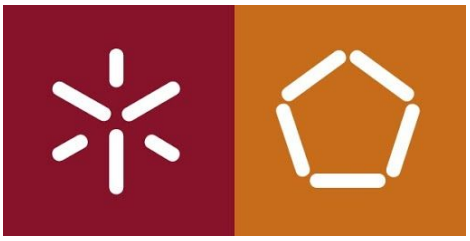


Escola de Engenharia
Departamento Informática
Mestrado Integrado em Engenharia Informática
Laboratórios de Informática III

Sistema de Gestão e Vendas

Grupo 50
Fernando Henrique Carvalho Lopes (A89472)
Luis Guilherme Guimarães de Araújo (A86772)
Pedro Almeida Fernandes (A89574)

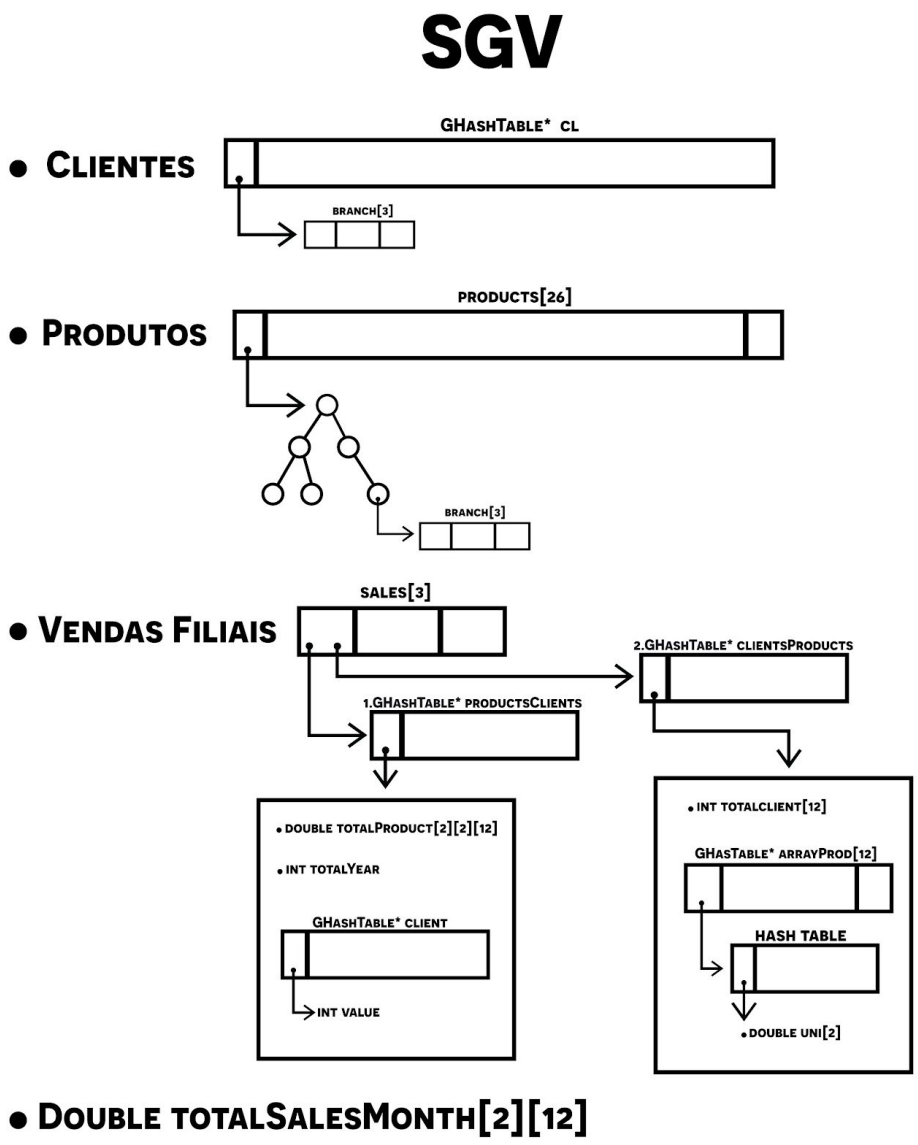
Braga, abril 2020

Índice

Introdução

Neste projeto tínhamos como objetivo realizar um sistema de gestão de vendas. Para o qual tivemos de usar estruturas de dados mais complexas do que o habitual, uma vez que a quantidade de dados com que tínhamos de trabalhar era de grandes proporções. Deste modo, as estruturas de dados mais básicas, como por exemplo simples arrays, não conseguiam suportar o tamanho da dimensão do ficheiro de vendas. Assim sendo, este trabalho baseou-se em encontrar uma resposta eficaz a este problema, para o qual foram usadas diversas estruturas de dados presentes na biblioteca GLib, entre as quais árvores balanceadas e *hash tables*.

Modularidade da estrutura de base de dados



Esta modularidade da estrutura geral (SGV) foi pensada com o intuito de dar resposta às perguntas/queries propostas pelos docentes da UC. Para as quais necessitamos de um acesso rápido e cirúrgico às informações pretendidas. Devido a grande quantidade de acessos aos clientes, decidimos usar uma *hash ta*ble. Para a necessidade de ordenação dos produtos, optamos por usar um array de árvores balanceadas uma vez que com a sua procura in order conseguimos obter todos os produtos por ordem alfabética. Como as vendas eram um ponto importante das queries em questão, foi necessário uma maior complexidade para obter de forma mais rápida e direta as informações necessárias aos problemas (informação mais detalhada no respectivo *header file*).

