Projeto Base Dados - 4 Entrega

Grupo 30 – Turno L04 – Taras Lykhenko

Abstract

Autores
MariaDuarte, 86474 - 33.3 % (20 horas)
Inês Lacerda, 86436 - 33.3 % (20 horas)
Francisco Rocha, 86414 - 33.3 % (20 horas)

---

1. Restrições de Integridade

```
--###########--------------- a) ---------------###########
CREATE OR REPLACE FUNCTION solicita_permit() RETURNS TRIGGER AS $body$
    DECLARE success INT;
    BEGIN

    success :=  (SELECT count(*) FROM audita NATURAL JOIN eventoEmergencia
              NATURAL JOIN vigia
              WHERE idCoordenador = new.idcoordenador
             AND camNum = new.camnum);
    IF success = 0 THEN
        RAISE EXCEPTION 'restriction a) violated when inserting %', new ;
    ELSE
      RETURN new;
    END IF;
  END;
$body$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS solicita_permit on solicita;
CREATE TRIGGER solicita_permit BEFORE INSERT OR UPDATE ON solicita
  FOR EACH ROW EXECUTE PROCEDURE solicita_permit();

--###########--------------- b) ---------------###########
CREATE OR REPLACE FUNCTION accionado_before_alocado() RETURNS TRIGGER AS $body$
    DECLARE success INT;
    DECLARE morada VARCHAR(255);
    BEGIN
    success := (SELECT count(*) FROM acciona
                WHERE new.numMeio = numMeio
                AND new.nomeEntidade = nomeEntidade
```

```
                AND new.numProcessoSocorro = numProcessoSocorro
            );
    IF success = 0 THEN
        RAISE EXCEPTION 'restriction b) violated';
        ELSE
                RETURN new;
    END IF;
  END;
$body$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS accionado_before_alocado on alocado;
CREATE TRIGGER accionado_before_alocado BEFORE INSERT OR UPDATE ON alocado
  FOR EACH ROW EXECUTE PROCEDURE accionado_before_alocado();
```

2. Índices

```
--Índices baseados em funções de dispersão(hash):
--      são os mais eficientes para testes de igualdade
--      Não podem suportar pesquisas por intervalo
--      Tem uma complexidade de O(1)

--Índices BTREE:
--      Adaptam-se a inserções/remoções
--      As entradas de dados no índice estão odernadas pelo valor da
 ↪  chave de pesquisa e é mantida uma estrutura
-- de pesquisa hierárquica que dirige as pesquisas para as páginas
 ↪  de entradas de dados corretas
--      Tem uma complexidade de O(log(n))

-- **BTREE é melhor que hash em casos que requerem vários hash
 ↪  codes.
-- A função de dispersão (hash) recebe como parâmetro o valor da
 ↪  chave de pesquisa e determina o contentor (bucket)
-- em que se situa o registo, se tivermos vários hash codes, temos
 ↪  que escolher um dos códigos para associar a um contentor.
-- Depois, se quisermos pesquisar, usando o hash code, temos que
 ↪  pesquisar todos os buckets, o que demoraria muito mais
-- tempo do que simplesmente usar uma btree.

--###########--------------- 1) ---------------###########
EXPLAIN SELECT dataHoraInicio, dataHoraFim
 FROM video V, vigia I
 WHERE V.camNum = I.camNum
        AND V.camNum = 10
```

```
       AND I.moradaLocal = 'Loures';

--Verificação da existençia dos índices e eliminação dos mesmos se
 ↪  existirem
DROP INDEX IF EXISTS video_cam;
DROP INDEX IF EXISTS vigia_cam;
DROP INDEX IF EXISTS vigia_mor;
-- Tendo em conta que existem três igualdades na condição WHERE, e
 ↪  assumindo que
--as tabelas "video" e "vigia" não sofrerão muitas
 ↪  alterações(inserções e remoções)
-- é necessário criar os seguintes indices de dispersão:
CREATE INDEX video_cam ON video USING HASH(camNum);
CREATE INDEX vigia_cam ON vigia USING HASH(camNum);
CREATE INDEX vigia_mor ON vigia USING HASH(moradaLocal);


--###########--------------- 2) ---------------###########
EXPLAIN SELECT sum(numVitimas)
FROM transporta T, eventoEmergencia E
WHERE T.numProcessoSocorro = E.numProcessoSocorro
GROUP BY numTelefone, instanteChamada

--Verificação da existençia dos índices e eliminação dos mesmos se
 ↪  existirem
DROP INDEX IF EXISTS trans_numP;
DROP INDEX IF EXISTS even_numP;
DROP INDEX IF EXISTS evento_numTinstC;
-- Apesar da existençia da igualdade na condição WHERE,
--como provavelmente as tabelas "transporta" e "eventoEmergencia"
--sofreram muitas alterações é mais eficiente escolher indices
 ↪  árvore B+
--devido a sua boa adptacao com inserções e remoções:
CREATE INDEX trans_numP ON transporta USING
 ↪  BTREE(numProcessoSocorro);
CREATE INDEX even_numP ON eventoEmergencia USING
 ↪  BTREE(numProcessoSocorro);
--Agora observando a condição GROUP BY é necessário criar um indice
 ↪  composto
--árvore B+ :
```

```
CREATE INDEX evento_numTinstC ON eventoEmergencia USING
 ↪  BTREE(numTelefone,instanteChamada);
```

