch, khả năng phân tích và độ chính xác cao.

Sau khi câu hỏi được xử lý và câu truy vấn được thực thi, giao diện sẽ hiển thị bao gồm nội dung câu hỏi của người dùng và nội dung câu trả lời phía bên dưới. Nội dung câu trả lời sẽ bao gồm 3 tab chuyển đổi.

1. Câu trả lời: Hiển thị mặc định khi hiển thị giao diện. Ở màn hình này sẽ đưa ra kết quả truy vấn dạng bảng cho người dùng.
2. Xem câu truy vấn: Màn hình này sẽ trả ra câu truy vấn cuối cùng để người dùng kiểm tra, người dùng có thể lưu lại câu truy vấn, khi đó nội dung câu hỏi và câu truy vấn sẽ được thêm vào phần quản lý câu hỏi mẫu.
3. Biểu đồ: Tùy thuộc vào yêu cầu của người dùng hệ thống sẽ hiển thị đưa ra biểu đồ tương ứng với yêu cầu.



Hình 3.9: Giao diện câu trả lời



Hình 3.10: Giao diện câu truy vấn SQL

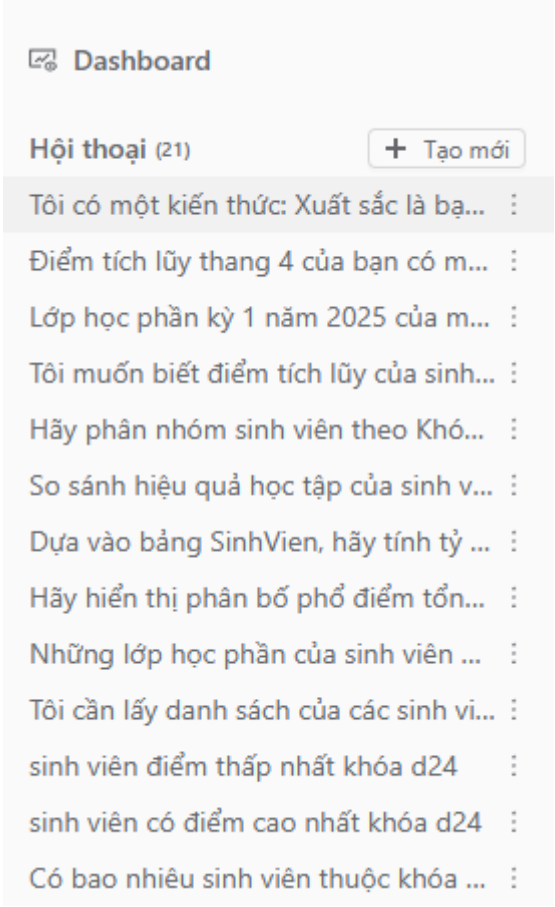


Hình 3.11: Giao diện biểu đồ trực quan

3.4.3. Triển khai các chức năng quản lý hỗ trợ

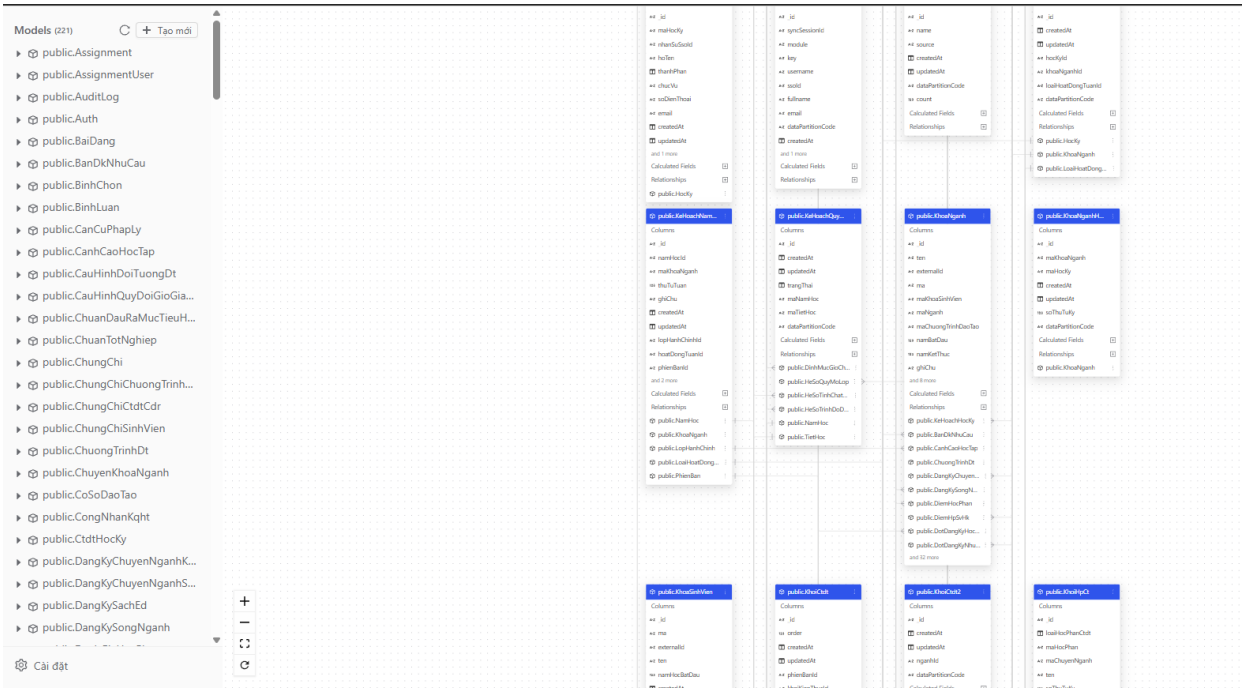
Để chuyển đổi từ một công cụ truy vấn đơn lẻ thành một hệ thống phân tích dữ liệu toàn diện, giao diện cần tích hợp một tập hợp các chức năng quản lý hỗ trợ nhằm nâng cao tính liên tục, khả năng tái sử dụng và độ an toàn của trải nghiệm người dùng.

