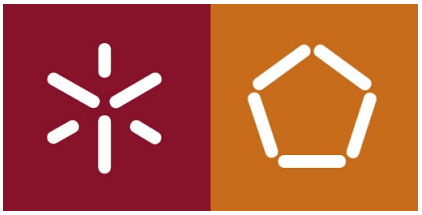


Sistema de Gestão de Vendas de uma Distribuidora

Laboratórios de Informática 3 - Grupo 2

João Pedro da Santa Guedes A89588
Luís Pedro Oliveira de Castro Vieira A89601
Pedro Miguel de Soveral Pacheco Barbosa A89529



19 de abril de 2020

Índice

1 Introdução

No âmbito da unidade curricular de Laboratórios de Informática 3 foi-nos proposta a implementação de um sistema de resposta a queries sobre a gestão de vendas de uma distribuidora com 3 filiais.

A primeira fase do projeto consistiu em implementar este sistema na linguagem de programação C. Tendo como objetivo tirar o máximo partido da eficiência e rápida execução das funcionalidades do nosso programa, tivemos também como foco alguns princípios da programação como modularidade e encapsulamento de dados, criação de código reutilizável, uma escolha otimizada das estruturas de dados bem como a sua reutilização e testes de performance e de profiling.

Estes objetivos vão de encontro aos conhecimentos adquiridos neste semestre na unidade curricular de Programação Orientada aos Objetos, sendo que a segunda fase deste projeto compreende a implementação do mesmo sistema na linguagem de programação Java.

À medida que fomos desenvolvendo o nosso trabalho, apercebemo-nos que as maiores dificuldades que encontramos foram descobrir quais as melhores estruturas de dados a implementar de modo a ter um programa não só rápido e eficaz, como também seguro, tendo em conta o grande volume de informação a ser tratada.

2 Ferramentas Auxiliares

Este projeto podia ser auxiliado por diversas ferramentas. No entanto, para construir as nossas estruturas de dados, decidimos criar os nossos próprios módulos em vez de utilizarmos a library glib, com o propósito de obtermos ferramentas mais intuitivas e simples e que apresentam todas as características necessárias para a realização do nosso projeto.

3 Estruturas de Dados

A abordagem feita às estruturas de dados é um dos passos mais importantes de todo o projeto pois é essencial tanto para organizar a quantidade de informação presente nos ficheiros de texto como para responder às queries do trabalho.

Assim, começamos por analisar como eram constituídos os produtos, os clientes e as vendas bem como o que nos era pedido em cada querie de forma a percebermos quais eram as informações às quais iríamos recorrer constantemente.

O nosso projeto tem por base quatro módulos principais, sendo eles o dos clientes, o dos produtos, o da faturação e o das filiais.

3.1 Tipos de Dados

A base de todo o trabalho são os dados dos clientes, dos produtos e das vendas. Para cada um deste tipo de dados criámos uma estrutura própria.

3.1.1 Clientes e Produtos

Foi-nos pedido no enunciado para criarmos um catálogo tanto de clientes como de produtos. Assim sendo, tanto a struct dos clientes como a dos produtos teve a mesma base. Originaram-se dois módulos diferentes, o dos produtos e o dos clientes.

```
struct catalogo_clientes {
  Catalogo catalogo;
};
```

```
struct catalogo_produtos{
  Catalogo catalogo;
};
```

Deste modo, um dos nossos principais desafios foi criar uma struct para o catálogo que fosse a mais eficiente possível e que, ao mesmo tempo, nos ajudasse mais a tratar de todas as informações disponíveis.

Chegámos à conclusão que a melhor estrutura seria um array de 26 posições, referentes às 26 letras do alfabeto, onde cada índice do mesmo aponta para uma árvore balanceada que irá conter o código do cliente ou do produto, conforme seja no módulo dos clientes ou dos produtos, de uma dada letra associada a esse índice.

Esta escolha permite-nos obter níveis de eficiência muito elevados em termos de funções tais como inserções, remoções, alterações de conteúdo que iriam ser recorrentemente necessárias ao longo do desenvolvimento deste projeto.

