Excel@FIT 2025
http://excel.fit.vutbr.cz

RAG agent for intuitive work with SQL databases

Marek Tenora*

Abstract
People often struggle with efficiently querying databases. This project develops a retrieval-augmented generation (RAG) system enabling natural language database queries with high accuracy, relevance, and factual correctness. The solution employs a graph-based architecture using LangGraph with a single advanced agent capable of both structured SQL querying and semantic vector search. Following a structured two-phase (research and respond) workflow, the agent analyzes questions, retrieves relevant database content, validates information, and provides comprehensive responses in Czech with proper citations.
Evaluation with a historical chronicle database demonstrates the system's capability to handle complex queries involving multiple-table joins while maintaining traceability and transparency. Differentiating between validated (SQL-retrieved) and contextual (vector-searched) information enhances the trustworthiness of responses. However, even strict instructions cannot fully eliminate occasional hallucinations by language models.
The implemented approach provides a useful framework for historians and humanities scholars and could serve as a model for developing natural language query systems across various complex database domains.

*xtenor02@stud.fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Efficiently querying SQL databases remains a challenge for many users, especially those without SQL expertise. Existing solutions like AskYourDatabase allow natural language queries, but often lack flexibility or deep semantic understanding. NotebookLM enables natural language interaction, but is not tailored for databases.

This work presents a web application that enables users to connect to any MySQL database, automatically index its contents, and query it in natural language (Czech). The backend uses FastAPI and LangGraph to orchestrate an advanced agent capable of both SQL and semantic vector search. The frontend is built in Vue.js. Users simply provide a connection string, and the system handles indexing and embedding records for fast, context-aware retrieval.

Evaluation using a historical database shows that the system reliably understands diverse database structures and handles complex queries, including multi-table joins. This flexibility and accuracy distinguish it from existing tools, making it suitable for historians, researchers, and anyone needing intuitive access to SQL data. Keep in mind that AI can make mistakes, and all the responses should be validated using the provided sources.

2. Implementation

Recent years have been remarkable for artificial intelligence, with new models consistently surpassing previous ones almost weekly. This rapid evolution has significantly impacted my project, leading me to rewrite the entire system logic around five times.

2.1 Agent

I've found agent systems to be a large leap forward, and I decided to use them. They allow the LLM to use multiple tools, reason, evaluate its own responses, and react based on them. The results of this project were not that promising before implementing agents.

After weeks of testing multiple approaches, I've come to use a single-agent workflow enhanced with another LLM, forming the final answer for the user, so the research agent can focus on the task.

Research agent can use multiple tools:

- SQLDatabaseToolKit - set of tools, enabling querying, retrieving database and table struc-

ture information, and checking SQL query correctness before it's executed.
- Vector database retriever tool - executes similarity search on the vector database, which helps the agent to find better context about the database and where the wanted information lies.

2.2 Prompting

Using effective prompts is essential. Refining the prompts took significant time on this project.

The prompt for the research agent contains:

- Always include formatted output in the response
- Use a vector database for hinting, never rely only on the results from it.
- Only results from the SQL database are validated and can be used for responding to the user
- Provide internal notes for the response assistant

Prompt for response assistant contains:

- Answer always in Czech language
- Format the response in markdown
- Add citations
- Never respond with only unvalidated information

2.3 Memory managment

Pasting the whole message history into the LLM isn't always the best and most efficient option. In my implementation, Research Agent gets full conversation history, and the respond assistant is using optimized memory, where only the messages from the last user question and the last Research Agent response are included.

This way, the response assistant is not distracted by all the previous messages, and the research agent can utilize research conducted in the past.

2.4 Evaluation

The system was tested mainly on an incomplete history database, where it really set a challenge for anyone to determine what's going on in there. Providing an evaluation is important as it is proof that the system is actually working. I used the framework Langsmith, which provides a web application where you can analyze your runs and see the results easily. Metrics for evaluating my project are:

- Correctness -
- Correctness (with pedantic judge) - Uses pydantic agent instead of LLM to evaluate the results.
- Relevance -

I made a dataset containing user questions and reference system answers based on the history database.

Acknowledgements

I would like to express my gratitude to my supervisor, Ing. Ondřej Klíma, Ph.D., for his assistance with the project. I appreciate his help in providing me with the history database and for connecting me with historians who offered valuable insights regarding the project's requirements.