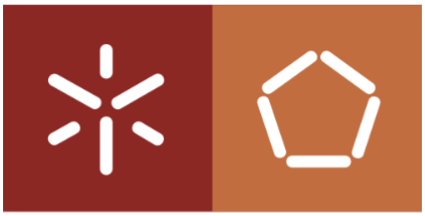


Universidade do Minho
Escola de Engenharia

Universidade do Minho

PROJETO LABORATÓRIOS DE INFORMÁTICA III
Ano letivo 2023/2024
Fase 1

Grupo 77

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Afonso Sousa (A104262), João Carvalho (A104533) e Rui Cruz (A104355)

Conteúdo

Capítulo 1

Introdução

Este projeto encontra-se atualmente em desenvolvimento no âmbito da unidade curricular de Laboratórios de Informática III, referente ao ano letivo 2023/2024.

A tarefa atribuída consiste na implementação de uma base de dados em memória destinada a armazenar informações fornecidas pelos docentes. Estes dados assemelham-se de perto à plataforma Booking.

O projeto está dividido em duas fases, sendo que a primeira fase concentra-se na implementação do parsing dos dados e no modo batch. Este modo envolve a execução sequencial de várias consultas sobre os dados, estando esses pedidos armazenados num ficheiro de texto cujo caminho é recebido como argumento.

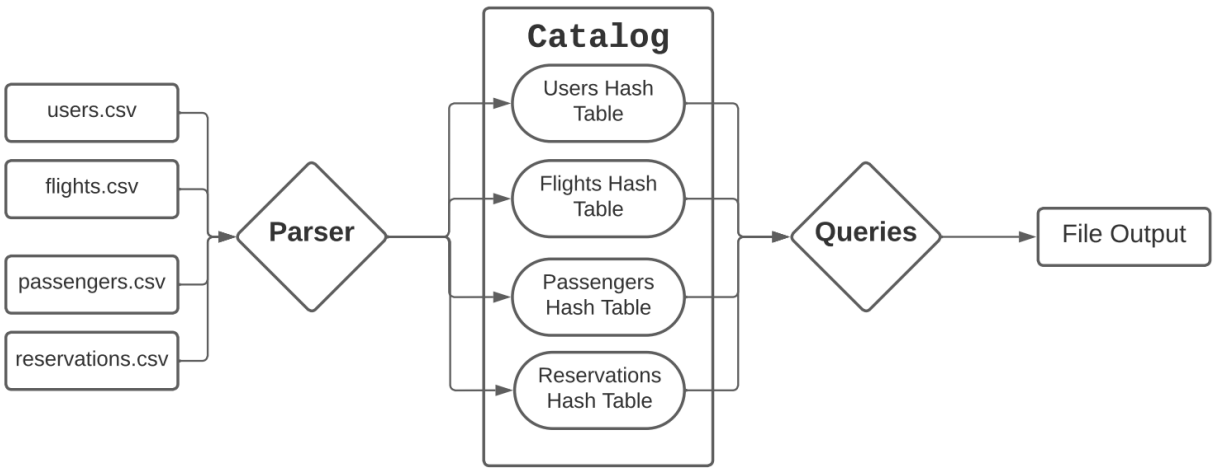
Assim, este processo exige não apenas a leitura e interpretação dos dados dos ficheiros, mas também o seu armazenamento em estruturas de dados versáteis e de rápido acesso.

Capítulo 2

Desenvolvimento

Como seria de antecipar, este projeto não se apresenta como uma tarefa fácil. Nesse sentido, o nosso grupo realizou uma planificação meticulosa antes de iniciar a codificação.

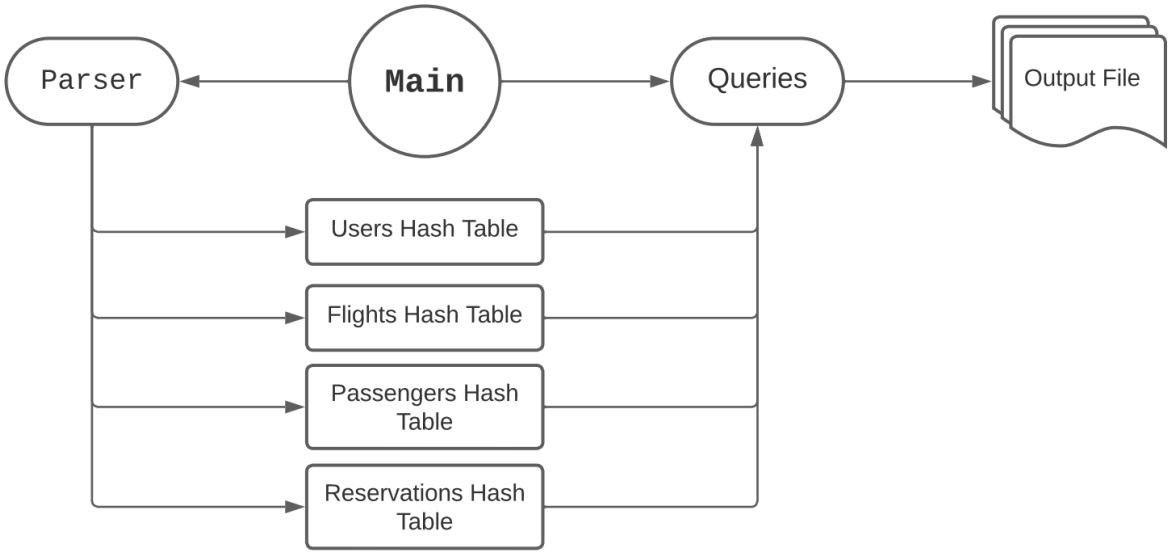
Começamos a imaginar uma arquitetura que delineava a interligação entre os diversos módulos e a forma como estes se comunicam entre si. Esta abordagem é crucial, uma vez que é fundamental que o encapsulamento e a modularidade estejam presentes em todas as fases do projeto. A arquitetura que concebemos assemelhou-se significativamente à idealizada pelos docentes:



