Instituto Superior Técnico

Projecto de Base de Dados, Parte 4

Professor: André Vasconcelos
Turno:BD22517957L08

Grupo:60

Nome	Número	Contribuição Trabalho [%]	Esforço [horas]
Madalena Pedreira	86466	0.33	5.5
Pedro Custódio	86496	0.33	5.5
Rita Fernandes	86508	0.33	5.5

Restrições de Integridade

1. a)

```
CREATE OR REPLACE FUNCTION verifica_coordenador()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.id_coordenador NOT IN (
     SELECT id_coordenador
     FROM audita
     WHERE data_auditoria <= NEW.data_hora_inicio
   )
   THEN RAISE EXCEPTION 'Nao e possivel o coordenador %  fazer essa solicitacao',
NEW.id_coordenador;
   END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER ch_coordenador BEFORE INSERT ON solicita
FOR EACH ROW EXECUTE PROCEDURE verifica_coordenador();
```

1. b)

```
CREATE OR REPLACE FUNCTION verifica_aloca()
RETURNS TRIGGER AS $$
BEGIN
   IF (SELECT num_meio
      FROM acciona
      WHERE num_meio = NEW.num_meio
      AND nome_entidade = NEW.nome_entidade
      AND  num_processo_socorro = NEW.num_processo_socorro) IS NULL THEN
      RAISE EXCEPTION 'O Meio %:% ainda nao foi accionado',NEW.nome_entidade,
NEW.num_meio;

      END IF;
   RETURN NEW;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER chk_aloca BEFORE INSERT ON alocado
FOR EACH ROW EXECUTE PROCEDURE verifica_aloca();
```

Índices

a)Consideramos adequado índices por hash para ambas as queries apresentadas. A necessidade torna-se evidente tendo em conta que ambas tratam de uma verificação de igualdade entre os dados.

Para a primeira query, ter-se-iam de aplicar índices sobre as tabelas Video e Vigia. Em relação à primeira tabela, como o ênfase é feito à questão da especificação do número da câmara, o

índice era constituído por num_Camara e agrupado por esse mesmo atributo. Já a tabela Vigia teria um índice composto por num_Camara e morada_Local, uma vez que a comparação para validar os dados que se procuram passam apenas por estes parâmetros, forçando um abandono das chaves primárias. Qualquer um dos índices sobre as tabelas seria agrupado por num_Camara, de maneira a tornar mais eficiente a procura, já que, por exemplo, para encontrar todas as câmaras número 10 num determinado local, uma vez encontrado o primeiro registo com esse número, a procura seria feita sequencialmente, pois todas as câmaras com esse número estariam seguidas, poupando inclusive gastos de I/O.

Para a segunda query, aplicar-se-iam índices sobre as tabelas Transporta e EventoEmergencia. Para a tabela Transporta seria necessário um índice simples, por num_Processo_Socorro, assim como para a tabela EventoEmergencia. Ambos os índices seriam agrupados por num_Processo_Socorro.

b) A avaliação teórica feita anteriormente não se vai reflectir como foi esperada uma vez procurada implementar no sistema, uma vez que o postgres não cobre a execução de índices hash compostos nem por agrupamentos em índices compostos como tinha sido preferido na secção anterior. Ora, em relação às tabelas Vigia teria de se recorrer a índices com btree com o mesmo conjunto de elementos compostos - sendo composto não pode ser agrupado. Já em relação às tabelas Video, Transporta e EventoEmergencia, ainda que tenham apenas um elemento a compor o índice (continuando portanto candidatas a uma indexação por hash no postgres), teve de se medir o quão vantajoso seria usar-se esse mecanismo em detrimento índices btree mas com agrupamentos. Preferiu-se a última opção, daí as alterações feitas às tabelas terem a forma que se segue:

```
ALTER TABLE vigia DROP CONSTRAINT pk_vigia;
ALTER TABLE video DROP CONSTRAINT pk_video CASCADE;
ALTER TABLE transporta DROP CONSTRAINT pk_transporta;
ALTER TABLE evento_emergencia DROP CONSTRAINT pk_evento_emergencia CASCADE;

CREATE INDEX idx_video ON video USING btree (num_Camara);
CLUSTER video USING idx_video;
CREATE INDEX idx_vigia ON vigia USING btree (num_Camara,morada_Local);

CREATE INDEX idx_transporta ON transporta USING btree (num_Processo_Socorro);
CLUSTER transporta USING idx_transporta;

CREATE INDEX idx_evento_emergencia ON evento_emergencia USING btree
(num_Processo_Socorro);
CLUSTER evento_emergencia USING idx_evento_emergencia;
```

Fez-se uma análise temporal em relação às várias fases de índices: antes de qualquer alteração (índices default), sem qualquer índice e, por fim, com a alteração anterior. A tabela que se segue condensa os resultados sobre os quais nos vamos basear para fazer a apreciação geral:

	Tempo antes de alteração de índices [ms]	Tempo sem índices [ms]	Tempo depois de alteração de índices [ms]
Query 1	0.946	1.437	0.554
Query 2	2.045	1.107	1.102

À partida conclui-se que houve um melhoramento em ambas as queries. No entanto, correndo o comando EXPLAIN antes de cada uma delas, verifica-se que para a segunda não foram usados os índices criados. Pode-se explicar este facto pelo número de dados ser pouco significativo tornando-se irrelevante recorrer a estruturas mais complexas.