3.1.2 Vendas

Arranjar uma forma de tratar das vendas de maneira a facilitar a resposta das queries foi um pouco mais complexo. Desta vez criámos dois módulos diferentes no nosso trabalho, um para a faturação e outro para as filiais.

Em relação ao primeiro módulo, em que tivemos de relacionar os produtos com as suas vendas, decidimos que o melhor seria criar um catálogo com a faturação. Tendo em conta o que nos pedia nas queries também criamos um estrutura organizada por modo de venda, se em regime normal ou promocional, por filial e por mês de onde obtinhamos a faturação, a quantidade de produtos e o número total de vendas. Desta forma, esta seria muito útil para organizarmos as nossas vendas segundo estes parâmetros para facilmente respondermos às queries acedendo rapidamente a informações que separavam as filiais, os meses, e os modos.

```
struct catalogo_faturacao{
  Catalogo catalogo;
};


struct fat_info{
  /*Modo -> N = 0 & P = 1*/
  double fat[MODO][FILIAL][MESES]; /*
  int qtd[MODO][FILIAL][MESES]; /*Qua
  int n_vendas[MODO][FILIAL][MESES];
};
```

O outro módulo foi o das filiais. Neste, decidimos dividi-lo em duas partes, uma em que relacionamos as filiais com os clientes e a outra na qual relacionamos as filiais com os produtos. Assim conseguimos facilmente obter informações que seriam úteis nas respostas às queries, como por exemplo, quais os produtos que foram comprados numa certa filial e que clientes compraram em todas as filiais.

Consequentemente, tivemos de criar estruturas para os clientes e estruturas para as filiais que, tal como aconteceu para os seus respetivos módulos, tiveram a mesma base.

Em relação aos clientes, começamos por criar uma estrutura "info cliente" que contém a quantidade de produtos que o cliente comprou e a sua faturação.

De seguida, relacionamos essa informação com as filiais. Assim, criamos uma estrutura "filial info produtos" na qual guardamos os clientes que compraram produtos num array de 12 posições, referentes aos 12 meses de um ano, em que cada índice aponta para uma árvore balanceada que contém os clientes que compraram nesse mês de uma forma ordenada e onde também utilizamos um array de 12 posições, referentes aos 12 meses de um ano, no qual em cada posição guardamos a quantidade total de compras realizadas pelo cliente nesse mês.

Seguidamente, criamos uma estrutura "filial cat clientes" para guardar as informações obtidas da estrutura acima referida. Deste modo, esta estrutura é formada por um catálogo, ou seja, um array de 26 posições em que cada índice do array, referente a uma letra do alfabeto, aponta para uma árvore balanceada que guarda as informações dos clientes começados por essa mesma letra.

Finalmente, elaboramos a última estrutura "filial c" que consiste num array de 3 posições, referentes às 3 filiais, em que cada índice do mesmo aponta para um array de 26 posições, referentes às letras do alfabeto, que funcionam como explicado na estrutura acima e, desta forma, ficam organizados por filial.

```c
/*FILIAL_clientes*/
struct filial_c{
  Filial_Catalogo_Clientes f_clientes[FILIAL];
};

/*Filial_Catalogo_Clientes*/
struct filial_cat_clientes{
  Catalogo catalogo;
};

/*f_info_c*/
struct filial_info_cliente{
  AVL produtos_comprados[MESES];
  int qtd_total_compras[MESES];
};

/*info_c*/
struct info_cliente{
  int qtd;
  double faturacao;
};
```

```c
/*FILIAL_produtos*/
struct filial_p{
  Filial_Catalogo_Produtos f_produtos[FILIAL];
};

/*Filial_Catalogo_Produtos*/
struct filial_cat_produtos{
  Catalogo catalogo;
};

/*f_info_p*/
struct filial_info_produtos{
  AVL clientes_compradores[MESES];
  int qtd_total_comprada[MESES];
  double faturacao_total_mes[MESES];
};

/*info_p*/
struct info_produto{
  int tipo_compra; /* 'N' = 0 & 'P = 1' & 'N+P' = 2*/
};
```

Em relação aos produtos, primeiramente criamos uma estrutura "info produto" que contém a o tipo de compra realizada se em regime normal, em

regime promocional, ou ambos.