Chức năng lịch sử không chỉ là danh sách các truy vấn đã thực hiện, mà là một công cụ học tập và quản lý công việc. Lịch sử được sắp xếp theo thứ tự thời gian, mỗi mục trong lịch sử có thể thực hiện lại câu truy vấn cho phép người dùng dễ dàng lặp lại hoặc tinh chỉnh các truy vấn trước đó.



Hình 3.12: Giao diện quản lý lịch sử truy vấn

Hiển thị schema và hỗ trợ ngữ cảnh cung cấp giao diện hiển thị chi tiết schema của cơ sở dữ liệu đang kết nối.



Hình 3.13: Giao diện quản lý schema

3.5. Thử nghiệm và đánh giá hệ thống

3.5.1. Kế hoạch và phương pháp thử nghiệm

Wren AI áp dụng kiểm thử theo kim tự tháp với ba tầng chính: Kiểm thử từng thành phần một, kiểm thử kết hợp giữa các thành phần với nhau, và kiểm thử toàn hệ thống. Cách tiếp cận này đảm bảo phát hiện lỗi sớm ở tầng thấp với chi phí thấp, đồng thời vẫn đảm bảo tính toàn vẹn của hệ thống qua các bộ test.

Hệ thống của tôi được chạy kiểm tra trên bộ gồm 1139 câu hỏi được khai thác trên hệ thống thủ công. Bộ đánh giá sẽ bao gồm: đánh giá dựa trên độ chính xác so với câu trả lời gốc đã có, và bên cạnh đó là hiệu suất phản hồi của của hệ thống.

Quy trình thực hiện đánh giá như sau:

- Bộ câu hỏi gồm 3 mức độ khác nhau: Mức độ dễ - câu hỏi chỉ truy vấn trên một bảng, mức độ trung bình - câu hỏi truy vấn 2 bảng, mức độ khó - câu hỏi truy vấn 3 bảng.

- Đối với phần kiểm tra hiệu năng sẽ thực hiện với mô phỏng với trường hợp 20 người, 30 người, 50 người, 100 người và 200 người để kiểm tra độ phản hồi của hệ thống.

3.5.2. Đánh giá độ chính xác của hệ thống

Với bộ câu hỏi mẫu gồm 1139 câu hỏi được thu thập và xây dựng dựa trên cơ sở dữ liệu đại học số cho thấy hệ thống đạt Execution Accuracy là 78.2% có nghĩa là trong tổng số 1139 câu hỏi có khoản 890 câu hỏi chuyển thành câu truy vấn được thực hiện đúng. Hình 3.14 là ví dụ "về câu hỏi liên quan đến điểm thống kê điểm tổng kết của các sinh viên trong kỳ trước"

