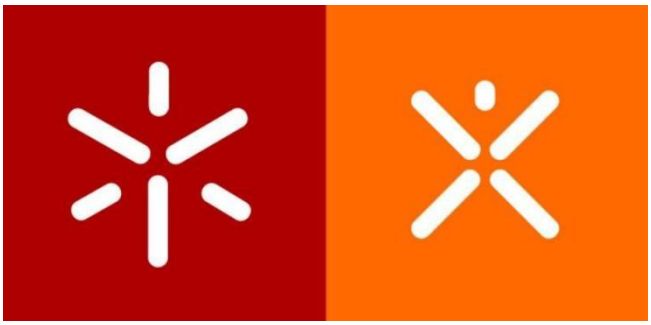

Universidade do Minho

Departamento de Informática

Licenciatura em Engenharia Informática

Laboratórios de Informática 3

Grupo 55 - Fase 2

Elementos do Grupo:

Guilherme Rio a100898

Diogo Cunha a100481

Rui Cerqueira a100537

Índice

Introdução

Este trabalho tem como objectivo a criação de código em C capaz de organizar e armazenar estructuras de dados complexas, mais especificamente de planeamento de viagens, sendo necessária ainda a implementação de ferramentas capazes de realizar operações predefinidas sobre esses dados (queries).

Módulos e estructuras de dados

- **Batch.c –** Este módulo é reponsável por iniciar o modo batch do programa, este é iniciado quando o programa recebe 2 argumentos, o path para os dados relativos aos voos, users, passengers e flights, assim como o path para as queries a ser executadas.

- **Interativo.c –** Este módulo é reponsável por iniciar o modo interactivo do programa, o qual é iniciado quando não são fornecidos argumentos ao mesmo.

- **Parser.c** – Este módulo é responsável por fazer o parsing do dataset fornecido.

- **Users.c –** Este módulo é responsável pela formatação do utilizador e por passar o mesmo ao seu respectivo catálogo, contém os getters necessários para obter todos os dados acerca de utilizadores.

- **Catálogo_users.c –** Este módulo é responsável por guardar os utilizadores na sua estructura de dados, optámos por utilizar uma hashtable para armazenar os utilizadores devido à necessidade frequente de acesso aos dados baseados no username do utilizador. O username do utilizador serve como chave única para cada utilizador na hashtable, facilitando um acesso rápido e eficiente às informações de cada utilizador.

- **User_stats.c –** Este módulo é responsável por guardar as "stats" necessárias para a resposta posterior das queries, para isso criamos uma hashtable com o nome do utilizador como chave para rápido acesso às stats de um utilizador, dentro desta temos guardado o seu número de reservas, o seu número de voos e o seu total gasto. Decidimos também guardar a sua lista de voos e lista de reservas, para estas decidimos usar Listas Ligadas em que cada nodo irá guardar os voos/reservas do utilizador.

- **Flights.c –** Este módulo é responsável pela formatação dos voos e por passar os mesmos ao seu respectivo catálogo, contém os getters necessários para obter os dados dos voos.

- **Catálogo_flights.c** – Este módulo é responsável por guardar os flights na sua estructura de dados, utilizámos uma hashtable para armazenar os dados devido à frequência de procura dos voos a partir do seu id, terá o seu id como chave.

- **Airport_stats.c –** Este módulo é responsável por guardar as "stats" necessárias para os aeroportos, para isto utilizamos uma hashtable com o id do aeroporto sendo a sua localização (ex: MAN), guardamos também o seu número de passageiros por ano, um array contendo todos os atrasos, o número de voos, e uma Lista Ligada que contém a lista de todos os voos.

- **Passengers.c –** Este módulo é responsável pela formatação dos passageiros e por passar os mesmos ao seu catálogo respectivo.

- **Catálogo_passengers.c** – Este módulo é responsável por guardar os dados relativos aos passageiros, para isso, escolhemos um Array.

- **Reservations.c –** Este módulo é responsável pela formatação das reservas e por inserir as mesmas na sua estructura de dados, para esta estructura decidimos usar uma hashtable pois procuramos pelas reservas através do seu id constantemente durante o projecto.

- **Catálogo_reservations –** Este módulo é responsável por guardar as reservas na sua estructura de dados, utilizámos uma hashtable para armazenar os dados por frequentementeprocurados reservas a partir do seu id, por isso utilizamos o seu id como chave.

