

Universidade do Minho
Departamento de Engenharia Informática

Relatório do Projecto de LI3

2018/2019



André Martins A84347



Luís Maia A84241

12 de Abril de 2019

Índice

| | | |
|----------|--|----------|
| 1 | Introdução | 2 |
| 2 | Objectivos | 2 |
| 3 | Estruturas e Análise dos Resultados | 3 |
| 4 | Conclusão | 7 |
| | Bibliografia | 8 |

1 Introdução

O projecto de C da disciplina de Laboratórios de Informática 3 tem por objectivo a exploração de forma pragmática, os desafios que se colocam a quem concebe e programa aplicações software (em qualquer linguagem), quando passamos a realizar a designada programação em larga escala, ou seja, aplicações com grandes volumes de dados e com mais elevada complexidade algorítmica e estrutural. Para que tal seja possível teremos que introduzir novos princípios de programação, mais adequados à programação em grande escala, designadamente: a modularidade e encapsulamento de dados usando construções da linguagem, a criação de código reutilizável, a escolha optimizada das estruturas de dados e reutilização e testes de performance.

2 Objectivos

O objectivo deste projecto, foi não só a sua realização, como também foi prioridade, o desenvolvimento das nossas capacidade de programação, modulação do código e reutilização do mesmo. O desenvolvimento destas capacidades, trará no futuro, a realização de melhores projectos, quer na questão de como são feitos, bem como no resultado final.

3 Estruturas e Análise dos Resultados

No inicio do projecto, começamos por implementar a leitura dos ficheiros e a validação dos seus dados, para isto usamos arrays, rapidamente reparamos que não seria a melhor solução devido a sua ineficiência, tempo de execução (load dos dados) tinha uma média de 350 segundos. Depois de discutirmos em grupo qual seria a estrutura, ou combinação de estruturas que traria mais vantagens em geral, principalmente na questão do tempo de tratamento de dados dos ficheiros, e por conseguinte, das queries propostas.

Escolhemos então usar árvores binárias de busca balanceadas (AVLs), da biblioteca GLib, combinadas com arrays. Estas estruturas têm respectivamente uma complexidade de $O(\log n)$ e $O(n)$.

No modulo de produtos foi usado, um array de tamanho 26 para representar as letras do alfabeto e em cada índice, esta presente uma AVL, com apenas produtos começados com essa letra. No modulo de clientes foi usado o mesmo tipo de estrutura para o tratamento dos dados. Nas vendas, pusemos os vários campos da venda quando validados numa struct para permitir uma mais fácil modularidade de dados. Com esta combinação de estruturas conseguimos diminuir o tempo de execução (load dos dados), para uma média de 3 segundos, uma grande melhoria comparado com o tempo da versão anterior.

```
//Estrutura de vendas
typedef struct vendas{
    char* prod;
    double preco;
    int unidades;
    char* tcompra;
    char* cliente;
    int mes;
    int filial;
}*Vendas;
```

Nesta estrutura é guardado cada elemento de uma linha de venda, um código produto (string), um preço, unidades, o tipo de compra, o código do cliente, o mês da compra e a filial onde a compra foi efectuada.

Para as queries foram usadas estruturas apropriadas para cada uma delas.

```
//Estrutura da querie 3
typedef struct q3{
    char* produto[7];
    char* tcompra[2];
    int nvendas;
    double lucro;
}*Q3;
```

Estrutura da querie 3 contem uma string do produto, o tipo de compra, o numero de vendas e o lucro total.

```
//Estrutura da querie 9
typedef struct q9{
    char* clientes;
    char* tcompra;
}*Q9;
```

Estrutura da querie 9 contem uma string de clientes e o tipo de compra.

```
//Estrutura da querie 10
typedef struct q10{
    char* produto;
    int nvendas;
}*Q10;
```

Estrutura da querie 10 contem uma string de produtos e o números de vendas desse mesmo produto.

```
//Estrutura da querie 7
typedef struct q7{
    int mes[12];
}*Q7;
```

Estrutura da querie 7 contem um array em que os índices representam os meses.

