Projeto de Bases de Dados – Parte 3

Professor Francisco Regateiro

Grupo 19 – Turno BDL05

Aluno	Percentagem relativa de contribuição (Esforço total)
Luís Xu (99100)	33,3 % (20 horas)
Artur Monteiro (99182)	33,3 % (20 horas)
Diogo Pereira (100670)	33,3 % (20 horas)

SQL:

```sql
SELECT nome
FROM (SELECT tin, COUNT(DISTINCT(nome_cat)) AS R FROM responsavel_por GROUP BY tin) AS L NATURAL JOIN retalhista
WHERE R IN (SELECT MAX(C) FROM (SELECT tin, COUNT(DISTINCT(nome_cat)) AS C FROM responsavel_por GROUP BY tin) AS LL);

SELECT DISTINCT(R.nome)
FROM retalhista R INNER JOIN (responsavel_por RP INNER JOIN categoria_simples CS ON RP.nome_cat = CS.nome) A ON A.tin = R.tin;

SELECT ean
FROM produto
WHERE ean NOT IN (SELECT ean FROM evento_reposicao);

SELECT ean
FROM (SELECT ean, COUNT(DISTINCT(tin)) AS C FROM evento_reposicao GROUP BY ean) AS B
WHERE C = 1;
```

Restrições de Integridade:

```sql
DROP TRIGGER IF EXISTS ri1 ON tem_outra;
DROP TRIGGER IF EXISTS ri4 ON evento_reposicao;
DROP TRIGGER IF EXISTS ri5 ON evento_reposicao;

CREATE OR REPLACE FUNCTION ri1()
RETURNS TRIGGER AS $$
    DECLARE nome_cat VARCHAR(80) := NEW.super_categoria;
BEGIN
    LOOP
        IF nome_cat = NEW.categoria
        THEN
            RAISE EXCEPTION 'ERRO';
        END IF;

        IF nome_cat IN (SELECT categoria FROM tem_outra)
        THEN
            SELECT super_categoria INTO nome_cat FROM tem_outra WHERE categoria = nome_cat;
        ELSE
            EXIT;
        END IF;
    END LOOP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER ri1 BEFORE INSERT OR UPDATE ON tem_outra
FOR EACH ROW EXECUTE PROCEDURE ri1();
```

```sql
CREATE OR REPLACE FUNCTION ri4() RETURNS TRIGGER AS $$
DECLARE unit INT;
BEGIN
    SELECT unidades INTO unit FROM planograma WHERE (ean = NEW.ean AND nro = NEW.nro AND num_serie = NEW.num_serie AND fabricante = NEW.fabricante);
    IF NEW.unidades > unit
    THEN
        RAISE EXCEPTION 'ERRO';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE    plpgsql;
CREATE TRIGGER ri4 BEFORE INSERT OR UPDATE ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE ri4();


CREATE OR REPLACE FUNCTION ri5() RETURNS TRIGGER AS $$
DECLARE category VARCHAR(80);
BEGIN
    SELECT cat INTO category FROM produto WHERE NEW.ean = produto.ean;
    IF category NOT IN (SELECT cat FROM produto NATURAL JOIN planograma WHERE nro = NEW.nro AND num_serie = NEW.num_serie AND fabricante = NEW.fabricante)
    THEN
        RAISE EXCEPTION 'ERRO';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER ri5 BEFORE INSERT OR UPDATE ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE ri5();
```

APP:

Todos os ficheiros necessários para o funcionamento da aplicação encontram-se na pasta web, estando os ficheiros htmls necessários na subdiretoria ‘templates’ tal como explicitado no enunciado. A página inicial (correspondente ao index.html) corresponde ao menu principal em que é apresentado ao utilizador todos os submenus que permitem a realização de todas as tarefas exigidas no enunciado. Tanto o menu ‘Tem outra’ como o menu ‘Responsável por’ não fazem nenhuma ação pedida pelo enunciado, mas achámos mais pertinente adicioná-los de modo a poder melhor visualizar algumas das tarefas implementadas. Foi também adicionado tanto a estes sub-menus como aos restantes um botão que permite voltar ao menu inicial tanto por razões estéticas como por razões de eficiência.

