**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. Quando a interrogação retorna uma elevada percentagem total de registos da tabela (neste caso 10%), torna-se desnecessária a implementação de um índice, dado que este levaria tanto tempo a aceder à tabela como no cenário em que não tivesse índice.

1.2. Índice B-Tree, uma vez que fazemos uma verificação entre um range de parâmetros de data\_hora.   
create index data\_hora\_idx on proposta\_de\_correcao using btree(data\_hora);

2. Índice único e primário (dado que a chave é primária, esta já assegura que o seu índice é único e como tal tem de ser usada com a estrutura B-Tree, uma vez que a estrutura hash não aceita UNIQUE constraints).  
create index anomalia\_id\_idx on incidencia(anomalia\_id);

3.1. Mesma situação que ocorre na alínea 1.1. É desnecessário criar um índice.  
3.2. Índice B-Tree, uma vez que fazemos uma verificação de um intervalo usando uma chave primária.  
create index anomalia\_id\_idx2 on correcao using btree(anomalia\_id);

4. Chave composta. 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);

Solução alternativa: Partial Indexasdasd  
create index anomalia\_id\_idx3 on anomalia(ts, tem\_anomalia\_redacao, lingua) where tem\_anomalia\_redacao is TRUE.

**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;