PROJETO DE BASES DE DADOS
PARTE 4

4º Entrega:
Restrições de Integridade
Índices
Modelo Multidimensional
Data Analytics

BD81795L03 - Grupo 11
Docente: Tomás Alves

84698 - André Fonseca
84705 - Catarina Custódio
84736 - Leonor Loureiro

Aluno	Horas	Contribuição
84698	6	33,(3)%
84705	6	33,(3)%
84736	6	33,(3)%

Restrições de Integridade

a)

```
create trigger trigger_fornece_prim before update on Supermercado.produto
   for each row execute procedure Supermercado.fornecedor_disjoint2();

create or replace function Supermercado.fornecedor_disjoint2() returns trigger
as $$
begin
   if exists (select * from Supermercado.fornece_sec
                                 where nif = new.forn_primario and ean = new.ean)
   then
          raise exception 'O fornecedor primario de um produto nao pode ser fornecedor secundario do mesmo produto.';
   end if;
   return new;
end
$$ language plpgsql;

create trigger trigger_fornece_sec before insert or update on Supermercado.fornece_sec
          for each row execute procedure Supermercado.fornecedor_disjoint();

create or replace function Supermercado.fornecedor_disjoint() returns trigger
as $$
begin
   if exists (select * from Supermercado.produto
                                 where forn_primario = new.nif and ean = new.ean)
   then
          raise exception 'O fornecedor primario de um produto nao pode ser fornecedor secundario do mesmo produto.';
   end if;
   return new;
end
$$ language plpgsql;
```

b)

```
create trigger trigger_evento_reposicao
before insert or update on Supermercado.evento_reposicao
          for each row execute procedure Supermercado.data_anterior_atual();

create or replace function Supermercado.data_anterior_atual() returns trigger
as $$
begin
   if (new.instante > CURRENT_TIMESTAMP)
   then
          raise exception 'O instante de reposicao deve ser anterior ou igual ao atual.';
   end if;
   return new;
end
$$ language plpgsql;
```

Índices

1.

Devem ser criados índices do tipo hash sobre a tabela *fornecedor* para o atributo *nif*, e sobre a tabela *produto* para o atributo *categoria*, uma vez que para um teste de igualdade, a procura com um índice hash é O(1), enquanto que para um índice B+ Tree é O(log n) e com um scan sequencial é O(n).

Executámos a instrução *explain analyse* sobre a query, usando cada combinação possível, para confirmar que a query tem menor custo e é mais rápida usando estes índices.

Teoricamente, deve também ser criado um índice do tipo hash sobre a tabela *produto* e sobre o atributo *forn_primario*, já que estamos a fazer uma comparação de igualdade. No entanto, ao repetirmos os testes não verificámos qualquer alteração no output do *explain* nem nos tempos de execução.

***create index** nif_idx **on** fornecedor **using hash**(nif);*
***create index** categoria_idx **on** produto **using hash**(categoria);*

2.

Teoricamente, devem ser criados índices do tipo hash sobre a tabela *fornece_sec* para o atributo *ean* e sobre a tabela *produto* para o atributo *ean*, uma vez que estamos a fazer um teste de igualdade entre P.ean e F.ean. No entanto, ao executarmos a instrução *explain analyse* sobre a query usando todas combinações diferentes dos vários tipos de índices estudadas para cada uma das tabelas e atributos, e usando somente scan sequenciais, verificámos que a query tem menor custo e é mais rápida quando usa o índice do tipo hash sobre a tabela fornece_sec e o índice B+ Tree sobre a tabela produto.

***create index** pk_produto **on** produto(ean);*
***create index** ean_idx **on** fornece_sec **using hash**(ean);*

Modelo Multidimensional

a) Definição das tabelas:

```
drop table if exists Supermercado.d_produto, Supermercado.d_tempo, Supermercado.f_reposicao;
drop domain if exists Supermercado.CEAN, Supermercado.NIFP, Supermercado.DAY, Supermercado.MONTH,
Supermercado.YEAR, Supermercado.CATEGORY, Supermercado.UNITS;

create domain Supermercado.CEAN as numeric(13,0);
create domain Supermercado.NIFP as numeric(9,0);
create domain Supermercado.DAY as numeric(2,0);
create domain Supermercado.MONTH as numeric(2,0);
create domain Supermercado.YEAR as numeric(4,0);
create domain Supermercado.CATEGORY as varchar(100);
create domain Supermercado.UNITS as int;


CREATE TABLE Supermercado.d_produto(
    cean Supermercado.CEAN,
    categoria Supermercado.CATEGORY,
    nif_fornecedor_principal Supermercado.NIF  NOT NULL,
  CONSTRAINT pk_d_produto PRIMARY KEY (cean)
);

CREATE TABLE Supermercado.d_tempo(
  dia Supermercado.DAY,
  mes Supermercado.MONTH,
  ano Supermercado.YEAR,
  CONSTRAINT pk_d_tempo PRIMARY KEY (dia ,mes, ano)
);
```

```
CREATE TABLE Supermercado.f_reposicao(
  cean Supermercado.CEAN,
  ano Supermercado.YEAR,
  mes Supermercado.MONTH,
  dia Supermercado.DAY,
  unidades Supermercado.UNITS,

  CONSTRAINT pk_f_reposicao PRIMARY KEY (cean, ano, mes, dia),
  CONSTRAINT fk_produto FOREIGN KEY (cean) REFERENCES Supermercado.d_produto(cean) ON DELETE
CASCADE,
  CONSTRAINT fk_ano FOREIGN KEY (ano, mes, dia) REFERENCES Supermercado.d_tempo(ano, mes, dia) ON
DELETE CASCADE
);
```

b) Carregar as tabelas:

```
INSERT INTO Supermercado.d_produto (cean, categoria, nif_fornecedor_principal)
SELECT ean, categoria, forn_primario
FROM Supermercado.produto;

INSERT INTO Supermercado.d_tempo (dia, mes, ano)
SELECT DISTINCT EXTRACT(DAY FROM instante),
  EXTRACT(MONTH FROM instante),
  EXTRACT(YEAR FROM instante)
FROM Supermercado.evento_reposicao;

INSERT INTO Supermercado.f_reposicao( cean, ano, mes, dia, unidades)
SELECT ean,
  EXTRACT(YEAR FROM instante),
  EXTRACT(MONTH FROM instante),
  EXTRACT(DAY FROM instante),
  unidades
FROM Supermercado.reposicao;
```

Data Analytics

```
SELECT categoria, ano, mes, COUNT(cean)
FROM Supermercado.f_reposicao NATURAL JOIN Supermercado.d_produto
WHERE nif_fornecedor_principal = 123455678
GROUP BY ROLLUP(categoria, ano, mes);
```

(testado em PostgreSQL 10.1)

Anexos:

schema: cria as tabelas da entrega anterior necessárias para carregar as tabelas do modelo multidimensional.
populate: popula as tabelas da entrega anterior com informação que possibilita testar coisas simples.
createStar: cria as tabelas do modelo multidimensional.
insertStar: carrega/popula as tabelas do modelo multidimensional com base nas tabelas encontradas no schema.
olap: query da Data Analytics, uma versão que usa o ROLLUP (apenas funciona em PostgreSQL mais recentes) e uma versão que gera um resultado semelhantes sem usar o ROLLUP.