Relatório de descrição da ferramenta de gerenciamento

1 Introdução

Com base na proposta postada na página da disciplina , foi requisitada a criação de um modelo relacional de banco de dados cujo qual serviria para implementação de um banco de armazenamento de referências científicas.

Os requisitos previam a criação das tabelas Coleções, Artigo, Autor e Citações cujas decisões de design dos atributos e relações serão apresentadas neste relatório ao longo das próximas seções.

2 Resolução dos requisitos

Nesta seção serão discutidos os métodos utilizados para solucionar os requisitos do trabalho, sendo estes divididos basicamente em quatro.

2.1 Escolha de diretório

Com a iniciativa de resolver os problemas de compatibilidade com o tratamento de caminhos entre sistemas operacionais, para facilitar a execução da aplicação foi implementado uma interface para seleção do diretório de csv's, para realizar esta seleção foi utilizada a biblioteca gráfica de interfaces simples chamada *TKInter*.

A seleção é feita através de uma função inclusa na biblioteca gráfica chamada *filedialog.askdirectory()* que atua quando o botão *Browse Folder* é acionado.

É possível ver o resultado da interface na figura 1. A maneira de realizar sua instalação está descrita no apêndice B.

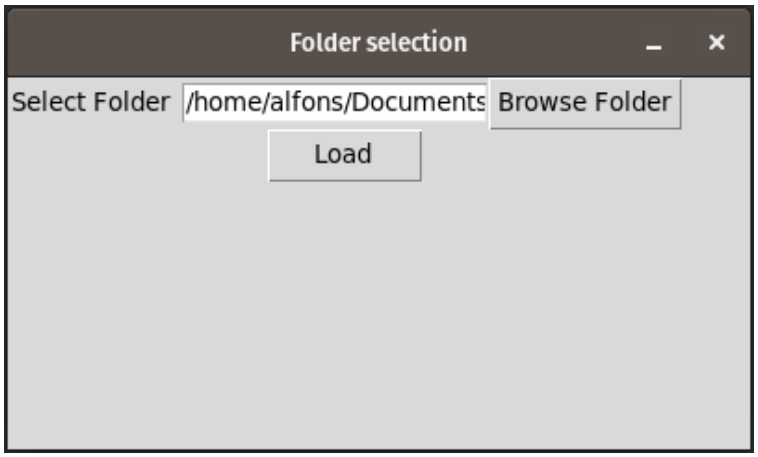

Figura 1: Resultado da Interface gráfica *Folder selection*

Então a partir da seleção o usuário aciona o botão de *Load* que realiza a chamada para a função de carregamento dos arquivos descrita na seção 2.2.

2.2 Carregamento dos arquivos .csv em memória

Para o carregamento dos arquivos, se optou por utilizar uma classe *Table* composta por três elementos:

- tableName: O nome da tabela, que é apenas o nome do arquivo com a exclusão do .csv no final;
- collumnNames: um dicionário que contém o nome de cada coluna, mapeado para a posição desta coluna no *array* e, consequentemente, na tabela de dados. Os nomes são obtidos pela leitura da primeira linha do arquivo;
- tableContent: uma matriz com todos os dados restantes do arquivo, separados assim como no arquivo fonte.

Apenas arquivos com terminação *.csv* são abertos, com leitura feita linha a linha, separando os atributos por meio da presença de uma vírgula (','') e nada mais. Para o funcionamento do algoritmo, o arquivo .csv deve seguir sua descrição literalmente: valores separados por **vírgula**.

Todos os valores são convertidos para seus devidos tipos por meio de tentativa e erro: caso se consiga convertê-lo, assim o faz, caso contrário, o valor permanece na forma de *string*.

2.3 Parseamento da *query* em forma de *string*

A *query* é recebida do terminal apenas como uma *string*. Para a utilização dela, se torna necessário separar esta linha de dados em palavras em um *array*. A separação foi feita com base em espaços na *query*, ou seja, o caractere que representa a diferença de uma palavra com outra é o de "espaço".

Portanto, a sintaxe da ferramenta criada tem como base o "espaço". Não se utiliza vírgulas, tampouco aspas, apenas espaços.

Após a separação em palavras, se utilizou uma máquina de estados para separar o *array* nas cláusulas *select*, *from*, *where* e *order by*. Esta funciona percorrendo o *array* e, sempre que encontra uma destas palavras, muda o estado do *loop* para o seu equivalente e adiciona todas as palavras seguintes em um *array* específico para a funcionalidade.

Todas as demais separações são feitas nas funções de cada operação, apenas caso esta operação não seja vazia.

2.4 Execução da query

A execução da *query*, neste trabalho, pôde ser separada em quatro condições base, sendo outras ações apenas uniões destas. A implementação será descrita abaixo:

2.4.1 Escolha das colunas a serem mostradas com *select*

Como forma de mapeamento da coluna à uma posição na matriz de dados, se optou por utilizar o dicionário de nomes das colunas da estrutura *Table*.

