INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

PROJETO DE BASE DE DADOS – PARTE 4

Grupo 39 | 4ª Feira: 11:00h – 12:30h

Docente: André da Silva Pereira

ALUNO	NÚMERO	HORAS	PERCENTAGEM RELATIVA
Daniel Pereira	89425	9h	33,3(3)%
Tiago Gonçalves	89547	9h	33,3(3)%
Tiago Barroso	89549	9h	33,3(3)%

RESTRIÇÕES DE INTEGRIDADE:

```
CREATE FUNCTION cancel_anomalia_traducao_zona_func()
RETURNS TRIGGER
AS
$$
DECLARE z1p1 POINT;
DECLARE z1p2 POINT;
DECLARE z2p1 POINT;
DECLARE z2p2 POINT;
BEGIN
   SELECT zona[0] INTO z1p1 FROM anomalia WHERE new.id = anomalia.id;
   SELECT zona[1] INTO z1p2 FROM anomalia WHERE new.id = anomalia.id;
   SELECT new.zona2[0] INTO z2p1;
            SELECT new.zona2[1] INTO z2p2;
   IF NOT (z1p1[0] < z2p2[0] OR z2p1[0] < z1p2[0] OR z1p2[1] > z2p1[1] OR z2p2[1] > z1p1[1]) THEN
      RAISE EXCEPTION '(RI-1) A zona da anomalia_tradução, % %, não se pode sobrepor à zona da
anomalia correspondente, % %', z2p1,z2p2,z1p1,z1p2;
   END IF;
   return new;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER cancel_anomalia_traducao_zona_insert BEFORE INSERT ON anomalia_traducao
FOR EACH ROW EXECUTE FUNCTION cancel_anomalia_traducao_zona_func();

CREATE TRIGGER cancel_anomalia_traducao_zona_update BEFORE UPDATE ON anomalia_traducao
FOR EACH ROW EXECUTE FUNCTION cancel_anomalia_traducao_zona_func();

CREATE FUNCTION cancel_anomalia_zona_func()
RETURNS TRIGGER
AS
$$
DECLARE z1p1 POINT;
DECLARE z1p2 POINT;
DECLARE z2p1 POINT;
DECLARE z2p2 POINT;
BEGIN
   IF EXISTS (SELECT * FROM anomalia_traducao WHERE anomalia_traducao.id = new.id) THEN
      SELECT zona2[0] INTO z2p1 FROM anomalia_traducao WHERE new.id = anomalia_traducao.id;
      SELECT zona2[1] INTO z2p2 FROM anomalia_traducao WHERE new.id = anomalia_traducao.id;
      SELECT new.zona[0] INTO z1p1;
      SELECT new.zona[1] INTO z1p2;
      IF NOT (z1p1[0] < z2p2[0] OR z2p1[0] < z1p2[0] OR z1p2[1] > z2p1[1] OR z2p2[1] > z1p1[1]) THEN
         RAISE EXCEPTION '(RI-1) A zona da anomalia_tradução, % %, não se pode sobrepor à zona da
anomalia correspondente, % %', z2p1,z2p2,z1p1,z1p2;
      END IF;
   END IF;
   return new;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER cancel_anomalia_zona_update BEFORE UPDATE ON anomalia
FOR EACH ROW EXECUTE FUNCTION cancel_anomalia_zona_func();

CREATE FUNCTION verify_utilizador_proc()
RETURNS TRIGGER
```

```
AS
$$
BEGIN
  IF (NOT EXISTS (SELECT email FROM utilizador_regular WHERE utilizador_regular.email = new.email)
AND NOT EXISTS (SELECT email FROM utilizador_qualificado WHERE utilizador_qualificado.email =
new.email)) THEN
     RAISE EXCEPTION '(RI-4) email, %, tem de figurar em utilizador_regular ou em
utilizador_qualificado', new.email;
          END IF;
  return new;
END;
$$
LANGUAGE plpgsql;

CREATE FUNCTION remove_utilizador_proc()
RETURNS TRIGGER
AS
$$
BEGIN
  IF (EXISTS (SELECT email FROM utilizador WHERE utilizador.email = old.email) AND NOT EXISTS
(SELECT email FROM utilizador_regular WHERE utilizador_regular.email = old.email) AND NOT EXISTS
(SELECT email FROM utilizador_qualificado WHERE utilizador_qualificado.email = old.email)) THEN
     RAISE EXCEPTION '(RI-4) email, %, tem de figurar em utilizador_regular ou em
utilizador_qualificado', old.email;
          END IF;
  return new;
END;
$$
LANGUAGE plpgsql;

CREATE FUNCTION verify_utilizador_qualificado_proc()
RETURNS TRIGGER
AS
$$
BEGIN
  IF (EXISTS (SELECT email FROM utilizador_regular WHERE utilizador_regular.email = new.email))
THEN
     RAISE EXCEPTION '(RI-5) email, %, nao pode figurar utilizador_regular ', new.email;
          END IF;
  return new;
END;
$$
LANGUAGE plpgsql;

CREATE FUNCTION verify_utilizador_regular_proc()
RETURNS TRIGGER
AS
$$
BEGIN
  IF (EXISTS (SELECT email FROM utilizador_qualificado WHERE utilizador_qualificado.email =
new.email)) THEN
     RAISE EXCEPTION '(RI-6) email, %, nao pode figurar utilizador_qualificado ', new.email;
          END IF;
  return new;
END;
$$
LANGUAGE plpgsql;

CREATE CONSTRAINT TRIGGER verify_utilizador AFTER INSERT ON utilizador
```

```
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE verify_utilizador_proc();

CREATE CONSTRAINT TRIGGER verify_utilizador_regular BEFORE INSERT ON utilizador_regular
FOR EACH ROW EXECUTE PROCEDURE verify_utilizador_regular_proc();

CREATE CONSTRAINT TRIGGER verify_utilizador_qualificado BEFORE INSERT ON utilizador_qualificado
FOR EACH ROW EXECUTE PROCEDURE verify_utilizador_qualificado_proc();

CREATE TRIGGER update_utilizador_regular BEFORE UPDATE ON utilizador_regular
FOR EACH ROW EXECUTE PROCEDURE verify_utilizador_regular_proc();

CREATE TRIGGER update_utilizador_qualificado BEFORE UPDATE ON utilizador_qualificado
FOR EACH ROW EXECUTE PROCEDURE verify_utilizador_qualificado_proc();

CREATE CONSTRAINT TRIGGER remove_utilizador_regular AFTER DELETE ON utilizador_regular
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE remove_utilizador_proc();

CREATE CONSTRAINT TRIGGER remove_utilizador_qualificado AFTER DELETE ON utilizador_qualificado
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE remove_utilizador_proc();
```

