Universidade do Minho

Licenciatura em Engenharia Informática

Laboratórios de Informática III

Relatório Fase 1 – Grupo 42

André Miranda – a104088

José Soares – a103995

Nuno Melo – a104446

Braga, Novembro 2023

Índice

1 – Introdução

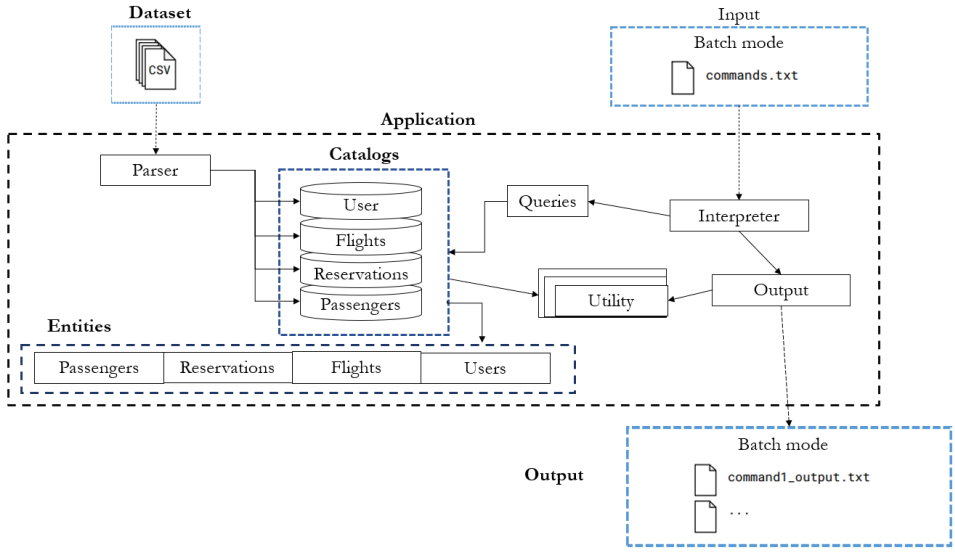
Serve o presente relatório para apresentação e discussão do projeto realizado no âmbito da unidade curricular de Laboratórios de Informática III do ano 2023/2024. O objetivo do projeto é desenvolver uma aplicação em C que permita analisar dados sobre utilizadores, voos, passageiros e reservas, usando estruturas de dados adequadas e eficientes para futuro tratamento de informações e tendo em conta aspetos como a modularidade e encapsulamento. O projeto foi dividido em várias etapas, que serão apresentadas e discutidas neste relatório.

2 – Desenvolvimento

2.1 – Criação da Arquitetura e definição do Pipeline

Na primeira etapa, foi necessário definir a arquitetura e o pipeline da aplicação, bem como os módulos que a compõem, para assim tornar a execução do projeto mais eficiente e organizada para o que iria ser o trabalho coletivo do nosso grupo.

Com base no guião fornecido, e numa primeira fase, definiu-se a seguinte arquitetura como apoio principal:



No desenvolvimento do nosso pipeline, concebemos a implementação de um parser genérico capaz de processar qualquer ficheiro no formato *.csv. Esse parser foi projetado para separar a informação em tokens, permitindo, assim um futuro armazenamento em estruturas de dados adequadas. Para isso, avançamos com a criação de entidades e estruturas específicas para cada catálogo, garantindo uma representação eficaz e organizada dos dados.

Em seguida, dedicamos esforços para a implementação do parser para os comandos, capacitando-o para interpretar e processar queries de maneira eficaz.

Além disso, como parte integrante do pipeline, desenvolvemos um módulo principal denominado "batch". Este módulo atua como o ponto central de execução, coordenando a interação entre os diferentes componentes do sistema. Adicionalmente, incorporamos um módulo de output projetado para a escrita de resultados, assegurando a saída adequada das informações processadas.

O conjunto integrado destas etapas — parser genérico, catálogos, parser de comandos, módulo batch, e módulo de output — compõe um pipeline completo e flexível, capaz de processar de forma eficiente os dados provenientes de diversos ficheiros *.csv, interpretar comandos/queries, e fornecer resultados de forma coerente e estruturada.

2.2 – Estruturas de Dados

Entrando agora com a parte de estruturas de dados, passamos à implementação das estruturas para entidades e catálogos. Inicialmente, tínhamos pensado em usar arrays para armazenar as entidades nos catálogos, mas depois de analisar o desempenho e a complexidade, optamos por usar tabelas de hash da biblioteca Glib.