De seguida, relacionamos essa informação com as filiais. Assim, criamos uma estrutura "filial info produto" na qual guardamos os produtos comprados por um cliente num array de 12 posições, referentes aos 12 meses de um ano, em que cada índice aponta para uma árvore balanceada que contém os produtos comprados por cada cliente nesse mês de uma forma ordenada e onde também utilizamos um array de 12 posições, referentes aos 12 meses de um ano, no qual em cada posição guardamos a quantidade total de produtos comprada nesse mês. Para concluir esta estrutura criamos um array de doubles de 12 posições, tal como nos outros, onde guardamos a faturação total efetuado no respetivo mês.

Igualmente aos clientes, criamos uma estrutura "filial cat clientes" para guardar as informações obtidas da estrutura acima referida. Deste modo, esta estrutura é formada por um catálogo, ou seja, um array de 26 posições em que cada índice do array, referente a uma letra do alfabeto, aponta para uma árvore balanceada que guarda as informações dos produtos começados por essa mesma letra.

Finalmente, elaboramos a última estrutura "filial p" que consiste num array de 3 posições, referentes às 3 filiais, em que cada índice do mesmo aponta para um array de 26 posições, referentes às letras do alfabeto, que funcionam como explicado na estrutura acima e, desta forma, ficam organizados por filial.

4 Queries

4.1 Query 1

Nesta query, relativa ao load dos ficheiros, usamos funções de inserção em árvores binárias balanceadas o que nos permite ordenar a informação, no nosso caso, pelos códigos de produtos e/ou clientes, dependendo da estrutura. Após validação dos códigos de produtos e clientes estes são inseridos nas diversas estruturas, nomeadamente catálogos de produtos e no de clientes, no catalálogo da faturação e nos dois catálogos das filiais, um organizado por produtos e outro por clientes. Todos estes são armazenados no SGV juntamente com o número total de produtos, o número total de clientes, o número de vendas válidas lidas e as válidas, e as diretorias de onde os ficheiros foram lidos. Existe também uma flag de validade para não permitir

a utilização das restantes queries sem o prévio load dos ficheiros, bem como para o caso de ser requisitado um novo load, ele destruir toda a informação antes alocada.

4.2 Query 2

Esta query pede uma listagem de todos os produtos começados por uma certa letra e o seu número total. Utilizando o catálogo dos produtos, calculamos o índex a aceder através do input do utilizador, onde contabilizamos o número total de produtos através da altura da árvore balanceada e inserimos o código de cada produto num array de strings de uma estrutura auxiliar.

4.3 Query 3

Esta query pretende obter o número total de vendas e o total faturado de um certo produto, num dado mês, devendo distinguir tais valores de acordo com o modo em que foi realizada, N (normal) ou P (promoção). Assim, fazendo uso das matrizes tridimensionais no catalogo da faturação, calculamos facilmente os valores pretendidos e alocamos uma estrutura criada para este efeito. Tal procedimento é feito quer para o cálculo do número de vendas, quer para o total faturado, nos dois modos.

4.4 Query 4

Atendendo ao pedido da query, que é o de obter a lista ordenada dos códigos dos produtos que ninguém comprou, bem como o seu número total, e fazer a distinção entre global e por filial, o grupo decidiu usar a seguinte estratégia. No caso de o utilizador pretender os produtos não comprados a nível global, o programa percorre o catálogo da faturação, e passando por todas as suas posições, isto é por cada uma das 26 letras do alfabeto, faz uma travessia inorder da árvore destinada, e cada vez que encontrar uma estrutura a NULL, uma vez que é assim que inicializamos cada estrutura de cada produto nessa árvore, então o produto não foi comprado,e é inserido num array de strings de uma estrutura auxiliar. Para contar o total de produtos, utilizamos um contador recursivo na chamadas das funções. Para cada filial o procedimento é o mesmo, mas ao invés de percorrer o catálogo da faturação, percorremoso catálogo das filais de produtos.