ÍNDICES:

1.1 - Não usamos nenhum índice secundário pois a condição não é muito seletiva (seleciona mais de 10% das entradas da tabela) logo dar scan sequencial é mais eficiente.

1.2 - Como neste caso a condição é bastante seletiva (seleciona menos de 0.001% das entradas da tabela), é mais vantajoso criar um índice para data_hora usando BTREE pois a query que temos é uma range query e BTREE suporta range queries, ao contrário da hash table.

CREATE INDEX index1 ON correcao USING BTREE(data_hora);

2 - Usamos um índice secundário para anomalia_id usando hash table pois só temos uma condição de igualdade na query. Nestas condições, é melhor escolher hash table pois esta estrutura suporta melhor queries com condições de igualdade comparado com BTREE.

CREATE INDEX index2 ON incidencia USING HASH(anomalia_id);

3.1 - Não usamos nenhum índice secundário pois a condição não é muito seletiva (seleciona mais de 10% das entradas da tabela) logo dar scan sequencial é mais eficiente.

3.2 - Como neste caso a condição é bastante seletiva (seleciona menos de 0.001% das entradas da tabela), é mais vantajoso criar um índice para anomalia_id usando BTREE pois a query que temos é uma range query e BTREE suporta range queries, ao contrário da hash table.