- **Hotel_stats.c –** Este módulo é responsável por guardar as "stats" acerca dos hotéis, para este usamos uma hastable com o id do hotel como chave, o número de reservas, a lista de reservas numa Lista Ligada, e guardamos também, o seu averageScore, e a sua soma dos ratings (usados para calcular o avgScore).

.

- **Stats_needed.c –** Este módulo é responsável por descobrir quais utilizadores, voos ou reservas irão necessitar de estatísticas para as queries.

- **Stats.c –** Este módulo é responsável pela criação das estructuras de dados para as stats.

- **Validation.c –** Este módulo é responsável pela validação de utilizadores, voos e reservas.

- **Handle.c –** Este módulo é responsável por ler as linhas do input e executar as respetivas queries.

- **Utils.c –** Contem funções auxiliares necessárias ao longo do programa.

Estructura

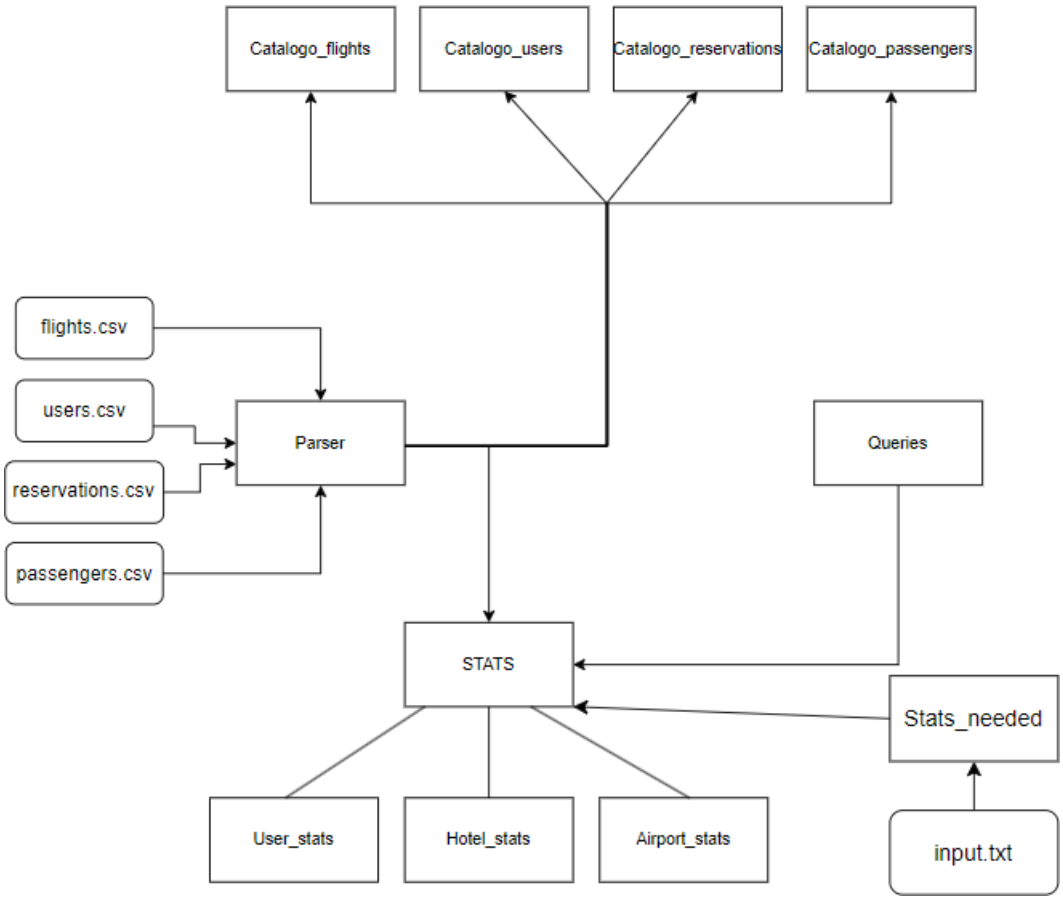

Figura 1 Estrutura do programa

Funcionamento

O programa começa então por verificar qual o modo de execução pretendido, batch ou interativo, caso seja escolhido o modo de operação batch, este irá inicializar e criar todos os catálogos para os dados fornecidos, e um catálogo adicional com os dados inválidos, criará também o catálogo das estatísticas.

