Grupo: 139

Turno: BD2L16

Docente: Daniel Mateus Gonçalves

|  |  |  |  |
| --- | --- | --- | --- |
| 95606 | João Pedro Antunes Aragonez | Esforço Total Estimado:  40h | Contribuição:  O projeto foi feito em conjunto, com a presença constante de todos os membros do grupo.  Existiu debate constante de ideias. |
| 95578 | Francisco Manuel Leal Mithá Ribeiro |
| 96732 | Diogo Artur Rainha Lopes |

1. IC’s

/\*1\*/

CREATE OR REPLACE FUNCTION verifica\_cat\_recursao\_trigger\_proc() RETURNS TRIGGER AS $$

BEGIN

    IF new.categoria = new.super\_categoria THEN

        RAISE EXCEPTION 'Uma Categoria não pode estar contida em si própria'

    END IF;

    RETURN new;

END

$$ LANGUAGE plpgsql;

CREATE CONSTRAINT TRIGGER verifica\_cat\_recursao BEFORE INSERT OR UPDATE ON tem\_outra

FOR EACH ROW EXECUTE FUNCTION verifica\_cat\_recursao\_trigger\_proc();

/\*2\*/

CREATE OR REPLACE FUNCTION verifica\_nr\_unid\_repostas\_trigger\_proc() RETURNS TRIGGER AS $$

DECLARE new\_units INTEGER := 0;

DECLARE prev\_units INTEGER := 0;

BEGIN

    SELECT new.unidades INTO new\_units, pl.unidades INTO prev\_units

    FROM planograma as pl

    WHERE pl.ean = new.ean AND pl.nro = new.nro AND pl.num\_serie = new.num\_serie

    AND pl.fabricante = new.fabricante

    IF new\_units > prev\_units OR new\_units < 0  THEN

        RAISE EXCEPTION 'O nr de unidades repostas num Evento de Reposicao nao pode exceder o numero de unidades especificado no Planograma ou ser inferior a 0';

    END IF;

    RETURN new;

END

$$ LANGUAGE plpgsql;

CREATE CONSTRAINT TRIGGER verifica\_nr\_unid\_repostas BEFORE INSERT OR UPDATE ON evento\_reposicao

FOR EACH ROW EXECUTE FUNCTION verifica\_nr\_unid\_repostas\_trigger\_proc();

/\*3\*/

CREATE OR REPLACE FUNCTION verifica\_cat\_produto\_trigger\_proc() RETURNS TRIGGER AS $$

BEGIN

    IF new.ean NOT IN (

        SELECT ean

        FROM tem\_categoria NATURAL JOIN prateleira

        WHERE nro = new.nro AND num\_serie = new.num\_serie AND fabricante = new.fabricante

    ) THEN

        RAISE EXCEPTION 'Um  Produto  so  pode  ser  reposto  numa  Prateleira  que  apresente  (pelo  menos)  uma  das Categorias desse produto.'

    END IF;

    RETURN new;

END

$$ LANGUAGE plpgsql;

CREATE CONSTRAINT TRIGGER verifica\_cat\_produto BEFORE INSERT OR UPDATE ON evento\_reposicao

FOR EACH ROW EXECUTE FUNCTION verifica\_cat\_produto\_trigger\_proc();

Nota: não foram incluídos os cascades.

1. SQL

/\*1 Qual o nome do retalhista (ou retalhistas)  responsáveis pela reposição do maior número de

categorias?  \*/

select nome

from (

    select nome, count(distinct nome\_cat) /\*distinct\*/

    from retalhista natural join responsavel\_por

    group by tin

    having count(distinct nome\_cat) >= all(

        select count(distinct nome\_cat)

        from retalhista natural join responsavel\_por /\*da para reutilizar esta parte em cima ?\*/

        group by tin

    )

) as lista;

/\*2 Qual o nome do ou dos retalhistas que são responsáveis por todas as categorias simples? \*/

select retalhista.nome

from categoria\_simples join responsavel\_por on categoria\_simples.nome = responsavel\_por.nome\_cat

join retalhista on retalhista.tin = responsavel\_por.tin

group by retalhista.nome

/\*3 Quais os produtos (ean) que nunca foram repostos? \*/

select ean

from produto

where ean not in (

    select ean from evento\_reposicao

);

