**Projeto de Bases de Dados**

**Parte 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Grupo:** 50 | | | |
| **Número** | **Nome** | **Contribuição** | **Esforço (em Horas)** |
| 93695 | Catarina Sofia dos Santos Sousa | 33% | 8h |
| 93743 | Nelson Alexandre Geada Trindade | 33% | 8h |
| 93754 | Rodrigo Rodrigues Major | 34% | 8h |

**Sala:** 1-29

**Turno:** 4ºFeira – 15h00

**Docente:** Carlota De Oliveira Lopes Dias

**Restrições de Integridade**

*--RI-100: um médico não pode dar mais de 100 consultas por semana na mesma instituição*

drop trigger if exists verifica\_medico\_trigger on consulta;

create or replace function verifica\_medico() returns trigger as $$

declare consultas decimal(20,2);

begin

  select count(\*) into consultas

  from consulta c

  where c.num\_cedula = new.num\_cedula

  and c.nome\_instituicao = new.nome\_instituicao

  and EXTRACT(YEAR from c.data) = EXTRACT(YEAR from new.data)

  and EXTRACT(WEEK from c.data) = EXTRACT(WEEK from new.data);

  if consultas >= 100 then

    raise exception 'O médico % não pode dar mais de 100 consultas por semana na mesma instituição.', new.num\_cedula;

    end if;

  return new;

 END;

 $$ Language plpgsql;

create trigger verifica\_medico\_trigger before insert on consulta

for each row execute procedure verifica\_medico();

*--RI-análise: numa análise, a consulta associada pode estar omissa; não estando, a especialidade*

*--da consulta tem de ser igual à do médico.*

drop trigger if exists verifica\_especialidade\_trigger on analise;

create or replace function verifica\_especialidade() returns trigger as $$

declare especialidade varchar(25);

begin

  select m.especialidade into especialidade

  from consulta c natural join medico m

  where c.num\_cedula = new.num\_cedula

  and c.num\_doente = new.num\_doente

  and c.data = new.data;

  if especialidade is not null and especialidade != new.especialidade then

    raise exception 'O médico % não tem a especialidade necessária para analisar.', new.num\_cedula;

    end if;

  return new;

 END;

 $$ Language plpgsql;

create trigger verifica\_especialidade\_trigger before insert on analise

for each row execute procedure verifica\_especialidade();

**Índices**

**Query 1**

Não é necessário criar nenhum índice porque é criado implicitamente para as chaves primárias.

Assim, apenas é preciso alterar a ordem dos campos das chaves primárias na declaração da tabela consulta para que o num\_doente seja o primeiro atributo. Concluindo, ao alterar a ordem dos campos das chaves primárias, o índice criado implicitamente é o único necessário para acelerar a execução desta query.

**Query 2**

Como esta query implica uma igualdade no atributo especialidade da tabela medico, então é útil criar um índice que seja apropriado para igualdades. Para isto, criamos um índice do tipo Hash porque funciona por "contentores" e guarda vários registos por cada contentor. Assim, através da função de dispersão o acesso fica otimizado, acelerando a realização desta query.

CREATE INDEX index\_especialidade ON medico USING HASH(especialidade)

**Query 3**

Blocos do disco são de 2KBytes e cada registo ocupa 1kByte, ou seja, cada bloco leva 2 registos. Seletividade de (1/6) = 0.16666667 ou seja, a probabilidade de um bloco não ter respostas é de (1-0.16666667)^2 (por serem dois registos por bloco), que é aproximadamente 69%. Logo, teremos de ler 31% dos blocos. Quanto menor for a percentagem de blocos a ler, maior é o benefício dos índices a reduzir leituras do disco. Assim, como teremos de ler 31% dos blocos, é útil utilizarmos um índice. Uma vez que a query implica uma igualdade no atributo especialidade da tabela medico, então é útil utilizar um índice do tipo Hash para acelerar a realização desta query, porque cada "contentor" aramazena um conjunto de entradas e através da função de dispersão, o acesso fica otimizado.

CREATE INDEX index\_especialidade ON medico USING HASH(especialidade)

**Query 4**

São criados índices implicitamente para as primary keys e assim, pode ser utilizado neste caso para acelerar a execução desta query, uma vez que o atributo num\_cedula é o primeiro atributo da chave primária na declaração da tabela consulta. Se o atributo num\_cedula não fosse o primeiro da chave primária, então tinhamos de alterar a ordem dos campos das chaves primárias.

Para além deste índice, deve-se criar um índice do tipo BTree para o atributo "data" da tabela consulta para otimizar a comparação entre as duas datas dadas. Este tipo de índice é o apropriado para acelerar esta query porque as folhas do índice estão sempre ordenadas, o que facilita a comparação entre as datas:

CREATE INDEX idx\_data on consulta USING B-TREE(data)

**Modelo Multidimensional**

DROP TABLE d\_tempo CASCADE;

DROP TABLE d\_instituicao CASCADE;

DROP TABLE f\_analise CASCADE;

DROP TABLE f\_presc\_venda CASCADE;

CREATE TABLE d\_tempo (

    id\_tempo *serial* PRIMARY KEY,

    dia int,

    dia\_da\_semana int,

    semana int,

    mes int,

    trimestre int,

    ano int

);

CREATE TABLE d\_instituicao (

    id\_inst *serial* PRIMARY KEY,

    nome varchar(255),

    tipo varchar(11),

    num\_regiao int,

    num\_concelho int,

*FOREIGN KEY* (nome) *REFERENCES* instituicao(nome),

*FOREIGN KEY* (num\_regiao) *REFERENCES* regiao(num\_regiao),

*FOREIGN KEY* (num\_concelho) *REFERENCES* concelho(num\_concelho)