Querie 1 :

Consiste em mudar um ficheiro, quanto o programa estiver a ser executado, não podendo ser reiniciado. Nesta querie é chamado, dependendo do ficheiro, a função que trata os dados da mesma. Apresenta o nome do ficheiro lido e o numero de clientes/produtos ou vendas validos.

Querie 2 :

Dado uma letra como input, apresenta 20 produtos de cada vez, começados por essa letra. Contem também um menu para ser possível navegar pelo resultado.

Querie 3 :

Dado um mês e um produto, é apresentado, o total de vendas e o total facturado, filial a filial ou globalmente.

Querie 5 :

Não necessita de input porque, os dados necessários já se encontram numa GTree, devolvendo a lista de clientes que realizaram compras nas 3 filiais, ordenada por ordem alfabética.

Querie 6 :

Determina o número de clientes registados que não realizaram compras bem como o número de produtos que ninguém comprou.

Querie 7 :

Dado um código de cliente, devolve uma tabela com o numero de compras desse cliente, separados por meses. apresenta também o numero total de produtos comprados.

Querie 8 :

Num intervalo de dois meses, devolve o numero de vendas e a facturação total desse tempo.

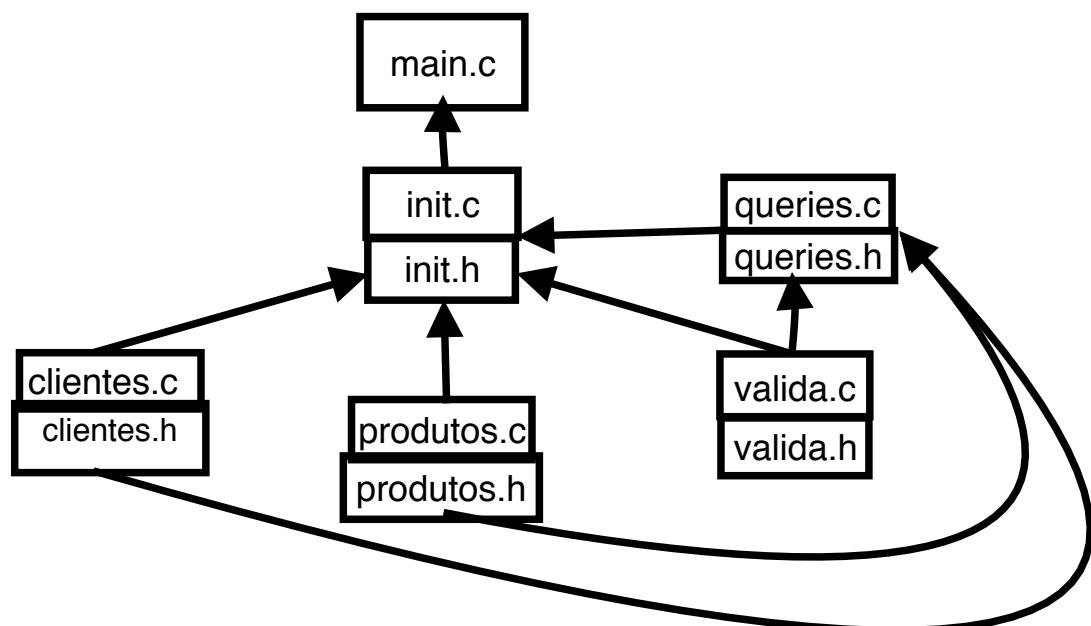
Querie 9 :

Dado um código de produto e uma filial, determina os códigos e o número total dos clientes que o compraram, diferenciando o tipo de compra (P ou N).

Querie 10 :

Dado um cliente e um mês, determina a lista de produtos que o cliente comprou nesse mês, caso tenho comprado.

Gráfico de dependências:



4 Conclusão

Apesar de termos chegado ao fim do projecto, concluímos que poderíamos ter tido melhor desempenho se, por exemplo, nas queries, em vez de fornecermos como input o array de estrutura de vendas em cada uma das queries, e assim percorrer o array para obter o resultado; deveríamos, ao fazer a validação das vendas ter colocado a informação que cada querie precisa, na estrutura própria da querie.

Quanto a modulação tambem poderia ter sido melhor planeada e executada. No geral o projecto correu relativamente bem, e conseguimos um bom resultado.

Bibliografia

<https://developer.gnome.org/glib/>