

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 )

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 --

);

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.

1. 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.

1. 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

**WHER**E 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

**WHER**E 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);