SGV.h

Ficheiro de modelação da estrutura de base de dados global. Este possui as funções de validação/obtenção de informação quer estas sejam produtos, clientes ou vendas.

É aqui que implementámos a estrutura **sgv**, a qual é a base para todas as tarefas propostas pelos docentes.

- **Int linesValidatedAndRead[6]** - Informação acerca da leitura das linhas e validação das mesmas.
    - Índice 0 - Número de linhas lidas no ficheiro dos clientes.
    - Índice 1 - Número de linhas validadas no ficheiro dos clientes.
    - Índice 2 - Número de linhas lidas no ficheiro dos produtos.
    - Índice 3 - Número de linhas validadas no ficheiro dos produtos.
    - Índice 4 - Número de linhas lidas no ficheiro das vendas.
    - Índice 5 - Número de linhas validadas no ficheiro das vendas.

- **CLIENTS clients** - Estrutura que guarda a informação dos clientes.

- **PRODUCTS products** - Estrutura que guarda a informação dos produtos.

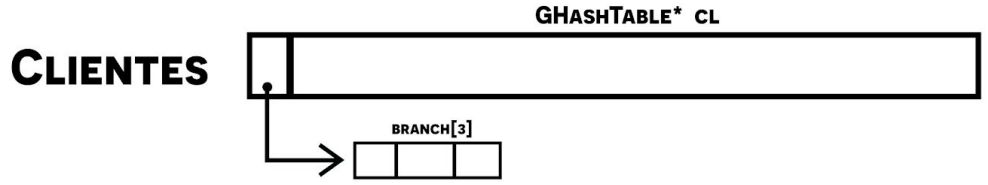
- **SALES sales** - Estrutura que guarda a informação das vendas.

- **Double totalSalesMonth[2][12]** - Array bidimensional que armazena informação total das vendas.
    - 1º Array - contém no índice 0 a faturação e no índice 1 o número de vendas.
    - 2º Array - cada índice corresponde a um mês.

```
struct sgv {
    int linesValidatedAndRead[6];
    CLIENTS clients;
    PRODUCTS products[26];
    SALES sales[3];
    double totalSalesMonth[2][12];
};
```

