

**Laboratórios de Informática III**

Projeto C – Sistema de Gestão de Recomendações

Projeto Realizado por:

Bernardo Saraiva - a93189

José Gonçalves - a93204

Rui Moreira - a93232

1. **Apresentação e conceitos gerais sobre o projeto**

Este projeto surgiu no no âmbito da cadeira de Laboratórios de Informática III, e tem como objetivo fundamental ajudar na consolidação experimental dos conhecimentos teóricos e práticos adquiridos anteriormente, através de um Sistema de Gestão de Recomendações.

Este trabalho surge como um desafio, o qual implica uma grande organização e utilização de conhecimentos de programação com o objetivo de conceber um programa com um bom desempenho, uma vez que é aplicado numa manipulação de ficheiros com grande volume de dados e grande complexidade algorítmica e estrutural.

Para tal, foram fornecidos três ficheiros, um de negócios, um de utilizadores e outro de reviews, que incluem toda a informação que é tratada neste projeto. É também importante salientar que foram utilizadas constantemente as funções da biblioteca do GLib.

Neste projeto foram introduzidas algumas noções importantes, sendo algumas das quais:

* **Modularidade e encapsulamento:** O código é dividido na parte pública (onde as funções são apresentadas nos *header files* e podem ser acedidas pelos outros ficheiros) e na parte privada (tudo o que é definido no ficheiro é apenas usado internamente). O código é dividido em partes, de forma isolada, para que o software seja o mais flexível possível e cada modificação implique as menores complicações para o resto do trabalho.
* **Criação de código reutilizável:** Neste projeto foi introduzida a noção de reutilização de código, o que, em conjunto com as noções anteriores, permite, por exemplo, chamar funções a várias *queries*.
* **Escolha otimizada de estruturas de dados e reutilização:** Uma vez que este projeto envolve a utilização de grandes volumes de dados, é fundamental a otimização do programa através das melhores escolhas a nível de estruturas de dados e reutilização de código.

1. **Apresentação dos Módulos e das funções utilizadas**
2. **Catálogo\_users, Catálogo\_business e Catálogo\_reviews**

Após uma leitura inicial do Enunciado, o nosso grupo decidiu começar por criar os módulos de catálogos dos três ficheiros, sendo estes o ***catalogo\_reviews, catalogo\_business e catalogo\_users***.

Os módulos referidos apresentam uma estrutura semelhante ao nível das funções das suas capacidades, pelo que todos contêm:

* Struct com as várias informações que cada um dos ficheiros fornece.
* Struct que compõe um catálogo dessas informações (um catálogo corresponde a um conjunto das structs referidas acima, através da utilização das **Hash Tables do glib**). É importante deixar claro que um catálogo é um conjunto de negócios/reviews/utilizadores, uma vez que esta designação será utilizada frequentemente ao logo do projeto e do relatório. Esta escolha foi deliberada pelos elementos do grupo, uma vez que nos pareceu a mais adequada e mais funcional para alcançar o resultado pretendido. Nas hash tables, utilizou-se como **key** para a dos businesses o business id, o user id para a hash table dos users e o review id para a hash table das reviews.
* Função de iniciar um catálogo, que aloca espaço para um catálogo e cria uma nova Hash Table (sendo esta Hash Table de reviews, negócios ou utilizadores).
* Função de free para um catálogo, que desaloca a memória previamente reservada para o catálogo.
* Função de load de um catálogo, que carrega o catálogo pretendido.
* Função que remove um negócio/review/user da Hash Table e função que remove uma chave (key) da Hash table.
* Função de free para um negócio/utilizador/review, que liberta o espaço alocado para as componentes desta estrutura.
* Função que cria o catálogo de negócios/users/reviews a partir da leitura de um ficheiro.
* Funções de get e set para as componentes da struct de negócios/users/reviews, para que seja possível conservar o encapsulamento e a modularidade.

1. **Table**

Este módulo não foi totalmente desenvolvido de uma vez só, pelo que foi sendo alterado e melhorado em conjunto com o módulo falado no ponto seguinte (sgr) e de acordo com as necessidades.

Neste módulo foi implementado um novo tipo de dados chamado TABLE, que é utilizado em todas as queries, sendo que o valor de retorno das queries é sempre deste tipo. Deste modo, a informação é apresentada numa tabela, sendo esta composta por um conjunto de linhas, sendo que criamos o tipo LINHA para esse efeito.

Este módulo contém as seguintes funções:

* Struct que define uma linha, e que é chamada de LINHA.
* Struct que define uma tabela, chamada de TABLE.
* Função que calcula o tamanho de uma linha.
* Função que inicializa uma TABLE (aloca a memoria necessária).
* Função que adiciona uma linha numa determinada TABLE, realoca a memória necessária para tal.
* Função que adiciona um cabeçalho numa TABLE.
* Função que cria uma LINHA (aloca a respetiva memória).
* Funções de get para LINHA.
* Funções de get para TABLE.

