Departamento de Informática
Escola de Engenharia
Universidade do Minho

Relatório Fase 1

Laboratórios de Informática III

Licenciatura em Engenharia Informática

2022/2023

Grupo 53

Diana Teixeira - a97516

Flávia Araújo - a96587

Marta Ribeiro - a95408

22/11/2022

Índice

Introdução

No âmbito da unidade curricular de “Laboratórios de Informática III”, foi proposto o desenvolvimento de um sistema de organização de dados na área de viagens, sendo os dados dessas, bem como os dados dos condutores e utilizadores, introduzidos em estruturas de dados e colocados posteriormente em catálogos e organizados de maneira otimizada para permitir um acesso mais rápido e resultados mais eficientes. Para auxílio da estruturação de dados, utilizámos estruturas pertencentes à biblioteca GLib2 da GNOME para a manipulação das coleções de dados, bem como estruturas da nossa própria autoria.

Nesta primeira fase, o nosso maior desafio encontrou-se no tipo de estrutura que iríamos usar ao longo do projeto – a nossa tentativa inicial foi através de *arrays*, porém isso demonstrou-se ineficiente, como será possível ler nas páginas adiante. Posto isto, decidimos utilizar *HashTables*, apesar de ainda nos encontrarmos em aberto para recorrer a *Balanced Binary Trees* nas situações futuras em que isso seja mais oportuno.

Estruturas de dados utilizadas

Como forma de armazenar a informação proveniente dos ficheiros .csv, optamos inicialmente por utilizar estruturas de *arrays*. Isto porque são estruturas de dados que ocupam menos memória; só teríamos necessidade de fazer inserções nos *arrays* na fase inicial (para preencher com os dados); e existiria a possibilidade de fazer *lookups*.

No entanto, após algumas dificuldades, e como vimos que estávamos a ficar num impasse, recorremos à *glib2*, passando assim à utilização das *HashTables* para manipular as coleções de dados. Foram utilizadas tanto para armazenar a informação dos *users,* como dos *drivers* e das *rides*. Temos noção de que é algo que pode custar mais no tempo de execução, e, por isso, é um dos pontos que procuramos vir a melhorar para a segunda fase do projeto.

Uma das opções poderá ser recorrer às árvores binárias em alguns cenários, por exemplo, em momentos onde seria necessário procurar informação por data.

Desenvolvimento da aplicação

	Hash Table	Arrays
Tempo de execução	**Maior**	**Menor**
Facilitam a procura?	**Não**	**Sim (são ordenados)**
Praticidade para lookups	**Sim**	**Não**
Possuimos as chaves	**Sim**	**Não**

A utilização de *HashTables* foi a técnica adotada nesta situação, tendo em conta percalços que surgiram com *Arrays*, porém procuramos mudar futuramente para conseguir melhorar o tempo de execução, pois o das *HashTables* não é o melhor possível, mesmo com todas as suas vantagens.

Modularidade e encapsulamento

Como forma de modularmos o nosso código, separamos os ficheiros em ficheiros *.c* e *.h*.

Nos ficheiros *.c*, temos o *main.c* que é o programa principal que utiliza os restantes módulos, e é o único que tem acesso a toda a informação e todos os *includes* dos *header files* e librarias, como por exemplo, *string.h*. Os restantes (*array.c, buffer.c, catalog.c, date.c, drivers.c, queries.c, rides.c* e *useres.c*), têm no início apenas o include do *header file* correspondente e em alguns casos, as librarias que são necessárias.

Para cada ficheiro *.c*, temos o seu *header file (.h)* correspondente (excetuando o *main.c*), onde estão as assinaturas das funções dos *.c* e estruturas de dados; e também os *includes* necessários para aquele módulo.

Posto isto, garantimos, então, que seguimos as regras do encapsulamento, mantendo todos os dados e funções de cada módulo, desconhecidas e imutáveis para os módulos que não necessitam de lhes ter acesso; tornando os dados viáveis, uma vez que se fornecem apenas cópias nos casos em que estes são necessários.

Evitamos o uso de variáveis globais por ser uma abordagem perigosa para o programa.

Modos de execução

Para esta primeira fase, apenas era necessária a implementação do modo *batch*, que consiste em receber dois parâmetros: o primeiro com o caminho para a pasta onde estão os ficheiros de entrada; e o segundo com o caminho para o ficheiro que contém a lista de comandos para as *queries*, no caso, o ficheiro com o nome *commands.txt*. Neste, cada linha tem o número da *query* que se pretende fazer, com os parâmetros necessários; e após a execução de cada linha, o resultado de cada linha, é guardado num ficheiro *commandN_output.txt* (na pasta Resultados), onde N corresponde ao número da linha no ficheiro *commands.txt*.

Pensamos também já numa base para posteriormente a aplicação poder funcionar com o modo interativo, estando essa parte em comentários no código.

Implementação das *queries*

Para a Fase 1, pelo menos 1/3 das *queries* teria de estar funcional. Em seguida apresentamos as *queries* que escolhemos, e o raciocínio para a sua implementação.

Query 4

A *query* 4, que calcula o preço médio das viagens (sem considerar gorjetas) numa determinada cidade, está totalmente implementada.