3. Modelo Multidimensional

```
DROP TABLE IF EXISTS factos;
DROP TABLE IF EXISTS d_evento;
DROP TABLE IF EXISTS d_meio;
DROP TABLE IF EXISTS d_tempo;

CREATE OR REPLACE FUNCTION getTime_id (tempo timestamp)
  RETURNS INTEGER AS $body$
  DECLARE dia INTEGER := extract( day FROM (tempo));
  DECLARE mes INTEGER := extract( month FROM (tempo));
  DECLARE ano INTEGER := extract( year FROM (tempo));
  BEGIN
    return dia  + mes*100 + ano * 10000 ;
  END;
$body$ LANGUAGE plpgsql;

/** Function that generates the time intervals
 * from a given minimum to a given maximum  **/
CREATE OR REPLACE FUNCTION genTimeIntervals(min timestamp,
                                            max timestamp)
  RETURNS TABLE (tempo timestamp) AS $body$
  BEGIN
        return QUERY SELECT current_date + i
                FROM generate_series(min - current_date,
                        max - current_date ) i;
  END;
$body$ LANGUAGE plpgsql;

CREATE TABLE d_evento(
  idEvento SERIAL UNIQUE,
  numTelefone NUMERIC(255) NOT NULL,
  instanteChamada TIMESTAMP NOT NULL,
  PRIMARY KEY(idEvento)
);

CREATE TABLE d_meio(
  idMeio SERIAL UNIQUE,
  numMeio INTEGER NOT NULL,
  nomeMeio VARCHAR(255),
  nomeEntidade VARCHAR(255) NOT NULL,
  tipo VARCHAR(255) NOT NULL,
  PRIMARY KEY(idMeio)
);

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)
);
```

```sql
CREATE TABLE factos(
  idFacto SERIAL UNIQUE,
  idEvento INTEGER,
  idMeio INTEGER,
  tempo_id INTEGER,
  PRIMARY KEY(idFacto),
  FOREIGN KEY(idEvento) REFERENCES d_evento(idEvento),
  FOREIGN KEY(idMeio) REFERENCES d_meio(idMeio),
  FOREIGN KEY(tempo_id) REFERENCES d_tempo(tempo_id)
);

INSERT INTO d_evento(numTelefone,instanteChamada)
SELECT numTelefone, instanteChamada FROM eventoEmergencia;

INSERT INTO d_meio(numMeio,nomeMeio,nomeEntidade,tipo)
SELECT numMeio,nomeMeio, nomeEntidade,'Apoio'
FROM meioApoio NATURAL JOIN meio;

INSERT INTO d_meio(numMeio,nomeMeio,nomeEntidade,tipo)
SELECT numMeio,nomeMeio, nomeEntidade,'Combate'
FROM meioCombate  NATURAL JOIN meio;

INSERT INTO d_meio(numMeio,nomeMeio,nomeEntidade,tipo)
SELECT numMeio,nomeMeio, nomeEntidade,'Socorro'
FROM meioSocorro  NATURAL JOIN meio;

INSERT INTO d_tempo(tempo_id,dia,mes,ano)
select getTime_id(timerange),
       extract(day FROM timerange),
       extract(month FROM timerange),
       extract(year FROM timerange)
from generate_series((SELECT min(instanteChamada) -- min
                      FROM eventoEmergencia),
                     (SELECT max(instanteChamada)
                      FROM eventoEmergencia), '1day') AS timerange;

INSERT INTO factos(idEvento,idMeio,tempo_id)
select idEvento,idMeio,getTime_id(instanteChamada) as tempo
FROM eventoEmergencia
NATURAL JOIN d_evento
JOIN d_tempo
ON (getTime_id(eventoEmergencia.instanteChamada) = d_tempo.tempo_id)
NATURAL JOIN d_meio NATURAL JOIN acciona;
```

4. Data Analytics

```sql
(SELECT ano,mes,
 sum(case when tipo  = 'Apoio' then 1 else 0 end) as Apoio,
  sum(case when tipo  = 'Combate' then 1 else 0 end) as Combate,
  sum(case when tipo  = 'Socorro' then 1 else 0 end) as Socorro
FROM d_tempo
NATURAL JOIN d_meio
```

```
NATURAL JOIN factos
WHERE idEvento = '15'
GROUP BY ano,mes ORDER BY ano, mes)

UNION

(SELECT ano, NULL,
 sum(case when tipo  = 'Apoio' then 1 else 0 end) as Apoio,
  sum(case when tipo  = 'Combate' then 1 else 0 end) as Combate,
  sum(case when tipo  = 'Socorro' then 1 else 0 end) as Socorro
FROM d_tempo
NATURAL JOIN d_meio
NATURAL JOIN factos
WHERE idEvento = '15'
GROUP BY ano ORDER BY ano)

UNION

(SELECT NULL, NULL,
 sum(case when tipo  = 'Apoio' then 1 else 0 end) as Apoio,
  sum(case when tipo  = 'Combate' then 1 else 0 end) as Combate,
  sum(case when tipo  = 'Socorro' then 1 else 0 end) as Socorro
FROM d_tempo
NATURAL JOIN d_meio
NATURAL JOIN factos
WHERE idEvento = '15')
```