Logo após efectuar a criação do catálogo para as estatísticas irá chamar a função start_stats_needed () que irá verificar no ficheiro de input, quais serão os identificadores que irão necessitar de stats, estas são guardadas numa estructura do catálogo de stats temporária pois só é necessária durante o parsingdos ficheiros.

O programa começa então por fazer o parsing de cada ficheiro, o parser lê então cada linha do ficheiro,e passa a mesma ao modulo respetivo através da função start_'type'_process () que irá retornar um inta avisar o parser se o utilizador é inválido, se receber a mensagem que é invalido irá de seguida imprimir a linha para o ficheiro de errors.

No módulo de cada tipo, este irá criar um uma instância desse tipo e irá verificar se os dados são validos, ou se não contém dados inválidos através do catálogo de dados inválidos, se for valido irá então passar o objecto ao respectivo catálogo, se for inválido irá efetuar a libertação da memória

alocada para essa instância, no caso de ser um utilizador ou um voo, irá adicionar os seusidentificadores ao catálogo de dados inválidos.

Ao mesmo tempo, o programa verifica se é necessário gerar estatísticas específicas para o elementoem análise, seja ele uma reserva, um voo ou relativo aos passageiros, conforme os dados são processados durante o parsing. Se for necessário criar a estatística será então passado ao modulo da estatistica respetivo.

Depois do parsing ser acabado, seguem-se as respostas as queries, e finalmente a libertação dememória.

Queries

- Querie 1 – Nesta querie temos 3 tipos de input
  - Para o input em que recebemos o id de uma reserva, começamos por procurar o id na hashtable correspondente e verificar se o mesmo existe, e após isso imprimimos os seus dados através de getters definidos nos respetivos módulos.
  - Para o input em que recebemos o id de um utilizador, começamos por procurar o id na hashtable correspondente e verificar se o mesmo existe, depois verificamos se a sua conta é ativa e caso seja, ainda temos de verificar se foram criadas as stats para o utilizador pois elas podem não ser criadas caso o utilizador não tenha nem voos nem reservas, e depois fazemos os respetivos prints.
  - Para o input em que recebemos o id de um voo, verificamos se o mesmo existe na sua hashtable, e imprimimos os seus respetivos dados.

- Querie 2 – Nesta querie temos 3 tipos de input, o caso em que so é requisitado reservas, um caso em que só são requisitados flights, e o final em que são requisitados ambos, começamos então por verificar que tipo nos é pedido, obtemos a respetiva lista, concatenando ambas se forem pedidos os dois tipos, ordenamos e efetuamos o print ao ficheiro.

- Querie 3 – Nesta querie apenas nos pedem o average rating de um hotel, o qual já temos guardados nas stats do hotel, ou seja, só temos de verificar se existem stats para esse hotel, e efetuar os prints respetivos.

- Querie 4 – Para esta querie começamos por verificar se existem stats para o hotel, e após isso efetuamos o sort da lista de reservas presentes nas stats, e efetuamos os prints dos dados necessários.

- Querie 5 – Para esta querie começamos por retirar as aspas presentes no input para conseguirmos efetuar a comparação entre datas, depois criamos uma Lista Ligada que será freed no final da função onde guardamos os voos que cumprem os requisitos a querie, e efetuamos os respetivos prints no final.

- Querie 6 – Para esta querie começamos por organizar a lista dos stats que criamos para os aeroportos pelo número de passageiros, e depois usamos um ciclo while para efetuar os prints enquanto uma variável i que começa com valor 1 é >= ao valor de N (top N).

- Querie 7 – Para esta querie começamos por organizar a lista dos stats dos aeroportos pela sua mediana e depois efetuamos o ciclo while como na querie 6 para efetuarmos prints apenas do top N medianas.

- Querie 8 – Para esta querie começamos por obter as reservas do hotel pedido, e depois iteramos essas reservas, usamos uma função auxiliar para verificar quantos dias da reserva se encontram dentro periodo fornecido, se o resultado for maior que 0, calculamos o preço total dessa reserva e adicionamos a receita, que será demonstrada no final.

Modo interactivo

O modo interactivo, começa por requisitar o caminho para o dataset, e de seguida o utilizador é recebido com o seguinte menu.