1. **SGR**

O sgr é o módulo no qual estão todas as funções das queries. É também um tipo de dados no qual é armazenada toda a informação relativa aos catálogos, garantindo a modularidade e encapsulamento. Este módulo possibilita também inicializar e destruir estruturas do tipo SGR.

Deste modo, passamos então a uma breve apresentação das funcionalidades das queries, das estratégias e das otimizações implementadas.

* Query 1 – Esta query denomina-se “load\_sgr” e é o núcleo de todo o projeto, já que, carrega para a memória toda a base de dados para que possa ser mais facilmente analisada. A estratégia usada para fazer este carregamento consiste em dividir a informação nos 3 catálogos referidos no enunciado do projeto. Cada um destes catálogos contém uma hashtable preenchida com uma estrutura de dados personalizada para cada tipo de dados a tratar. Dados estes que são localizáveis através do campo id presente na sua estrutura, já que, desempenham o papel de key.
* Query 2 - A query business\_started\_by\_letter tem como objetivo determinar a lista, bem como a quantidade de negócios existentes em que o nome começa por determinada letra. No que toca à implementação esta seleção e contagem efetua-se através da progressão do catálogo dos businesses através da função foreach disponiblizada pelo glib. A cada iteração deste ciclo, e, portanto, a cada business name é verificado se este começa com a letra selecionada e, em caso afirmativo, é diretamente adicionado à TABLE e o contador aumenta uma unidade.
* Query 3 - A querry business\_info tem como objetivo determinar toda a informação associada a um business. Informação esta que se encontra dividida em 2 catálogos: businesses(nome, cidade, estado e business\_id) e reviews(stars, numero total de reviews). Para a apresentação deste resultado foi necessário dividir a querry em 2 etapas. A primeira consiste em procurar no catálogo businesses o elemento com o business\_id recebido como argumento e obter os primeiros campos da informação necessária. De seguida, é necessário percorrer todos os reviews para preencher o resto dos dados necessários. Assim por cada value da hashtable é verificado se o review corresponde ao business correto, caso isto aconteça o contador de reviews é aumentado em uma unidade, bem como as stars também são somadas para que no fim seja possível calcular a média. Por fim basta adicionar todos os dados recolhidos à TABLE.
* Query 4 – Esta query é chamada de businesses\_reviewed e tem como objetivo determinar a lista dos negócios aos quais um determinado utilizador fez review. Para tal, foi utilizada uma struct idealizada para esta query, na qual é armazenada toda a informação necessária ao longo da query e que necessita de ser utilizada ao longo das funções. Esta função implica o cruzamento de dados de dois catálogos, pelo que, inicialmente procura no catálogo dos businesses cada um dos business id’s que o user com o user id (que é passado como argumento) fez review e guarda-se essa informação numa lista ligada singular, sendo esta pertencente à biblioteca glib. Seguidamente, procura-se no catálogo cada um dos businesses com o business ID presente na lista ligada anterior e colocamos na nova lista ligada. Finalmente, percorremos ambas as listas ligadas (que têm o mesmo tamanho) e imprimimos na tabela duas colunas, uma com o id e outra com o nome. Escolheu-se este modo de percorrer as listas porque apenas percorre as listas uma vez, pelo que não compromete a eficácia da query (percorre em tempo linear).
* Query 5 – Esta query determina a lista dos negócios com um número de stars superior ou igual ao que é passado como argumento numa dada cidade. Para tal, desenvolveu-se uma função que devolve uma Singly Linked List com os business id’s dos negócios com tantas ou mais estrelas. Seguidamente, utilizou-se uma função que procura na hash table o business id que está na lista ligada da função anterior (porque o business id é a key) e adiciona na tabela juntamente com o nome do negócio. Uma das otimizações realizadas nesta query foi ao utilizar a função prepend em vez de append, sendo que para o append é necessário percorrer a lista na sua totalidade (tempo linear de cada vez que adiciona), o que causa sérios problemas de performance. Por sua vez, o prepend adiciona no início da lista, sendo assim em tempo constante.
* Query 6 – Para esta query, começamos por percorrer a hash table dos businesses e criar uma hash table de structs (struct bus\_aux), que obtém as informações acerca da média das stars das reviews daquele negócio, em que a key é o business id. Em seguida, uma vez que apenas é possível aceder ao nome do negócio através do ficheiro dos businesses, a outra função cria uma hash table na qual cada value é uma lista ligada de structs e junta à struct da função anterior o nome do negócio.
* Query 7 – Para esta query, começamos por percorrer o catálogo dos reviews e criamos uma HashTable nova em que cada chave é o userID e para cada user temos a lista dos businesses aos quais deu review. De seguida percorremos esta HashTable e retiramos todos os users só com um business reviewed. Depois percorremos, esta hashtable e, para cada user, percorremos a sua lista de businesses e vamos verificar se tem businesses de estados diferentes. Se sim, adicionamos à Table.
* Query 8 - Começamos por percorrer o catálogo dos reviews e criamos uma HashTable nova que cada chave é o userID e para cada user temos a lista dos businesses aos quais deu review. De seguida percorremos esta HashTable e retiramos todos os users só com um business reviewed. Depois, percorremos esta hashtable e para cada user, percorremos a sua lista de businesses e vamos verificar se tem businesses de estados diferentes. Se sim adicionamos à Table.
* Query 9 - Esta query, dada uma palavra, determina a lista de reviewID que a referem no campo text. Para isto, percorremos o catálogo dos reviews e verificamos se o campo text contém a palavra dada. Em caso afirmativo adicionamos à tabela.

