PROJETO DE BASES DE DADOS
PARTE 4

Grupo 20
Turno: 3ª-feira 8h30
Docente: Paulo Carreira

		Horas	% de trabalho realizado
Joana Mendonça	83597	12	~33%
Gonçalo Guerra	83899	12	~33%
Filipe Colaço	84715	12	~33%

Restrições de Integridade

RI-1: a zona da anomalia_tradução não se pode sobrepor à zona da anomalia correspondente

É usada aqui a função ?#, que verifica se duas variáveis do tipo box se intercetam.

```
create or replace function doCheckZona2()
    returns trigger as
$BODY$
begin
    if exists(select 1 from anomalia as na
              where an.anomalia_id = new.anomalia_id and an.zona ?# new.zona2) then
        raise exception 'zona2 sobrepõe-se a zona';
    end if;

    return new;
end;
$BODY$
LANGUAGE plpgsql;

create trigger check_zona2
    before insert on anomalia_traducao
    for each row
    execute procedure doCheckZona2();
```

RI-4: email de utilizador tem de figurar em utilizador_qualificado ou utilizador_regular

A implementação desta restrição passa por requerer que, aquando da inserção de um valor na tabela `utilizador`, o `email` desse `utilizador` também seja inserido ou em `utilizador_regular` ou em `utilizador_regular`. Para fazer a distinção entre regular e qualificado na tabela `utilizador`, é também adicionado uma nova coluna `type` a `utilizador`.

```
create or replace function doCheckUtilizador()
    returns trigger as
$BODY$
begin
    if new.type = 'Q' then
        insert into utilizador_qualificado values (new.email);

    elsif new.type = 'R' then
        insert into utilizador_regular values (new.email);

    else
        raise exception '"%" não é um tipo válido de utilizador', new.type;
    end if;

    return new;
end;
$BODY$
LANGUAGE plpgsql;

create trigger check_utilizador
    after insert on utilizador
    for each row
    execute procedure doCheckUtilizador();
```

RI-5: email não pode figurar em utilizador_regular
RI-6: email não pode figurar em utilizador_qualificado

Nesta restrição, é feita uma simples procura na tabela oposta de onde estamos a tentar inserir.

```
create or replace function doCheckUtilizadorQR()
    returns trigger as
$BODY$
begin
    if TG_TABLE_NAME = 'utilizador_qualificado' and
        exists(select 1 from utilizador_regular as ur where new.email = ur.email) then
        raise exception key '"%" já existe em utilizador_regular', new.email;
    end if;

    if TG_TABLE_NAME = 'utilizador_regular' and
        exists(select 1 from utilizador_qualificado as uq where new.email = uq.email)
        then
        raise exception key '"%" já existe em utilizador_qualificado', new.email;
    end if;

    return new;
end;
$BODY$
LANGUAGE plpgsql;

create trigger check_utilizador_qualificado
    before insert on utilizador_qualificado
    for each row
    execute procedure doCheckUtilizadorQR();

create trigger check_utilizador_regular
    before insert on utilizador_regular
    for each row
    execute procedure doCheckUtilizadorQR();
```

Índices

1.1.

```
create index data_hora_idx on proposta_de_correcao using hash(data_hora);
```

2. Índice único, por a chave de pesquisa conter a chave candidata.

```
create unique index anomalia_id_idx on incidencia(anomalia_id);
```

3.1.

```
create index index anomalia_id_idx2 on correcao using btree(anomalia_id);
```

4. De notar que a solução não é compatível com o PostgreSQL, visto não suportar índices agrupados (clustered).

```
create clustered index anomalia_id_idx3 on anomalia(ts, tem_anomalia_redacao, lingua);
```

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 unique,
    email varchar(255) not null unique,
    type char(1) not null
);

create table d_tempo (
    id_tempo serial not null unique,
    dia int not null,
    dia_da_semana varchar(9) not null,
    semana int not null,
    mes int not null,
    trimestre int not null,
    ano int not null
);

create table d_local (
    id_local serial not null unique,
    latitude numeric(8,6) not null,
    longitude numeric(9,6) not null,
    nome varchar(255) not null
);

create table d_lingua (
    id_lingua serial not null unique,
    lingua varchar(255) not null
);

create table f_anomalia (
    id_utilizador int not null,
    id_tempo int not null,
    id_local int not null,
    id_lingua int not null,
    tipo_anomalia varchar(255) not null,
    com_proposta boolean not null
);
create index idx_f_anomalia on f_anomalia(id_utilizador, id_tempo, id_local, id_lingua);
```

Data Analytics

Decidimos implementar um cube através de união de clausulas group by, dado que a versão de PostgreSQL do sistema de base de dados do IST é anterior à introdução da função cube do PostgreSQL.

```
select tipo_anomalia, dia_da_semana, lingua, total_anomalias
from (
    (select tipo_anomalia, dia_da_semana, lingua, count(*) as total_anomalias
    from f_anomalia natural join
        d_tempo natural join d_lingua
    group by tipo_anomalia, dia_da_semana, lingua
    order by tipo_anomalia, dia_da_semana, lingua)
    union all

    (select tipo_anomalia, dia_da_semana, null, count(*) as total_anomalias
    from f_anomalia natural join
        d_tempo
    group by tipo_anomalia, dia_da_semana
    order by tipo_anomalia, dia_da_semana)
    union all

    (select null, dia_da_semana, lingua, count(*) as total_anomalias
    from f_anomalia natural join
        d_tempo natural join d_lingua
    group by lingua, dia_da_semana
    order by dia_da_semana, lingua)
    union all

    (select tipo_anomalia, null, lingua, count(*) as total_anomalias
    from f_anomalia natural join
        d_lingua
    group by tipo_anomalia, lingua
    order by tipo_anomalia, lingua)
    union all

    (select tipo_anomalia, null, null, count(*) as total_anomalias
    from f_anomalia
    group by tipo_anomalia
    order by tipo_anomalia)
    union all

    (select null, dia_da_semana, null, count(*) as total_anomalias
    from f_anomalia natural join
        d_tempo
    group by dia_da_semana
    order by dia_da_semana)
    union all

    (select null, null, lingua, count(*) as total_anomalias
    from f_anomalia natural join
        d_lingua
    group by lingua
    order by lingua)
    union all

    (select null, null, null, count(*) as total_anomalias
    from f_anomalia)
) as R;
```