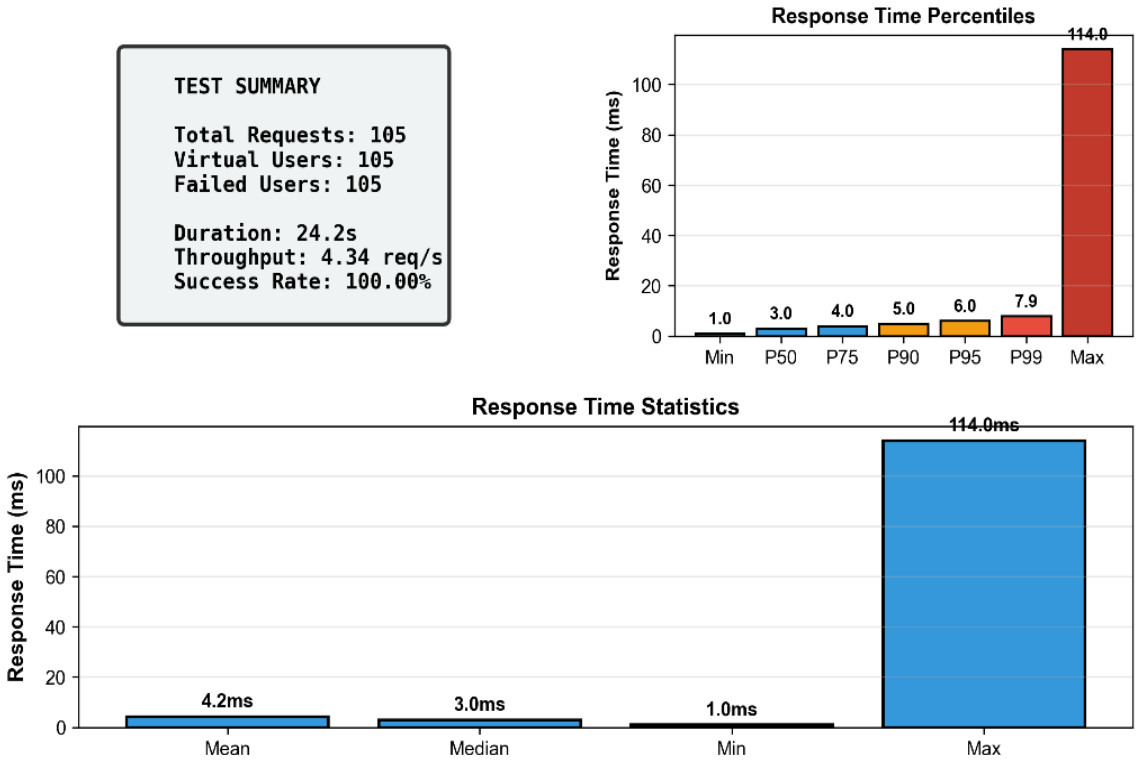


Hình 3.14: Ví dụ về kết quả bước Text-to-SQL với truy vấn "Hiển thị phổ điểm tổng kết của các sinh viên trong kỳ vừa qua"

3.5.3. Đánh giá hiệu suất của hệ thống

Hiệu suất truy vấn được đo lường qua thời gian phản hồi, thông lượng, và mức sử dụng tài nguyên. Các chỉ số quan trọng bao gồm thời gian phản hồi trung bình, độ trễ phân vị 95/99, và số truy vấn mỗi giây.

Kết quả thực hiện với câu hỏi "Tìm điểm thành phần của các học phần sinh viên có mã B23DCCC144". Để thực hiện kiểm tra hiệu suất hệ thống sử dụng công cụ artillery để mô phỏng các thao tác thực thi trên trang web. Hình 3.15 là hình ảnh các biểu đồ báo cáo hiệu suất của hệ thống: Thông lượng, P95, P99 của 105 người dùng.



Hình 3.15: Hình ảnh báo cáo của artillery với câu hỏi "Tìm điểm thành phần của các học phần sinh viên có mã B23DCCC144" với truy vấn của 105 người.

Kiểm thử tải và kiểm thử căng đánh giá khả năng xử lý tải của hệ thống, xác định các điểm nghẽn, và kiểm tra độ ổn định dưới áp lực cao. Các kịch bản kiểm thử bao gồm người dùng đồng thời, tải liên tục, và kiểm thử tăng đột ngột. Tôi thực hiện chạy script phần mềm artillery với 20 người, 50 người, 100 người, 200

người truy vấn cùng lúc. Bảng dưới thể hiện các thông số sau khi thực hiện khảo sát.

Bảng 3.2: Đánh giá khả năng chịu tải của hệ thống

Người dùng đồng thời	Tổng số yêu cầu	Thời gian thực hiện (s)	Thông lượng (req/s)	Tỷ lệ thành công (%)	Phản hồi TB (ms)	Median (ms)	P95 (ms)	P99 (ms)	Tỷ lệ lỗi (%)
20	20	9.5	10.57	100.00	7.6	4.0	12.1	100.5	0.00
50	50	16.2	30.88	100.00	1616.7	645.6	4965.3	5168.0	0.00
100	100	41.7	23.98	100.00	12179.1	10201.2	29445.4	30647.0	0.00
200	200	71.0	28.17	70.85	25384.0	23162.4	55843.6	58122.9	29.15

3.5.4. So sánh hiệu suất các mô hình ngôn ngữ lớn

Để đánh giá và lựa chọn mô hình ngôn ngữ lớn (LLM) phù hợp cho hệ thống Text-to-SQL tiếng Việt, chúng tôi tiến hành so sánh bốn mô hình phổ biến hiện nay: GPT-4o-mini, DeepSeek-V3/R1, Claude 3.7 Sonnet và Gemini 3 Pro. Việc lựa chọn này dựa trên khả năng hỗ trợ của Wren AI và sự phổ biến của các model trong cộng đồng.

Bộ dữ liệu test: Từ tập 1,139 câu hỏi đã xây dựng trong Mục 3.2.3, chúng tôi lựa chọn 100 câu hỏi đại diện phân bố đều theo ba mức độ phức tạp để đảm bảo tính đại diện và khách quan trong đánh giá.

Các tiêu chí đánh giá: Chúng tôi đánh giá các mô hình theo năm tiêu chí chính. Thứ nhất là khả năng SQL/Coding được đo bằng chỉ số HumanEval, đánh giá khả năng sinh code chính xác. Thứ hai là lập luận logic thông qua benchmark GPQA (Graduate-Level Google-Proof Q&A), đo khả năng reasoning phức tạp. Thứ ba là độ

chính xác Text-to-SQL thực tế trên dữ liệu tiếng Việt của hệ thống. Thứ tư là chi phí sử dụng tính theo triệu tokens. Cuối cùng là tốc độ xử lý đo bằng tokens/giây, ảnh hưởng trực tiếp đến trải nghiệm người dùng.

