Projeto de Bases de Dados, Parte 4

Trabalho realizado por:

Afonso Vasconcelos, 90698
Pedro Leitão, 90764
Rodrigo Rosa, 90777

Contribuição para o projeto:

90698	33.3(3)%	10 horas
90764	33.3(3)%	10 horas
90777	33.3(3)%	10 horas

Grupo 16

Turno: Sexta-feira, 8:30h (BD8179577L06)

Docente: Paulo Carreira

Restrições de Integridade:

RI-1:

```
CREATE OR REPLACE FUNCTION check_anomalia_traducao()
RETURNS TRIGGER AS $BODY$
BEGIN
IF EXISTS
        (SELECT zona
        FROM anomalia
        NATURAL JOIN anomalia_traducao
        WHERE (NEW.anomalia_id = anomalia_id AND NEW.zona2&&zona) )
        THEN RAISE EXCEPTION 'Sobreposicao de anomalias';
        ELSE RETURN NEW;
        END IF;
END;
$BODY$ LANGUAGE plpgsql;
CREATE CONSTRAINT TRIGGER checkAnomalia
AFTER INSERT ON anomalia_traducao
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE check_anomalia_traducao();
```

RI-4:

```
CREATE OR REPLACE FUNCTION check_email_utilizador()
RETURNS TRIGGER AS $BODY$
BEGIN
        IF(NOT EXISTS(SELECT email FROM utilizador_regular WHERE (NEW.email = email))
                AND
                NOT EXISTS(SELECT email
                        FROM utilizador_qualificado WHERE (NEW.email = email) ) )
                THEN RAISE EXCEPTION
                        'Email não pertence a utilizador regular ou qualificado';
                END IF;
                 RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;
CREATE CONSTRAINT TRIGGER checkEmail
AFTER INSERT ON utilizador
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE check_email_utilizador();
```

RI-5:

```
CREATE OR REPLACE FUNCTION check_email_qualificado()
RETURNS TRIGGER AS $BODY$
BEGIN
        IF(EXISTS(SELECT email FROM utilizador_regular WHERE (NEW.email = email)))
                THEN RAISE EXCEPTION 'Email já pertence a um utilizador regular';
        END IF;
        RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;
CREATE CONSTRAINT TRIGGER checkQualificado
AFTER INSERT ON utilizador_qualificado
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE check_email_qualificado();
```

RI-6:

```
CREATE OR REPLACE FUNCTION check_email_regular()
RETURNS TRIGGER AS $BODY$
BEGIN
        IF(EXISTS(SELECT email FROM utilizador_qualificado WHERE (NEW.email = email)))
                THEN RAISE EXCEPTION 'Email já pertence a um utilizador qualificado';
        END IF;
        RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;
CREATE CONSTRAINT TRIGGER checkRegular
AFTER INSERT ON utilizador_regular
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE check_email_regular();
```

Índices:

1.

1.1.Neste caso, se a percentagem de **linhas** que satisfazem a condição imposta for baixa, um **índice b+tree** sobre a coluna **data_hora** da tabela **proposta_de_correcao** será o mais indicado, pois trata-se de uma pesquisa sobre um intervalo pequeno de dados, caso contrário, o índice deixa de ser útil.

Código:

```
CREATE INDEX data_hora_idx ON proposta_de_correcao USING BTREE (data_hora);
```

1.2. De modo a acelerar a execução do pedido será necessária a criação de um **índice b+tree** sobre a coluna **data_hora** da tabela **proposta_de_correcao**, visto estarmos perante uma pesquisa sobre um intervalo pequeno de dados, logo um índice b+tree é bastante eficiente para pesquisas deste tipo.

Código:

```
CREATE INDEX data_hora_idx ON proposta_de_correcao USING BTREE (data_hora);
```

2. Neste caso, como estamos perante um teste de igualdade, de modo a otimizar o pedido, deve ser definido um **índice do tipo hash** sobre a coluna **anomalia_id**, da tabela **incidência** .

Código:

```
CREATE INDEX data_hora_idx ON incidencia USING HASH (anomalia_id);
```

3.

3.1.Neste caso, se a percentagem de linhas que satisfazem a condição imposta for baixa, um **índice b+tree** sobre a coluna **anomalia_id** da tabela **correcao** será o mais indicado, pois trata-se de uma pesquisa sobre um intervalo pequeno de dados, caso contrário, o índice não é muito útil(sendo inútil caso essa percentagem seja superior a 90%).

Código:

```
CREATE INDEX anomalia_idx ON correcao USING BTREE (anomalia_id);
```

3.2. De modo a acelerar a execução do pedido, será necessária a criação de um **índice b+tree** sobre a coluna **anomalia_id** da tabela **correcao**, visto estarmos perante uma pesquisa sobre um intervalo cujas linhas que nele se encontram serem poucas, sendo por isto um **índice b+tree** bastante eficiente para executar essa pesquisa.

Código:

```
CREATE INDEX anomalia_idx ON correcao USING BTREE (anomalia_id);
```

4. Deve ser criado um **índice composto do tipo b+tree** sobre as colunas <**language, ts**> da tabela **anomalia**, visto tratar-se de uma pesquisa numa combinação de campos.