4.5 Query 5

Para esta questão é-nos pedida a lista ordenada de códigos dos clientes que compraram em todas as filiais. Para tal efeito averiguamos que bastava aceder à primeira filial de clientes e, percorrendo o catálogo por completo, inserir num array de strings de uma estrutura auxiliar todos os clientes que, com o auxilio de uma função própria, eram confirmados como clientes que compraram em todas as filiais.

4.6 Query 6

Esta query pretende obter o número total de produtos que ninguém comprou, bem como o número de clientes registados que não realizaram nenhuma compra. Relativamente ao número de produtos que ninguém comprou, simplesmente é realizada uma travessia da àrvore de cada posição do catálogo da faturação e averiguada se a estrutura em cada nodo é NULL, aumentando assim o contador recursivo. Já o número de clientes é calculado de forma similar. É realizada uma travessia de cada árvore do catálogo de cada filial, e através de uma função auxiliar, verificamos se o cliente fez ou não compras. Caso não tenha feito alguma compra, o contador recursivo é incrementado.

4.7 Query 7

Nesta query é-nos pedido que apresentemos uma tabela com o total de produtos comprados (a soma das quantidades)em cada mês, e distinguindo as filiais. Como recebemos um código de cliente, somos capazes de, no catálogo de cada filial aceder diretamente ao nodo do cliente na árvore e obter a estrutura. Caso a estrutura não seja NULL, alocamos numa matriz bidimensional, na filial e no mês especifico, de uma estrutura auxiliar, o total de produtos correspondente.

4.8 Query 8

De forma a responder à query, calcular o total faturado e o total de vendas registadas entre um intervalo de dois meses dados pelo utilizador, decidimos que em cada posição do catálogo da faturação é feita uma travessia posorder onde é calculado o total faturado acedendo à estrutura de cada produto,

e é contabilizado o total de vendas registadas fazendo uso de um contador recursivo.

4.9 Query 9

Nesta query, dado um produto e uma filial, tivemos que determinar os clientes, e o número total, que o compraram, distinguindo entre os dois modos de venda, normal ou promocional. Para tal acedemos diretamente ao nodo do produto no respetivo catálogo da filial requisitada e, acedendo à estrutura desse produto, percorremos a AVL dos clientes que o compraram, fazendo uma travessia inorder da mesma, de forma a inserir os clientes num dos arrays de strings de uma estrutura auxiliar, dependendo do tipo de compra dado pelo inteiro na estrutura de cada cliente comprador. Assim , se o inteiro for igual a 0, corresponde a N, se for 1 corresponde a P, e se for 2 significa que o cliente comprou esse produto das duas formas. O total de clientes é dado pelo tamanho da AVL dos clientes compradores.

4.10 Query 10

O objetivo desta query era, dado um código de cliente e um mês, determinar a lista de produtos que o cliente mais comprou, por quantidade. Desta forma, acedemos diretamente ao nodo do cliente dentro do catálogo de cada filial, e fizemos uma travessia inorder dos produtos por ele comprados, inserindo o seu código juntamente com a quantidade comprada desse produto numa estrutura auxiliar. Usamos uma função que ordena um array dessas estruturas pela quantidade comprada de cada produto e retornamos ao utilizador esse resultado.

4.11 Query 11

De forma a responder a esta query e obter a lista dos produtos mais vendidos, começamos por percorrer o catálogo da faturação e, de seguida, cada árvore de forma inorder inserindo numa estrutura auxiliar tanto o código de produto, bem como a quantidade que foi comprada desse produto em cada filial ao longo dos 12 meses. Seguidamente, ordenamos um array dessas estruturas e percorrendo o catálogo das filiais correspondente ao produto, para o número pedido, averiguamos o número de clientes que comprou cada produto e atualizamos na estrutura auxiliar.