Môi trường test: Tất cả các test được thực hiện trên cùng môi trường Wren AI với PostgreSQL 16, MDL đã được chuẩn hóa đầy đủ theo semantic layer, và temperature = 0.1 để đảm bảo tính nhất quán.

Bảng 3.3: Bảng so sánh tổng hợp các mô hình ngôn ngữ lớn

Tiêu chí	GPT-4o-mini	DeepSeek-V3 / R1	Claude 3.7 Sonnet	Gemini 3 Pro
SQL/Coding (HumanEval)	70%	~80%+	85%	80%
Lập luận logic (GPQA)	Trung bình	Rất cao	Rất cao	Cao
Độ chính xác Text-to-SQL	Khá (Dễ sai ở bảng lớn)	Xuất sắc	Xuất sắc	Rất tốt
Chi phí (per 1M tokens)	$0.15	$0.14 - $0.28	$3.00	Tùy chọn miễn phí
Tốc độ (Tokens/sec)	Rất nhanh	Trung bình	Nhanh	Cực nhanh

3.6. Cải tiến và tối ưu hóa

3.6.1. Phân tích điểm yếu của hệ thống

Mặc dù việc áp dụng Wren AI kết hợp với PostgreSQL 15 đã giúp hệ thống hướng đến khả năng cho phép người dùng không cần biết SQL mà vẫn truy vấn dữ liệu, kết quả thử nghiệm cho thấy còn nhiều hạn chế kỹ thuật và trải nghiệm cụ thể:

Độ chính xác vẫn còn hạn chế, đặc biệt với câu hỏi phức tạp từ việc chuyển ngôn ngữ tiếng Việt sang câu truy vấn SQL là một dạng của phân tích ngữ nghĩa, hệ thống thường gặp khó khăn khi câu hỏi người dùng thường cấu trúc phức tạp nhiều

mệnh đề, điều kiện lồng nhau, nhiều ràng buộc. Một số kết quả thử nghiệm cho thấy SQL sinh ra đúng cú pháp, thực thi được nhưng trả về dữ liệu sai so với ý định.

Tốc độ phản hồi không ổn định với những bảng dữ liệu lớn. Việc thực thi các truy vấn với nhiều bảng, nhiều quan hệ, nhiều cột gây ra trường hợp overhead trước khi sinh SQL. Nếu không có cơ chế tối ưu, thời gian chờ sẽ lâu, ảnh hưởng đến người dùng. Với nhiều truy vấn phức tạp việc thực thi trên cơ sở dữ liệu cũng trở thành vấn đề, phải có cách tối ưu câu truy vấn để cải thiện độ trễ khi thực thi trên cơ sở dữ liệu.

Wren AI sử dụng nội bộ một ngôn ngữ trung gian riêng để người dùng dễ tương tác, trong trường hợp SQL sinh ra không tương thích với cú pháp SQL sẽ dẫn đến lỗi khi chạy hoặc kết quả không như mong đợi.

Khi sinh câu truy vấn SQL lỗi, thông báo lỗi thường kỹ thuật gây khó hiển và khó tự sửa. Nếu phản hồi chậm hoặc trả về dữ liệu không chính xác, người dùng khó tin tưởng hệ thống để thực hiện các truy vấn quan trọng.

3.6.2. Các giải pháp cải tiến

Để khắc phục các hạn chế về độ chính xác, hiệu suất, và trải nghiệm người dùng ở hệ thống hiện tại, chúng tôi đề xuất các giải pháp sau:

- Tối ưu prompt và cung cấp context schema đầy đủ: Trong prompt gửi đến Wren AI đính kèm metadata chi tiết của database: tên bảng, cột, kiểu dữ liệu, ràng buộc, mô tả quan hệ, giúp hệ thống hiểu rõ cấu trúc. Việc đó giúp giảm trường hợp nhầm bảng/cột, giảm sai sót và cải thiện độ chính xác và tính đúng đắn của SQL.
- Cập nhật bảng ánh xạ giữa các thuật ngữ nghiệp vụ và bảng/cột thực trong cơ sở dữ liệu, làm giàu thêm bộ từ đồng nghĩa, mô tả các cột bảng để tăng tính đúng đắn khi thực hiện trích xuất dữ liệu.
- Làm giàu kho câu hỏi mẫu: Trong quá trình thử nghiệm có thể xây dựng thêm kho câu hỏi mẫu từ những câu đã prompt từ đó có thể cải thiện độ chính xác của LLM.