);

CREATE TABLE f\_presc\_venda (

    id\_presc\_venda int PRIMARY KEY,

    id\_medico int,

    num\_doente int,

    id\_data\_registo int,

    id\_inst int,

    substancia varchar(255),

    quant int,

*FOREIGN KEY* (id\_presc\_venda) *REFERENCES* prescricao\_venda(num\_venda),

*FOREIGN KEY* (id\_medico) *REFERENCES* medico(num\_cedula),

*FOREIGN KEY* (id\_data\_registo) *REFERENCES* d\_tempo(id\_tempo),

*FOREIGN KEY* (id\_inst) *REFERENCES* d\_instituicao(id\_inst)

);

CREATE TABLE f\_analise (

    id\_analise int PRIMARY KEY,

    id\_medico int,

    num\_doente int,

    id\_data\_registo int,

    id\_inst int,

    nome varchar(255),

    quant int,

*FOREIGN KEY* (id\_analise) *REFERENCES* analise(num\_analise),

*FOREIGN KEY* (id\_data\_registo) *REFERENCES* d\_tempo(id\_tempo),

*FOREIGN KEY* (id\_inst) *REFERENCES* d\_instituicao(id\_inst)

);

**ETL**

*-- d\_tempo*

With temp as (

  SELECT DISTINCT data from prescricao\_venda pv

  UNION

  SELECT DISTINCT data from analise a

)

INSERT INTO d\_tempo(dia, dia\_da\_semana, semana, mes, trimestre, ano)

SELECT extract(day from data),

    extract(dow from data),

    extract(week from data),

    extract(month from data),

    extract(quarter from data),

    extract(year from data)

FROM temp sub;

*-- d\_instituicao*

INSERT INTO d\_instituicao(nome, tipo, num\_regiao, num\_concelho)

SELECT inst.nome,

    inst.tipo,

    inst.num\_regiao,

    inst.num\_concelho

FROM instituicao inst;

*-- f\_presc\_venda*

With temp as (

  SELECT num\_cedula, num\_doente, id\_tempo, substancia, num\_venda

  FROM prescricao\_venda pv

  INNER JOIN d\_tempo dt

  ON (

    dt.ano = extract(year from pv.data) and

    dt.mes = extract(month from pv.data) and

    dt.dia = extract(day from pv.data)

    )

)

INSERT INTO f\_presc\_venda(id\_presc\_venda, id\_medico, num\_doente, id\_data\_registo, id\_inst, substancia, quant)

SELECT t.num\_venda, num\_cedula, num\_doente, id\_tempo, id\_inst, substancia, quant

FROM temp t

INNER JOIN (

  SELECT num\_venda, id\_inst, quant

  FROM venda\_farmacia vf

  INNER JOIN d\_instituicao di

  ON vf.inst = di.nome

  ) t2

ON t.num\_venda = t2.num\_venda;

*-- f\_analise*

with temp as(

  SELECT num\_analise, num\_cedula, num\_doente, id\_tempo, inst, nome, quant

  FROM analise a

  INNER JOIN d\_tempo dt

  ON (

    dt.ano = extract(year from a.data) and

    dt.mes = extract(month from a.data) and

    dt.dia = extract(day from a.data)

  )

)

INSERT INTO f\_analise(id\_analise, id\_medico, num\_doente, id\_data\_registo, id\_inst, nome, quant)

SELECT num\_analise, num\_cedula, num\_doente, id\_tempo, id\_inst, t.nome, quant

FROM temp t

INNER JOIN d\_instituicao di

ON di.nome = t.inst;

**Queries OLAP**

*--1*

SELECT especialidade, mes, ano, count(\*)

FROM f\_analise a

    INNER JOIN medico ON a.id\_medico = medico.num\_cedula

    INNER JOIN d\_tempo ON a.id\_data\_registo = d\_tempo.id\_tempo

WHERE d\_tempo.ano >= 2017

    AND d\_tempo.ano <= 2020

    AND a.nome = 'glicemia'

GROUP BY CUBE(especialidade, mes, ano)

ORDER BY especialidade, ano, mes ASC;~

*--2*

WITH TEMP AS(

    SELECT substancia, num\_concelho, dia\_da\_semana, mes, SUM(quant) AS sum\_quant, COUNT(\*) AS c

    FROM f\_presc\_venda v

        NATURAL JOIN d\_instituicao

        INNER JOIN d\_tempo on v.id\_data\_registo = d\_tempo.id\_tempo

    WHERE num\_regiao = 2

        AND trimestre = 1

        AND ano = 2020

    GROUP BY ROLLUP(substancia, num\_concelho, dia\_da\_semana, mes)

    ORDER BY substancia, mes, dia\_da\_semana asc

)

SELECT substancia, num\_concelho, dia\_da\_semana, mes, sum\_quant,

CASE

WHEN dia\_da\_semana IS NULL AND mes IS NULL THEN

    CAST(CAST(c AS FLOAT)/90 AS FLOAT(3))

WHEN dia\_da\_semana IS NOT NULL AND mes IS NULL THEN

    CAST(CAST(c AS FLOAT)/13 AS FLOAT(3))

WHEN dia\_da\_semana IS NOT null AND mes IS NOT null THEN

    CAST(CAST(c AS FLOAT)/4 AS FLOAT(3))

END AS media

FROM TEMP sub