As tabelas de hash proporcionam um acesso rápido e eficiente (O(1)), especialmente em operações de busca, e adaptam-se dinamicamente ao crescimento do conjunto de dados, evitando realocações frequentes de memória. Ou seja, com esta escolha iriamos conseguir um sistema mais eficiente e com melhor desempenho em comparação com a abordagem inicial de usar arrays.

Já com as mãos à obra, decidimos implementar quatro catálogos distintos utilizando tabelas de hash para armazenar dados relacionados a utilizadores, voos, passageiros e reservas. Ao abordar a questão específica dos passageiros, deparamo-nos com o problema recorrente da possível presença de elementos repetidos e colisões frequentes. Após uma análise cuidadosa, decidimos prosseguir com a criação de uma tabela de hash dedicada aos passageiros. Nessa abordagem, a chave foi definida como a combinação única do identificador de voo e do identificador de usuário, proporcionando uma solução para lidar com as características dessa categoria específica.

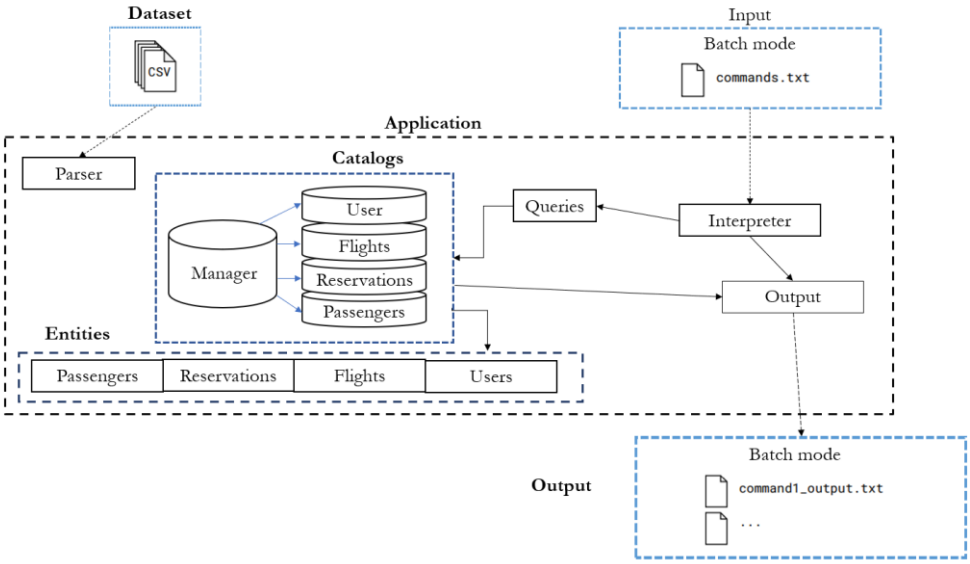
Ao iterar sobre as tabelas de hash, surgiu um novo percalço com a suposta iteração que não estava a ser precisa nem tão pouco correta, resolvemos então adaptar a ideia inicial de arrays à estrutura de cada catálogo.

Além da tabela de hash, adicionamos um array que continha as chaves válidas e o número correspondente de chaves para cada catálogo. Esta abordagem híbrida permite uma iteração mais eficiente e simplifica o processo de manipulação de dados. O array de chaves válidas oferece uma estrutura ordenada e acessível para percorrer os elementos, enquanto a tabela de hash continua a ser a principal estrutura para recuperação rápida e eficiente de dados. Essa combinação de arrays e tabelas de hash visa otimizar a operação geral do sistema, fornecendo uma solução equilibrada para a iteração e manipulação de dados nos diferentes catálogos.

No que toca, ao "interpretador" decidimos criar uma estrutura para cada input do ficheiro de comandos, e assim formar de uma só vez, uma *struct command* em que guardávamos o numero da query, a existência de flag (0 ou 1), o número de argumentos e consequentemente os próprios argumentos.

Para concluir, e de modo a "encapsular" os catálogos decidimos criar um catálogo principal, em que nele estavam contidos todos os outros catálogos relativos ás entidades.

No final, acabamos com uma estrutura, um pouco diferente mas mais capaz e completa.



2.3 – Queries

Nesta primeira fase do projeto, decidimos optar por fazer as queries 1, 2, 3, 4, 7 e 9, e foram escolhidas porque devido ao tempo e também à fase em que estávamos, estas realmente eram as mais indicadas.

Query 1 – Para a primeira query, e já depois de termos um comando criado, era necessário diferenciar o tipo de entidade, para então percorrer o catálogo associado à entidade através do array, e assim armazenar as informações pedidas para serem dadas como resposta ao comando.

