# Projeto de Bases de Dados

# Parte 4

## Supermarket Management

Grupo: 33

Turno: BD81795L05, Quarta-feira às 11:30, sala 1-27

Docente de Laboratório: Carlos Mendes

| Número | Nome | Contribuição |
|---|---|---|
| 79062 | Heydi Marques | 33,3% (8h) |
| 80908 | André Batista | 33,3% (8h) |
| 82396 | Sandro Anjos | 33,3% (8h) |

TÉCNICO LISBOA

# Restrições de Integridade

### a) O fornecedor (primário) de um produto não pode existir na relação fornece_sec para o mesmo produto.

```
create or replace function soFornPrimario() returns trigger
as $$
    declare nifPrim numeric;
    begin
        SELECT forn_primario INTO nifPrim FROM produto P WHERE P.ean = new.ean;
        if new.nif = nifPrim then
          raise exception 'O fornecedor % já é o fornecedor primário de %.', new.nif, new.ean;
        end if;
        return new;
    end
$$ language plpgsql;

drop trigger if exists soFornPrimario_trigger on fornece_sec;
create trigger soFornPrimario_trigger before insert or update on fornece_sec
for each row execute procedure soFornPrimario();

create or replace function jaFornSecundario() returns trigger
as $$
    declare nifSec numeric;
    begin
        SELECT nif INTO nifSec FROM fornece_sec F WHERE F.ean = new.ean;
        if new.forn_primario = nifSec then
                raise exception 'O fornecedor % já é o fornecedor secundário de %.',
        new.forn_primario, new.ean;
        end if;
        return new;
    end
$$ language plpgsql;

drop trigger if exists jaFornSecundario_trigger on produto;
create trigger jaFornSecundario_trigger before update on produto
for each row execute procedure jaFornSecundario();
```

### b) O instante mais recente de reposição tem de ser sempre anterior ou igual à data atual.

```
create or replace function instanteEvento() returns trigger
as $$
    declare data date;
    begin
        SELECT CURRENT_DATE INTO data;
        if new.instante > data then
                raise exception 'Instante de reposição (%) superior à data atual.', new.instante;
        end if;
        return new;
    end
$$ language plpgsql;

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

# Índices

1. **Liste o nif e nome de todos os fornecedores primários da categoria "Frutos".**

```
drop index if exists fornecedor_index;
drop index if exists produto1_index;
drop index if exists produto2_index;

create index fornecedor_index on fornecedor(nif) using hash;
create index produto1_index on produto(forn_primario) using hash;
create index produto2_index on produto(categoria) using btree;
```

Nesta query, relacionam-se as colunas nif da tabela fornecedor, e forn_primario da tabela produto. Uma vez que são duas tabelas diferentes, a utilização de um índice primário do tipo hash para as respetivas colunas nif e forn_primario como search key seria o método mais eficiente.

Já para a categoria da tabela produto, o mais eficiente é utilizar um índice do tipo B+Tree. A coluna "categoria" será organizada de forma alfabética e saber-se-á que, na primeira vez em que for diferente de 'Frutos', poderá interromper-se a procura.

2. **Liste o número fornecedores secundários de cada produto com mais de 1 fornecedor secundário.**

```
drop index if exists fornece_sec_index;
drop index if exists fornece_sec2_index;
drop index if exists produto3_index;

create index fornece_sec_index on fornece_sec(nif) using btree;
create index fornece_sec2_index on fornece_sec(ean) using hash;
create index produto3_index on produto(ean) using hash;
```

Esta query começa por selecionar o ean do qual se deseja obter a informação. Este ean está presente em tabelas diferentes, logo criar-se um índice primário do tipo hash para cada uma das tabelas aumentará a sua eficiência para a operação que iguala estas duas colunas. Não justificando a necessidade de uma organização alfabética nem numérica (ean são únicos), um índice do tipo hash é a melhor opção.

Para contar o número de nifs da tabela fornece_sec recorre-se a um índice do tipo B+Tree, para que estes estejam ordenados e a operação seja acelarada.

# Modelo Multidimensional

**Tabela d_produto contém os detalhes de cada produto:**

```
drop table if exists d_produto cascade;
        create table d_produto
                (ean                        numeric(13,0)  not null,
                 categoria                  varchar(30)    not null,
                 nif_fornecedor_principal   numeric(9,0)   not null,
                 constraint pk_d_produto primary key(ean));
```

**Tabela d_tempo contém as datas:**

```
drop table if exists d_tempo cascade;
        create table d_tempo
               (data     date      not null,
                dia      integer   not null,
                mes      integer   not null,
                ano      integer   not null,
                constraint      pk_d_tempo     primary key(data));
```

**Tabela fact_reposicao é a tabela de factos:**

```
drop table if exists fact_reposicao cascade;
        create table fact_reposicao
                (ean              numeric(13,0)  not null,
                 data             date           not null,
                 unidades         numeric(2,0)   not null,
                 idEvento         serial         not null,
                 constraint  pk_fact_reposicao   primary key(idEvento),
                 constraint  fk_fact_reposicao_d_produto foreign key(ean) references
                 d_produto(ean),
                 constraint fk_fact_reposicao_d_tempo  foreign key(data) references
                 d_tempo(data));
```

**Populate d_produto:**

```
INSERT INTO d_produto
        SELECT ean, categoria, forn_primario
        FROM produto;
```

**Populate d_tempo:**

```
INSERT INTO d_tempo
        SELECT distinct instante, date_part('day',instante), date_part('month',instante),
        date_part('year',instante)
        FROM evento_reposicao;
```

**Populate fact_reposicao:**

```
INSERT INTO fact_reposicao
        SELECT P.ean, T.data, R.unidades
        FROM d_produto P, d_tempo T, reposicao R
        WHERE P.ean = R.ean and T.data = R.instante;
```

# Data Analytics

SQL OLAP para obter o número de reposições de produtos do fornecedor com NIF 123 455 678 para cada categoria, com rollup por ano e mês.

**Solução com ROLLUP:**

```
SELECT categoria, ano, mes, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
GROUP BY ROLLUP (categoria, ano, mes)
WHERE nif_fornecedor_principal = '123455678'
ORDER BY ano desc, mes desc, categoria, numero_reposicoes;
```

**Solução com GROUPING SETS:**

```
SELECT categoria, ano, mes, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
GROUP BY GROUPING SETS ( (categoria, ano, mes), (categoria, ano), (categoria), () )
WHERE nif_fornecedor_principal = '123455678'
ORDER BY ano desc, mes desc, categoria, numero_reposicoes;
```

**Solução com UNION (usada para testar no PostgreSQL):**

```
(SELECT categoria, ano, mes, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = '123455678'
GROUP BY categoria, ano, mes)
UNION
(SELECT categoria, ano, null, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = '123455678'
GROUP BY categoria, ano)
UNION
(SELECT categoria, null, null, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = '123455678'
GROUP BY categoria)
UNION
(SELECT null, null, null, count(ean) as numero_reposicoes
FROM fact_reposicao NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = '123455678')
ORDER BY ano desc, mes desc, categoria, numero_reposicoes;
```