TÉCNICO LISBOA

Ano Letivo 2021/2022

Projeto BD – Parte 3

Professora Daniela Machado

24 de junho de 2022

Nome	**Número de aluno**	Contribuição relativa	Esforço total (horas)
Martim Santos	95638	33.3%	15
Diogo Adegas	96854	33.3%	15
Tomás Nunes	96915	33.3%	15

Grupo 149 – Turno L15

Base de Dados

(ficheiro populate.sql)

```sql
DROP TABLE IF EXISTS evento_reposicao;
DROP TABLE IF EXISTS responsavel_por;
DROP TABLE IF EXISTS retalhista;
DROP TABLE IF EXISTS planograma;
DROP TABLE IF EXISTS prateleira;
DROP TABLE IF EXISTS instalada_em;
DROP TABLE IF EXISTS ponto_de_retalho;
DROP TABLE IF EXISTS IVM;
DROP TABLE IF EXISTS tem_categoria;
DROP TABLE IF EXISTS produto;
DROP TABLE IF EXISTS tem_outra;
DROP TABLE IF EXISTS super_categoria;
DROP TABLE IF EXISTS categoria_simples;
DROP TABLE IF EXISTS categoria;


CREATE TABLE categoria (
    nome_cat    VARCHAR(50) NOT NULL,
    PRIMARY KEY (nome_cat)
);


CREATE TABLE categoria_simples (
    nome_cat    VARCHAR(50) NOT NULL,
    PRIMARY KEY (nome_cat),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);


CREATE TABLE super_categoria (
    nome_cat    VARCHAR(50) NOT NULL,
    PRIMARY KEY (nome_cat),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);


CREATE TABLE tem_outra (
    super_categoria     VARCHAR(50) NOT NULL,
    categoria           VARCHAR(50) NOT NULL,
    PRIMARY KEY (categoria),
    FOREIGN KEY (super_categoria) REFERENCES super_categoria(nome_cat),
    FOREIGN KEY (categoria) REFERENCES categoria(nome_cat),
    CONSTRAINT CHK_category CHECK (super_categoria != categoria) -- RI-RE5 --
);
```

```sql
CREATE TABLE produto (
    ean         CHAR(13)        NOT NULL,
    nome_cat    VARCHAR(50)     NOT NULL,
    descr       VARCHAR(200),
    PRIMARY KEY (ean),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);


CREATE TABLE tem_categoria (
    ean         CHAR(13)       NOT NULL,
    nome_cat    VARCHAR(50)    NOT NULL,
    PRIMARY KEY (ean, nome_cat),
    FOREIGN KEY (ean) REFERENCES produto(ean),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);


CREATE TABLE IVM (
    num_serie       SERIAL      NOT NULL,
    fabricante      VARCHAR(50) NOT NULL,
    PRIMARY KEY (num_serie, fabricante)
);


CREATE TABLE ponto_de_retalho (
    nome_pr     VARCHAR(50) NOT NULL,
    distrito    VARCHAR(50) NOT NULL,
    concelho    VARCHAR(50) NOT NULL,
    PRIMARY KEY (nome_pr)
);


CREATE TABLE instalada_em (
    num_serie   SERIAL      NOT NULL,
    fabricante  VARCHAR(50) NOT NULL,
    nome_pr     VARCHAR(50) NOT NULL,
    PRIMARY KEY (num_serie, fabricante),
    FOREIGN KEY (num_serie, fabricante) REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY (nome_pr) REFERENCES ponto_de_retalho(nome_pr)
);


CREATE TABLE prateleira (
    nro         INTEGER     NOT NULL,
    num_serie   SERIAL      NOT NULL,
    fabricante  VARCHAR(50) NOT NULL,
    altura      FLOAT,
    nome_cat    VARCHAR(50) NOT NULL,
```

```
    PRIMARY KEY (nro, num_serie, fabricante),
    FOREIGN KEY (num_serie, fabricante) REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);

CREATE TABLE planograma (
    ean         CHAR(13)    NOT NULL,
    nro         INTEGER     NOT NULL,
    num_serie   SERIAL      NOT NULL,
    fabricante  VARCHAR(50) NOT NULL,
    faces       INTEGER,
    unidades    INTEGER     NOT NULL,
    loc         VARCHAR(50),
    PRIMARY KEY (ean, nro, num_serie, fabricante),
    FOREIGN KEY (ean) REFERENCES produto(ean),
    FOREIGN KEY (nro, num_serie, fabricante) REFERENCES prateleira(nro, num_serie,
fabricante)
);

CREATE TABLE retalhista (
    tin         CHAR(9)     NOT NULL,
    nome_ret    VARCHAR(50) NOT NULL,
    PRIMARY KEY (tin),
    UNIQUE (nome_ret)    -- RI-RE7 --
);

CREATE TABLE responsavel_por (
    nome_cat    VARCHAR(50) NOT NULL,
    tin         CHAR(9)     NOT NULL,
    num_serie   SERIAL      NOT NULL,
    fabricante  VARCHAR(50) NOT NULL,
    PRIMARY KEY (num_serie, fabricante),
    FOREIGN KEY (num_serie, fabricante) REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY (tin) REFERENCES retalhista(tin),
    FOREIGN KEY (nome_cat) REFERENCES categoria(nome_cat)
);

CREATE TABLE evento_reposicao (
    ean         CHAR(13)    NOT NULL,
    nro         INTEGER     NOT NULL,
    num_serie   SERIAL      NOT NULL,
    fabricante  VARCHAR(50) NOT NULL,
    instante    TIMESTAMP   NOT NULL,
    unidades    INTEGER     NOT NULL,
    tin         CHAR(9)     NOT NULL,
    PRIMARY KEY (ean, nro, num_serie, fabricante, instante),
    FOREIGN KEY (ean, nro, num_serie, fabricante) REFERENCES planograma(ean, nro,
num_serie, fabricante),
    FOREIGN KEY (tin) REFERENCES retalhista(tin)                        );
```

