**Base de Dados 2021/2022**

**Projeto BD - Parte 3**

Grupo nº 137, Laboratório BD2L10, Professor Flávio Martins

|  |  |  |  |
| --- | --- | --- | --- |
| Aluno | Número | Percentagem (%) | Esforço Total (h) |
| Rúben Nobre | 99321 | 33 | 10 |
| Alexandra Pato | 97375 | 33 | 10 |
| Teresa Costa | 99177 | 33 | 10 |

**Arquitetura da aplicação web**

A aplicação web encontra-se disponível online no seguinte link: <https://web2.tecnico.ulisboa.pt/ist199321/app.cgi/>.

Para o desenho da mesma optou-se por minimizar o número de páginas web, sendo que a nossa aplicação apenas recorre a três ficheiros .html.

Ao carregar no link mencionado é levado à pagina inicial onde é possível ver-se todos os retalhistas na base de dados, também como inserir e remover retalhistas, removendo também os seus produtos.

Ao carregar no botão “IVMs” no topo da página o utilizador é levado à página onde é possível ver todas as IVMs na base de dados.

Ao carregar no botão “Mais info.” referente a uma das IVMs listadas o layout da página sofre uma alteração: agora também aparecem as informações referentes aos eventos de reposição da IVM solicitada também como os produtos por categoria dessa IVM.

Ao carregar no botão “Categories” no topo da página o utilizador é levado à página onde é possível ver todas as categorias, remover uma categoria, adicionar uma nova categoria, e adicionar uma nova sub-categoria, tendo a super-categoria que ser especificada.

**Relações entre os vários ficheiros**

1. O ficheiro populate.sql contém:
   1. A informação para a criação das tabelas para preencher a base de dados;
   2. Algumas implementações das IC de ICs.sql (pois caso contrário uma categoria não seria automaticamente adicionada a um produto por has\_category aquando da criação do produto))
   3. A informação para a inserção das entradas de teste nas tabelas criadas
2. O ficheiro ICs.sql deve ser executado após populate.sql. Contém:
   1. Triggers e funções para garantir cumprimento das IC das várias tabelas de **1.a**: RI-1, RI-4, RI-5, e RI-RE6
3. O ficheiro view.sql deve ser executado após populate.sql. Contém:
   1. A vista para as Vendas, de acordo com o ponto **4. Vistas**
   2. A vista para os eventos de reposição de uma dada ivm, para uso no website
4. O ficheiro queries.sql deve ser executado após populate.sql. Contém:
   1. O código sql para as consultas sucintas à base de dados, de acordo com o pedido pelo ponto **3. SQL**
5. O ficheiro analytics.sql deve ser executado após populate.sql. Contém:
   1. O código para as consultas OLAP a partir da vista de **3.a**, de acordo com o ponto **6. Consultas OLAP**
6. Os ficheiros web/templates/\*.html contém templates para as várias páginas do site, de funcionamento já referido em **Arquitetura da Aplicação Web**
7. O ficheiro web/app.cgi contém as funções necessárias para acesso à base de dados.

**Índices**

Assumindo que as entradas de cada tabela são relativamente

estáticas e não há modificações frequentes:

**7.1**

CREATE INDEX retailer\_name\_index ON retailer USING hash(retailer\_name);

* hash(nome) em retalhista:   
   Em "DISTINCT", compara-se apenas os valores de igual hash code (e, por isso, de menores sub-grupos da tabela) para averiguar igualdade, diminuindo-se o número de entradas a comparar/percorrer e por isso dimiuindo o tempo de execução

CREATE INDEX category\_name\_index ON responsible\_for USING hash(category\_name);

* hash(nome\_cat) em responsavel\_por:   
   Compara-se apenas categorias com o mesmo hash code (e, por isso, de menores sub-grupos da tabela) que 'Frutos', diminuindo-se o número de entradas a comparar/percorrer e por isso dimiuindo o tempo de execução

CREATE INDEX responsible\_for\_tin\_index ON responsible\_for USING hash(retailer\_tin);

* hash(tin) em responsavel\_por:   
   Compara-se apenas os tin de responsavel\_por com o mesmo hash code (e, por isso, de menores sub-grupos da tabela) que o do tin dos retalhistas, diminuindo-se o número de entradas a comparar/percorrer e por isso dimiuindo o tempo de execução

CREATE INDEX retailer\_tin\_index ON retailer USING hash(retailer\_tin);

* hash(tin) seria preferível ao default btree(tin) em retalhista:  
   Compara-se apenas os tin de responsavel\_por com o mesmo hash code (e, por isso, de menores sub-grupos da tabela) que o do tin dos retalhistas, diminuindo-se o número de entradas a comparar/percorrer e por isso dimiuindo o tempo de execução

**7.2**

CREATE INDEX product\_category\_name\_index ON product USING hash(category\_name);

CREATE INDEX has\_category\_name\_index ON has\_category USING hash(category\_name);

* hash(cat) em produto, hash(nome) em tem\_categoria:  
   Usando a mesma hash function, facilita-se comparação “P.cat = T.nome” e o “GROUP BY” pois reduz-se a comparação e a divisão dos agrupamentos apenas a entradas com um mesmo hash code (e, por isso, de menores sub-grupos das suas respetivas tabelas), diminuindo-se o número de entradas a comparar/percorrer e por isso dimiuindo o tempo de execução

CREATE INDEX product\_descr\_index ON product(product\_descr);

* btree(desc) em produto:  
   Usando btree, reduz-se procura a entradas no ramo da árvore de palavras iniciadas por 'A' que, sendo a primeira letra do alfabeto, é um ramo que não demora muito a ser encontrado quando comparado a ramos mais elaborados; assim, diminui-se o número de ramos a percorrer e entradas a comparar/percorrer e por isso dimiui-se o tempo de execução

**Observações**

* Considerámos, tanto por lógica como por simplificação, que um produto apenas pode ter categorias simples
* Considerámos que, ao apagar uma categoria, o seu ascendente direto deveria tornar-se antecessor das descendentes diretas da categoria apagada
* Considerámos que apenas deveria ser considerada a categoria principal de uma prateleira e não os seus descendentes, aquando da colocação de um produto numa prateleira (ex: produto de categoria “Iogurte” pode ser colocado numa prateleira “Iogurte”, mas não numa prateleira “Laticínios”)