**Projeto de Bases de Dados**

**Parte 3**

**Turno Prático 04,** Prof. Rodrigo Sousa

|  |  |  |  |
| --- | --- | --- | --- |
| Grupo 67 | | | |
| Nome | Nº | Esforço (Horas) | Contribuição (%) |
| Joana Brito | 96037 | 20 | 33% |
| Gonçalo Gonçalves | 99226 | 20 | 33% |
| João Marques | 99243 | 20 | 33% |

## 

### 1. Base de Dados

A base de dados e o código necessário ao preenchimento da mesma com dados de teste é apresentado no ficheiro populate.sql.

### 2. Restrições de Integridade

No ficheiro ICs.sql é possível observar o código relativo à implementação das restrições de integridade. Recorreu-se à utilização de Triggers para a verificação das condições pedidas antes da inserção ou atualização de certos dados numa determinada tabela.

### 3. SQL

No ficheiro queries.sql são apresentadas as consultas SQL para cada uma das situações apresentadas no enunciado.

### 4. View

O ficheiro view.sql contém as instruções para a criação da *view*, que resume as informações mais importantes sobre as vendas, combinando informações de diferentes tabelas do modelo. A função dow (“day of week”) foi chamada para a seleção do dia da semana de cada *instance*.

### 5. Desenvolvimento da Aplicação

De seguida segue a explicação da arquitetura web, mais precisamente, como efetuar as operações na aplicação desenvolvida.

Inserir Categoria

Ir ao menu principal, carregar no botão “Categoria”, que leva ao sub-menu das categorias. Dentro deste menu, é possível pressionar o botão "Inserir Categoria", escrever o nome da categoria e submeter para que esta seja criada.

Nota: Ao inserir uma categoria assume-se sempre que esta é uma categoria simples, pelo que apenas se insere a nova categoria nas tabelas “Categoria” e “Categoria Simples”.

Remover Categoria

Ir ao menu principal, carregar no botão “Categoria”, que nos leva ao sub-menu das categorias. Dentro este menu, carregar no botão "Remover Categoria" e escrever o nome da categoria e submeter para que esta seja removida.

Nota: Ao remover uma categoria também são apagados todos os dados onde o nome da categoria são Foreign Key, através de ON DELETE CASCADE.

Inserir Retalhista

Ir ao menu principal, carregar em “Retalhista”, que nos leva ao sub-menu dos retalhistas. Dentro do menu, carregar em "Inserir Retalhista" e escrever o *tin* e nome do retalhista para que este seja criado.

Remover Retalhista

Ir ao menu principal, carregar em Retalhista, que nos leva ao sub-menu dos retalhistas. No sub-menu, carregar no botão "Remover Retalhista", escrever o *tin* do retalhista a remover e submeter.

Nota: Ao remover um retalhista é removido não só o retalhista, mas também todos os eventos de reposição feitos por este e todas as relações *responsavel\_por* com o seu *tin*.

Listar todos os eventos de reposição de uma IVM

Ir ao menu principal, carregar no botão “IVM”, que nos leva ao sub-menu das IVMs. Dentro do menu das IVMs, inserir o número de série e o fabricante da IVM cujos eventos de reposição queremos listar e, por fim, carregar no botão “Listar”.

Listar todas as sub-categorias de uma super-categoria

Ir ao menu principal, carregar no botão “Categoria”, que nos leva ao sub-menu das Categorias. Dentro do menu das categorias, carregar no botão “Listar Categorias” e inserir o nome da categoria cujas sub-categorias queremos listar. Por último, carregar no botão “Listar sub-categorias”.

Atenção: não foi possível desenvolver esta operação de forma completa, visto que o botão está implementado, mas não executa a operação correta, listando todas as categorias.

### 6. Consultas OLAP

Para as consultas de análise de dados foi submetido o ficheiro analytics.sql. A solução apresentada recorre às instruções ROLLUP e GROUP BY.

### 7. Índices

De seguida são apresentadas as instruções de criação dos índices SQL de forma a melhorar os tempos de consulta para cada um dos casos requeridos.

drop INDEX if exists idx\_nome\_retalhista;

drop INDEX if exists idx\_nome\_categoria;

drop INDEX if exists idx\_descr;

drop INDEX if exists idx\_nome;

1.

CREATE INDEX idx\_nome\_retalhista ON retalhista USING hash(nome\_retalhista);

CREATE INDEX idx\_nome\_categoria ON responsavel\_por USING hash(nome\_categoria);

2.

CREATE INDEX idx\_descr ON produto USING hash(descr);

CREATE INDEX idx\_nome ON tem\_categoria USING hash(nome\_categoria);

Justificação:

CREATE INDEX idx\_nome\_retalhista ON retalhista USING hash(nome\_retalhista);

* hash(nome\_retalista) 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, diminuindo o tempo de execução.

CREATE INDEX idx\_nome\_categoria ON responsavel\_por USING hash(nome\_categoria);

* hash(nome\_categoria) 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, diminuindo o tempo de execução.

CREATE INDEX idx\_descr ON produto USING btree(descr);

* btree(descr) em produto: usando btree, reduz-se a 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, diminui-se o tempo de execução.

CREATE INDEX idx\_nome ON tem\_categoria USING hash(nome\_categoria);

* Hash(nome\_categoria) em tem\_categoria: usando a hash function, facilita-se a comparação “P.cat = T.nome”, pois reduz a comparação apenas a entradas com o mesmo hash code (e, por isso, de menores sub-grupos da tabela), diminuindo-se o número de entradas a comparar/percorrer e, por isso, diminuindo o tempo de execução.