**Projeto De Bases De Dados, Parte 4**

Grupo no. 55

Turno: Quarta 11:00 – 12:30

Professor: Taras Lykhenko

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Número** | **Percentagem da**  **Contribuição** | **Esforço (horas)** |
| Daniela Lopes | 86403 | 33,(3)% | 5 |
| Afonso Vieira | 86458 | 33,(3)% | 5 |
| Madalena Galrinho | 87546 | 33,(3)% | 5 |

**Índices**

Para testar a influência do uso dos índices populou-se a base de dados usando um script que

gerou, consoante a alínea:

-> 1 000 coordenadores, 1 000 câmaras, 100 moradaLocal, 1 000 vigia e 6 000 vídeos

(100 dos quais filmados pela câmara 10 em Loures)

-> 6 000 eventos de emergência, 6 000 processoSocorro, 6 000 entidades, 6 000 meios,

6 000 meios de socorro e 100 transporta.

Para a análise dos índices necessários, analisou-se o comportamento e desempenho das queries sem o uso de índices (excepto o índice por defeito da chave primária) através do uso do comando EXPLAIN do PostgreSQL.

**a)**

**1 - Liste todos os vídeos filmados pela câmara 10 em Loures**

QUERY PLAN

-------------------------------------------------------------------------------------

Nested Loop (cost=0.27..127.07 rows=32 width=16)

-> Index Only Scan using vigia\_pkey on vigia i (cost=0.27..8.29 rows=1 width=4)

Index Cond: ((moradalocal = 'Loures'::text) AND (numcamara = 10))

-> Seq Scan on video v (cost=0.00..118.46 rows=32 width=20)

Filter: (numcamara = 10)

-------------------------------------------------------------------------------------

E PRECISO DIZER SE SAO PRIMARIOS OU SECUNDARIOS???????????????

Tabela: Video, Atributos: numCamara, Hash, Desagrupado, Denso

Tabela: Vigia, Atributos: numCamara , moradaLocal, Hash, Agrupado, Denso

Ambos são Hash porque estamos a tratar de comparações diretas, não havendo comparações

com intervalos ou booleanos.

Ambos são densos pois, apesar de ocupar mais espaço, é mais rápido pois não tem de fazer procura na Tabela e no nosso caso não faria sentido usar índices esparsos pois não queremos aceder a conteúdos da tabela ordenados e dentro de um certo intervalo (com o índice a apontar para o início do intervalo).

Na tabela video escolhemos apenas o atributo numCamara pois é o único que é necessário para as comparações feitas na query, sendo os outros atributos desnecessários e iria ser muito ineficiente. É desagrupado pois não está de acordo com a primary key da Tabela.

Na tabela vigia escolhemos os atributos numCamara e moradaLocal, que correspondem à primary key da tabela, pois ambos são necessários para as comparações e­­ assim esta tabela não tem de ser acedida cada vez que a query é feita. É Agrupado pois está de acordo com a primary key da Tabela, pelo que segue a ordem da mesma.

**2 - Liste o número de vítimas transportado por meios de socorro em cada evento de emergência.**

QUERY PLAN

----------------------------------------------------------------------------------

HashAggregate (cost=89.72..91.72 rows=200 width=60) Group Key: e.numtelefone, e.instantechamada -> Hash Join (cost=13.82..85.96 rows=502 width=60) Hash Cond: (e.numprocessosocorro = t.numprocessosocorro) -> Seq Scan on eventoemergencia e (cost=0.00..64.90 rows=590 width=60) -> Hash (cost=11.70..11.70 rows=170 width=8) -> Seq Scan on transporta t (cost=0.00..11.70 rows=170 width=8)

----------------------------------------------------------------------------------

Tabela: Transporta, Atributos: numProcessoSocorro, Hash, Desagrupado, Denso

Tabela: EventoEmergencia, Atributos: numProcessoSocorro, Hash, Desagrupado, Denso

Analogamente, nas duas tabelas escolhemos o atributo numProcessoSocorro pois é o único preciso nas comparações feitas na query. Ambos os índices são desagrupados e densos pelos motivos descritos acima.

**b)**

**1 - Liste todos os vídeos filmados pela câmara 10 em Loures**

CREATE INDEX video\_idx on video USING HASH(numCamara);

Este índice tem por objetivo acelarar a execução de V.numCamara = I.numCamara e de

V.numCamara = 10, ambas condições de igualdade, pelo que o uso de uma função de dispersão

hash permite que a procura seja feita em 0(1).

CREATE INDEX vigia\_idx on vigia USING HASH(numCamara, moradaLocal);

ESTE INDICE TORNARIA A PROCURA MAIS RAPIDA!!!!?????????

Não é necessário criar o outro índice associado à interrogação 1 pois o índice que

seria mais vantajoso para esta tabela seria o correspondente ao índice da primary key,

o qual já existe, apesar de o original ter btree em vez de hash

**2 - Liste o número de vítimas transportado por meios de socorro em cada evento de emergência.**

CREATE INDEX transporta\_idx on transporta USING HASH(numProcessoSocorro);

CREATE INDEX evento\_emergencia\_idx on eventoEmergencia USING HASH(numProcessoSocorro);

Ambos estes índices têm por objetivo acelarar a execução da comparação de igualdade

T.numProcessoSocorro = E.numProcessoSocorro uma condição de igualdade, pelo que o uso

de uma função de dispersão hash permite que a procura seja feita em 0(1).

**Modelo Multidimensional**

create table d\_evento(

idEvento serial not null,

numTelefone varchar(15) not null,

instanteChamada timestamp not null,

primary key(idEvento)

);

create table d\_meio(

idMeio serial not null,

numMeio integer not null,

nomeMeio varchar(80) not null,

nomeEntidade varchar(200) not null,

tipo varchar(80) not null,

primary key(idMeio)

);

create table d\_tempo(

dia integer not null,

mes integer not null,

ano integer not null,

primary key(dia, mes, ano)

);

create table facts(

idEvento serial not null,

idMeio serial not null,

dia integer not null,

mes integer not null,

ano integer not null,

foreign key(idEvento) references d\_evento(idEvento),

foreign key(idMeio) references d\_meio(idMeio),

foreign key(dia, mes, ano) references d\_tempo(dia, mes, ano)

);

**Data Analytics**

/\*com rollup\*/

select tipo, count(\*)

from facts natural join d\_meio

where idEvento = 15

group by tipo, (group by ano, mes

union

group by ano);

/\*sem rollup\*/

select tipo, count(\*)

from facts natural join d\_meio

where idEvento = 15

group by tipo, rollup(ano, mes);