Modelo Multidimensional

Para definir o esquema em estrela foram criadas as seguintes tabelas:

```
CREATE TABLE d_evento (
   id_evento serial,
   num_telefone varchar(15) not null,
   instante_chamada timestamp not null,
   constraint pk_d_evento primary key(id_evento),
   constraint fk_d_evento foreign key(num_telefone, instante_chamada)
         references evento_emergencia(num_telefone, instante_chamada),
   unique(num_telefone,instante_chamada));

CREATE TABLE d_meio (
   id_meio serial,
   num_meio integer not null ,
   nome_meio varchar(30) not null,
   nome_entidade varchar(30) not null,
   tipo varchar(30) not null,
   constraint pk_d_meio primary key(id_meio),
   constraint fk_d_meio foreign key(num_meio,nome_meio,nome_entidade)
         references meio(num_meio,nome_meio,nome_entidade),
   unique(num_meio,nome_meio,nome_entidade,tipo)
);

CREATE TABLE d_tempo (
   id_tempo serial,
   dia integer not null ,
   mes integer not null ,
   ano integer not null ,
```

```
    constraint pk_d_tempo primary key(id_tempo),
    unique(dia,mes,ano));

CREATE TABLE d_factos (
    id_evento integer not null,
    id_meio integer not null,
    id_tempo integer not null,
    constraint pk_d_factos primary key(id_evento,id_meio,id_tempo),
    constraint fk_d_factos_evento foreign key(id_evento)
        references d_evento(id_evento),
    constraint fk_d_factos_meio foreign key(id_meio)
        references d_meio(id_meio),
    constraint fk_d_factos_tempo foreign key(id_tempo)
        references d_tempo(id_tempo));
```

Cada registo da tabela d_factos corresponde, então, à data em que um meio foi usado num dado evento de emergência.

Após a criação das tabelas foi necessária a inserção dos dados no novo esquema. A inserção dos registos da tabela d_tempo foi feita no populate inicial, pois nesta tabela residem todas as datas dos anos 2018 e 2019, numeradas sequencialmente. Para as restantes tabelas do modelo multidimensional o carregamento a partir das tabelas iniciais foi o que se segue:

```
INSERT INTO d_meio(num_meio,nome_meio,nome_entidade,tipo)
        SELECT num_meio, nome_meio, nome_entidade, 'combate'
        FROM meio_combate NATURAL JOIN meio;

INSERT INTO d_meio(num_meio,nome_meio,nome_entidade,tipo)
        SELECTnum_meio, nome_meio, nome_entidade, 'apoio'
        FROM meio_apoio NATURAL JOIN meio;

INSERT INTO d_meio(num_meio,nome_meio,nome_entidade,tipo)
        SELECT num_meio, nome_meio, nome_entidade, 'socorro'
        FROM meio_socorro NATURAL JOIN meio;

INSERT INTO d_meio(num_meio,nome_meio,nome_entidade,tipo)
        SELECT num_meio, nome_meio, nome_entidade, 'nao especificado'
        FROM (SELECT num_meio, nome_entidade
                FROM meio
                EXCEPT
                SELECT num_meio, nome_entidade
                FROM meio_combate
                EXCEPT
```

```
        SELECT num_meio, nome_entidade
        FROM meio_apoio
        EXCEPT
        SELECT num_meio, nome_entidade
        FROM meio_socorro) AS t NATURAL JOIN meio;


INSERT INTO d_evento(num_telefone, instante_chamada)
    SELECT num_telefone, instante_chamada
    FROM evento_emergencia;


INSERT INTO d_factos(id_evento,id_meio,id_tempo)
    SELECT e.id_evento, m.id_meio, t.id_tempo
    FROM (d_evento NATURAL JOIN evento_emergencia) AS e,
        (d_meio NATURAL JOIN acciona) AS m,
        d_tempo AS t
    WHERE (SELECT  date_part('day', instante_chamada)
        FROM d_evento
        WHERE id_evento=e.id_evento) = t.dia
    AND   (SELECT  date_part('month', instante_chamada)
        FROM d_evento
        WHERE id_evento=e.id_evento) = t.mes
    AND   (SELECT  date_part('year', instante_chamada)
        FROM d_evento
        WHERE id_evento=e.id_evento) = t.ano;
```

Data Analytics

```
SELECT tipo, ano, mes, COUNT(d_factos)
FROM d_factos NATURAL JOIN d_evento NATURAL JOIN d_tempo NATURAL JOIN d_meio
GROUP BY tipo, ano, mes
UNION
SELECT tipo, ano, null, COUNT(d_factos)
FROM d_factos NATURAL JOIN d_evento NATURAL JOIN d_tempo NATURAL JOIN d_meio
GROUP BY tipo, ano
UNION
SELECT tipo, null, null, COUNT(d_factos)
FROM d_factos NATURAL JOIN d_evento NATURAL JOIN d_meio
GROUP BY tipo;
```