Figura 2 Menu inical

Se o utilizador selecionar a opção help, irá encontar uma explicação e exemplo de cada query com umsistema de paginação para uma navegação mais facil.

Figura 3 Menu de ajuda

Se o utilizador seleccionar a opção Run Query,ser-lhe-á requisitado a query que o mesmo pretende executar, de seguida, o dataset será loaded e a query executada e displayed com um sistema de paginação, no final a data será cleared e o utilizador voltará para o menu principal.

Figura 4 Query Display

Testes funcionais e de desempenho

Vamos agora abordar o desempenho do nosso programa e o seu modulo de testes desenvolvido, paraos testes efetuados no programa foi utilizado o dataset large, com dados inválidos, e as 500 queries fornecidas pelos professores.

Na seguinte imagem podemos visualiar que memória usada pelo nosso program foi 4566 MB, e um tempo de execução de 100s, isto devido ao espaço alocado não so para os dados todos, mas como paraas estatisticas necessárias, a parte mais intensiva do nosso programa é a volta do parsing que trata de validação, criação das entidades, e das suas estatisticas, para que haja menos trabalho posteriormente a responder às queries.

	Máquina 1 (VM)	Máquina 2	Máquina 3
CPU	Intel Core i7-1165G7	Intel Core i5-1135G7	Intel Core i7-8750H
RAM	3/8	4/8	6/12
Cores/Threads	16GB (10 Disponibilizados para a VM)	8GB	16GB
Disco	1TB NVMe Intel	512 GB SSD PCIe	128 GB NVMe
OS	Ubunto Linux	Ubunto Linux	Ubunto Linux
Memória Utilizada	4566 MB	4566 MB	4566 MB
Tempo de execução 500 queries	98.487s	95.225s	120.242s

Figura 5 Desempenho

Podemos visualizar também, na tabela abaixo o tempo de execução das queries nos computadores dosvários membros do grupo, para este teste usamos o seguinte input:

```
1 Book0002718540
2 RPacheco1587
3 HTL1201
4 HTL1504
5 LHR "2021/07/01 23:24:19" "2021/08/30 10:27:43"
6 2022 272
7 153
8 HTL301 2023/01/16 2023/06/109 "Mateus Sim"
9 "Mateus Sim"
```

Conseguimos observar que a querie mais demorada foi a querie 4, pois está tem de consultar todas as reservas efetuadas num hotel e efetuar várias comparações de modo a ordenar essa lista corretamente, o mesmo se aplica a querie 5 so que com os aeroportos, a querie 9 apensar de não ter sido implementada na sua totalidade, também é bastante demora, pois é a única que não tira proveito das estatiscas criadas previamente, as restantes queries, maioritariamente fazem consulta de dados, ou para o caso deste input, lidam com uma quantidade bastente pequena dos mesmos.
Para todas estas estruturas auxiliares das estatisticas decidimos usar Listas Ligadas.

	Máquina 1	Máquina 2	Máquina 3
Querie 1	0.339ms	0.176ms	0.163ms
Querie 2	0.139ms	0.104ms	0.253ms
Querie 3	0.08ms	0.037ms	0.112ms
Querie 4	540.340ms	438.380ms	1128.170ms
Querie 5	226.401ms	128.495ms	456.941ms
Querie 6	0.119ms	0.114ms	0.219ms
Querie 7	0.0122ms	0.058ms	0.114ms
Querie 8	209.127ms	159.232ms	483.101ms
Querie 9	446.132ms	395.379	563.945ms

Figura 6 Tempos de execução das queries

Conclusão

Como podemos ver ao longo do relátorio a fase mais intensiva do nosso projecto é a fase do parsing pois preparámos tudo para a resolução das queries nesse momento, gerando isto limitações no modo interactivo do nosso programa, para trabalho futuro seria então melhorar esse aspecto, e finalizar a implementação das queries de maneira mais eficaz, assim como melhorar o uso de memória fazendo algo como em vez de alocar espaço para os dados de Passengers, associar os mesmos aos utilizadores e flights já alocados usando pointers.

Resumindo, apesar de este projecto ser bastante desafiador, com uma implementação complexa, e de termos sido confrontados várias vezes com obstáculos inesperados, estamos satisfeitos com o trabalho realizado e com as melhorias realizadas em relação à primeira fase, que cumpre os principais objectivos deste projecto.