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- Database creation

```
create table categoria (
    nome varchar(255) not null,
    primary key (nome)
);

create table categoria_simples (
  nome varchar(255) not null,
  primary key (nome),
  foreign key (nome) references categoria(nome)
);

create table super_categoria (
  nome varchar(255) not null,
  primary key (nome),
  foreign key (nome) references categoria(nome)
);

create table tem_outra (
    super_categoria varchar(255) not null,
    categoria varchar(255) not null,
    primary key (categoria),
    foreign key (categoria) references categoria(nome),
    foreign key (super_categoria) references super_categoria(nome),
    unique (super_categoria, categoria) --RI-RE4
);

create table produto (
    ean numeric(13, 0) not null,
    cat varchar(255) not null,
    descr varchar(255),
    primary key (ean),
    foreign key (cat) references categoria(nome)
);

create table tem_categoria (
    ean numeric(13, 0) not null,
    nome varchar(255) not null,
    foreign key (nome) references categoria(nome),
    foreign key (ean) references produto(ean)
);

create table IVM (
    num_serie integer not null,
    fabricante varchar(255) not null,
```

```
    primary key (num_serie, fabricante)
);

create table ponto_de_retalho (
    nome varchar(255) not null,
    distrito varchar(255),
    concelho varchar(255),
    primary key (nome)
);

create table instalada_em (
    num_serie integer not null,
    fabricante varchar(255) not null,
    locale varchar(255),
    primary key (num_serie, fabricante),
    foreign key (num_serie, fabricante) references IVM(num_serie, fabricante),
    foreign key (locale) references ponto_de_retalho(nome)
);

create table prateleira (
    nro integer not null,
    num_serie integer not null,
    fabricante varchar(255) not null,
    altura integer not null,
    nome varchar(255) not null,
    primary key (nro, num_serie, fabricante),
    foreign key (num_serie, fabricante) references IVM(num_serie, fabricante),
    foreign key (nome) references categoria(nome)
);

create table planograma (
    ean numeric(13, 0) not null,
    nro integer not null,
    num_serie integer not null,
    fabricante varchar(255) not null,
    faces integer not null,
    unidades integer not null,
    loc varchar(255) not null,
    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 integer not null,
    nome varchar(255) not null,
    primary key (tin),
```

```
    unique (nome) --RI-RE7
);

create table responsavel_por (
    nome_cat varchar(255) not null,
    tin integer not null,
    num_serie integer not null,
    fabricante varchar(255) not null,
    primary key (nome_cat, num_serie, fabricante),
    foreign key (nome_cat) references categoria(nome),
    foreign key (tin) references retalhista(tin),
    foreign key (num_serie, fabricante) references IVM(num_serie, fabricante)

);

create table evento_reposicao (
    ean numeric(13, 0) not null,
    nro integer not null,
    num_serie integer not null,
    fabricante varchar(255) not null,
    instante timestamp not null,
    unidades integer not null,
    tin integer 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)
);
```

2- 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 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 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 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.

3- SQL

```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)
    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
        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;
```

4- 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;
```

5- 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/

6- OLAP

```sql
--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), ())
```

7- Índices:

```sql
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.

```sql
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.