Bases de Dados

Instituto Superior Técnico – LEIC-A

Projeto – 3ª Entrega – Grupo 42

Turno BD2L15 – Prof. Daniela Machado

Nome	Nº de aluno	Esforço estimado (horas)
David Belchior	95550	12
Francisco Sousa	95579	12
Mariana Charneca	95635	12

1. Base de Dados

Para permitir que a inserção das entradas na base de dados cumpra as restrições de integridade, o ficheiro **populate.sql** foi desdobrado em dois ficheiros:

- O ficheiro **create_tables.sql** contém a criação de todas as tabelas da base de dados;
- O ficheiro **populate.sql** faz a inserção das entradas nas respetivas tabelas, contendo informação necessária para a demonstração do funcionamento das consultas SQL e OLAP e, igualmente, das restrições de integridades (através de entradas que violam, intencionalmente, essas restrições).

Esclarecimentos sobre as tabelas da base de dados:

- A coluna **cat** na tabela **produto** não foi definida, pois introduz informação redundante (a tabela **tem_categoria** já relaciona a tabelas em causa com a tabela **categoria**).
- A restrição **RI-RE5**, apesar de se encontrar implicitamente assegurada pela restrição **RI-RE2**, foi representada.
- Devido à existência de dependências circulares entre tabelas, as restrições de integridade **RI-RE1** e **RI-RE6** não puderam ser definidas.
    - **RI-RE1**: As tabelas **categoria_simples** e **super_categoria** dependem diretamente da tabela **categoria**, pelo que, antes da inserção de entradas nas duas primeiras tabelas implicam a existência de entradas com o mesmo **nome** na última categoria mencionada (devido à existência de uma foreign key para essa tabela). Como tal, não é possível que uma entrada na tabela **categoria** já exista, previamente, com o mesmo nome numa das outras tabelas, pelo que a restrição não pôde ser assegurada.
    - **RI-RE6:** O mesmo se aplica a esta restrição, pois a relação **tem_categoria** implica a existência de um **produto**, o qual, segundo a restrição, só poderia existir com a presença de uma entrada contendo o seu nome na tabela **tem_categoria**, o que é impossível.

2. Restrições de Integridade

O ficheiro **ICs.sql** contém as restrições de integridade pedidas no enunciado e, adicionalmente, o trigger, já mencionado, que adiciona uma entrada nova com o **nome** da nova entrada da tabela **categoria_simples** ou **super_categoria** na tabela **categoria**, caso ainda não exista.

3. SQL

As consultas SQL, acompanhadas das situações que as originaram, encontram-se no ficheiro **queries.sql**.

4. Vistas

A vista **vendas (ean, cat, ano, trimestre, dia_mes, dia_semana, distrito, concelho, unidades)** encontra-se definida no ficheiro **view.sql**.

Considerou-se que vendas que, mesmo possuindo os mesmos atributos nas colunas selecionadas na vista, tenham valores diferentes noutras colunas não exibidas são distintas entre si – por outras palavras, por exemplo, cada evento de reposição corresponde a uma venda independente.

5. Desenvolvimento da Aplicação

A aplicação que permite demonstrar algumas das operações aplicáveis sobre a base de dados, escrita em Flask (framework sobre Python) encontra-se no ficheiro **app.cgi** e pode ser acedida no website https://web2.tecnico.ulisboa.pt/~ist195579/app.cgi/.

Esta aplicação permite as seguintes operações:

- **Inserir e remover uma categoria**, dado um **nome**;

- **Inserir e remover uma subcategoria**, dado um nome e a sua **supercategoria**;
    - Na inserção, poderá ser feita a promoção de uma categoria simples para supercategoria, enquanto que, na remoção, caso uma categoria não seja supercategoria de nenhuma outra categoria, poderá ser despromovida a categoria simples.
- **Inserir um retalhista**, dado o seu TIN e nome;
- **Remover um retalhista**, dado o seu TIN;
- **Listar todos os eventos de uma reposição de uma IVM** (dado o seu número de série e o seu fabricante);
- **Listar, a todos os níveis de profundidade, todas as subcategorias de uma supercategoria** (dado o seu nome).

É importante mencionar que, em todas as operações de remoção, todas as entradas existentes noutras tabelas que dependem direta ou indiretamente (nomeadamente, por meio de foreign keys) das entradas removidas são também removidas.

6. OLAP

As consultas OLAP requisitadas, acompanhadas das situações que as originaram, encontram-se no ficheiro **analytics.sql**.

A escolha do agrupamento em **CUBE** permite uma análise relativamente profunda e relevante dos artigos vendidos, uma vez que mostra o total:

- **Em cada combinação de dia da semana e concelho**;
- **Por dia da semana**, acumulando os totais em todos os concelhos;
- **Por concelho**, acumulando o total de cada dia da semana;
- **Globalmente**, acumulando todos os totais existentes.

7. Índices

7.1. Para esta query, considerámos boa ideia utilizar um **índice hash** no **tin** na tabela **responsavel_por**.
Tendo em conta que o **tin** do retalhista é único, ao criar a hash table o tempo deprocura diminui para **O(1)**.

```
CREATE INDEX  tin_index ON responsavel_por USING hash(tin);
```

7.2. Para esta query, propomos a criação de um **índice hash** para o **nome** na tabela **tem_categoria**, pois a igualdade no **WHERE** toma vantagem do uso deste (tal como o **GROUP BY**).

```
CREATE INDEX  nome_index ON tem_categoria USING hash(nome);
```

Propomos também a criação de um **índice btree** para a **desc** na tabela **produto**, pois melhoraria a velocidade de procura de descrições começadas por 'A'.

```
CREATE INDEX  produto ON tem_categoria USING hash(desc);
```