Para a tarefa de introdução e remoção de categorias na base de dados foram criados dois sub-menus um que nos lista o conteúdo da tabela de super-categorias e outro que lista o conteúdo das categorias-simples. Optámos por esta solução para haver simplicidade de perceção do tipo de categoria de cada uma, evitando assim uma lista com todas as categorias nela. Em cada uma destas páginas há a opção de tanto adicionar uma nova categoria como a de remover uma já existente. Para remover apenas é necessário clicar no botão enquanto para adicionar é se reencaminhado para outra página onde se pode colocar o nome da categoria nova onde se pode escrever o nome da categoria a adicionar. Na página das super-categorias é ainda possível adicionar e remover sub-categorias da categoria escolhida sendo para a primeira feito da mesma maneira que das outras e para a segunda feito com um dropdown visto que nos pareceu ser a maneira que mais evitaria erros do utilizador. É ainda possível listar as sub-categorias de uma super sendo isto feito por SQL de uma forma recursiva para apanhar todos os níveis de sub-categorias. A nível da remoção de categorias foi feita a remoção de todas as dependências, dessa categoria, existentes nas tabelas da base de dados.

De modo a inserir e remover retalhistas fizemos, tal como para as categorias, um sub-menu que lista todos os retalhistas existentes na base de dados e nos permite tanto adicionar um

novo retalhista escrevendo o seu tin e o seu nome, como remover um retalhista à escolha tal como a todos os seus produtos.

É também importante de referir que para as ações de remoção referidas acima foi escolhido fazê-lo através de um botão à frente de cada nome de tanto categoria como retalhista de modo a facilitar a escolha como de diminuir a possibilidade de erro do utilizador.

Por fim a ação de listagem das IVMs é feita novamente numa página à parte que mostra todas as IVMs presentes na base de dados. É também possível nesta página fazer a listagem dos eventos de reposição de cada IVM. Esta opção leva-nos para uma nova página onde é apresentada as categorias existentes nos eventos de reposição associados a esta IVM tal como o número de produtos total repostos dessa categoria. Escolhemos apresentar assim os dados devido à ambiguidade existente no enunciado nesta ação, tendo sido esta a forma que entendemos ser a mais correta de apresentar os dados.

O uso de um background e a maneira como todos os dados/botões existentes estão centrados na tela foi tanto uma escolha estética como de eficiência para o rato poder estar sempre centrado na tela.

Link: http://web2.tecnico.ulisboa.pt/~ist199100/test.cgi/menu

Análise de dados:

```sql
CREATE OR REPLACE VIEW vendas(ean, cat, ano, trimestre, mes, dia_mes, dia_semana, distrito, concelho, unidades) AS
    SELECT ean, cat, EXTRACT(YEAR FROM instante), EXTRACT(QUARTER FROM instante),
        EXTRACT(MONTH FROM instante), EXTRACT(DAY FROM instante),
        EXTRACT(DOW FROM instante), distrito, concelho, unidades
    FROM evento_reposicao NATURAL JOIN produto NATURAL JOIN instalada_em NATURAL JOIN ponto_de_retalho;
```

Índices:

7.1)

CREATE INDEX rindex ON responsavel_por using hash(tin);

CREATE INDEX rindex_aux ON responsavel_por using hash(nome_cat);

Foram criados dois índices para organizar as colunas tin e nome_cat com uma Hash pois as condições trata-se de comparações diretas.

7.2)

CREATE INDEX pindex ON produto using hash(cat);

CREATE INDEX pindex_aux ON produto using btree(cat, descr);

Foram criados dois índices para organizar as colunas cat e a combinação de cat e nome_cat com uma Hash e BTree, respetivamente, pois as condições consistem numa comparação direta e um intervalo de dados.