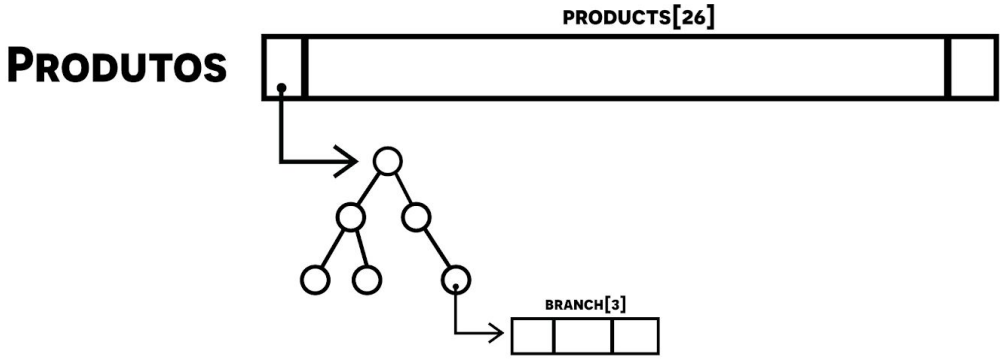
Clients.h

Este ficheiro contém a estrutura de manipulação de clientes. Optamos por guardar a informação relativa aos clientes numa *hash table*, visto que os mesmos serão acedidos várias vezes no decorrer da validação das vendas e devido ao seu rápido acesso. A estrutura é formada por uma key e um value, sendo a key o próprio código do cliente e o value um array de 3 posições, uma para cada filial, onde guardamos a informação relativa à efetuação de compras por parte desse cliente nessa filial. Decidimos desta forma para uma fácil detecção de compras por parte de um clientes numa filial.



Products.h

Este ficheiro contém a estrutura de manipulação de produtos. Decidimos usar um array de 26 posições, uma para cada letra do alfabeto, contendo em cada índice uma árvore balanceada cuja key representa um produto e o value um array de três posições, um para cada filial, contendo um valor que representa a realização de uma compra ou não, tal como temos na estrutura dos clientes. Esta decisão foi feita de modo a organizar os produtos por ordem alfabética e ainda assim, com a divisão em letras, continuar com uma alta rapidez no acesso aos mesmos.

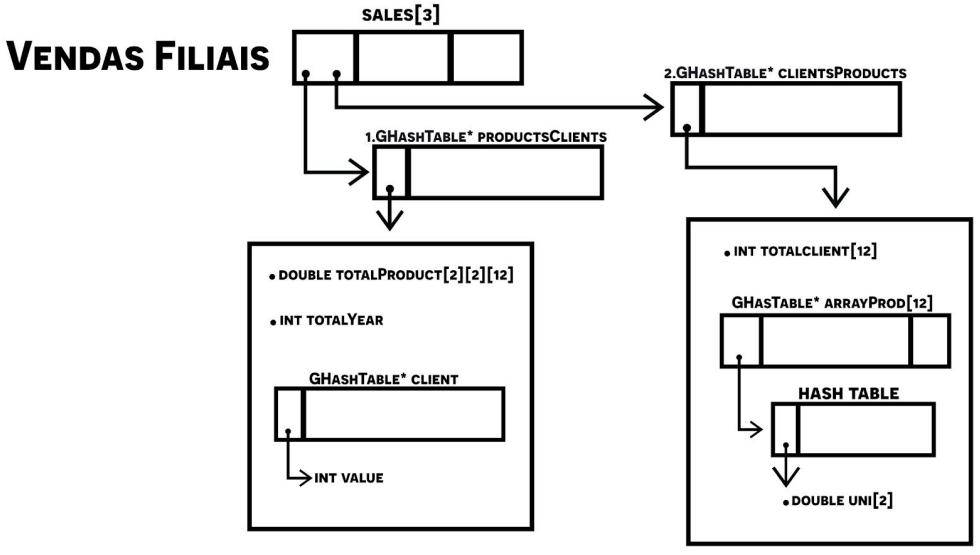


Sales.h

Este ficheiro contém a estrutura de manipulação de vendas. Elaboramos para esta fase do projeto uma estrutura que fosse de rápido acesso e que nos permitisse realizar as tarefas, apresentadas pelos docentes, de uma forma mais direta. Para tal escolhemos dividir as vendas num array de três posições, um para cada filial. Nestas posições será introduzida uma estrutura, SALES, contendo duas *hash tables*, uma que relaciona clientes com produtos, e outra produtos com clientes.