O desafio inicial consistiu em identificar as estruturas ideais para armazenar todos os dados. Foi unânime a ideia de que as tabelas de dispersão (hash tables) eram ideais para as estruturas principais, uma vez que oferecem um tempo de acesso constante. Criamos, assim, três hash tables - uma para os utilizadores, outra para os voos e uma terceira para as reservas. Para os passageiros foi criado um array de listas.

Para aproximar as três primeiras referidas decidimos agrupá-las numa estrutura principal denominada de catálogo, onde cada uma das hash tables representaria uma tabela da base de dados.

Posteriormente, foi necessário abordar o parsing dos dados. Desenvolvemos uma solução, provavelmente não é das melhores, mas foi a que inicialmente fez mais sentido para nós, visto que o input consiste num ficheiro CSV separado apenas por pontos e vírgulas. Criamos um parser geral que poderia ser invocado por outras funções noutro ficheiro, capazes de utilizar o output do parser, contribuindo para a modularização do código.



Finalmente, para concluir esta fase inicial, optamos por abordar as queries que desenvolvemos ao longo desta primeira fase. Apesar de não ter sido adquirido qualquer resultado após a implementação das queries, achamos que o raciocínio esteja de acordo com o pretendido. De modo a analisar o ficheiro input.txt (ficheiro que inclui os comandos) foi desenvolvido um módulo chamado interpreter que lê cada linha desse ficheiro e interpreta esse comando. Explicaremos agora como implementamos cada uma das queries.

É importante referir que caso uma query no modo batch contenha a flag 'F', o seu resultado deverá ser apresentado com o formato field: value, para além da indicação do número do registo no seu início (através de --- n ---, onde n é o número do registo).

Exemplo de output quando o comando não tem a flag F:

```
F000000123;2023/10/06;flight
R000000456;2023/10/02;reservation
F000000121;2023/10/01;flight
```

Exemplo de output quando o comando tem a flag F:

```
--- 1 ---
id: F000000123
date: 2023/10/06
type: flight

--- 2 ---
id: R000000456
date: 2023/10/02
type: reservation

--- 3 ---
...
```

Query 1: Esta query consiste em "Listar o resumo de um utilizador, voo, ou reserva, consoante o identificador recebido por argumento" .

A *query1* realiza consultas no sistema, processando informações sobre reservas, voos e utilizadores com base em identificadores. Para reservas é calculado o custo total. Para voos, obtém dados e calcula informações como o número de passageiros. Para utilizadores, extrai e processa dados, calculando estatísticas. Como é evidente, cada dataset requer um conjunto de funções auxiliares, tais como: *check_if_reserv_flight_or_user*, *calculate_num_passageiros* e *calculate_NumAndPrice_Reservations*; para facilitar a obtenção de dados e operações.

Query 3: A query consiste em "Apresentar a classificação média de um hotel, a partir do seu identificador."

Ou seja, é necessário calcular a classificação média de um hotel (utilizamos uma função auxiliar) a partir do seu id. O resultado é devolvido à query 3.

Query 4: "Listar as reservas de um hotel, ordenadas por data de início (da mais recente para a mais antiga). Caso duas reservas tenham a mesma data, deve ser usado o identificador da reserva como critério de desempate (de forma crescente)."

O processo é efetuado através de funções auxiliares para manipular a lista de reservas associadas ao hotel (*sort_flights_by_date_and_flight_id*). Depois das reservas estarem ordenadas corretamente, serão então imprimidas no ficheiro de output.

Query 5: "Listar os voos com origem num dado aeroporto, entre duas datas, ordenados por data de partida estimada (da mais antiga para a mais recente). Um voo está entre e caso a sua respetiva data estimada de partida esteja entre e (ambos inclusivos). Caso dois voos tenham a mesma data, o identificador do voo deverá ser usado como critério de desempate (de forma crescente)."

