Projeto de Bases de Dados

LEIC-A 2017/2018

Grupo 30

Parte 4

Restrições de Integridade, Índices, Modelo Multidimensional & *Data Analytics*

Turno BD2251795L09 - Sexta-Feira 12:30
Professor Paulo Carreira

83467	Francisco Neves	12 horas
83468	Francisco Catarrinho	12 horas
83550	Pedro Santos	12 horas

1 Restrições de Integridade

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

```
CREATE TABLE fornece_sec (
    ...
    CONSTRAINT not_in_forn_prim CHECK (NOT forn_prim_sec(ean, nif)),
);

CREATE OR REPLACE FUNCTION forn_prim_sec(_ean char(13), _nif char(9))
RETURNS BOOLEAN
AS $$
DECLARE result BOOLEAN;
BEGIN
    SELECT EXISTS (
        SELECT 1 FROM produto
        WHERE produto.ean = _ean AND produto.forn_primario = _nif
    ) INTO result;
    RETURN result;
END;
$$ LANGUAGE plpgsql;
```

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

```
CREATE TABLE evento_reposicao (
    ...
    instante timestamp NOT NULL CHECK (instante <= localtimestamp),
);
```

2 Índices

Para testar a influência do uso dos índices populou-se a base de dados usando um *script* que gerou 600 000 produtos (1 000 dos quais da categoria 'Frutos'), 300 000 fornecedores, 100 000 fornecimentos secundários e 300 000 categorias.

Para a análise dos índices necessários, analisou-se o comportamento e desempenho das *queries* sem o uso de índices (excepto o índice por defeito da chave primária) através do uso do comando **EXPLAIN** do *PostgreSQL*.

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

a) Tipo de índices que faria sentido criar.

```
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
HashAggregate  (cost=97324.76..97335.44 rows=1068 width=81)
Group Key: f.nif, f.nome
->  Nested Loop  (cost=0.42..97319.42 rows=1068 width=81)
     ->  Seq Scan on produto p  (cost=0.00..89514.82 rows=1068 width=71)
           Filter: (categoria = 'Frutos'::bpchar)
     ->  Index Scan using fornecedor_pkey on fornecedor f  (cost=0.42..7.30 rows=1 width=81)
           Index Cond: (nif = p.forn_primario)

Average Execution time: 5030.019 ms
NEW TIME with indexes: 6.701 ms
```

Tendo em conta estes resultados, foram criados os índices seguintes:

1) Tabela: produto, **Atributo(s):** categoria - **HASH:** Desagrupado, Denso

Este índice tem como objetivo acelerar a execução do **P.categoria = "Frutos"** já que esta é uma *point query* envolvendo uma condição de igualdade. O uso de uma função de dispersão (hash) permite que a procura

seja realizada aproximadamente em $O(1)$.

2) Tabela: fornecedor, **Atributo(s):** nif - **HASH:** Desagrupado, Denso

Este índice tem como objetivo acelerar a execução do **F.nif = P.forn_primario** já que esta é uma condição de igualdade. O uso de uma função de dispersão (*hash*) permite que a procura seja realizada em $O(1)$. De notar que é usada uma BTREE implicitamente pelo *PostgreSQL* já que o atributo nif é chave primária da tabela Fornecedor.

b) Criação dos índices em SQL.

```
CREATE INDEX categoria_produto_index ON produto USING HASH(categoria);
CREATE INDEX nif_fornecedor_index ON fornecedor USING HASH(nif);
```

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

a) Tipo de índices que faria sentido criar.

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------
GroupAggregate  (cost=12124.80..81894.00 rows=101032 width=24)
Group Key: p.ean
Filter: (count(f.nif) > 1)
->  Merge Join  (cost=12124.80..79873.36 rows=101032 width=24)
     Merge Cond: (p.ean = f.ean)
     ->  Index Only Scan using produto_pkey on produto p  (cost=0.42..64584.08 rows=601024 width=14)
     ->  Materialize  (cost=12124.33..12629.49 rows=101032 width=24)
           ->  Sort  (cost=12124.33..12376.91 rows=101032 width=24)
                 Sort Key: f.ean
                 ->  Seq Scan on fornece_sec f  (cost=0.00..1654.32 rows=101032 width=24)

Average Execution Time: 1510.540 ms
NEW TIME with indexes: 94.641 ms
```

Tendo em conta estes resultados, foram criados os índices seguintes:

1) Tabela: fornece_sec, **Atributo(s):** ean - **BTREE:** Agrupado, Esparso

Este índice tem como objetivo acelerar a execução do **P.ean = F.ean** eliminando o *Sort* que era previsto pelo *query plan*.

2) Tabela: produto, **Atributo(s):** ean - **BTREE:** Agrupado, Esparso

Este índice tem como objetivo acelerar a execução do **GROUP BY P.ean**. De notar que este índice é usado implicitamente pelo PostgreSQL já que este atributo é chave primária da tabela.

b) Criação dos índices em SQL.

