Projeto Bases de Dados - Parte 3

Grupo 5:

- Diogo Fonseca nº 99065 - 33,3% - 11 horas
- João Ponces de Carvalho nº 99091 - 33,3% - 11 horas
- João Marques nº 99092 - 33,3% - 11 horas

Turno: BDL06 **Docente: Pedro Dias**

Esquema Base de Dados (parte 1):

```sql
create table categoria(
    nome varchar(32) not null unique,
    constraint pk_categoria primary key(nome)
);

create table categoria_simples(
    nome varchar(32) not null unique,
    constraint pk_categoria_simples primary key(nome),
    constraint fk_categoria_simples foreign key(nome)
        references categoria(nome)
);

create table super_categoria(
    nome varchar(32) not null unique,
    constraint pk_super_categoria primary key(nome),
    constraint fk_super_categoria foreign key(nome)
        references categoria(nome)
);

create table tem_outra(
    super_categoria varchar(32) not null,
    categoria varchar(32) not null unique,
    constraint pk_tem_outra primary key(categoria),
    constraint fk_tem_outra_super foreign key(super_categoria)
        references categoria(nome),
    constraint fk_tem_outra_categoria foreign key(categoria)
        references categoria(nome)
);
```

```sql
create table produto(
    ean char(13) not null unique,
    cat varchar(32) not null,
    descr varchar(80) not null,
    constraint pk_produto primary key(ean),
    constraint fk_produto foreign key(cat)
        references categoria(nome)
);

create table tem_categoria(
    ean char(13) not null unique,
    nome varchar(32) not null,
    constraint pk_tem_categoria PRIMARY KEY(ean, nome),
    constraint fk_tem_categoria_ean foreign key(ean)
        references produto(ean),
    constraint fk_tem_categoria_nome foreign key(nome)
        references categoria(nome)
);

create table IVM(
    num_serie int not null,
    fabricante varchar(32) not null,
    constraint pk_ivm primary key(num_serie, fabricante)
);
```

```sql
create TABLE responsavel_por(
    nome_cat varchar(32) not null,
    tin varchar(32) not null,
    num_serie int not null,
    fabricante varchar(32) not null,
    CONSTRAINT pk_responsavel_por PRIMARY KEY(num_serie, fabricante),
    CONSTRAINT fk_responsavel_por_categoria FOREIGN KEY(nome_cat)
        references categoria(nome),
    CONSTRAINT fk_responsavel_por_tin FOREIGN KEY(tin)
        references retalhista(tin),
    CONSTRAINT fk_responsavel_por_ivm FOREIGN KEY(num_serie, fabricante)
        references IVM(num_serie, fabricante)
);

CREATE TABLE evento_reposicao(
    ean char(13) not null,
    nro int not null,
    num_serie int not null,
    fabricante varchar(32) not null,
    instante TIMESTAMP not null,
    unidades int not null,
    tin varchar(32) not null,
    constraint pk_evento_reposicao PRIMARY KEY(ean, nro, num_serie, fabricante, instante),
    CONSTRAINT fk_evento_reposicao_planograma FOREIGN KEY(ean, nro, num_serie, fabricante)
        references planograma(ean, nro, num_serie, fabricante),
    CONSTRAINT fk_evento_reposicao_tin FOREIGN KEY(tin)
        references retalhista(tin)
);
```

Esquema Base de Dados (parte 2):

```
create table ponto_de_retalho(
    nome varchar(32) not null unique,
    distrito varchar(17) not null,
    concelho varchar(32) not null,
    constraint pk_ponto_de_retalho primary key(nome)
);

create table instalada_em(
    num_serie int not null,
    fabricante varchar(32) not null,
    local_de_instalacao varchar(32) not null,
    constraint pk_instalada_em primary key(num_serie, fabricante),
    constraint fk_instalada_em_ivm foreign key(num_serie, fabricante)
        references IVM(num_serie, fabricante),
    constraint fk_instalada_em_local foreign key(local_de_instalacao)
        references ponto_de_retalho(nome)
);

create table prateleira(
    nro int not null,
    num_serie int not null,
    fabricante varchar(32) not null,
    altura decimal(5,2) not null,
    nome varchar(32) not null,
    constraint pk_prateleira primary key(nro, num_serie, fabricante),
    constraint fk_prateleira_ivm foreign key(num_serie, fabricante)
        references ivm(num_serie, fabricante),
    constraint fk_prateleira_nome foreign key(nome)
        references categoria(nome)
);

create table planograma(
    ean char(13) not null,
    nro int not null,
    num_serie int not null,
    fabricante varchar(32) not null,
    faces int not null,
    unidades int not null,
    loc varchar(32) not null,
    constraint pk_planograma PRIMARY KEY(ean, nro, num_serie, fabricante),
    constraint fk_planograma_ean FOREIGN KEY(ean)
        references produto(ean),
    constraint fk_planograma_prateleira FOREIGN KEY(nro, num_serie, fabricante)
        REFERENCES prateleira(nro, num_serie, fabricante)

CREATE TABLE retalhista(
    tin varchar(32) not null unique,
    nome varchar(32) not null unique,
    CONSTRAINT pk_retalhista PRIMARY KEY(tin)
);
```

Restrições de Integridade:

(RI-1)

```
create or replace function chk_re_1_proc()
returns trigger as
$$
begin
   if new.super_categoria = new.categoria then
      raise exception 'Uma categoria não pode estar contida em si propria';
   end if;

   return new;
end;
$$ language plpgsql;

drop trigger if exists chk_re_1 on tem_outra;
create trigger chk_re_1
before insert or update on tem_outra
for each row execute procedure chk_re_1_proc();
```