4.12 Query 12

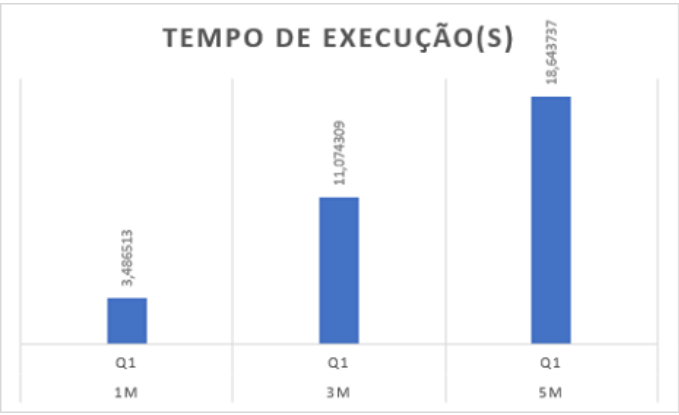
Esta query pede que determinemos, dado um cliente, uma lista de produtos com o número total pedido pelo utilizador, ordenada pelo total faturado, de forma decrescente, durante o ano. Para tal, acedemos diretamente ao nodo do cliente em cada filial e fizemos uma travessia inorder da AVL dos produtos comprados inserindo numa estrutura auxiliar o código do cliente e o valor gasto nesse produto.

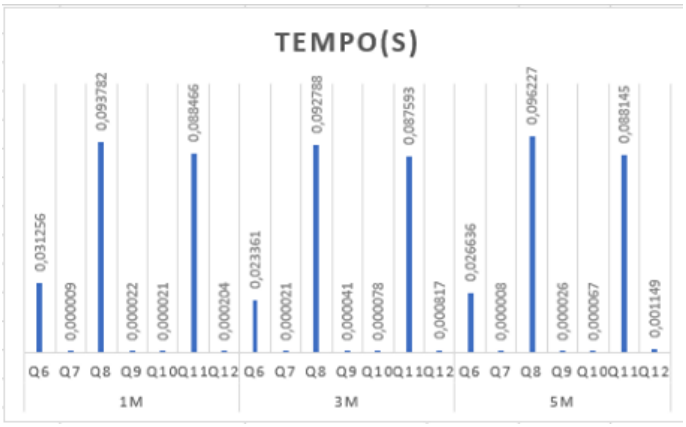
4.13 Query 13

Por fim, a query 13 pede que mostremos ao utilizador as informações lidas aquando do load dos ficheiros, nomeadamente os caminhos para os ficheiros, a quantidade total de produtos, a quantidade total de clientes, o número de vendas lindas e o número de vendas válidas. A estratégia que decidimos utilizar para esta última querie foi a de criar variáveis no sgv que guardem estas informações e que depois acedemos para apresentar ao utilizador.

5 Testes e Resultados

Depois de todo o trabalho estar concluído, fizemos testes para os ficheiros de 1, 3 e 5 milhões. Os resultados obtidos estão apresentados, pela mesma ordem, nas tabelas abaixo.





6 Conclusão

Finalizado o projeto, retiramos algumas conclusões essenciais. Uma foi o uso da modularidade e abstração de dados de modo a obtermos um código seguro, de maneira a proteger não só do utilizador como de alguém que pretenda aceder e alterar dados. Outra conclusão foi a de o uso de estruturas auxiliares ter facilitado tanto o acesso às informações como a comunicação entre o módulo e a vista. Ao longo da realização do projeto também nos deparamos com alguns problemas. Um deles não conseguimos resolver, sendo um segmentation fault no sistema operativo macOS na função "altera faturação". Apesar de todas as tentativas de modificação do código e outras estratégias (recorrendo também ao uso de diferentes debuggers e valgrind) não conseguimos perceber qual era o erro que ocorria neste sistema operativo sendo que no linux não demonstrava problemas. Finalmente, não pudemos deixar de abordar o facto de que a realização do trabalho na linguagem de programação C não facilita o trabalho do programador, obrigando-nos a criar quase tudo de raiz. Assim sendo, esperamos que na próxima fase do projeto, sendo a sua realização em Java, que carece de mais funcionalidades, o trabalho seja mais simples de realizar.