```
CREATE INDEX ean_forn_sec_index ON fornece_sec USING BTREE(ean);
CREATE INDEX ean_produto_index ON produto USING BTREE(ean);
```

3 Modelo Multidimensional

3.1 Esquema em Estrela

```
CREATE TABLE d_produto (
    ean                     char(13)   NOT NULL,
    categoria               char(70)   NOT NULL,
    nif_fornecedor_principal char(9)   NOT NULL,
    PRIMARY KEY (ean)
);
```

```sql
CREATE TABLE d_tempo (
    tempo_id   integer    NOT NULL,
    dia        integer    NOT NULL,
    mes        integer    NOT NULL,
    ano        integer    NOT NULL,
    PRIMARY KEY (tempo_id)
);

CREATE TABLE facts (
    ean       char(13)    NOT NULL,
    tempo_id  integer     NOT NULL,
    nro       integer     NOT NULL,
    lado      char(20)    NOT NULL,
    altura    char(20)    NOT NULL,
    operador  char(70)    NOT NULL,
    unidades  integer     NOT NULL,
    FOREIGN KEY (ean)         REFERENCES d_produto(ean),
    FOREIGN KEY (tempo_id)    REFERENCES d_tempo(tempo_id)
);
```

3.2 Intruções SQL para carregar o esquema

```sql
-- Funcao auxiliar que gera um id para a dimensao tempo a partir de um instante.
CREATE OR REPLACE FUNCTION date_to_int(instante timestamp)
RETURNS integer AS
$$
DECLARE result integer;
           dia integer;
           mes integer;
           ano integer;
BEGIN
    dia := date_part('day', instante)::int;
    mes := date_part('month', instante)::int;
    ano := date_part('year', instante)::int;
    result := ano * 10000 + mes * 100 + dia;
    RETURN result;
END;
$$ LANGUAGE plpgsql;

-- Funcao para carregar a tabela da dimensao produto.
CREATE OR REPLACE FUNCTION load_d_produto()
RETURNS void AS
$$
BEGIN
    INSERT INTO d_produto(ean, categoria, nif_fornecedor_principal)
        SELECT ean, categoria, forn_primario FROM produto;
END;
$$ LANGUAGE plpgsql;

-- Funcao para carregar a tabela da dimensao tempo.
CREATE OR REPLACE FUNCTION load_d_tempo()
RETURNS void AS
$$
BEGIN
    INSERT INTO d_tempo(tempo_id, dia, mes, ano)
        SELECT DISTINCT date_to_int(instante) AS tempo_id,
                        date_part('day', instante) AS dia,
                        date_part('month', instante) AS mes,
                        date_part('year', instante) AS ano
        FROM evento_reposicao;
END;
$$ LANGUAGE plpgsql;

-- Funcao para carregar a tabela de factos.
CREATE OR REPLACE FUNCTION load_facts()
```

```
RETURNS void AS
$$
DECLARE cur CURSOR FOR SELECT * FROM reposicao;
        c_ean      char(13);
        c_nro      integer;
        c_lado     char(20);
        c_altura   char(20);
        c_operador char(70);
        c_instante timestamp;
        c_unidades integer;
BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO c_ean, c_nro, c_lado, c_altura, c_operador, c_instante, c_unidades;
        EXIT WHEN NOT FOUND;

        INSERT INTO facts (ean, tempo_id, nro, lado, altura, operador, unidades)
            VALUES (
                c_ean,
                ( SELECT date_to_int(c_instante)),
                c_nro,
                c_lado,
                c_altura,
                c_operador,
                c_unidades
            );
    END LOOP;
    CLOSE cur;

END;
$$ LANGUAGE plpgsql;

-- Carregamento do esquema em estrela.
DO $$ BEGIN
    PERFORM load_d_produto();
    PERFORM load_d_tempo();
    PERFORM load_facts();
END $$;
```

4 Data Analytics

1) 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, ano, mes, COUNT(ean) AS nmr_reposicoes FROM facts AS f
JOIN (
    SELECT nif_fornecedor_principal AS nif, ean, categoria
    FROM d_produto
    UNION
    SELECT nif, ean, categoria
    FROM d_produto
    JOIN fornece_sec AS f USING (ean)
) AS p USING (ean)
JOIN d_tempo USING (tempo_id)
WHERE nif = '123455678'
GROUP BY categoria, ROLLUP(ano, mes);
```