Building a RAG-Based Question Answering System

Using the MongoDB Movies Dataset for AI-Powered Chat Systems

Problem Statement

Intelligent Movie Q&A

Develop a system to answer user questions about movies.

Hybrid Database Access

Combine MongoDB (NoSQL) and PostgreSQL (SQL) for structured data.

Context-Aware Answers

Utilize Retrieval-Augmented Generation (RAG) for semantic responses.

Business Use Cases

AI Chat Assistant

For OTT platforms to enhance user experience.

Intelligent FAQ

For cinemas or streaming applications.

Semantic Search

Across movie metadata and reviews.

User Feedback Insights

To inform business decisions.

Data Migration & Pre-processing

Goal: Transform complex, semi-structured MongoDB data into flat tables for PostgreSQL.

01

Lists Exploded

Normalized into separate tables (genres, cast).

02

Nested Dicts Flattened

Into new columns (e.g., imdb_rating, imdb_votes).

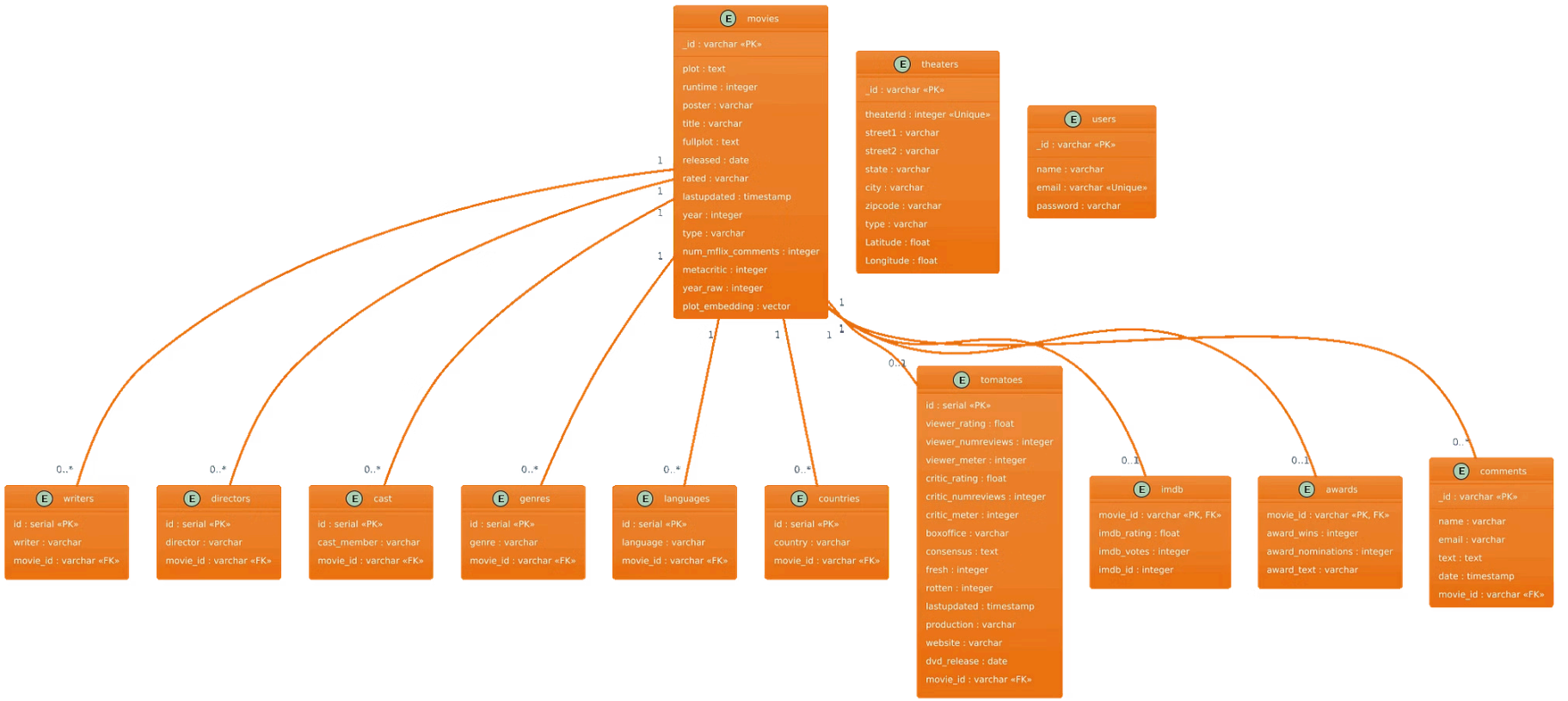
03

Data Cleaning

Missing list values filled with "Unknown"; coordinates extracted to Lat/Long.