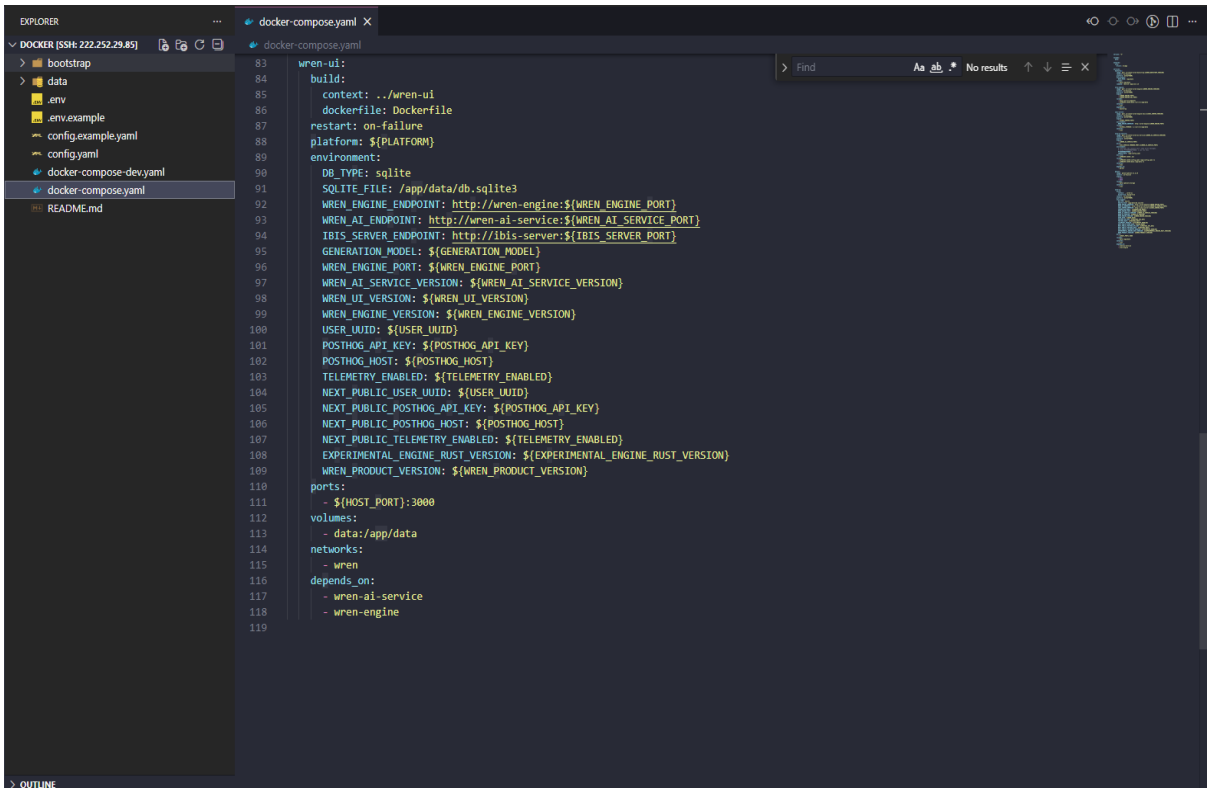
- Bổ sung thêm các hướng dẫn để định hướng cho Wren AI trong việc tạo truy vấn SQL, vì các hướng dẫn này mô giúp Wren AI hiểu mô hình dữ liệu và quy tắc nghiệp vụ của hệ thống hiện tại.
- Thiết lập giám sát, thu thập feedback và cải tiến liên tục: ghi lại câu hỏi và câu truy vấn đã sinh, lỗi, kết quả, thời gian thực thi, cho phép người dùng đánh giá và sửa câu truy vấn sẽ được dùng làm ví dụ trong qua trình cải thiện kết quả sau này.

3.7. Triển khai và hướng dẫn sử dụng

3.7.1. Hướng dẫn cài đặt và cấu hình

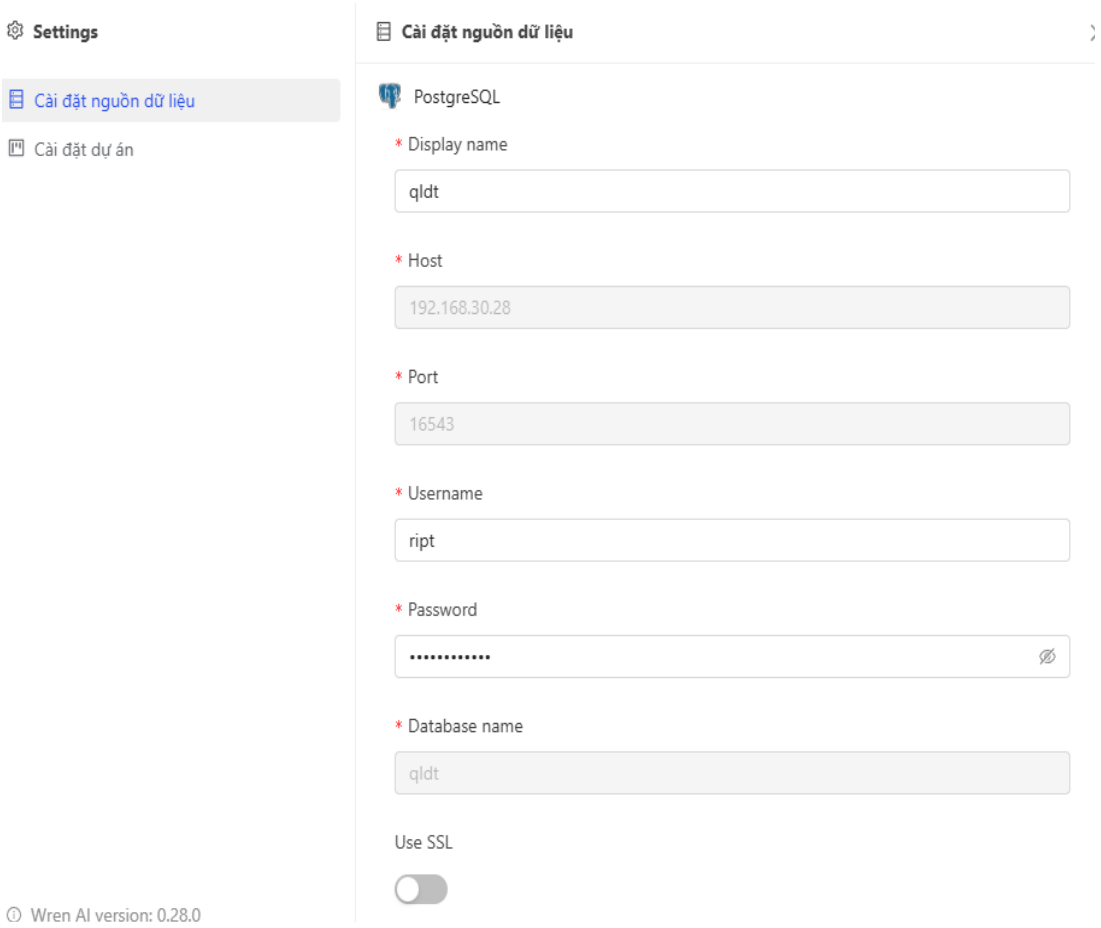
Để triển khai công cụ trên môi trường thực tế các bước để cài đặt và cấu hình hệ thống như sau:

Bước 1, trên máy chủ cài đặt Docker phiên bản mới nhất, sau đó thực hiện tải mã nguồn từ github và thực hiện cài đặt bằng docker compose.



Hình 3.16: Cấu hình công cụ Wren AI trong file docker compose

Bước 2, sau khi server được khởi tạo thành công mở giao diện Wren UI và thực hiện cài đặt kết nối đến cơ sở dữ liệu nguồn, để đảm bảo những vấn đề thực thi các câu truy vấn không được phép chỉ cung cấp username với quyền đọc dữ liệu.



Hình 3.17: Giao diện cài đặt cơ sở dữ liệu nguồn

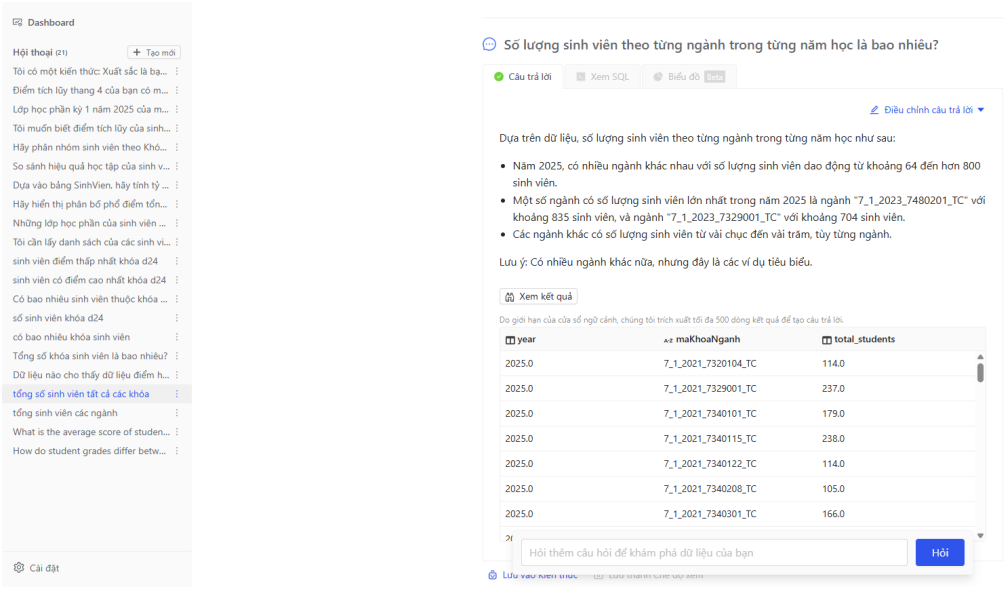
3.7.2. Hướng dẫn sử dụng cho người dùng cuối

Giao diện người dùng được thiết kế đơn giản, dễ tiếp cận hướng tới người dùng không biết SQL với các hướng dẫn như sau:

1. Truy cập giao diện Wren AI UI trên ứng dụng trình duyệt web sau khi triển khai và kết nối đến cơ sở dữ liệu nguồn.
2. Trên giao diện hiển thị có thanh lịch sử các truy vấn cũ và một giao diện khung chat chính sử dụng để nhập câu hỏi bằng ngôn ngữ tiếng Việt, sau khi nhập sau câu hỏi sẽ có nút gửi câu hỏi.

3. Sau khi nhập và gửi câu hỏi Wren AI sẽ thực hiện các nghiệp vụ để sinh ra câu SQL và kết quả thực thi trả về.
4. Sau khi có kết quả có thể kiểm tra kết quả trả về, xem báo cáo.