O processo é efetuado através de funções auxiliares para manipular a lista de reservas associadas ao hotel (*sort_flights_by_date_and_flight_id*). Depois dos voos estarem ordenados corretamente, serão então imprimidos no ficheiro de output.

Query 6: "Listar o top N aeroportos com mais passageiros, para um dado ano. Deverão ser contabilizados os voos com a data estimada de partida nesse ano. Caso dois aeroportos tenham o mesmo valor, deverá ser usado o nome do aeroporto como critério de desempate (de forma crescente)."

Para atingir esse propósito, a resolução da query envolve várias etapas que serão detalhadas a seguir.

1. Obtenção dos Aeroportos Existentes: A primeira etapa envolve a obtenção de uma lista de todos os aeroportos requeridos pela query. Esta informação é crucial para a análise subsequente, pois permite a identificação de origens e destinos dos voos.

2. Verificação da Data de Partida: Para cada voo, o código verifica se a data estimada de partida corresponde ao ano fornecido como parâmetro. Isso garante que apenas voos do ano desejado sejam considerados na análise.

3. Cálculo do Número de Passageiros: Para cada voo considerado, o código utiliza a função *calculate_num_passageiros* para determinar o número de passageiros associados a esse voo.

4. Ordenação dos Resultados: Após a colheita dos dados, são ordenadas as origens e os respectivos passageiros (*sort_arrays*). Isso permite a identificação dos aeroportos mais movimentados durante o ano em questão.

Query 7: "Listar o top N aeroportos com a maior mediana de atrasos. Atrasos num aeroporto são calculados a partir da diferença entre a data estimada e a data real de partida, para voos com origem nesse aeroporto. O valor do atraso deverá ser apresentado em segundos. Caso dois aeroportos tenham a mesma mediana, o nome do aeroporto deverá ser usado como critério de desempate (de forma crescente)."

O raciocínio utilizado não fugiu muito daquele que foi usado na query 6. Em primeira instância foi necessário obter uma lista de todos os aeroportos. São utilizados 3 arrays, um para calcular o total de atrasos, outro que calcula o tempo de atraso e finalmente um que calcula a mediana de cada voo. Depois das iterações dos voos e dos cálculos é feita uma ordenação dos arrays.

Query 8: "Apresentar a receita total de um hotel entre duas datas (inclusive), a partir do seu identificador. As receitas de um hotel devem considerar apenas o preço por noite (*price_per_night*) de todas as reservas com noites entre as duas datas. E.g., caso um hotel tenha apenas uma reserva de 100€/noite de 2023/10/01 a 2023/10/10, e quisermos saber as receitas entre 2023/10/01 a 2023/10/02, deverá ser retornado 200€ (duas noites). Por outro lado, caso a reserva seja entre 2023/10/01 a 2023/09/02, deverá ser retornado 100€ (uma noite)."

O algoritmo utilizado nesta query é mais simples. Utilizamos uma função auxiliar que dado um id de uma reserva calcula a diferença de dias entre as datas de "check-in" e de "check-out" sabendo que o custo de uma reserva é dado por: price_per_night × número de noites + $\frac{price\_per\_night \times número\ de\ noites}{100}$ × imposto da cidade.

Capítulo 3

Dificuldades sentidas

Como já mencionado, este trabalho não é de fácil execução, o que implica a presença e a expectativa de muitas adversidades ao longo do processo. A maior complexidade foi, sem dúvida, a implementação cuidadosa do encapsulamento e da modularidade e a construção da dualidade parser<->queries. A realidade é que o tópico do encapsulamento é bastante extenso e complexo, e, sendo esta a primeira vez que o abordamos, é natural que existam falhas. Assumimos o compromisso de melhorar este aspeto na segunda fase.

Outra dificuldade encontrada foi a realização de mudanças atómicas no código sem compreender se estávamos a afetar alguma parte do código. É importante destacar que um dos pontos ressaltados pelo encapsulamento e modularidade é a garantia de efetuar essas alterações sem grandes problemas associados - indicando que, provavelmente, não implementamos esses conceitos da forma mais eficaz.

Capítulo 4

Conclusões

Resumindo, apesar das diversas dificuldades e dos novos desafios apresentados por este projeto, acreditamos estar à altura desta tarefa. No decorrer do planeamento, tomamos as decisões corretas, o que nos permitiu construir uma base sólida a partir da qual conseguiremos expandir e adaptar para lidar com as queries mais complexas da 2ª fase. Infelizmente os resultados obtidos nesta primeira fase não foram os esperados, ainda existem alguns problemas incógnitos, no entanto, sentimo-nos preparados para concluir o projeto e aprimorar ainda mais o que já realizamos até agora.