Query 2 - Para esta query, e porque utilizamos tabelas de hash, o acesso feito a um utilizador era feito em tempo constante o que nos permitia depois, consoante o argumento passado pelo comando criado pelo input, decidir qual dos catálogos deveria ser percorrido para assim ir buscar as informações necessárias à query.

Query 3 – Dado o identificador de um hotel, percorremos o array das reservas válidas, no catálogo das reservas, verificando se em cada reserva, o hotel era o mesmo. Caso fosse igual, incrementamos um contador e somávamos o rating desse hotel a um acumulador, no final, apenas dividimos o valor do acumulador (soma dos ratings) pelo contador (número de reservas para o hotel).

Query 4 – Nesta query, o nosso sistema através do identificador de um hotel, recupera toda a informação de todas as reservas associadas a esse hotel através do iteração sobre a tabela de hash com o array, e com base numa struct auxiliar, utilizamos o algoritmo de ordenação *Quick Sort* para ordenar em ordem decrescente a data de entrada. O resultado era a lista ordenada de reservas de um hotel.

Query 7 – Para listar um numero N de aeroportos com a maior mediana de atrasos, foi necessário percorrer o catálogo dos voos e associar numa struct apropriada, cada um dos aeroportos a uma lista de atrasos. No fim da iteração, e utilizando o algoritmo de ordenção Quick Sort, a lista de aeroportos formada era posta em ordem em relação à mediana de atrasos.

Query 9 – Através de um prefixo passado como argumento pelo comando, era feita uma iteração pelo catálogo dos utilizadores, armazenando todos os que tinham um id com prefixo correspondente ao fornecido, de seguida, era feita a ordenação (Quick Sort) onde a lista era ordenada de forma crescente pelo nome do utilizador.

2.4 – Dificuldades

Como dificuldades sentidas no projeto temos a apontar os seguintes tópicos:

- Estruturas de Dados: Arrays vs Tabelas de Hash
- Implementação das Tabelas de Hash com a glib

No primeiro tópico, e como já foi referido neste relatório, a nossa grande dificuldade foi em decidir como avançar com os catálogos, dado que ainda não sabíamos nada sobre tabelas de hash, e o array à partida era a estrutura a que mais estávamos familiarizados. No entanto, e devido as vantagens do uso de uma tabela de hash, decidimos em conjunto aprender mais sobre esta estrutura de dados e começar a implementar.

De seguida, e colocando as mãos à obra, sentimos enormes dificuldades em saber como organizar e fazer a manutenção das tabelas, o processo não foi fácil mas acho que no final, consegui-mos controlar a situação e ultrapassar as dificuldades.

2.5 – Aspetos a melhorar

Tempos de Execução e Reallocs: Foi observado que os tempos de execução, especialmente relacionados aos reallocs, estão prejudicar diretamente a eficiência de nosso sistema. A fim de otimizar o desempenho, vamos tentar encontrar estratégias alternativas e ajustar a alocação de memória de forma mais eficaz.

Algoritmos de Queries: Os algoritmos atuais das nossas queries apresentam imperfeições que afetam a precisão e a eficácia de nossas consultas. Vamos tentar rever os aspetos referentes à forma como armazenamos os dados e também à formo como acedemos aos mesmos.

Estrutura dos Passageiros e Users: A estrutura atual dos passageiros pode ser simplificada e integrada à estrutura dos utilizadores. Esta combinação não apenas simplificará o código, mas também facilitará a manutenção e o entendimento geral da lógica do sistema.

Parsing Incorreto das Queries: Identificamos que o parsing das queries não está totalmente correto, o que pode levar a interpretações inadequadas e resultados imprecisos. Para isso, na segunda fase, vamos melhorar o parser e garantir que não existam erros ao criar os comandos.

Encapsulamento: O encapsulamento, que garante a proteção e isolamento adequados dos componentes do sistema, precisa de ser fortalecido no nosso programa. Para esta primeira fase, vimos que ainda não era necessário, no entanto sabemos o que tem de ser feito e já estamos neste momento a pensar em formar para implementar este conceito no nosso projeto.

3 – Conclusão

Em conclusão, o projeto foi uma experiência enriquecedora e desafiadora, que contribuiu para o desenvolvimento das nossas competências na área da programação em C. Aprofundamos os nossos conhecimentos sobre estruturas de dados, reconhecendo a importância de analisar as características especificas de cada uma, e também a maneira como construímos e desenvolvemos um projeto de grupo. As lições aprendidas com este projeto certamente serão aplicadas em trabalhos futuros, enriquecendo assim o nosso percurso profissional na área da engenharia de software.