5. Nếu kết quả lỗi có thể thực hiện xử lý lỗi bằng cách cải thiện câu hỏi. Nếu câu SQL không hợp lệ có thể chỉnh sửa câu truy vấn để đảm bảo được kết quả đúng.



Hình 3.18: Giao diện truy vấn và lịch sử truy vấn của người dùng

3.7.3. Hướng dẫn vận hành và bảo trì

Để hệ thống hoạt động ổn định và bền vững cần thiết lập quy trình vận hành và bảo trì như sau:

1. Kiểm tra và bảo trì định kỳ: Theo dõi các truy vấn sinh ra, những lỗi cú pháp, hiệu năng truy vấn dữ liệu, có cơ chế cache dữ liệu hiệu quả. Cập nhật schema khi cơ sở dữ liệu nguồn thay đổi cần bổ sung các mô tả tránh ứng dụng tạo ra những kết quả sai. Sao lưu dữ liệu định kỳ bao gồm cơ sở dữ liệu, cấu hình, log, cache.
2. Giám sát hiệu suất của cơ sở dữ liệu: Theo dõi các chỉ số thời gian trung bình thực thi câu truy vấn, số lượng kết nối, thời gian trả kết quả,. Điều chỉnh cấu hình nếu cần.
3. Cập nhật tinh chỉnh model, prompt của Wren AI khi dữ liệu thay đổi.

3.8. Kết luận chương 3

Chương 3 đã trình bày một cách hệ thống quy trình triển khai hệ thống chuyển đổi ngôn ngữ tiếng Việt sang câu truy vấn SQL dựa trên nền tảng Wren AI tích hợp với Postgres 15. Thông qua việc mô tả kiến trúc tổng thể, cơ chế tích hợp giữa các thành phần, và phương pháp phát triển giao diện người dùng theo nguyên tắc lấy người dùng làm trung tâm, nghiên cứu đã làm rõ cách thức hoạt động của từng module trong hệ thống cũng như khả năng tương tác hiệu quả với cơ sở dữ liệu quan hệ. Đặc biệt, kết quả thử nghiệm trước và sau cải tiến đã chứng minh tính khả thi của mô hình Text-to-SQL với sự tiến bộ rõ rệt về độ chính xác cú pháp và ngữ nghĩa, đồng thời quy trình phát triển lặp với chu trình phản hồi liên tục cho thấy hiệu quả của phương pháp tối ưu hóa được áp dụng. Các hoạt động triển khai và tài liệu hướng dẫn kỹ thuật đã được hệ thống hóa, tạo điều kiện cho việc nhân rộng giải pháp trong thực tế. Dựa trên nền tảng này, chương 4 sẽ tiến hành đánh giá toàn diện hiệu năng hệ thống, phân tích khả năng ứng dụng thực tiễn, đồng thời đề xuất các hướng nghiên cứu mở rộng nhằm nâng cao chất lượng và phạm vi áp dụng của công cụ trong bối cảnh doanh nghiệp.

KẾT LUẬN VÀ KIẾN NGHỊ

Đề tài đã đạt được mục tiêu ban đầu trong việc thiết kế và triển khai một công cụ vận dụng trí tuệ nhân tạo nhằm thực hiện chuyển đổi tự động từ câu hỏi bằng tiếng Việt sang câu truy vấn SQL, từ đó mang lại một giải pháp truy xuất thông tin có tính ứng dụng cao cho bối cảnh Đại học số ở Việt Nam. Quá trình nghiên cứu đã thực hiện công tác hệ thống hóa các tri thức liên quan đến xử lý ngôn ngữ tiếng Việt, các tiếp cận trong bài toán Text-to-SQL, đồng thời làm rõ những đặc trưng riêng của tiếng Việt khi áp dụng vào lĩnh vực giáo dục đại học số. Trên phương diện ứng dụng, hệ thống đã cho thấy khả năng rút ngắn đáng kể thời gian cần thiết cho việc tra cứu thông tin, đồng thời tạo điều kiện thuận lợi cho các nhà quản lý trong công tác đưa ra quyết định dựa trên phân tích dữ liệu. Đề tài cũng đưa ra đóng góp trong việc xây dựng mô hình dữ liệu chuẩn hóa dành cho môi trường Đại học số, có thể được sử dụng như tài liệu tham khảo cho các công trình nghiên cứu có tính chất tương đồng trong thời gian tới.

Song song với những thành tựu đạt được, công trình nghiên cứu vẫn còn tồn tại một số hạn chế cần được cải thiện. Về khía cạnh độ chính xác, hệ thống thể hiện sự khó khăn khi xử lý các câu truy vấn mang tính phức tạp cao, đặc biệt là những truy vấn đòi hỏi việc liên kết nhiều nguồn dữ liệu, sử dụng truy vấn lồng nhau và thực hiện các phép tính thống kê ở mức độ nâng cao. Nguyên nhân xuất phát từ việc mô hình ngôn ngữ chưa trải qua quá trình huấn luyện chuyên sâu dành riêng cho lĩnh vực giáo dục với ngữ liệu tiếng Việt. Năng lực giải quyết những câu hỏi thiếu rõ ràng, duy trì mạch thông tin qua nhiều lượt trao đổi, và ứng phó với sự đa dạng trong cách thức diễn đạt còn chưa được hoàn thiện. Về phương diện triển khai thực tế, việc phụ thuộc vào giao diện lập trình ứng dụng (API) của các mô hình ngôn ngữ thương mại đặt ra những thách thức liên quan đến chi phí vận hành cũng như vấn đề bảo mật khi cần truyền tải thông tin đến các bên cung cấp dịch vụ bên ngoài.