CREATE INDEX index3 ON correcao USING BTREE(anomalia_id);

4 - Escolhemos um índice com chave composta para lingua, ts e tem_anomalia_redacao pois temos uma condição de seleção para estes três atributos. É utilizado BTREE em vez de hash table pois há condições de desigualdade e hash table só suporta eficientemente queries que só utilizam condições com igualdades. Se ts1 e ts2 forem próximos um do outro (ex: 1 semana) e a tabela tiver anomalias registadas ao longo de alguns anos, ordenar primeiro por ts é mais vantajoso porque a condição do timestamp é mais seletiva logo duas linhas com línguas iguais vão estar mais próximas uma da outra. O atributo tem_anomalia_redacao é o menos seletivo pois só tem 2 valores possíveis, logo é o último pelo qual é ordenado.

CREATE INDEX index4 ON anomalia USING BTREE(ts, lingua, tem_anomalia_redacao);

MODELO MULTIDIMENSIONAL:

```
CREATE TABLE d_utilizador
  (id_utilizador        SERIAL          NOT NULL,
   email                VARCHAR(255)    NOT NULL,
   tipo                 VARCHAR(255)    NOT NULL,
   PRIMARY KEY(id_utilizador));

CREATE TABLE d_tempo
  (id_tempo             SERIAL          NOT NULL,
   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,
   PRIMARY KEY(id_tempo));
```

```
CREATE TABLE d_local
  (id_local              SERIAL              NOT NULL,
   latitude              FLOAT               NOT NULL,
   longitude             FLOAT               NOT NULL,
   nome                  VARCHAR(255)        NOT NULL,
   PRIMARY KEY(id_local));

CREATE TABLE d_lingua
   (id_lingua            SERIAL              NOT NULL,
    lingua               VARCHAR(255)        NOT NULL,
    PRIMARY KEY(id_lingua));

CREATE TABLE f_anomalia
   (id_utilizador        SERIAL              NOT NULL,
    id_tempo             SERIAL              NOT NULL,
    id_local             SERIAL              NOT NULL,
    id_lingua            SERIAL              NOT NULL,
    tipo_anomalia        VARCHAR(255)        NOT NULL,
    com_proposta         BOOLEAN             NOT NULL,
    PRIMARY KEY(id_utilizador, id_tempo, id_local, id_lingua),
    FOREIGN KEY(id_utilizador) REFERENCES d_utilizador(id_utilizador) ON DELETE CASCADE ON
UPDATE CASCADE,
    FOREIGN KEY(id_tempo) REFERENCES d_tempo(id_tempo) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY(id_local) REFERENCES d_local(id_local) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(id_lingua) REFERENCES d_lingua(id_lingua) ON DELETE CASCADE ON UPDATE
CASCADE);

INSERT INTO d_utilizador(email, tipo) (SELECT email, 'utilizador_qualificado' FROM
utilizador_qualificado);
INSERT INTO d_utilizador(email, tipo) (SELECT email, 'utilizador_regular' FROM utilizador_regular);

INSERT INTO d_tempo(dia, dia_da_semana, semana, mes, trimestre, ano) (SELECT DISTINCT
EXTRACT(DAY FROM ts), EXTRACT(DOW FROM ts), EXTRACT(WEEK FROM ts), EXTRACT(MONTH FROM
ts), EXTRACT(QUARTER FROM ts), EXTRACT(YEAR FROM ts) FROM anomalia);

INSERT INTO d_local(latitude, longitude, nome) (SELECT latitude, longitude, nome FROM local_publico);

INSERT INTO d_lingua(lingua) (SELECT DISTINCT lingua FROM anomalia);


INSERT INTO f_anomalia(id_utilizador, id_tempo, id_local, id_lingua, tipo_anomalia, com_proposta) (

   SELECT id_utilizador, id_tempo, id_local, id_lingua, 'anomalia traducao', TRUE
         FROM anomalia NATURAL JOIN anomalia_traducao
                  INNER JOIN incidencia ON incidencia.anomalia_id = anomalia.id
                  INNER JOIN item ON item.id = incidencia.item_id
                  INNER JOIN d_utilizador ON incidencia.email = d_utilizador.email
                  INNER JOIN d_local ON d_local.latitude = item.latitude AND d_local.longitude =
item.longitude
                  INNER JOIN d_lingua ON anomalia.lingua = d_lingua.lingua
