Projeto de Base de Dados

Parte 4

|  |  |  |  |
| --- | --- | --- | --- |
| Prof. André Vasconcelos | | | |
| **Grupo 33** | **Turno L03** | **Segunda às 08:00** | |
| Aluno | | Contribuição (%) | Esforço(horas) |
| 86426 | Gabriel Figueira | 33 |  |
| 86461 | Lívio Costa | 33 |  |
| 86499 | Pedro Carvalho | 33 |  |

**Restrições de Integridade**

**a)**

CREATE OR REPLACE FUNCTION chk\_coordenador\_solicita()

RETURNS TRIGGER AS $BODY$

BEGIN

IF NOT EXISTS (select moradaLocal from (select NEW.\*) as t natural join vigia where moradaLocal IN (select moradaLocal from audita natural join eventoemergencia where idCoordenador = NEW.idCoordenador)) THEN

RAISE EXCEPTION 'Video nao solicitado'

USING HINT = 'So pode solicitar videos de camaras colocadas num local cujo accionamento de meios esteja a ser (ou tenha sido) auditado';

END IF;

RETURN NEW;

END;

$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER chk\_coordenador\_solicita BEFORE INSERT OR UPDATE ON Solicita FOR EACH ROW EXECUTE PROCEDURE chk\_coordenador\_solicita();

**b)**

CREATE OR REPLACE FUNCTION chk\_aloc\_meioapoio()

RETURNS TRIGGER AS $BODY$

BEGIN

IF NOT EXISTS (SELECT numMeio,nomeEntidade,numProcessoSocorro FROM Acciona NATURAL JOIN MeioApoio where nummeio = NEW.numMeio and nomeentidade = NEW.nomeEntidade and numprocessosocorro = NEW.numProcessoSocorro) THEN

RAISE EXCEPTION 'Meio nao accionado'

USING HINT = 'O meio de apoio só pode ser alocado se tiver sido accionado';

END IF;

RETURN NEW;

END;

$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER chk\_alocar\_meioapoio BEFORE INSERT OR UPDATE ON Alocado FOR EACH ROW EXECUTE PROCEDURE chk\_aloc\_meioapoio();

**Índices**

Como a query implica aceder a campos com atributos específicos, numCamara=10 e moradaLocal = ‘Loures’, escolhemos criar um índice de hash tanto na tabela vigia como na tabela vídeo, visto que se pretendem fazer testes de igualdade.

**1.**

De acordo query apresentada, estaremos a fazer dois testes de igualdade e assumindo que uma câmara só existe num local então para acelerar esta interrogação decidimos escolher um índice baseado em funções de dispersão(hash).

Como a query implica um acesso a campos de acordo com valores específicos, numCamara=10 e moradaLocal = ‘Loures’, escolhemos criar um ***índice de hash*** tanto na tabela ***Vigia*** como na tabela ***Video***, visto que se pretendem fazer testes de igualdade.

Em ambas as tabelas o índice é criado sobre o atributo <numCamara>.

As instruções para criação dos índices são as seguintes:

* create index hs\_video\_camara on video using hash (numCamara);
* create index hs\_vigia\_camara on vigia using hash (numCamara);

Nesta query temos testes de igualdade quando o numProcessoSocorro é comparado entre as duas tabelas e uma pesquisa em range pelos campos numTelefone,instanteChamada pelo que escolhemos uma hash para ambas as tabelas sobre numProcessoSocorro e uma btree composta por numTelefone,instanteChamada.

create index hs\_transporta on transporta using hash(numProcessoSocorro);

create index hs\_eventoemergencia on eventoemergencia using hash(numProcessoSocorro);

create index bt\_eventoemergencia on eventoemergencia(numTelefone,instanteChamada);

**2.**

Nesta query temos testes de igualdade quando o numProcessoSocorro é comparado entre as duas tabelas e uma pesquisa em range pelos campos numTelefone, instanteChamada que se encontram no “group by” pelo que decidimos escolher um índice ordenado(btree).

Na tabela ***Transporta*** criámos um ***índice btree*** sobre o atributo <numProcessoSocorro>.

Na tabela ***EventoEmergencia*** criámos um ***índice btree*** sobre os atributos <numProcessoSocorro, numTelefone, instanteChamada>. Este índice é composto visto que os atributos “numTelefone” e “instanteChamada” estão no “group by” porque de alguma forma queremos acelerar a execução tendo estes atributos em conta.

As instruções para criação dos índices são as seguintes:

* create index bt\_transporta on transporta using btree (numProcessoSocorro);
* create index bt\_eventoemergencia on eventoemergencia using btree (numProcessoSocorro, numTelefone, instanteChamada);

**Modelo Multidimensional**

drop table if exists facto;

drop table if exists d\_tempo;

drop table if exists d\_meio;

drop table if exists d\_evento;

create table d\_evento(

idEvento serial not null,

numTelefone varchar(15) not null,

instanteChamada timestamp not null,

constraint pk\_d\_evento primary key(idEvento)

);

create table d\_meio(

idMeio serial not null,

numMeio integer not null,

nomeMeio varchar(200) not null,

nomeEntidade varchar(200) not null,

tipo varchar(20) not null,

constraint pk\_d\_meio primary key(idMeio)

);

create table d\_tempo(

idTempo serial not null,

dia integer not null,

mes integer not null,

ano integer not null,

constraint pk\_d\_tempo primary key(idTempo)

);

create table facto(

idEvento integer not null,

idMeio integer not null,

idTempo integer not null,

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

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

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

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

);

**Data Analytics**

select mes, ano, idevento, tipo, count(1) as numeromeios

from facto

natural join d\_meio

natural join d\_tempo

where idevento = 15

group by rollup(ano,mes),tipo,idevento

order by mes,ano;