Base de Dados 2018/2019 Taguspark

Projeto de Bases de Dados, Parte 4

Grupo 36

Turno BD817957L05

Professora Raquel Casteleiro

Número	Nome	% esforço por aluno	Esforço em horas
71060	Bruno Rodrigues	31%	10
85254	Alexandre Caldeira	31%	10
87665	Iulian Puscasu	38%	12

Restrições de Integridade

a)

```
CREATE OR REPLACE FUNCTION verifica_solicita()
RETURNS TRIGGER AS $verifica_solicita$
        DECLARE
                localSolicitado VARCHAR(255);
                localValido VARCHAR(255);
        BEGIN
                SELECT morada INTO localSolicitado
                        FROM Vigia
                        WHERE NEW.num_camara = num_camara;
                SELECT DISTINCT morada  INTO localValido
                        FROM audita AS A , eventoemergencia AS e
                        WHERE a.num_processo_socorro = e.num_processo_socorro
                        AND new.id_coordenador = a.id_coordenador
                        AND morada = localSolicitado;
                IF localValido IS NULL THEN
                        RAISE EXCEPTION 'O coordenador % nao pode solicitar um
video da camara %.', NEW.id_coordenador, NEW.num_camara;
                END IF;
                RETURN NEW;
        END;
$verifica_solicita$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_solicita BEFORE INSERT OR UPDATE ON solicita
        FOR EACH ROW EXECUTE PROCEDURE verifica_solicita();
```

b)

```
CREATE OR REPLACE FUNCTION verifica_meio()
RETURNS TRIGGER AS $verifica_meio$
        DECLARE
                meioAlocado INTEGER;
        BEGIN
                SELECT COUNT(*) INTO meioAlocado
                        FROM acciona NATURAL JOIN meio
                        WHERE NEW.num_meio = num_meio
                        AND NEW.nome_entidade = nome_entidade
                        AND NEW.num_processo_socorro = num_processo_socorro;
                IF meioAlocado <> 1 THEN
                        RAISE EXCEPTION 'O meio numero % da entidade % nao foi
accionado no processo %.', NEW.num_meio, NEW.nome_entidade,
NEW.num_processo_socorro;
                END IF;
                RETURN NEW;
        END;
$verifica_meio$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_meio BEFORE INSERT OR UPDATE ON alocado
        FOR EACH ROW EXECUTE PROCEDURE verifica_meio();
```

Índices

a)
1 - Na primeira interrogação vamos aceder aos atributos numCamara, moradaLocal da tabela vigia e ao atributo numCamara da tabela video.
Como todos os acesso são para verificar igualdade de valores, podemos usar um indice de hash, que é mais eficiente a procurar valores, mas pior a inserir e remover. Como só queremos procurar valores, esta é a melhor opção.
Além disso, os atributos numCamara e moradaLocal da tabela vigia estão sempre a ser acedidos em conjunto, por isso faz sentido pô-los no mesmo index.
Assim, temos um index para numCamara da tabela video e um index para numCamara e moradaLocal da tabela vigia

2 - Na segunda interrogação, vamos aceder ao atributo numProcessoSocorro da tabela transporta e aos atributos numProcessoSocorro, numTelefone, instanteChamada da tabela EventoEmergencia.
Para o atributo numProcessoSocorro de ambas as tabelas apenas queremos procurar o valor, mas para numTelefone e instanteChamada da tabela EventoEmergencia queremos comparar valores e ver qual é maior, o que não funciona num indice hash. No entanto, vamos ter um indice hash para a coluna numProcessoSocorro de cada tabela.
Para criar a tabela com o resultado pedido, primeiro acedemos a todas as instâncias dos atributos numProcessoSocorro das duas tabelas e só depois é que acedemos a numTelefone e instanteChamada (para ordenar a tabela), logo faz sentido que tenhamos um index separado para numTelefone e instanteChamada, (mas estando estes os dois no mesmo index).
Este index vai ser do tipo b-tree pois este tipo de index suporta muitas funcionalidades é é bom para uso geral.

