UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA

TEXT TO SQL

INTRODUÇÃO A BANCO DE DADOS

Prof. Leandro Batista de Almeida

Discente:
Enzo Westphal Tacla

CURITIBA
2025

1. Conexão com os Bancos de Dados

A aplicação permite fazer conexão com dois bancos de dados diferentes, MySQL Workbench e PostgreSQL. A conexão com o MySQL Workbench é feita utilizando a biblioteca *MySQLdb.* A função *connectMysql()* é responsável por estabelecer a conexão utilizando parâmetros, host, usuário, senha e nome do banco de dados, que foram pré-definidos. Para o PostgreSQL, utilizando a biblioteca *psycopg2* a conexão é feita pela função *connectPostgresql()* utilizando os mesmos parâmetros pré-definidos citados acima. No caso dos dois bancos de dados existe um *try catch* para imprimir uma mensagem em caso de erro.

Na interface o usuário pode escolher com qual banco de dados deseja se conectar, chamando a função de conexão apropriada, armazenando o objeto da conexão no estado da sessão, *st.session_state.connection*, para que outras operações possam ser realizadas sem ter que se reconectar.

2. Carregamento do schema

Após a conexão com o banco de dados de preferência ser executada, a aplicação precisa entender a estrutura do banco de dados para poder gerar consultas SQL mais precisas. Esse processo também é diferente para os dois bancos de dados.

Para o MySQL a função *getMysqlSchema* executa o comando *SHOW TABLES* para obter todas as tabelas do banco de dados. Em seguida o comando *SHOW CREATE TABLES* é executado, que retorna a declaração *CREATE TABLE*, que auxilia para um Modelo de Linguagem Grande (LLM) entender a estrutura da tabela, incluindo tipos de dados e chaves primárias, por exemplo.

Já para o PostgreSQL não possui um comando como *SHOW CREATE TABLE.* Portanto a função *getPostgresqlSchema* consulta o *information_schema.tables* para listar as tabelas do schema. Para cada tabela uma consulta SQL é feita para conseguir detalhes das colunas, como: nome, tipo de dado, chave primária, etc.

O schema formatado é armazenado em *st.session_state.schema* e pode ser visualizado na interface dentro de um *st.expander*.

3. Interface de Consulta

A interface de consulta foi feita utilizando a biblioteca *Streamlit*. Ela é composta por 2 áreas principais, barra lateral e área de consulta. A barra lateral

contém os botões para o banco de dados de preferência. Além disso, também é exibido o status da conexão e um botão para visualizar o schema do banco de dados selecionado. Já na área de consulta, após a conexão com o banco de dados é exibido um campo para que o usuário digite sua pergunta em linguagem natural e um botão para gerar a resposta que inicia a sequência de geração de SQL e resposta.

4. Conversão da Linguagem Natural para SQL

Essa parte do projeto é realizada na função *generateSQL*, num processo que envolve 3 etapas.

1. **Construção do prompt:** um prompt de texto detalhado é criado e contém 3 partes: Instrução, Schema e Pergunta do usuário. Instrução: uma ordem para o LLM gerar um consulta SQL com base nas informações fornecidas, é instruído a responder apenas com o código SQL. Schema: a string de schema armazenada em *st.session_state.schema*, é inserida no prompt, dando ao modelo o contexto necessário sobre as tabelas, colunas e seus tipos de dados. E por fim, a pergunta do usuário em linguagem natural é adicionada ao final do prompt.
2. **Comunicação com o LLM:** o prompt é enviado para um modelo de linguagem, nesse caso o llama 3, modelo com 7 bilhões de parâmetro, através da biblioteca ollama, escolhido muito por conta de sua simplicidade de instalação e eficiência satisfatória na interpretação dos prompts. A API, ollama.chat, é configurada com temperatura 0.0, o que torna a saída do modelo mais determística e pontual, reduzindo a chance de alucinações.
3. **Resposta:** a resposta da API contém o código SQL gerado, que é retornado pela função.

Após a geração do código SQL a consulta é executada e os resultados são apresentados, novamente em um processo de 3 etapas

1. **Execução da consulta:** A biblioteca *pandas* é utilizada para executar a consulta SQL gerada no banco de dados conectado. A função *pd.read_sql_query(sql, st.session_state.connection)* envia a query para o banco e retorna os resultados diretamente em um *DataFrame*.
2. **Geração da Resposta em Texto:** o *DataFrame* resultante junto com a pergunta original são enviados para o LLM novamente, por meio da função *GenerateTextResponse.* Então o prompt pede para o modelo para gerar uma resposta em linguagem natural, concisa e amigável.
3. **Exibição na Interface:** A resposta em texto é exibida para o usuário, já o SQL gerado e os dados retornados são expostos em uma tabela abaixo.

5. Como rodar o programa

O programa original foi feito no Linux Ubuntu.

Ao instalar o arquivo src, que contém os códigos. No diretório raiz, deve ser criado um ambiente virtual, isso pode ser feito utilizando os seguintes comandos:
python3 -m venv venv
source venv/bin/activate

Após isso, as bibliotecas devem ser instaladas. Bibliotecas utilizadas: MySQLdb, psycopg2, streamlit, pandas, também foi usado o modelo llama 3 de 7 bilhões de parâmetros.

Para mudar o banco de dados basta mudar as variáveis nos arquivos MySQLConnection.py e PGConnection.py. Exemplo:

DB_HOST_MYSQL = "localhost"
DB_USER_MYSQL = "seu_usuario_mysql"
DB_PASSWORD_MYSQL = "sua_senha_mysql"
DB_NAME_MYSQL = "University"

Por último, dentro do ambiente virtual basta digitar o comando: streamlit run Interface.py

LInk para o repositório do github: https://github.com/enzowtacla/Text-to-SQL