Licenciatura Bolonha em Engenharia Informática e de Computadores
Disciplina: Base de Dados

Projeto de Bases de Dados - Parte 4

Grupo 7

Turno BD225179577L04 (Segunda-feira, 8:00-9:30, Lab 10)

Docente do Laboratório Tiago Oliveira

Catarina Carreiro, 92438, 33%, 17 horas
Cristiano Clemente, 92440, 33%, 17 horas
Mónica Jin, 92532, 34%, 17 horas

RI-100

```
CREATE OR REPLACE FUNCTION ri_100_proc ()
    RETURNS TRIGGER
    AS $$
DECLARE
    count_consultas integer;
BEGIN
    SELECT
        COUNT(num_cedula) INTO count_consultas
    FROM
        consulta
    WHERE
        date_part('week', data) = date_part('week', NEW.data)
        AND nome_instituicao = NEW.nome_instituicao;
    IF count_consultas < 100 THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'O medico % atingiu o seu limite semanal de
100 consultas na instituicao %.', NEW.num_cedula,
NEW.nome_instituicao;
    END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER ri_100
    BEFORE INSERT ON consulta
    FOR EACH ROW
    EXECUTE PROCEDURE ri_100_proc ();
```

RI-análise

```
CREATE OR REPLACE FUNCTION ri_analise_proc ()
    RETURNS TRIGGER
    AS $$
DECLARE
    medico_especialidade varchar(100);
BEGIN
    SELECT
        especialidade INTO medico_especialidade
    FROM
        medico
    WHERE
        num_cedula = NEW.num_cedula;
    IF NEW.num_cedula IS NULL AND NEW.num_doente IS NULL AND
NEW.data IS NULL THEN
        --consulta omissa RETURN NEW;
    ELSIF NEW.num_cedula IS NOT NULL
            AND NEW.num_doente IS NOT NULL
            AND NEW.data IS NOT NULL THEN
            --consulta nao omissa IF medico_especialidade =
NEW.especialidade THEN
                RETURN NEW;
            ELSE
                RAISE EXCEPTION 'A analise e de % mas o medico e
especialista em %.', NEW.especialidade, medico_especialidade;
            END IF;
    ELSE
        RAISE EXCEPTION 'A analise com o numero % nao tem os dados
corretos.', NEW.num_analise;
    END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER ri_analise
    BEFORE INSERT ON analise
    FOR EACH ROW
    EXECUTE PROCEDURE ri_analise_proc ();
```

Índices

1

De modo a tornar a comparação direta “num_doente = <um_valor>” bastaria alterar a declaração da tabela consulta para
`consulta(num_doente, num_cedula, data, nome_instituicao)`
de forma a aproveitar o índice implícito associado à chave primária.

2

Criando um novo índice
```
CREATE INDEX medico_especialidade ON medico USING
HASH(especialidade);
```
passamos a poder procurar de forma eficiente os médicos por especialidade.
A hashtable torna a comparação direta “ especialidade = “Ei” ” mais eficiente.
A chave primária num_cedula em nada é relevante para a query em causa.

3

Criando um novo índice
CREATE INDEX medico_especialidade ON medico(especialidade);
passamos a poder procurar de forma eficiente os médicos por especialidade.
A chave primária num_cedula em nada é relevante para a query em causa.
O método DEFAULT do create index é BTree e este método mantém os dados ordenados.
Uma vez que cada bloco tem 2 registos, então quando um índice mapeia para um bloco há 3 possibilidades:

1) Índice aponta para o registo 1 do bloco
2) Índice aponta para o registo 2 do bloco
3) Índice aponta para para ambos os registos (caso ideal)

Como todos os médicos estão uniformemente distribuídos pelas 6 especialidades, criar um índice para a especialidade pode não ser eficiente uma vez que a criação de índices tem custos.
Porém, ao adotarmos o método BTree, estamos a ordenar os dados e isto leva a que registos com a mesma especialidade fiquem adjacentes, portanto a probabilidade de obtermos a possibilidade 3 é mais elevada. Assim, escolheu-se criar um índice para a especialidade usando o método BTree.

4

A declaração da tabela consulta passaria a ser
`consulta(num_cedula, data, num_doente, nome_instituicao)`
de forma a aproveitar o índice implícito associado à chave primária.