                  INNER JOIN d_tempo ON EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND
EXTRACT(DOW FROM anomalia.ts) = d_tempo.dia_da_semana AND EXTRACT(WEEK FROM anomalia.ts)
= d_tempo.semana AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND EXTRACT(QUARTER
FROM anomalia.ts) = d_tempo.trimestre AND EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
                  WHERE anomalia.id IN (SELECT anomalia_id AS id FROM correcao)
         UNION
         SELECT id_utilizador, id_tempo, id_local, id_lingua, 'anomalia traducao', FALSE
                  FROM anomalia NATURAL JOIN anomalia_traducao
```

```
                INNER JOIN incidencia ON incidencia.anomalia_id = anomalia.id
                INNER JOIN item ON item.id = incidencia.item_id
                INNER JOIN d_utilizador ON incidencia.email = d_utilizador.email
                INNER JOIN d_local ON d_local.latitude = item.latitude AND d_local.longitude =
item.longitude
                INNER JOIN d_lingua ON anomalia.lingua = d_lingua.lingua
                INNER JOIN d_tempo ON EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND
EXTRACT(DOW FROM anomalia.ts) = d_tempo.dia_da_semana AND EXTRACT(WEEK FROM anomalia.ts)
= d_tempo.semana AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND EXTRACT(QUARTER
FROM anomalia.ts) = d_tempo.trimestre AND EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
                WHERE anomalia.id NOT IN (SELECT anomalia_id AS id FROM correcao)
        UNION
        SELECT id_utilizador, id_tempo, id_local, id_lingua, 'anomalia redacao', FALSE
                FROM anomalia
                INNER JOIN incidencia ON incidencia.anomalia_id = anomalia.id
                INNER JOIN item ON item.id = incidencia.item_id
                INNER JOIN d_utilizador ON incidencia.email = d_utilizador.email
                INNER JOIN d_local ON d_local.latitude = item.latitude AND d_local.longitude =
item.longitude
                INNER JOIN d_lingua ON anomalia.lingua = d_lingua.lingua
                INNER JOIN d_tempo ON EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND
EXTRACT(DOW FROM anomalia.ts) = d_tempo.dia_da_semana AND EXTRACT(WEEK FROM anomalia.ts)
= d_tempo.semana AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND EXTRACT(QUARTER
FROM anomalia.ts) = d_tempo.trimestre AND EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
                WHERE anomalia.id NOT IN (SELECT anomalia_id AS id FROM correcao)
                AND anomalia.id NOT IN (SELECT id FROM anomalia_traducao)
        UNION
        SELECT id_utilizador, id_tempo, id_local, id_lingua, 'anomalia redacao', TRUE
                FROM anomalia
                INNER JOIN incidencia ON incidencia.anomalia_id = anomalia.id
                INNER JOIN item ON item.id = incidencia.item_id
                INNER JOIN d_utilizador ON incidencia.email = d_utilizador.email
                INNER JOIN d_local ON d_local.latitude = item.latitude AND d_local.longitude =
item.longitude
                INNER JOIN d_lingua ON anomalia.lingua = d_lingua.lingua
                INNER JOIN d_tempo ON EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND
EXTRACT(DOW FROM anomalia.ts) = d_tempo.dia_da_semana AND EXTRACT(WEEK FROM anomalia.ts)
= d_tempo.semana AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND EXTRACT(QUARTER
FROM anomalia.ts) = d_tempo.trimestre AND EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
                WHERE anomalia.id IN (SELECT anomalia_id AS id FROM correcao)
                AND anomalia.id NOT IN (SELECT id FROM anomalia_traducao)
)
```

DATA ANALYTICS:

```
SELECT tipo_anomalia, lingua, dia_da_semana, COUNT(*)
FROM f_anomalia NATURAL JOIN d_lingua NATURAL JOIN d_tempo
GROUP BY CUBE (tipo_anomalia, lingua, dia_da_semana)
ORDER BY (tipo_anomalia, lingua, dia_da_semana)
```