1. **View**

O módulo view incorpora todo o agregado de funções que permitem consultar o estado de funções ao logo da sua execução, quer seja para mostrar os resultados de uma função, para descobrir erros ou fazer debugging.

Estas funções aplicam-se a tabelas, linhas, structs de businesses/reviews/users e hash tables.

1. **Interpretador**

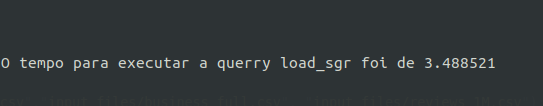
O módulo do interpretador contém todas as funções que interagem com o utilizador e chamam as funções criadas no módulo sgr, ou seja, para executar as queries,

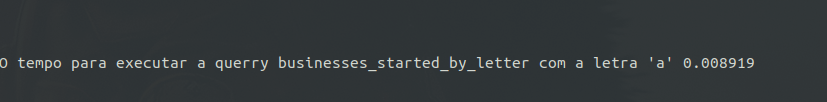
Foram desenvolvidas para este modulo algumas funções, sendo as mais importantes:

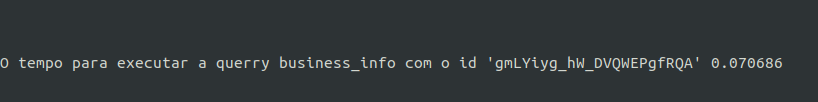
* Uma função que faz o parsing do comando recebido.
* Uma função que conta o número de palavras do comando recebido.
* Função que conta as palavras existentes numa string separadas por um determinado caracter.
* Uma função que calcula a posição de um delimitador.
* Função que converte uma string para um conjunto de palavras.
* Função que devolve o argumento x de um comando passado.
* Uma que interpreta o comando, executando a query pedida com os argumentos passados.
* Por fim, a função final de interpretador.

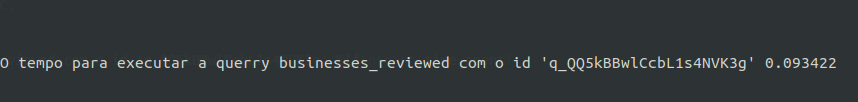
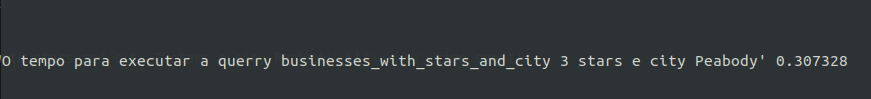
1. **Resultados**

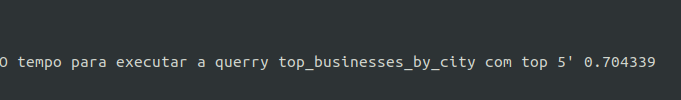
Em seguida apresentam-se alguns dos resultados obtidos e os respetivos tempos de execução de cada uma das queries.

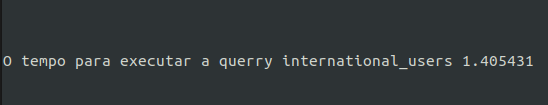


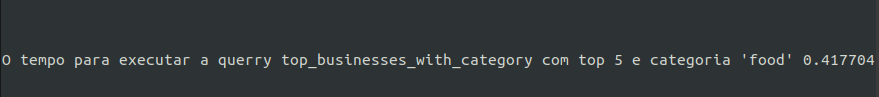
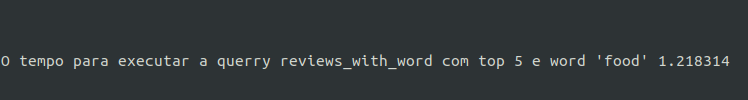












1. **Conclusão**

Considera-se que este projeto conseguiu concretizar o objetivo de manipular e tratar grandes volumes de informação, através de um Sistema de Gestão de Recomendações, e com o devido uso dos conceitos de modularidade e encapsulamento com devida correção.

As maiores dificuldades deste projeto passaram pelo funcionamento da biblioteca do glib e as suas funcionalidades, assim como a organização e tratamento do grande volume de dados para que cada query pudesse ser o mais otimizada possível.