star_schema

```
DROP TABLE IF EXISTS d_tempo CASCADE;
CREATE TABLE d_tempo(
    id_tempo SERIAL PRIMARY KEY,
    dia INTEGER NOT NULL,
    dia_da_semana INTEGER NOT NULL,
    semana INTEGER NOT NULL,
    mes INTEGER NOT NULL,
    trimestre INTEGER NOT NULL,
    ano INTEGER NOT NULL);

DROP TABLE IF EXISTS d_instituicao CASCADE;
CREATE TABLE d_instituicao(
    id_inst SERIAL PRIMARY KEY,
    nome CHAR(100) NOT NULL REFERENCES instituicao(nome) ON DELETE CASCADE ON
UPDATE CASCADE,
    tipo CHAR(100) NOT NULL,
    num_regiao INTEGER NOT NULL, --REFERENCES regiao(num_regiao) ON DELETE CASCADE
ON UPDATE CASCADE,
    num_concelho INTEGER NOT NULL --REFERENCES concelho(num_concelho) ON DELETE
CASCADE ON UPDATE CASCADE);

DROP TABLE IF EXISTS f_presc_venda CASCADE;
CREATE TABLE f_presc_venda(
    id_presc_venda INTEGER PRIMARY KEY REFERENCES prescricao_venda(num_venda) ON
DELETE CASCADE ON UPDATE CASCADE,
    id_medico INTEGER NOT NULL REFERENCES medico(num_cedula) ON DELETE CASCADE ON
UPDATE CASCADE,
    num_doente INTEGER NOT NULL,
    id_data_registo INTEGER NOT NULL REFERENCES d_tempo(id_tempo) ON DELETE CASCADE
ON UPDATE CASCADE,
    id_inst INTEGER NOT NULL REFERENCES d_instituicao(id_inst) ON DELETE CASCADE ON
UPDATE CASCADE,
    substancia CHAR(100) NOT NULL,
    quant INTEGER NOT NULL);

DROP TABLE IF EXISTS f_analise CASCADE;
CREATE TABLE f_analise(
    id_analise INTEGER PRIMARY KEY REFERENCES analise(num_analise) ON DELETE
CASCADE ON UPDATE CASCADE,
    id_medico INTEGER REFERENCES medico(num_cedula) ON DELETE SET NULL ON UPDATE
CASCADE,
    num_doente INTEGER,
    id_data_registo INTEGER NOT NULL REFERENCES d_tempo(id_tempo) ON DELETE CASCADE
ON UPDATE CASCADE,
    id_inst INTEGER NOT NULL REFERENCES d_instituicao(id_inst) ON DELETE CASCADE ON
UPDATE CASCADE,
    nome CHAR(100) NOT NULL,
    quant INTEGER NOT NULL);
```

Nota:Para implementar a chave substituta Id_presc_venda usamos o atributo num_venda da tabela presc_venda (já definida na E3) mas que neste caso passa a ser UNIQUE.

```
INSERT INTO d_instituicao
SELECT nextval('d_instituicao_id_inst_seq'), instituicao.* FROM
instituicao;

INSERT INTO d_tempo
SELECT nextval('d_tempo_id_tempo_seq'), date_part('day',
data_registo), date_part('dow', data_registo), date_part('week',
data_registo), date_part('month', data_registo), (date_part('month',
data_registo)-1)/3 + 1,date_part('year', data_registo)
FROM prescricao_venda NATURAL JOIN venda_farmacia;

INSERT INTO d_tempo
SELECT nextval('d_tempo_id_tempo_seq'), date_part('day',
data_registo), date_part('dow', data_registo), date_part('week',
data_registo), date_part('month', data_registo), (date_part('month',
data_registo)-1)/3 + 1,date_part('year', data_registo)
FROM analise;

INSERT INTO f_presc_venda
SELECT num_venda, num_cedula, num_doente, id_tempo, id_inst,
substancia, quant
FROM prescricao_venda NATURAL JOIN venda_farmacia JOIN d_instituicao
ON venda_farmacia.inst = d_instituicao.nome, d_tempo
WHERE date_part('day', data_registo) = d_tempo.dia AND
date_part('month', data_registo) = d_tempo.mes AND date_part('year',
data_registo)=d_tempo.ano;

INSERT INTO f_analise
SELECT num_analise, num_cedula, num_doente, id_tempo, Id_inst,
analise.nome, quant
FROM analise JOIN d_instituicao ON inst = d_instituicao.nome,
d_tempo
WHERE date_part('day', data_registo) = d_tempo.dia AND
date_part('month', data_registo) = d_tempo.mes AND date_part('year',
data_registo)=d_tempo.ano;
```

```
SELECT especialidade, ano, mes, COUNT(id_analise)
FROM f_analise JOIN medico ON id_medico=num_cedula JOIN
d_tempo ON id_data_registo=id_tempo
WHERE f_analise.nome = 'glicemia' AND ano BETWEEN 2017 AND
2020
GROUP BY GROUPING SETS(especialidade, ano, mes);


WITH total_presc_venda(substancia, num_concelho, mes,
dia_da_semana, quant_total, n_prescricoes)AS(
    SELECT substancia, num_concelho, mes,dia_da_semana,
SUM(quant), COUNT(id_presc_venda)
    FROM f_presc_venda JOIN d_tempo ON
id_data_registo=id_tempo NATURAL JOIN d_instituicao
    WHERE trimestre=1 AND ano=2020 AND num_regiao=0
--assumimos que Lisboa tem num_regiao=0
    GROUP BY substancia, num_concelho, mes, dia_da_semana
)
SELECT substancia, num_concelho, mes, dia_da_semana,
SUM(quant_total), SUM(n_prescricoes)/4 AS
n_medio_prescricoes_diario
FROM total_presc_venda
GROUP BY substancia, num_concelho, mes, dia_da_semana
UNION
SELECT substancia, num_concelho, mes, NULL, SUM(quant_total),
SUM(n_prescricoes)/30 AS n_medio_prescricoes_diario
FROM total_presc_venda
GROUP BY substancia, num_concelho, mes
UNION
SELECT substancia, num_concelho, NULL, NULL, SUM(quant_total),
SUM(n_prescricoes)/90 AS n_medio_prescricoes_diario
FROM total_presc_venda
GROUP BY substancia, num_concelho
UNION
SELECT substancia, NULL, NULL, NULL, SUM(quant_total),
SUM(n_prescricoes)/90 AS n_medio_prescricoes_diario
FROM total_presc_venda
GROUP BY substancia;
```