Código:

```
CREATE INDEX language_ts_idx ON anomalia USING BTREE (language, ts);
```

Modelo Multidimensional:

```
DROP TABLE IF EXISTS d_utilizador CASCADE;
DROP TABLE IF EXISTS d_tempo CASCADE;
DROP TABLE IF EXISTS d_local CASCADE;
DROP TABLE IF EXISTS d_lingua CASCADE;
DROP TABLE IF EXISTS f_anomalia CASCADE;

CREATE TABLE d_utilizador(
  id_utilizador SERIAL NOT NULL,
  email VARCHAR(254) NOT NULL,
  tipo VARCHAR(11) 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 NUMERIC(9,6) NOT NULL,
  longitude NUMERIC(9,6) NOT NULL,
  nome VARCHAR(30) NOT NULL,
  PRIMARY KEY(id_local)
);

CREATE TABLE d_lingua(
  id_lingua SERIAL NOT NULL,
  lingua CHAR(3) NOT NULL,
  PRIMARY KEY(id_lingua)
);

CREATE TABLE f_anomalia(
  id_utilizador INTEGER NOT NULL,
  id_tempo INTEGER NOT NULL,
  id_local INTEGER NOT NULL,
  id_lingua INTEGER NOT NULL,
  tipo_anomalia VARCHAR(9) 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, 'regular' FROM utilizador_regular;
INSERT INTO d_utilizador(email, tipo)
  SELECT email, 'qualificado' FROM utilizador_qualificado;

INSERT INTO d_tempo(dia, dia_da_semana, semana, mes, trimestre, ano)
  SELECT EXTRACT(DAY FROM ts) AS dia,
  date_part('dow', ts),
  EXTRACT(WEEK FROM ts) AS semana,
  EXTRACT(MONTH FROM ts) AS mes,
  date_part('quarter', ts),
  EXTRACT(YEAR FROM ts) AS ano
  FROM anomalia;

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

INSERT INTO d_lingua(lingua)
  SELECT 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, 'Tradução', False
  FROM d_utilizador
    NATURAL JOIN incidencia
    NATURAL JOIN anomalia
    NATURAL JOIN anomalia_traducao
    NATURAL JOIN item
    NATURAL JOIN d_local
    NATURAL JOIN d_lingua
    NATURAL JOIN d_tempo
    WHERE EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND
EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
  EXCEPT(
    SELECT id_utilizador, id_tempo, id_local, id_lingua, 'Tradução', False
    FROM d_utilizador
      NATURAL JOIN incidencia
      NATURAL JOIN anomalia
      NATURAL JOIN anomalia_traducao
      NATURAL JOIN item
      NATURAL JOIN d_local
      NATURAL JOIN d_lingua
      NATURAL JOIN proposta_de_correcao
      NATURAL JOIN d_tempo
  );

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, 'Tradução', True
  FROM d_utilizador
    NATURAL JOIN incidencia
    NATURAL JOIN anomalia
    NATURAL JOIN anomalia_traducao
    NATURAL JOIN item
    NATURAL JOIN d_local
    NATURAL JOIN d_lingua
    NATURAL JOIN d_tempo
  WHERE EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND
EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano;
```

```
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, 'Redação', False
  FROM d_utilizador
    NATURAL JOIN incidencia
    NATURAL JOIN anomalia
    NATURAL JOIN item
    NATURAL JOIN d_local
    NATURAL JOIN d_lingua
    NATURAL JOIN d_tempo
  WHERE tem_anomalia_redacao = True AND
    EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND
EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano
  EXCEPT(
    SELECT id_utilizador, id_tempo, id_local, id_lingua, 'Redação', False
    FROM d_utilizador
      NATURAL JOIN incidencia
      NATURAL JOIN anomalia
      NATURAL JOIN item
      NATURAL JOIN d_local
      NATURAL JOIN d_lingua
      NATURAL JOIN d_tempo
      NATURAL JOIN proposta_de_correcao
    );

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, 'Redação', True
    FROM d_utilizador
    NATURAL JOIN incidencia
    NATURAL JOIN anomalia
    NATURAL JOIN item
    NATURAL JOIN d_local
    NATURAL JOIN d_lingua
    NATURAL JOIN d_tempo
    NATURAL JOIN proposta_de_correcao
  WHERE tem_anomalia_redacao = True AND
    EXTRACT(DAY FROM anomalia.ts) = d_tempo.dia AND EXTRACT(MONTH FROM anomalia.ts) = d_tempo.mes AND
EXTRACT(YEAR FROM anomalia.ts) = d_tempo.ano;
```

Data Analytics:

```
SELECT tipo, '' AS lingua, '' AS dia_da_semana, COUNT(*)
FROM d_utilizador NATURAL JOIN f_anomalia
GROUP BY tipo
UNION
SELECT '', lingua, '', COUNT(*)
FROM d_lingua NATURAL JOIN f_anomalia
GROUP BY lingua
UNION
SELECT '', '', CAST(dia_da_semana AS VARCHAR(11)), COUNT(*)
FROM d_tempo NATURAL JOIN f_anomalia
GROUP BY dia_da_semana
ORDER BY dia_da_semana, lingua, tipo;
```