Dado um nome de coluna, este dicionário retorna a posição dela na tabela, assim tornando possível acessar apenas as colunas desejadas sem a utilização de *loops* para encontrar *strings* iguais às digitadas.

No caso de seleção por completo ('*'), o programa retorna todas as colunas.

2.4.2 Seleção da tabela (*from*)

Para selecionar a tabela, bastou parsear a *query* a partir da palavra *from*, adicionando as palavras seguintes em um *array* de nome *selectFrom*.

Com isto em mãos, basta percorrer o *array* de tabelas até encontrar a tabela com nome igual ao digitado, para então utilizá-la futuramente.

2.4.3 Filtragem com *where*

Esta etapa foi, sem dúvidas, a mais trabalhosa de todo o projeto. Pelo fato da entrada do terminal ser apenas uma *strig*, uma forma de mapear caracteres à uma sintaxe python haveria de ser feita.

Como solução à este problema, se optou por utilizar um dicionário em python. Este mapeando cada caractere à uma função de comparação equivalente, assim automatizando grande parte da aplicação e economizando processamento e linhas de código, por descartar a necessidade de comparações do tipo *if - else* para achar qual ação fazer.

O resultado foi um dicionário da forma demonstrada na figura 2.

```
self.operatorsDict = {}
self.operatorsDict["="]   = lambda a, b: a == b
self.operatorsDict["!="]  = lambda a, b: a != b
self.operatorsDict["<"]   = lambda a, b: a < b
self.operatorsDict[">"]   = lambda a, b: a > b
self.operatorsDict[">="]  = lambda a, b: a >= b
self.operatorsDict["<="]  = lambda a, b: a <= b
self.operatorsDict["&"]   = lambda a, b: a & b
self.operatorsDict["|"]   = lambda a, b: a | b
self.operatorsDict["and"] = lambda a, b: a and b
self.operatorsDict["or"]  = lambda a, b: a or b
```

Figura 2: Dicionário para mapeamento de operadores lógicos.

Com isto em mãos, apenas se torna necessário separar corretamente a *query* na cláusula *where* e tratar as condições para quando a comparação é entre colunas ou entre coluna e um valor. Para o caso de valores, estes também tiveram de ser convertidos para serem comparados.

2.4.4 Ordenação com *order by*

Para a ordenação do resultado a partir de um atributo da relação, se utilizou o algoritmo de ordenação *merge-sort*. Para seu funcionamento, basta passar a tabela que se deseja ordenar e com qual atributo se deseja fazer as comparações, para a função. O retorno é uma matriz ordenada de forma crescente. Tratativas adicionais podem ser feitas a partir desse ponto.

Com esta implementação, para tabelas relativamente pequenas ao menos, o tempo de resposta da ordenação se demonstrou rápido.

2.4.5 União das funções

Como as saídas e entradas das funções descritas anteriormente são da forma de uma matriz de dados, operações mais complexas são epenas uniões destas. Deste modo, uma *query* com todas as operações (*select*, *from*, *where*, *order by*) apenas dá como entrada de uma função, a saída da logicamente anterior.

Após a saída final, só se precisou imprimir as informações em tela de forma legível, o que foi feito com sucesso, assim finalizando o trabalho.

3 Sintaxe da query

A sintaxe escolhida para a ferramenta foi simples: palavras separadas por **espaço** e fim da *query* com **;**.

Qualquer escrita fora desse padrão resultará em uma mensagem de erro, estas quais tratadas em pontos do código descritos abaixo.

3.1 Tratativa de erros

Apesar de não ser requisito do trabalho, tratativas para erros tiveram de ser implementadas para que a experiência se tornasse mais fluida. Diversas tratativas foram feitas para erros diversos, sendo a maioria presente na função *isQuerySintaxOk*.

Outras tratativas só eram possíveis após processamento mais aprofundado dos dados, e, portanto, só foram feitas em pontos mais distantes

A Como executar a aplicação

A execução é simples: basta abrir o arquivo adendo deste relatório, posicionar os arquivos *.csv* das tabelas, com sintaxe correta, dentro de uma pasta de preferência, e, em seu terminal, executar o comando:

```
python3 main.py
```

Isto abrirá a interface de escolha de diretório para carregamento. O escolhido pode ser qualquer um que contenha um arquivo *.csv*, por motivos descritos na seção 2.2. Basta, então, pressionar *Ok* e logo após o botão de *Load*. Caso seja necessário o usuário precisa reiniciar o programa para carregar outro diretório de csv's.

Isto carregará as tabelas em memória e permitirá que o usuário comece sua experiência em uma ferramenta de banco de dados no terminal. As *queries* aceitas são descritas na seção 3, e para terminar o programa, basta digitar *quit;*.

B Instalação da interface gráfica

A interface gráfica *TKInter* vem instalada por padrão em alguns kits de desenvolvimento dos sistemas Linux para python, caso não a tenha, a instalação pode ser realizada de duas formas:

Via gerenciador de pacotes:

```
sudo apt install python3-tk
```

Via comando *pip*:

```
pip install tk
```