b)
1-
```
DROP INDEX vigia_index;
DROP INDEX video_index;

CREATE INDEX vigia_index ON vigia(numCamara, moradaLocal)
USING hash;

CREATE INDEX video_index ON video(numCamara)
USING hash;
```

2-

```
DROP INDEX transporta_index;
DROP INDEX evento_lookup_index;
DROP INDEX evento_sort_index;

CREATE INDEX transporta_index ON transporta(numProcessoSocorro)
USING hash;

CREATE INDEX evento_lookup_index ON
EventoEmergencia(numProcessoSocorro)
USING hash;

CREATE INDEX evento_sort_index ON EventoEmergencia(numTelefone,
instanteChamada)
USING btree;
```

Modelo Multidimensional

```
DROP TABLE IF EXISTS fact_table CASCADE;
DROP TABLE IF EXISTS d_tempo CASCADE;
DROP TABLE IF EXISTS d_meio CASCADE;
DROP TABLE IF EXISTS d_evento CASCADE;

CREATE TABLE d_tempo(
        id_data SERIAL NOT NULL,
        dia NUMERIC(2,0) NOT NULL,
        mes NUMERIC(2,0) NOT NULL,
        ano NUMERIC(4,0) NOT NULL,

        PRIMARY KEY(id_data) );

CREATE TABLE d_evento(
        id_evento SERIAL NOT NULL,
        num_telefone CHAR(9) NOT NULL,
        instante_chamada TIMESTAMP NOT NULL,

        PRIMARY KEY(id_evento));

CREATE TABLE d_meio(
        id_meio SERIAL NOT NULL,
        num_meio INTEGER,
        nome_meio VARCHAR(255),
        nome_entidade VARCHAR(255) NOT NULL,
        tipo VARCHAR(255),

        PRIMARY KEY(id_meio));
```

```
CREATE TABLE fact_table(
	id_evento SERIAL NOT NULL ,
	id_meio SERIAL NOT NULL,
	id_data SERIAL NOT NULL,

	FOREIGN KEY(id_evento) REFERENCES d_evento(id_evento),
	FOREIGN KEY(id_meio) REFERENCES d_meio(id_meio),
	FOREIGN KEY(id_data) REFERENCES d_tempo(id_data) );

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

INSERT INTO d_meio(num_meio, nome_meio, nome_entidade, tipo)
SELECT num_meio, nome_meio, nome_entidade, 'meio de apoio'
FROM meioapoio NATURAL JOIN meio;

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

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

CREATE OR REPLACE FUNCTION year_iteration(date, date) RETURNS VOID
AS $year_iteration$
	DECLARE
		d1 date := $1;
		d2 date :=$2;
	BEGIN

	LOOP
		INSERT INTO d_tempo values (default, extract(day FROM d1),
EXTRACT(month FROM d1), EXTRACT(year FROM d1));

		d1 := d1 + 1;
		EXIT WHEN d1 > d2;
	END LOOP;

	END;
$year_iteration$ LANGUAGE plpgsql;
```

```
SELECT year_iteration('2018-01-01 00:00:00','2018-12-31 23:59:59');

INSERT INTO fact_table(id_evento, id_meio, id_data)
SELECT id_evento, id_meio, id_data
        FROM eventoemergencia AS e, alocado AS al, d_evento AS de, d_meio AS
dm, d_tempo AS dt
        WHERE al.num_processo_socorro = e.num_processo_socorro
        AND de.instante_chamada = e.instante_chamada
        AND dm.num_meio = al.num_meio
        AND dt.dia = extract(day FROM de.instante_chamada)
        AND dt.mes = extract(month FROM de.instante_chamada)
        AND dt.ano = extract(year FROM de.instante_chamada);
```

Data Analytics

```
SELECT tipo, ano, mes, COUNT(id_meio)
        FROM fact_table NATURAL JOIN d_meio NATURAL JOIN d_tempo
        WHERE id_evento = 15
        GROUP BY ROLLUP(tipo, ano, mes);
```