Output: Pre-processed CSV files for each entity, ready for SQL import.

ER Diagram



Project Lifecycle Workflow

Data Extraction & Pre-processing

MongoDB connection, flatten with pandas, save CSVs.

Database Injection

Create PostgreSQL schema, insert CSV data.

Embedding & Vectorization

Extract text, generate embeddings, store in FAISS.

Query-Answering App

Streamlit UI, classify queries, route to SQL or FAISS+LLM.

Query Paths

Structured Query Path

- Natural-language question
- LLM-generated SQL
- Execute on Postgres
- Return results

Semantic Query Path (RAG)

- Query
- FAISS similarity search
- Augment prompt with retrieved plots
- LLM generates natural-language answer

Preprocessing & Embeddings

Creating a Semantic Knowledge Base:

1 Extract Text

Fullplot/plot text from Postgres movies table.

2 Convert to Vectors

Dense vectors using OpenAIEmbeddings.

3 Store in FAISS

For fast similarity search.

4 Semantic Search

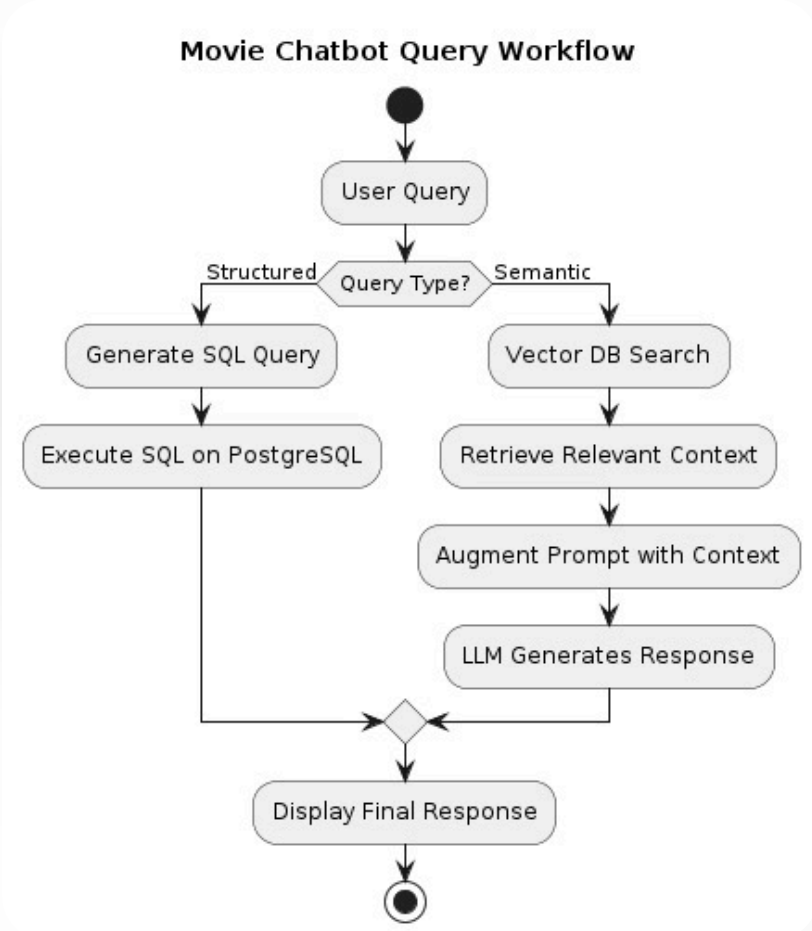
Similar movies have "close" vectors.

Outcome: A powerful vector DB (`faiss_movie_index`) for semantic Q&A.

RAG Workflow

How the Pipeline Works:

- Import packages (LangChain, OpenAI, FAISS, Streamlit, psycopg2, SQLAlchemy).
- Load pre-built FAISS index and GPT-3.5-Turbo.
- Classify user query (Structured vs Semantic).
- Structured path: Generate SQL → query Postgres → return data.
- Semantic path: Similarity search in FAISS → augment prompt → LLM generates answer.
- `answer_user_query()` orchestrates the whole process.



Technical Stack & Future Scope

Technical Stack

- Databases: MongoDB, PostgreSQL
- Data Engineering: Python, Pandas, psycopg2
- NLP & LLMs: LangChain, OpenAI, FAISS
- UI: Streamlit

Skills Learned

- Data migration
- Relational modeling
- RAG pipeline
- Prompt engineering
- LLM integration

Future Scope

- Advanced recommendation models
- Multi-modal (images, trailers)
- Fine-tune embeddings for domain specificity

Thank You / Q&A