Restrições de Integridade

(ficheiro ICs.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;

-------------------------------------------------------------
-- (RI-1) Uma Categoria não pode estar contida em si própria --
-------------------------------------------------------------

CREATE OR REPLACE FUNCTION nao_contem() RETURNS TRIGGER AS
$$
BEGIN
    IF NEW.super_categoria = NEW.categoria THEN
        RAISE EXCEPTION 'Categoria nao pode estar contida em si própria';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER ri1
BEFORE UPDATE OR INSERT ON tem_outra
FOR EACH ROW EXECUTE PROCEDURE nao_contem();

-------------------------------------------------------------------------------
-- (RI-4) O número de unidades repostas num Evento de Reposição              --
--        não pode exceder o número de unidades especificado no Planograma   --
-------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION nao_excede() RETURNS TRIGGER AS
$$
DECLARE max_unidades INTEGER;
BEGIN
    SELECT unidades INTO max_unidades
    FROM planograma
    WHERE planograma.ean = NEW.ean
      AND planograma.nro = NEW.nro
      AND planograma.num_serie = NEW.num_serie
      AND planograma.fabricante = NEW.num_serie;

    IF NEW.unidades > max_unidades THEN
        RAISE EXCEPTION 'Numero de unidades não deve excede as especificadas no
Planograma: %', max_unidades;
    END IF;

    RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER ri4
BEFORE UPDATE OR INSERT ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE nao_excede();


-----------------------------------------------------------------
-- (RI-5) Um Produto só pode ser reposto numa Prateleira que    --
--        apresente (pelo menos) uma das Categorias desse produto --
-----------------------------------------------------------------

CREATE OR REPLACE FUNCTION apresenta_categoria() RETURNS TRIGGER AS
$$
DECLARE
    categ_prateleira VARCHAR(50);
    categs_produto CURSOR FOR
        SELECT nome FROM tem_categoria WHERE ean = NEW.ean;
    categoria VARCHAR(50);
BEGIN
    SELECT nome
    INTO categ_prateleira
    FROM prateleira
    WHERE nro = NEW.nro
        AND num_serie = NEW.num_serie
        AND fabricante = NEW.fabricante;

    OPEN categs_produto;
    LOOP
        FETCH NEXT FROM categs_produto INTO categoria;
        EXIT WHEN NOT FOUND;

        IF categoria = categoria_prat THEN
            RETURN NEW;
        END IF;

    END LOOP;
    CLOSE categs_produto;

    RAISE EXCEPTION 'Produto não pode ser reposto nesta Prateleira.';
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER ri5
BEFORE UPDATE OR INSERT ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE apresenta_categoria();
```

SQL

(ficheiro queries.sql)

```
-----------------------------------------------------------------
-- Qual o nome do retalhista (ou retalhistas) responsáveis pela --
--          reposição do maior número de categorias?            --
-----------------------------------------------------------------
SELECT nome_ret
FROM retalhista NATURAL JOIN responsavel_por
GROUP BY nome_ret
HAVING COUNT(DISTINCT nome_cat) >= ALL (
        SELECT COUNT(DISTINCT nome_cat)
        FROM responsavel_por
        GROUP BY tin );
-----------------------------------------------------------------
-- Qual o nome do ou dos retalhistas que são responsáveis por --
--               todas as categorias simples?                  --
-----------------------------------------------------------------
SELECT nome_ret
FROM retalhista
        NATURAL JOIN responsavel_por
        NATURAL JOIN categoria_simples
GROUP BY nome_ret
HAVING COUNT(DISTINCT nome_cat) = (
        SELECT COUNT(*)
        FROM categoria_simples );

