Projeto DB - Parte 3

99228 - Gonçalo Carmo - 12h00 - 33.33%
99252 - João Penedo - 12h00 - 33.33%
95655 - Pedro Beato - 12h00 - 33.33%

Turno: BD2L17 - Daniela Falcão Machado

Grupo: 143

Restrições de Integridade

RI-1:

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

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER cat_cant_contain_self_trigger
BEFORE UPDATE OR INSERT ON tem_outra
FOR EACH ROW EXECUTE PROCEDURE cat_cant_contain_self_proc();
```

RI-4:

```
CREATE OR REPLACE FUNCTION replaced_cant_exceed_plan_proc()
RETURNS TRIGGER AS
$$
BEGIN
   IF NEW.unidades > (SELECT unidades FROM planograma WHERE NEW.ean =
planograma.ean
                                    AND NEW.nro = planograma.nro
                                    AND NEW.num_serie = planograma.num_serie
                                    AND NEW.fabricante = planograma.fabricante)
      THEN
      RAISE EXCEPTION 'O numero de unidades repostas num Evento de Reposicao nao
pode exceder o numero de unidades especificadas no Planograma';
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

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

RI-5:

```
CREATE OR REPLACE FUNCTION cat_checker(cat1 VARCHAR(80), cat2
VARCHAR(80)) /*cat 1 - inicialmente categoria repoosta, cat2 - categoria da
prateleira*/
RETURNS BOOLEAN
AS
$$
DECLARE
   cat_temp VARCHAR(80);
BEGIN
   IF (cat1 = cat2) THEN
      RETURN FALSE;
   ELSE
      SELECT super_categoria INTO cat_temp FROM tem_outra WHERE categoria
= cat1;
      IF (NULL = cat_temp) THEN
         RETURN TRUE;
      ELSE
         RETURN cat_checker(cat_temp, cat2);
      END IF;
   END IF;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION cant_replace_where_no_cat_proc()
RETURNS TRIGGER AS
$$
DECLARE
cat1 VARCHAR(80) = nome FROM tem_categoria WHERE tem_categoria.ean =
NEW.ean;
cat2 VARCHAR(80) = nome FROM prateleira WHERE NEW.nro = prateleira.nro
                        AND NEW.num_serie = prateleira.num_serie
                        AND NEW.fabricante = prateleira.fabricante;

BEGIN
   IF cat_checker(cat1, cat2)
      THEN
      RAISE EXCEPTION 'Um produto so pode ser reposto numa Prateleira que
apresente, pelo menos uma das categorias desse Produto';
      /*Assumindo que um produto pode estar inserido em varias categorias mas
uma prateleira so pode ter uma ????*/
   END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

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

SQL

1.
```
SELECT nome
FROM retalhista
INNER JOIN responsavel_por ON retalhista.tin = responsavel_por.tin
GROUP BY nome
HAVING COUNT(*) >= ALL (
    SELECT COUNT(*)
    FROM retalhista
    INNER JOIN responsavel_por ON retalhista.tin = responsavel_por.tin
    GROUP BY nome);
```

2.
```
SELECT DISTINCT nome
FROM retalhista
INNER JOIN responsavel_por ON retalhista.tin = responsavel_por.tin
WHERE nome_cat IN (SELECT nome FROM categoria_simples);
```

3.
```
SELECT ean
FROM produto
WHERE ean NOT IN (SELECT ean FROM evento_reposicao);
```

4.
```
SELECT ean
FROM evento_reposicao
GROUP BY ean
HAVING COUNT(ean) = 1;
```

Vistas

```
CREATE VIEW Vendas(ean, cat, ano, trimestre, mes, dia_mes, dia_semana, distrito,
concelho, unidades)
AS
SELECT produto.ean, tem_categoria.nome,
EXTRACT(YEAR FROM evento_reposicao.instante),
EXTRACT(QUARTER FROM evento_reposicao.instante),
EXTRACT(MONTH FROM evento_reposicao.instante),
EXTRACT(DAY FROM evento_reposicao.instante),
EXTRACT(DOW FROM evento_reposicao.instante),
ponto_de_retalho.distrito,
ponto_de_retalho.concelho,
evento_reposicao.unidades
FROM produto
INNER JOIN tem_categoria ON produto.ean = tem_categoria.ean
INNER JOIN evento_reposicao ON produto.ean = evento_reposicao.ean
INNER JOIN instalada_em ON evento_reposicao.num_serie =
instalada_em.num_serie
INNER JOIN ponto_de_retalho ON instalada_em.local_ponto_de_retalho =
ponto_de_retalho.nome;
```

Aplicação

Funcionalidade da Aplicação:

Existe um menu principal com as opções: “Ver Categorias”, Ver “Retalhistas”, “Ver IVMs”. Ao selecionar “Ver Categorias” passa para uma página onde estão expostas todas as categorias na base de dados onde se pode inserir uma nova categoria sem categoria “pai”, remover uma categoria, ou expandir uma categoria. Ao selecionar “expandir” encontra-se uma página com todas as categorias que são “filhas” da categoria selecionada e as mesmas opções encontradas no menu anterior, com uma exceção, se selecionar “inserir nova”, insere uma categoria que é “filha” da categoria expandida.

Voltando ao menu principal, se selecionar “Ver Retalhistas” consegue-se ver uma lista de todos os retalhistas presentes na base de dados e as opções para remover ou adicionar um novo retalhista.

Novamente no menu principal, ao selecionar “Ver IVMs” é redirecionado para uma página que apresenta todas as IVMs no sistema e uma opção para ver todas as reposições feitas nessa IVM identificadas pela categoria e quantidade reposta.

Relação entre ficheiros:

A aplicação inicia no ficheiro “index.html”, carregando em “Ver Categorias”, passa-se para o ficheiro “categorias.html”, se uma categoria for expandida entra-se no ficheiro “expand_cat.html” e se a categoria expandida for simples entra-se em “cat_simples.html”, se for optado adicionar uma nova categoria sem pai, passa-se para o ficheiro “new_cat_root.html”, se uma categoria for adicionada como filha de outra, vai-se para o ficheiro “new_cat_in.html”.

No Menu principal, selecionando “Ver Retalhistas” entra-se no ficheiro “retalhistas.html”, se for escolhido adicionar um novo retalhista, passa-se para o ficheiro “new_ret.html”.

Novamente no Menu principal, escolhendo a opção “Ver IVMs”, passa-se para o ficheiro “ivm.html”, optando ver as reposições de uma IVM entra-se no ficheiro “reposicoes.html”

Link:
http://web2.ist.utl.pt/~ist199228/app.cgi/

Nota: Quando se remove um retalhista ou uma categoria, são também removidos todos os eventos de reposição que lhes estavam associados, para poder respeitar as restrições foreign keys.

Índices

7.1 -

CREATE INDEX t_2 ON responsavel_por(tin)

Índice ordenado, porque indexa a coluna pela qual os dados estão fisicamente ordenados no ficheiro e faz sentido criar índices sobre o atributo “tin”, pois é este sobre o qual recai a conexão entre as duas tabelas e tabela “responsavel_por”, pois é nela que “tin” é uma foreign key, sendo assim a melhor maneira de reduzir acessos ao disco na realização da procura em questão.

7.2 -

CREATE INDEX t_3 ON tem_categoria(ean, nome);

Índice não ordenado, porque indexa uma coluna que não aquela pela qual os dados estão ordenados e faz sentido criar índices sobre o atributo “ean”, “nome” e tabela “tem_categoria”, porque a tabela não tem primary key, logo é o melhor local para criar um índice, já que não tem um por defeito.