/\*4 Quais os produtos (ean) que foram repostos sempre pelo mesmo retalhista? \*/

select ean

from (

    select ean, count(distinct tin) /\*necessário este count ?? \*/

    from evento\_reposicao

    group by ean

    having count(distinct tin) = 1

) as lista;

1. View

create view vendas(ean, cat, ano, trimestre, mes, dia\_mes, dia\_semana, distrito, concelho, unidades) as

    select ean, 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 (produto natural join evento\_reposicao natural join instalada\_em) j1 join ponto\_de\_retalho pr on j1.locale = pr.nome;

1. Explicação do funcionamento da aplicação:

* ***categorias.html***: Mostra todas as categorias e possui 3 botões que permitem voltar ao menu inicial(*index.html)*, mostrar todas as super categorias(*super.html*) e mostrar todas as categorias simples(*simples.html*).
* ***eventos\_reposicao.html***: Mostra todos eventos de reposição da IVM selecionada (em *ivms.html*) , disponibilizando a sua data, unidades repostas, descrição do produto e respetiva categoria. Possui dois botões, um que retorna ao menu(*index.html)*, e um que retorna as IVM’s(*ivms.html).*
* ***index.html:*** Permite navegar entre as páginas das categorias(*categorias.html*), das IVM’s(*ivms.html*) e retalhistas(*retalhistas.html*).
* ***simples.html:*** Mostra todas as categorias simples, permite inserir(*inserir\_cat\_simples.html*) novas categorias simples e remover categorias existentes.
* ***super.html:*** Mostra todas as super categorias, permite inserir(*inserir\_cat\_super.html*) novas super categorias, remover categorias existentes e listar as subcategorias(*sub\_cat.html*) de cada super categoria.
* ***sub\_cat.html:*** Mostra todas as subcategorias da super categoria selecionada (em *super.html)* e possui um botão que permite retornar a página das super categorias.
* ***retalhistas.html:*** Mostra todos os retalhistas, permite inserir novos retalhistas(*inserir\_retalhista.html) e remover retalhistas já existentes.*
* ***inserir\_cat\_simples.html:*** Insere uma nova categoria simples, especificando uma super categoria e o nome da nova categoria.
* ***inserir\_cat\_super.html:*** Insere uma nova super categoria, especificando o nome da nova categoria.
* ***inserir\_retalhista.html:*** Insere um novo retalhista, especificando o nome do novo retalhista e o seu TIN.
* ***erros.html:*** Caso algum dos inserir falhe esta página é apresentada.

Link para a aplicação: <http://web2.tecnico.ulisboa.pt/~ist196732/app.cgi/>

1. OLAP

--1

select dia\_semana, concelho, sum(unidades)

from vendas

where DATEFROMPARTS(ano, mes, dia\_mes) >= DATEFROMPARTS(2012, 3, 12) AND

      DATEFROMPARTS(ano, mes, dia\_mes) <= DATEFROMPARTS(2010, 7, 21)

group by cube (dia\_semana, concelho);

--cube ou grouping sets ((dia\_semana), (concelho), ())

--2

select concelho, cat, dia\_semana, sum(unidades)

from vendas

where distrito = 'Lisboa'

group by cube(concelho, cat, dia\_semana);

--cube ou grouping sets ((concelho), (cat), (dia\_semana), ())

1. Índices:

create index i1 on responsavel\_por using hash(nome\_cat);

Sendo que é feito uma comparação de igualdade com o nome da categoria do tabela *responsável\_por* optamos por fazer um índice de hash na foreign key “nome\_cat” do mesmo, já que é o mais eficiente para comparações.

create index i2 on tem\_categoria(nome);

create index i3 on produto using hash(cat);

Foram criados dois índices, um relativo ao “nome” da tabela tem\_categoria com hash já que é feito comparação de igualdade, e um segundo para “cat” do produto pela mesma razão.