Projeto de Bases de Dados
Parte 4

Dezembro 2017

Número	Nome	Contribuição	Esforço
76221	Emanuel Pereira	10%	2 horas
80832	Margarida Ferreira	50%	6 horas
83532	Miguel Marques	40%	4 horas

Grupo 42,
Turno BD2251795L07 (4ª feira, 11h),
Professor Miguel Amaral

1 Restrições de Integridade

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

A restrição é garantida com recurso a dois *triggers*, um sobre a tabela `produto` e outro sobre a tabela `fornece_sec`.

O primeiro, sobre a tabela `produto` impede que seja adicionado a um produto um fornecedor primário que já esteja registado como secundário para o mesmo produto (ou seja, existe uma entrada na tabela `fornece_sec` para esse fornecedor e produto), quer na criação de uma nova entrada na tabela, quer na alteração de uma já existente.

O segundo *trigger*, sobre a tabela `fornece_sec` impede que seja adicionado como fornecedor secundário o fornecedor primário de um produto, seja através da adição de uma nova entrada na tabela ou da atualização de uma entrada já existente.

Estes *triggers* são adicionados à base de dados através do seguinte troço de código:

```
CREATE OR REPLACE FUNCTION check_forn_sec_proc()
RETURNS TRIGGER AS $BODY$
BEGIN
    IF EXISTS(SELECT nif
              FROM fornece_sec
              WHERE fornece_sec.nif = NEW.forn_primario
              AND NEW.ean = fornece_sec.ean)
    THEN
        RAISE EXCEPTION '% e fornecedor secundario de %',
            NEW.forn_primario, NEW.ean
        USING HINT = 'Remova como fornecedor secundario do produto';
    END IF;
    RETURN NEW;
END;
$BODY$  LANGUAGE    plpgsql;

CREATE OR REPLACE FUNCTION check_forn_prim_proc()
RETURNS TRIGGER AS $BODY$
BEGIN
    IF EXISTS(SELECT forn_primario
              FROM produto
              WHERE produto.forn_primario = NEW.nif
              AND NEW.ean = produto.ean)
    THEN
        RAISE EXCEPTION '% e fornecedor primario de %', NEW.nif, NEW.ean
        USING HINT = 'Altere o fornecedor primario do produto';
    END IF;
    RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER check_forn_sec
BEFORE INSERT OR UPDATE ON produto
FOR EACH ROW EXECUTE PROCEDURE check_forn_sec_proc();
```

```
CREATE TRIGGER check_forn_prim
BEFORE INSERT OR UPDATE ON fornece_sec
FOR EACH ROW EXECUTE PROCEDURE check_forn_prim_proc();
```

b) O instante mais recente de reposição tem de ser sempre anterior ou igual à data atual:
A restrição é assegurada por uma restrição do tipo *check*, conseguida através da inserção da linha:

```
CHECK (instante < CURRENT_TIMESTAMP)
```

No *statement* `CREATE TABLE` da tabela `fornece_sec`. Pode ser inserido no fim, junto das restrições de chave primária, como restrição de tabela, ou na linha de declaração do atributo textttinstante, depois do tipo de dados.[1]

Alternativamente, o *check* pode ser adicionado posteriormente à criação da tabela, através da linha:

```
ALTER TABLE evento_reposicao ADD CHECK (instante < CURRENT_TIMESTAMP);
```

2 Índices

2.1 Consulta 1

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

```
select distinct F.nif, F.nome
from fornecedor F, produto P
where F.nif = P.forn_primario and P.categoria = 'Frutos'
```

São criados dois índices (um deles automaticamente pelo Postgres), para otimizar cada uma das condições de igualdade da cláusula *where*.

Para executar `F.nif = P.forn_primario`, que neste caso funciona como a condição de junção de um *join*, uma das colunas, `fornecedor.nif` ou `produto.forn_primario`, vai ter de ser percorrida sequencialmente. Por cada um dos elementos dessa coluna, vai ser encontrado o elemento igual na outra. Assim, a execução é otimizada se houver um índice numa destas colunas.