- ***Hash Table* 1 (productsClients)** é constituída por uma key, o qual corresponde ao código de um produto, e pelo seguinte value, estrutura **prodCl**:
    - **Double totalProduct[2][2][12]** - Neste array tridimensional está contida a informação do produto dividida em modo de compra, número de vendas ou faturação e por fim meses. Está representado o modo de compra no primeiro array, compra normal (índice 0) ou promoção (índice 1). No segundo array está representado a faturação (índice 0) e o número de vendas (índice 1). Por fim, no terceiro array está dividido por meses.
    - **Int totalYear** - Contém o número total de unidades compradas do produto.
    - **GHashTable* client** - Inclui todos os clientes que compraram este produto. A key corresponde ao cliente e o value ao modo de compra, compra normal (valor 1), compra em promoção (valor 2), ambos (valor 3).

- ***Hash Table* 2 (clientsProducts)** é constituída por uma key, o qual corresponde ao código de um cliente, e pelo seguinte value, estrutura **clProd**:
    - **Int totalClient[12]** - Array dividido em meses, que possui o número de unidades compradas por um cliente.
    - **GHastable* arrayProd[12]** - Esta estrutura contém um array de 12 posições, correspondentes aos meses. Em cada índice existe uma *hash table* cuja key é um código de produto que o cliente comprou, e o value, **uni[2]**, um array de duas posições, na qual a posição 0 representa a faturação e na posição 1 as unidades compradas, resultantes da interação do cliente com o produto.



Defines.h

Neste *header file* estão contidos todos os defines usados para a realização do menu, de forma a facilitar a compreensão e aumentar a simplicidade do código do mesmo.

Files.h

Neste *header file* estão as funções necessárias para a verificação e obtenção dos ficheiros introduzidos pelo utilizador.

Gets.h

Neste *header file* faz-se a gestão de variáveis apresentadas pelo utilizador para parâmetros, utilizados no menu 1.

Interface.h

Neste *header file* estão realizadas todas a queries realizadas no decorrer do projeto de gestão de vendas.

Menu.h

Neste *header file* estão presentes ambas as implementações de menus do trabalho. Tudo o que é de parte gráfica e embelezamento do trabalho encontra-se aqui.

Resultados dos Testes Realizados

Queries	Vendas1_M	Vendas3_M	Vendas5_M	Input
Query 1	2,304430	7.074237	15.990111	Dados iniciais
Query 2	0,001584	0.000857	0.000955	Letra I
Query 3	0,000005	0.000003	0.000004	AA1415 ; 7
Query 4.1	0,042417	0.016294	0.039685	Filial 1
Query 4.2	0,026870	0.027232	0.022245	Filial 2
Query 4.3	0,029220	0.029970	0.021850	Filial 3
Query 4.4	0,029759	0.018634	0.023016	Global
Query 5	0,003140	0.003211	0.003222	-
Query 6	0,013623	0.009532	0.011305	-
Query 7	0,000003	0.000002	0.000003	A1415
Query 8	0,000001	0,000001	0.000001	1...12
Query 9	0,000006	0,000140	0.000022	AA1415 ; 3
Query 10	0,000022	0,000025	0.000028	A1415 ; 7
Query 11	0,284243	0,269487	0.270325	300000
Query 12	0,000063	0,002168	0.000294	A1415 ; 300000
Destruição	1,135049	2,378877	3.788626	-

Conclusão

Com este trabalho aprendemos que que há sempre maneiras de melhorar a eficiência do nosso código. Durante a sua realização fomos modificando as estruturas de dados e funções diversas vezes de modo a conseguir uma melhor solução perante todos os obstáculos que fomos encontrando.