(RI-4)

```
create or replace function chk_re_4_proc()
returns trigger as
$$
declare
   x int := (select unidades
      from planograma
      where (
         ean = new.ean and
         nro = new.nro and
         num_serie = new.num_serie and
         fabricante = new.fabricante
         )
      );
begin
   if new.unidades > x then
      raise exception 'Numero de unidades repostas excede o planograma';
   end if;
   return new;
end;
$$ language plpgsql;

drop trigger if exists chk_re_4 on evento_reposicao;
create trigger chk_re_4
before insert or update on evento_reposicao
for each row execute procedure chk_re_4_proc();
```

(RI-5)

```
create or replace function chk_re_5_proc()
returns trigger as
$$
begin
   if new.ean not in (
      select ean
      from tem_categoria
      where nome = (
         select nome
         from prateleira
         where new.nro = nro and new.num_serie = num_serie and new.fabricante =
               fabricante
      )
   ) then
      raise exception 'Produto nao pode ser colocado na prateleira';
   end if;
   return new;
end;
$$ language plpgsql;

drop trigger if exists chk_re_5 on evento_reposicao;

create trigger chk_re_5
before insert or update on evento_reposicao
for each row execute procedure chk_re_5_proc();
```

SQL:

(1)

```
select nome
from(
   select tin, count(*)
   from (
      select distinct nome_cat, tin from responsavel_por
   ) c
   group by (tin)
   having count(*) >= all(
      select count(*)
      from (
         select distinct nome_cat, tin from responsavel_por
      ) b
      group by tin)
) a
natural join retalhista;
```

(2)

```
select nome
from (
   select count(distinct nome_cat), tin
   from responsavel_por
   group by (tin)
   having count(distinct nome_cat) = any(
      select count(*)
      from categoria_simples)
) a
natural join retalhista;
```

(3)

```
select ean
   from produto
   where ean not in
   (select ean
      from evento_reposicao);
```

(4)

```
select ean
from (select distinct ean, tin from evento_reposicao) a
group by (ean)
having count(*) = 1;
```

Vistas:

```
drop view if exists Vendas;
create view Vendas(
        ean, cat, ano, trimestre, mes, dia_mes,
        dia_semana, distrito, concelho, unidades)
as
select e.ean, t.nome,
        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 e join tem_categoria t
        on e.ean = t.ean
natural join
instalada_em i join ponto_de_retalho p
        on i.local_de_instalacao = p.nome
```

Índices:

(1)

create index tin_index on responsavel_por using using hash(tin);

Para esta query criamos um índice do tipo hash para o atributo "tin" da tabela "responsavel_por". Isto porque, por ser uma foreign key, não possui nenhum tipo de índice associado. Outro motivo relevante é a comparação realizada no fim da query sobre os dois atributos referidos, onde um índice do tipo hash será mais eficiente. Não há necessidade de criar um índice para o atributo "nome_cat", pois o mesmo está presente na mesma tabela do atributo referido anteriormente, já indexado.

(2)

create index nome_index on tem_categoria using hash(nome);
create index descr_cat_index on produto(descr, cat);

Para esta query sugerimos criar dois índices . O primeiro, de tipo hash, é sobre o atributo "nome" da tabela "tem_categoria" e terá como principal consequência uma maior rapidez na realização da comparação. O mesmo, por não ser uma primary key, não possui nenhum índice associado e levaria a uma menor eficiência. O segundo índice é sobre dois atributos, "descr" e "cat", ambos da tabela "produto". Apesar de já existir um índice criado para o atributo "nome" na tabela "tem_categoria", como este e o atributo "cat" não se encontram na mesma tabela, não iríamos tirar proveito do índice criado. Assim, crê-se que um índice para ambos os atributos, que são foreign keys, irá beneficiar bastante a query desejada em termos de complexidade temporal.

Aplicação Web:

Os ficheiros criados para o funcionamento da aplicação encontram-se na pasta "web" dentro do zip da entrega final. Os ficheiros html estão incluídos na subdiretoria "templates".

Quanto à estrutura da aplicação em si, salienta-se o menu inicial com quatro opções. Estas são **"Gestão de Categorias"**, **"Listagem de Sub-categorias"**, **"Gestão de Retalhistas"** e **"Listagem de Eventos de Reposição"**.

Escolhendo a opção **"Gestão de Categorias"**, podemos observar uma tabela com todas as categorias existentes na base de dados. A partir da mesma é possível executar duas ações distintas: remover a própria categoria e adicionar a esta uma subcategoria que já se encontre inserida na base de dados. A tarefa de adicionar uma nova categoria ao sistema pode ser feita na caixa de texto debaixo da tabela, onde se encontra também um botão com uma drop down list para se escolher o tipo de categoria a adicionar.

Se selecionada a opção **"Listagem de Sub-categorias"**, podemos visualizar uma tabela com todas as super categorias presentes na base de dados. Cada uma terá um botão do seu lado direito que permitirá ao utilizador ver uma tabela com todas as suas subcategorias.

Se se optar por **"Gestão de Retalhistas"**, será exibida uma tabela com todos os retalhistas já na base de dados. É permitido ao utilizador a remoção de qualquer retalhista à sua escolha. Também lhe será disponibilizada a opção de adicionar um novo trabalhador que deverá possuir um nome e um TIN, a preencher nas devidas caixas de texto.

Por fim, se for escolhido **"Listagem de Eventos de Reposição"**, serão apresentadas todas as IVMs que existem, tendo cada uma a opção de visualizar os respectivos eventos de reposição.

É de se referir que no fim da realização da cada operação existe uma confirmação visual em caso de sucesso e um botão que permite o retorno do utilizador ao menu inicial, exceto em caso de erro. Neste caso o utilizador terá de utilizar as setas do seu browser para voltar até onde pretende.

Link da aplicação web: https://web2.ist.utl.pt/ist199092/app.cgi/