Interpretador

Grupo No. 8

David Carvalho Nº27973
Diogo Marques Nº.27972
Gonçalo Vidal Nº27984

Licenciatura em Engenharia Sistemas Informáticos

2ºano

Barcelos |, 2024

Índice de Figuras

Não foi encontrada nenhuma entrada do índice de ilustrações.

Índice

1. Introdução

Este relatório descreve o desenvolvimento do projeto "CQL Interpreter", realizado no âmbito da unidade curricular de Processamento de Linguagens da Licenciatura em Engenharia de Sistemas Informáticos, no Instituto Politécnico do Cávado e do Ave, durante o ano letivo 2024/2025.

O objetivo principal do projeto foi implementar um interpretador para a linguagem CQL (Comma Query Language), projetada para operar sobre ficheiros CSV. A aplicação foi desenvolvida em Python, utilizando a biblioteca PLY (Python Lex-Yacc), permitindo aos utilizadores realizar operações de consulta e manipulação de dados em ficheiros CSV através de comandos.

2. Desenvolvimento

O desenvolvimento da aplicação foi realizado em **Python**, utilizando a biblioteca **PLY** (Python Lex-Yacc).

2.1. Estrutura do projeto

- **lexer.py**: define o analisador léxico da linguagem CQL;
- **parser.py**: define a gramática sintática da linguagem e as ações semânticas associadas;
- **interpreter.py**: contém as funções que executam os comandos definidos pela linguagem;
- **filesCSV.py**: responsável pelo carregamento e exportação das tabelas;
- **cql_interpreter.py:** ponto de entrada da aplicação que liga os componentes lexer, parser e interpreter;
- **/test:** pasta com os testes da aplicação;
- **/data:** pasta com os ficheiros csv;
- **/input:** pasta com os ficheiros de entrada da aplicação.
- **/output:** pasta com os ficheiros de saída da aplicação;

2.2. Analisador Léxico (lexer.py)

O ficheiro ***lexer.py*** define os tokens da linguagem, como palavras-chave (SELECT, FROM, WHERE, IMPORT, etc.), operadores (=, <>, <, >, <=, >=), símbolos de pontuação (;, ,, (,)), identificadores e literais (strings e números).

Através da biblioteca PLY, foram criadas expressões regulares para cada tipo de token, permitindo identificar corretamente os elementos da linguagem.

2.3. Analisador Sintático (parser.py)

No ***parser.py***, é definida a gramática da linguagem CQL, em conjunto com as ações semânticas associadas a cada regra. A análise sintática constrói diretamente estruturas representativas das instruções, sob a forma de uma árvore de sintaxe abstrata (AST).

As regras suportam a totalidade dos comandos descritos no enunciado, incluindo:

- **Configuração e gestão de tabelas:**
  - IMPORT TABLE nome FROM "ficheiro.csv";
  - EXPORT TABLE nome AS "ficheiro.csv";
  - RENAME TABLE nome1 nome2;
  - DISCARD TABLE nome;
  - PRINT TABLE nome;
- **Consultas:**
  - SELECT * FROM tabela;
  - SELECT col FROM tabela;
  - SELECT * FROM tabela WHERE col > 5 AND col2 = 2;

  Para todas as consultas é possível limitar o numero de resultados utilizando “LIMIT X” no final do comando

- **Criação de tabelas:**
  - CREATE TABLE nome SELECT * FROM tabela;
  - CREATE TABLE nome SELECT * FROM tabela WHERE coluna > 1;
  - CREATE TABLE nome FROM tabela1 JOIN tabela2 USING(coluna)
- **Procedimentos:**
  - PROCEDURE nome DO ... END;
  - CALL nome;

As instruções reconhecidas são convertidas e depois processados pelo interpretador.

2.4. Interpretador (interpreter.py)

O interpreter.py é responsável pela execução das ações descritas na AST. Cada tipo de instrução tem uma função correspondente que interage com os dados em memória.

As operações implementadas incluem:

- Descartar, renomear e imprimir tabelas em memoria.
- Seleção de colunas específicas ou todas de uma tabela em memoria, com a possibilidade de limitar os resultados;
- Criação de novas tabelas em memória a partir de queries;
- Criação e execução de procedimentos.

A estrutura de dados usada para armazenar as tabelas e os procedimentos e foi um dicionário, em que cada tabela e procedimento é identificada por uma chave que é criada a partir do nome da tabela ou pelo nome do procedimento.

2.5. Gestor dos ficheiros CSV (filesCSV.py)

O módulo **filesCSV.py** é responsável pela importação, exportação de ficheiros CSV, funcionando como um interface entre os dados em ficheiro e as estruturas de dados em memória utilizadas pela linguagem aplicação.

- Importar tabelas: No processo de importação, a aplicação lê o conteúdo de um ficheiro CSV e guarda o conteúdo num array. Durante esta leitura, o sistema ignora linhas que começam por #, tratadas como comentários, e interpreta corretamente os campos delimitados por aspas, permitindo, por exemplo, que valores contendo vírgulas internas sejam lidos como uma única célula.
- Exportar tabelas: No processo de exportação, os dados que se encontram em memória são escritos de volta num ficheiro CSV.

2.6. Ponto de entrada da aplicação (interpreter_cql.py)

O ficheiro interpreter_cql.py é o ponto de entrada da aplicação. Permite dois modos de operação:

- **Modo por ficheiro:** o utilizador fornece um ficheiro .fca com comandos CQL, que são lidos e processados sequencialmente;
- **Modo interativo:** o utilizador escreve comandos diretamente no terminal, que são processados imediatamente.

Em ambos os modos a aplicação carrega todos os procedimentos presentes no ficheiro *"procedures.fca"*.

Este módulo inicializa o lexer e parser, e o interpretador, e controla a entrada/saída dos dados. Os ficheiros CSV são lidos da pasta data/, os comandos são carregados da pasta input/ e os resultados de exportações são escritos na pasta output/.

3. Bibliografia

- https://docs.nestjs.com/
- https://nextjs.org/docs
- https://www.typescriptlang.org/docs/
- https://react.dev/
- https://pactumjs.github.io/guides/integration-testing.html

4. Conclusão

O projeto DoATask apresenta-se como uma solução inovadora e eficaz para fomentar o voluntariado, respondendo a uma necessidade real de centralização e acessibilidade das oportunidades de participação cívica. Através de uma plataforma intuitiva e bem estruturada, foram integradas funcionalidades que permitem não só a criação e gestão de comunidades e tarefas, como também a valorização do esforço dos voluntários por meio de um sistema de recompensas com pontos e moedas.

A especificação do sistema, suportada por uma arquitetura robusta e baseada em tecnologias modernas como Next.js, NestJS e PostgreSQL, garante escalabilidade, segurança e uma boa experiência de utilizador. Os diagramas apresentados demonstram de forma clara a lógica de funcionamento da aplicação e a interação entre os seus componentes.

Com o DoATask, torna-se possível não apenas facilitar o acesso ao voluntariado, mas também dinamizar e fortalecer os laços comunitários, promovendo a solidariedade através de uma abordagem tecnológica e motivadora. Este relatório reflete o empenho da equipa no desenvolvimento de uma aplicação funcional, relevante e com potencial de impacto social positivo.