**Projeto de Bases de Dados**

**Parte 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Número de Aluno** | **Percentagem de Contribuição** | **Esforço em horas** |
| Joana Maria Leal Coutinho | 87666 | 33.3% |  |
| João Rafael Pinto Soares | 87675 | 33.3% |  |
| Pedro M. S. P. Rodrigues | 87696 | 33.3% |  |

Grupo Nº 3

Turno BD817957L08 (6ª feira, 8:30 – 10:00)

Professor André Vasconcelos

**Implementação das Restrições de Integridade:**

a)

**create or replace function check\_solicita()** returns trigger as $$

begin

if not exists(

select \*

from Audita natural join EventoEmergencia natural join Vigia

where idCoordenador = new.idCoordenador and numCamara = new.numCamara

) then

raise exception 'Um coordenador apenas pode solicitar videos de camaras colocadas num local cujo accionamento de meios esteja a ser (ou tenha side) auditado por ele proprio.';

end if;

return new;

end

$$ language plpgsql;

**create trigger check\_solicita\_constraint before insert or update on Solicita**

for each row execute procedure check\_solicita();

b)

**create or replace function check\_acciona()** returns trigger as $$

begin

if not exists(

select \*

from Acciona

where numMeio = new.numMeio and nomeEntidade = new.nomeEntidade and numProcessoSocorro = new.numProcessoSocorro

) then

raise exception 'Um Meio de Apoio só pode ser alocado a Processos de Socorro para os quais tenha sido accionado.';

end if;

return new;

end

$$ language plpgsql;

**create trigger check\_alocado\_acciona\_constraint before insert or update on Alocado**

for each row execute procedure check\_acciona();

**Índices:**

a)

No primeiro query, são utilizadas as tabelas *Video* e *Vigia*. É feito o produto cartesiano entre estas, selecionando entradas cujo *numCamara* para cada uma das tabelas é igual. São escolhidas, em particular, linhas referentes ao *numCamara* 10 e à *moradaLocal* “Loures”. Assim, nesta query apenas são usadas comparações de igualdade, nomeadamente, entre as colunas *numCamara* e *moradaLocal*. Pelo contexto do problema, é de esperar que uma câmara vigíe um número muito reduzido de locais.

Tendo tudo isto em conta, o melhor tipo de índice para a tabela *Video* é primário usando Hash sobre o atributo *numCamara*. Para a tabela *Vigia*, o melhor tipo de índice é também primário usando Hash sobre o atributo *numCamara*. Para este tipo de query, não é necessário colocar um índice secundário para qualquer uma das tabelas.

Este tipo de índices permitiria obter complexidade de na seleção de *numCamara* para cada tabela, tendo depois complexidade de (sendo o número de linhas da tabela *Vigia*) no pior caso. No entanto, pelas razões explicadas acima, esta complexidade será na maioria dos casos .

No segundo query, são utilizadas as tabelas  *Transporta*  e  *EventoEmergencia*. É feito o produto cartesiano entre estas, selecionando, agrupadas conjuntamente pelo *numTelefone* e  *instanteChamada*, entradas cujo *numProcessoSocorro* para cada uma das tabelas é igual. Assim, tal como na primeira query, apenas são usadas comparações de igualdade.

Deste modo e assumindo que a tabela *EventoEmergencia* está ordenada pelos atributos *numTelefone* e  *instanteChamada*, o melhor tipo de índice para a tabela *Transporta* é primário usando Hash sobre o atributo *numProcessoSocorro*. Para a tabela *EventoEmergencia*, o melhor tipo de índice é também primário usando Hash sobre o atributo *numProcessoSocorro*.

b)

create index video\_idx on video using hash(numCamara);

create index vigia\_idx on vigia using hash(numCamara);

**Modelo Multidimensional:**

drop table d\_evento cascade;

drop table d\_meio cascade;

drop table d\_tempo cascade;

drop table facts cascade;

**create table d\_evento**

(idEvento **serial** **not null**,

numTelefone **numeric(9) not null** check(numTelefone > 0),

instanteChamada **timestamp** **not** **null**,

**unique(numTelefone, instanteChamada)**,

constraint pk\_d\_evento primary key(idEvento));

**create table d\_meio**

(idMeio **serial** **not** **null**,

numMeio **numeric** **not** **null** check(numMeio >= 0),

nomeMeio **varchar(80) not null**,

nomeEntidade **varchar(80) not null**,

tipo **varchar(80) not null**,

constraint pk\_d\_meio primary key(idMeio));

**create table d\_tempo**

(idTempo **serial** **not** **null**,

dia **numeric(2) not null**,

mes **numeric(2) not null**,

ano **numeric(4) not null**,

constraint pk\_d\_tempo primary key(idTempo));

**create table facts**

(idEvento **serial**,

idMeio **serial**,

idTemp**o serial,**

constraint pk\_facts primary key(idEvento, idMeio, idTempo),

constraint fk\_facts\_d\_evento foreign key(idEvento) references d\_evento(idEvento),

constraint fk\_facts\_d\_meio foreign key(idMeio) references d\_meio(idMeio),

constraint fk\_facts\_d\_tempo foreign key(idTempo) references d\_tempo(idTempo));

**create or replace function get\_dates() returns void as $$**

declare

min\_date date;

max\_date date;

begin

Select min(instanteChamada)::timestamp::date into min\_date

From EventoEmergencia;

Select max(instanteChamada)::timestamp::date into max\_date

From EventoEmergencia;

while min\_date <= max\_date LOOP

insert into d\_tempo(dia, mes, ano) values (EXTRACT(day from min\_date), EXTRACT(month from min\_date), EXTRACT(year from min\_date));

min\_date := min\_date + interval '1 day';

end loop;

end;

$$ language plpgsql;

insert into d\_evento(numTelefone, instanteChamada)

select numTelefone, instanteChamada

from EventoEmergencia;

insert into d\_meio(numMeio, nomeEntidade, nomeMeio, tipo)

select \*, 'MeioCombate'

from Meio natural join MeioCombate;

insert into d\_meio(numMeio, nomeEntidade, nomeMeio, tipo)

select \*, 'MeioApoio'

from Meio natural join MeioApoio;

insert into d\_meio(numMeio, nomeEntidade, nomeMeio, tipo)

select \*, 'MeioSocorro'

from Meio natural join MeioSocorro;

select get\_dates();

insert into facts

select

idEvento, idMeio, idTempo

from EventoEmergencia natural join Acciona natural join d\_evento natural join d\_meio natural join d\_tempo

Where EXTRACT(day from instanteChamada) = dia and

EXTRACT(month from instanteChamada) = mes and

EXTRACT(year from instanteChamada) = ano;

**Data Analytics:**

Select **ano, mes**, tipo, count(tipo)

From facts NATURAL JOIN d\_meio NATURAL JOIN d\_tempo

Where idEvento = 15

Group by tipo, ano, mes

**UNION**

Select **ano, null**, tipo, count(tipo)

From facts NATURAL JOIN d\_meio NATURAL JOIN d\_tempo

Where idEvento = 15

Group by tipo, ano, mes

**UNION**

Select **null, null**, tipo, count(tipo)

From facts NATURAL JOIN d\_meio NATURAL JOIN d\_tempo

Where idEvento = 15

Group by tipo, ano, mes;