Esta *query* é chamada através do comando *4 <city>* e apresenta apenas o *preco_medio* calculado como *double*, com 3 casas decimais.

Iterando pelas *rides*, aferindo o id do condutor; vai à *Hash* dos *drivers* encontrar o condutor correspondente, ver a classe do carro; e também a distância e cidade (*rides*).

Após isto, testa-se se a cidade na qual o condutor se deslocou naquela viagem, é igual à cidade passada como argumento na *query*. Se sim, existe um contador que é incrementado e o preço dessa viagem é calculado conforme a classe do carro (*Basic, Green, Premium*) e adicionado ao preço das viagens anteriores. Se não, continua a iterar pelas *rides*.

No final, para obter a média, é dividido o preço total de todas as viagens naquela cidade pelo contador.

Query 5

A *query* 5, que calcula o preço médio das viagens (sem considerar gorjetas) num determinado intervalo de tempo, está totalmente implementada.

Esta *query* é chamada através do comando *5 <data A> <data B>* e apresenta apenas o *preco_medio* calculado como *double*, com 3 casas decimais.

Iterando pelas *rides*, aferindo o id do condutor; vai à *Hash* dos *drivers* encontrar o condutor correspondente, ver a classe do carro; e também a distância e a data (*rides*).

Após isto, testa-se se a data na qual o condutor se deslocou naquela viagem, está compreendida entre as datas passadas como argumento na *query*. Se sim, existe um contador que é incrementado e o preço dessa viagem é calculado conforme a classe do carro (*Basic, Green, Premium*) e adicionado ao preço das viagens anteriores. Se não, continua a iterar pelas *rides*.

No final, para obter a média, é dividido o preço total de todas as viagens naquela cidade pelo contador.

Query 6

A *query 6*, que calcula a distância média das viagens, numa cidade, num determinado intervalo de tempo, está totalmente implementada.

Esta *query* é chamada através do comando *6 <city> <data A> <data B>* e apresenta apenas a *distancia_media* calculado como *double*, com 3 casas decimais.

Iterando pelas *rides,* são aferidas a data e a cidade.

Após isto, testa-se se a cidade na qual o condutor se deslocou naquela viagem, é igual à passada como argumento na *query;* e depois se a data daquela viagem está entre os valores passados também como argumento na *query*. Se sim, existe um contador que é incrementado e a distância dessa viagem é adicionada às distâncias das viagens anteriores. Se não, continua a iterar pelas *rides*.

No final, para obter a média, é dividida a distância total de todas as viagens naquela cidade entre aquelas datas, pelo contador.

Query 8

A *query 8*, que apresenta a lista das viagens nas quais o utilizador e o condutor são do género passado como parâmetro, e têm perfis com X ou mais anos; ordenando-a das contas mais antigas para as mais recentes, não está totalmente implementada, faltando a ordenação da informação.

Esta *query* é chamada através do comando *8 <gender> <X>* e apresenta o *id_condutor, nome_condutor, username_utilizador* e *nome_utilizador.*

Iterando pelas *rides,* é aferido o username do condutor e o id do utilizador; bem como depois os géneros de cada um e os anos de conta.

Após isto, testa-se se os géneros do utilizador e condutor são iguais ao género passado como argumento na *query*. Se sim, afere-se o nome do condutor, o nome do utilizador e o *username* do utilizador. No final, é escrito no ficheiro de *output* os id's nomes e *username* do condutor e utilizador. Se não, continua a iterar pelas *rides*.

Query 9

A *query 9*, que apresenta a lista das viagens nas quais o passageiro deu gorjeta, num intervalo de tempo, ordenando-a por ordem de distância percorrida, não está totalmente implementada, faltando a ordenação da informação.

Esta *query* é chamada através do comando *9 <data A> <data B>* e apresenta o *id_viagem, data_viagem, distancia, cidade* e *valor_gorjeta.*

Iterando pelas *rides,* é aferida a data da viagem.

Após isto, testa-se se a data daquela viagem está entre as datas passadas como argumento na *query*. Se sim, afere-se se foi dada uma gorjeta na viagem. No caso de haver gorjeta, é escrito no ficheiro de *output* o id, distância, cidade e a data. Se não, continua a iterar pelas *rides*.

As restantes *queries* (1,2,3 e 7), ainda não estão implementadas, sendo um dos objetivos para a segunda fase fazê-lo, assim como completar as que ainda não estão totalmente funcionais (8 e 9).

Conclusão e trabalho futuro

Após a realização desta primeira fase do projeto, sentimos que conseguimos atingir os objetivos que tínhamos em mente, apesar dos desafios que fomos encontrando ao longo do mesmo.

Para o trabalho futuro, pretendemos:

- Diminuir ao tempo de execução do programa, implementando estruturas de dados mais indicadas para o que nos é pedido;

- Melhorar o encapsulamento dos dados;

- Completar também as *queries* que não estão totalmente funcionais, e implementar as restantes;

- Pôr o programa a ser executado também com o modo Interativo, bem como o módulo de paginação para os resultados apresentados que são longos;

- Fazer uma análise do desempenho do nosso programa.