-------------------------------------------------------
-- Quais os produtos (ean) que nunca foram repostos? --
-------------------------------------------------------
SELECT ean
FROM produto
WHERE ean NOT IN (SELECT ean FROM evento_reposicao);
-------------------------------------------------------------
-- Quais os produtos (ean) que foram repostos sempre pelo --
--                  mesmo retalhista?                      --
-------------------------------------------------------------
SELECT ean
FROM evento_reposicao
GROUP BY ean
HAVING COUNT(DISTINCT tin) = 1;
```

Vistas

(ficheiro view.sql)

```
DROP VIEW IF EXISTS vendas;

-----------------------------------------------------------------
```

```
--------------|            VIEW vendas            |---------------
-------------------------------------------------------------------

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

Desenvolvimento da Aplicação

(pasta web/)

A aplicação é constituída por uma pasta de templates (templates/). Existe o template base que que é responsável pela criação do menu no topo da página. Todos os outros templates são responsáveis por uma view diferente. O index.html é a view da HomePage apresentada inicialmente.

Para as views que tem como função listar, é apenas apresentada a tabela pretendida na página. Para as views que são responsáveis por inserir ou remover, é apresentado um forms a preencher pelo usuário de modo para a execução do comando.

Link: https://web2.tecnico.ulisboa.pt/~ist196915/app.cgi/

1. Inserir/Remover Categorias e Sub-categorias
   - Através da Homepage, é possível clicar em “Criar Categoria” para criar uma nova categoria. Aquando o botão é pressionado, o utilizador é redirecionado para uma página onde lhe é pedido o Nome da Categoria e o seu Tipo (“super” ou “simples”).
   - Para remover categorias, é necessário inserir o nome da categoria a remover. Não é possível remover uma Categoria que esteja associada a outra para evitar quebrar hierarquias de categorias.
2. Inserir/Remover Retalhista com todos os seus produtos, garantindo que esta operação seja atómica
   - Do mesmo modo que com a Categoria, existe um “Criar Retalhista”. Ao clicar no mesmo, surge um menu que requer preenchimento com TIN, Nome do Retalhista e Nome da Categoria pela qual ele é responsável. A criação é atómica.
   - Para remover o retalhista, clica-se na HomePage em “Remover Retalhista”. Este evento realiza a remoção do retalhista por completo da DataBase, i.e apaga todos os elementos ao qual ele está associado de modo a ele deixar de existir por completo. Esta operação é atómica.

3. Listar todos os eventos de reposição de uma IVM, apresentando o número de unidades repostas por categoria de produto
   . Ao pressionar “Listar Eventos de Reposição” são apresentados todos os eventos de reposição por IVM. É apresentada também uma outra tabela que agrupa o número de unidades de reposição por categoria. Esta abordagem permite manter toda a informação de cada evento tal como apresentar o número somado de unidades repostas por categoria de produto.
4. Listar todas as sub-categorias de uma super-categoria, a todos os níveis de profundidade
   . Ao pressionar “Listar SubCategorias”, é apresentada uma tabela com todas as sub-categorias de Super Categoria inserida.

Índices

7.1 -

```
SELECT DISTINCT R.nome
FROM retalhista R, responsavel_por P
WHERE R.tin = P.tin AND P. nome_cat = 'Frutos'
```

Esta query pode ser otimizada criando índices hash sobre a coluna dos nomes das categorias (nome_cat) na relação responsável_por. Através de índices hash será possível encontrar rápidas correspondências entre uma cadeia de caracteres (nomes de uma categorias). É possível ainda assim otimizar ainda mais a query criando um segundo índice para a tabela tin.

```
CREATE INDEX cat_index ON responsavel por USING hash(nome_cat);
CREATE INDEX tin_index ON responsavel_por(tin);
```

7.2 -

```
SELECT T.nome, COUNT(T.ean)
FROM produto P, tem_categoria T
WHERE p.cat = T.nome AND P. desc like ‘A%’
GROUP BY T.nome
```

Executando o commando “explain analyze”, chegamos à conclusão que um índice sobre a descrição do produto optimiza a query. Como queremos encontrar as entradas com nome da categoria igual nas tabelas tem_categoria, podemos também otimizar através da criação de índices sobre as colunas cat na tabela produto e nome na tabela tem_categoria.

```
CREATE INDEX prod_index ON produto(nome_cat, descr);
CREATE INDEX tem_cat_index ON tem_categoria(nome);
```