Laboratórios de Informática III

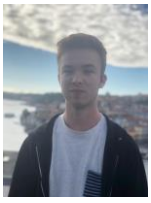
Relatório do Trabalho Prático - Fase 2

LEI - 2022/2023

Grupo 25

Diogo costa Ivo Ribeiro Martim Ribeiro

A95460 A96726 A96113



Universidade do Minho

Índice

1 Introdução

Na segunda e última fase do projeto, foi-nos solicitada a conclusão do trabalho, assim como as considerações relativas aos aspetos de desempenho da solução desenvolvida. Mais especificamente: a totalidade das queries funcionais, o modo de operação interativo, validações sobre o formato dos ficheiros de dados, suporte para ficheiros de entrada com uma ordem de grandeza superior e evolução dos aspetos relacionados com a modularidade, encapsulamento e qualidade do código.

Durante o relatório iremos discutir os resultados obtidos.

2 Arquitectura do projeto

O projeto foi organizado de forma a respeitar as boas práticas de modularidade e encapsulamento do código, de modo a melhorar a organização e a facilitar o eventual trabalho de manuntenção futuro. O código foi assim dividido em 4 partes: parsing dos dados, criação dos catálogos, interpretação dos comandos e resolução das queries.

A seccão de parsing lê os dados para a memória, verifica se os mesmos passam nas validações e, caso isto se verifique, guarda-os nos respetivos catálogos de cada ficheiro.

A secção de interpretação de comandos lê cada linha do ficheiro de comandos e chama a respetiva query com os dados disponibilizados nesse ficheiro e ainda os catálogos necessários para o funcionamento de uma determinada query.

A secção das queries recebe a informação passada pelo interpretador de comandos e executa o conjunto de funções necessárias para obter os resultados pretendidos pelas mesmas.

Todos estes módulos e funções encontram-se devidamente documentados, aumentando a organização e contribuindo para uma mais fácil interpretação do código.

3 Estrutura de dados

Estratégias

No início desta segunda fase optamos por continuar com o método que até agora estávamos a usar que era o de guardar toda a informação lida nos ficheiros de input em hastables e depois usar essa informação passando a informação das mesma para GLists a cada chamada de uma query, como era de especular este modelo estava a usar muita memória

mesmo e então chegamos a um ponto onde até mesmo com o dataset regular não conseguíamos fazer todas as querie uma única vez.

Então decidimos procurar outras alternativas e começamos por reestruturar as nossas estruturas de dados para ao invés de guardarem apontadores para todos os campos de informação, substituir os possíveis de substituir por variáveis menos custosas em termos de memória, tal como inteiros, caracteres,…, Apesar de melhorias não estávamos ainda a conseguir correr queries como as 7, 8 e 9 é mesmo a dos top n condutores e utilizadores era muito lenta é muito dispendiosa.

Então é com o objetivo de conseguir responder a todas as queries e facilitar a execução de queries como a dos top n decidimos recorrer a alocação de memória para arrays destas nossas estruturas. Estruturas essas que foram modificadas e acrescentada uma nova para a execução da query 7.

Com esta estratégia de guardar a informação das hashtables em listas de estruturas conseguimos poupar tanto em termos de memória como em tempo de execução e então já conseguimos passar nos testes para o dataset regular.

Apesar de conseguirmos realizar as queries para o dataset regular, com este método estávamos a usar muita memória e então não estávamos a conseguir executar o large dataset por estarmos a alocar mais memória que o permitido.

Para evitar este problema decidimos então adaptar o nosso programa para que este leia os ficheiros de input valide e guarde a informação útil dos mesmos em hashtables como fazíamos no início e posteriormente guardamos essa informação em quatro novos ficheiros auxiliares. Após guardar esta informação nestes novos ficheiros libertamos toda a memória alocada anteriormente, mas hashtables e destruímos as mesmas. De seguida Lemos estes novos ficheiros e inserimos a informação dos mesmos em arrays das estruturas correspondentes. Desta forma podemos executar todas as queries apenas com a informação destes arrays alocada.

A memória estes arrays e libertada no fim de executarmos todas as queries .

Desta maneira conseguimos ter uma resposta rápida a qualquer uma das queries e ainda conseguimos executar o nosso código para um dataset maior.

Eficiência e melhoramentos à 1º fase

De maneira a melhorar o código, o grupo implementou algumas mudanças.

Foi decidido que os dados que eram lidos dos ficheiros mas que não tinham utilidade apenas estariam a ocupar memória. Daí, campos como os comentários de uma viagem, deixaram de ser guardados e passaram a ser ignorados.

4 Queries

Na segunda fase deste trabalho concluímos então as restantes 6 queries(4 á 9).

Na query 4 e 5, nós criamos duas funções "mediaporcity"e "preçoentredatas"que irão calcular o preço médio das viagens numa determinada cidade e num determinado intervalo de tempo respetivamente. Ambas as funçoes irão buscar informaçao a um array de estruturas(Ride) e filtrá-lo de acordo com o objetivo da query. Para a query 4, filtrámos de acordo com a cidade escolhida e na query 5 de acordo com o intervalo de tempo escolhido. Depois de filtar o array, tirámos os preços de todos os arrays, somamos e dividimos pelo numero respetivo.

A query 6 é bastante semelhante anterior já que vai buscar os dados a mesma estrutura(o array de Rides), e esta filtra o array pela cidade e pelo intervalo de tempo simultaneamente e retira a distância em vez do preço.

Na query 7 pretendemos buscar o top N condutores de uma cidade a partir da sua avaliação média, para isso vamos buscar os dados a um ficheiro com as informações já ordenado consoante a avaliação média de cada driver. A partir desse ficheiro, filtrámos todos os drivers que pertencem a cidade escolhida e tirámos o top N.

A query 8 pretende listar todas as viagens nas quais o utilizador e o condutor são do género escolhido e tem idade maior ou igual á escolhida,

para isso também vai buscar a um ficheiro com todas as rides ordenados pelo id da viagem, e vai filtrar e listar todas as viagens que apresentam os parâmetros referidos anteriormente.

A query 9 é bastante parecida com a anterior, só que em vez de filtrar pelo género e idade, vamos filtrar pelo intervalo de datas escolhidas e pela gorjeta.

5 Modo de operação interativo

O nosso modo iterativo vai ser um género de um menu baseado em scanfs e printfs, onde primeiro escolhes o número da query que queres executas e digitas os parâmetros que te pedirem consoante o numero da query escolhida. Depois disso, a query é executada e o número de comando é incrementado.

6 Validações sobre os ficheiros de dados

De modo a validar os dados recebidos, foram criada várias funções isoladas que validavam cada categoria, por exemplo, se uma data era válida, se a classe de um carro estava correta, etc. Assim, por cada linha do ficheiro lida, as funções respetivas para cada ficheiro eram chamadas e, apenas se devolvessem *True* para todas, a linha seria inserida num catálogo. Caso contrário, esta linha era descartada.

7 Conclusão

A conclusão deste trabalho de programação em C demonstra a importância da gestão e armazenamento de dados. O programa desenvolvido permitiu uma melhor organização e controle dos dados, otimizando o processo de gerenciamento e garantindo o encapsulamento de dados e a integridade das informações.