Dựa trên những hạn chế đã được nhận diện, nghiên cứu đưa ra các định hướng phát triển cho giai đoạn tiếp theo. Về việc nâng cao độ chính xác, cần triển khai công

tác huấn luyện bổ sung cho mô hình với tập ngữ liệu tiếng Việt chuyên ngành giáo dục, xây dựng hệ thống từ điển thuật ngữ có độ phong phú cao hơn, và thiết kế các cơ chế xử lý ngữ cảnh ở mức độ tiên tiến. Việc tích hợp phương pháp học tăng cường dựa trên phản hồi từ người sử dụng, kết hợp với việc phát triển module có khả năng tự động phát hiện và điều chỉnh lỗi trong truy vấn, sẽ giúp hệ thống có khả năng tự hoàn thiện qua thời gian sử dụng. Về khía cạnh mở rộng chức năng, hệ thống có tiềm năng được bổ sung các năng lực phân tích có tính dự báo, hệ thống đưa ra khuyến nghị thông minh, cùng với khả năng tạo lập các biểu đồ và bảng điều khiển có tính tương tác. Những tính năng này sẽ hỗ trợ đắc lực trong việc đưa ra các quyết định dựa trên dữ liệu được trực quan hóa một cách nhanh chóng và hiệu quả.

TÀI LIỆU THAM KHẢO

[1] Chính phủ Việt Nam, "Quyết định số 749/QĐ-TTg về phê duyệt Chương trình chuyển đổi số quốc gia đến năm 2025, định hướng đến năm 2030," Hà Nội, Việt Nam, 2020.

[2] Bộ Giáo dục và Đào tạo, "Kế hoạch chuyển đổi số ngành Giáo dục và Đào tạo giai đoạn 2021-2025," Hà Nội, Việt Nam, 2021.

[3] N. T. Anh và P. V. Nam, "Challenges in database management for Vietnamese universities," in *Proc. Int. Conf. Education Technology*, 2023, pp. 145-152.

[4] T. H. Minh, "IT workload in higher education institutions," *Journal of Information Systems in Education*, vol. 15, no. 2, pp. 67-78, 2023.

[5] T. Yu *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3911-3921.

[6] D. Gao *et al.*, "Text-to-SQL empowered by large language models: A benchmark evaluation," *arXiv preprint arXiv:2308.15363*, 2024.

[7] W. Lei *et al.*, "Re-examining the role of schema linking in text-to-SQL," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6943-6954.

[8] J. Curran, "Enabling Natural Language Prompting with AtScale's Semantic Layer and Generative AI," AtScale Technical White Paper, Aug. 2024. [Online]. Available: https://www.atscale.com/blog/enable-natural-language-prompting-with-semantic-layer-genai/. [Accessed: Feb. 4, 2026].

[9] J. P. Nascimento *et al.*, "Bridging the gap: Text-to-SQL for real-world databases," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2025, pp. 156-169.

[10] Y. Wang *et al.*, "DBCopilot: A multi-database adaptive text-to-SQL framework," *arXiv preprint arXiv:2501.02450*, 2025.

[11] H. Chen, "ASKSQL: Cost-effective text-to-SQL with Vietnamese language support," in *Proc. ACL Workshop on NLP for Low-Resource Languages*, 2025, pp. 78-85.

[12] L. Zhang *et al.*, "SPS-SQL: Semantic prompt scaffolding for small models," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025, pp. 2341-2355.

[13] L. Dou et al., "MultiSpider: Towards Benchmarking Multilingual Text-to-SQL Semantic Parsing," arXiv preprint arXiv:2212.13492, Dec. 2022.

[14] K. T. Pham, T. H. Nguyen, J. Jo, Q. V. H. Nguyen, and T. T. Nguyen, "Multilingual text-to-SQL: Benchmarking the limits of language models with collaborative language agents," arXiv:2509.24405, Sep. 2025. [Online]. Available: https://arxiv.org/abs/2509.24405.

[15] Wren AI, "Wren AI: Open-source Text-to-SQL Solution," *GitHub Repository*, 2024. [Online]. Available: https://github.com/Canner/WrenAI

[16] K. Curran and J. Mariani, "Improving text-to-SQL accuracy with semantic layer," AtScale Technical Report, 2024.

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung đề án qua phần mềm Turnitin một cách trung thực và đạt kết quả mức độ tương đồng 18% toàn bộ nội dung đề án. Bản đề án kiểm tra qua phần mềm là bản cứng đề án đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Hà Nội, ngày …… tháng …… năm ……

HỌC VIÊN CAO HỌC

(Ký và ghi rõ họ tên**)**

Bùi Khắc Ngọc



NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN CAO HỌC
(Ký và ghi rõ họ tên)	(Ký và ghi rõ họ tên)
PGS.TS. NGÔ QUỐC DŨNG	Bùi Khắc Ngọc