É criado automaticamente pelo *Postgres*, um índice primário na coluna `nif` da tabela `fornecedor`, do tipo `b-tree` (`fornecedor_pkey`). O ideal seria um índice do tipo `hash`, mas a sua criação não se justifica, tento em conta a existência do índice `b-tree`.

Para otimizar a segunda condição de igualdade da cláusula *where*, `P.categoria = 'Frutos'`, é criado um índice na coluna `categoria` da tabela `produto` do tipo `hash` (`prod_categoria_idx`).

Código SQL:

```
CREATE INDEX prod_categoria_idx ON produto USING hash (categoria);
-- CREATE INDEX fornecedor_pkey ON fornecedor USING hash(nif);
--  nao e necessario; o postgres cria automaticamente um do tipo b-tree
```

[1] Esta restrição está definida no ficheiro `schema.sql`, entregue na parte 3 do projeto.

2.2 Consulta 2

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

```
select ean, count(nif)
from produto P, fornece_sec F
where P.ean = F.ean
group by P.ean
having count(nif) > 1;
```

Como na consulta 1, a condição de igualdade da cláusula *where*, `P.ean = F.ean`, funciona como a condição de junção de um *join*. Ao executar o *join*, uma das colunas `produto.ean` ou `fornecedor.ean` vai ter de ser percorrida sequencialmente, enquanto são encontrados os elementos correspondentes na outra. Esta procura pode ser otimizada utilizando um índice.

O *Postgres* cria automaticamente um índice primário na coluna `ean` da tabela `produto` do tipo `b-tree` (`produto_pkey`), que otimiza o *statement group by*. Para o *join*, o ideal seria um índice do tipo `hash`, mas a não se justifica a criação de um novo índice, dada a existência do índice `b-tree`.

Código SQL (só comentários):

```
-- CREATE INDEX produto_pkey ON produto USING b-tree (ean);
--  nao e necessario; o postgres cria automaticamente um do tipo b-tree
```

3 Modelo Multidimensional

```
-- Assegura que as tabelas nao existem antes de serem criadas
DROP TABLE IF EXISTS d_produto;
DROP TABLE IF EXISTS d_tempo;
DROP TABLE IF EXISTS reposicoes;

SELECT DISTINCT ean AS cean,
                categoria,
                forn_primario AS nif_fornecedor_principal
INTO d_produto
FROM produto NATURAL JOIN reposicao;
-- Usamos NATURAL JOIN para impedir que estejam presentes produtos
--  que nao foram repostos.

ALTER TABLE d_produto ADD PRIMARY KEY (cean);

SELECT DISTINCT to_number(to_char(instante, 'YYYYMMDD'), '99999999') AS data_id,
                CAST(date_part('day',   instante) AS NUMERIC(2)) AS dia,
                CAST(date_part('month', instante) AS NUMERIC(2)) AS mes,
                CAST(date_part('year',  instante) AS NUMERIC(4)) AS ano
                -- Os casts sao necessarios para que os dados nao sejam
                --  guardados em floating point.
INTO d_tempo
FROM reposicao;
-- Usamos reposicao e nao evento_reposicao para evitar que haja entradas,
--  para as quais nao ha uma entrada de d_produto correspondente.
```

```
ALTER TABLE d_tempo ADD PRIMARY KEY (data_id);

-- Tabela de factos:
SELECT DISTINCT ean AS cean,
                to_number(to_char(instante, 'YYYYMMDD'), '99999999') AS data_id
INTO reposicoes
FROM produto NATURAL JOIN reposicao;

ALTER TABLE reposicoes ADD PRIMARY KEY (cean, data_id);
```

4 Data Analytics

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

```
SELECT categoria, NULL as ano, NULL as mes, COUNT(cean)
FROM reposicoes NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = 123455678
GROUP BY categoria
UNION
SELECT categoria, ano, mes, COUNT(cean)
FROM reposicoes NATURAL JOIN d_produto NATURAL JOIN d_tempo
WHERE nif_fornecedor_principal = 123455678
GROUP BY categoria, mes, ano;
```