****

**Projeto de Bases de Dados - Parte 4**

**Grupo 5**

Turno: L06 Seg. 14:00-15:30

Docente de Laboratório: Tiago Oliveira

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Número | % Contribuição | Esforço (horas) |
| Carolina Pereira | 92433 | 25 | 14 |
| Francisco Figueiredo | 89443 | 25 | 9 |
| Tomás Sequeira | 92565 | 25 | 10 |
| Vicente Lorenzo | 92569 | 25 | 10 |

**Restrições de Integridade**

create or replace function triggerCemConsultas() returns

trigger as $$

declare nConsultas integer;

begin

select count(\*) into nConsultas from consulta

where(num\_cedula=new.num\_cedula AND nome=new.nome AND DATE\_PART('week',data\_consulta)=DATE\_PART('week',new.data\_consulta));

if nConsultas > 100 then

raise exception 'Médico já tem 100 consultas para essa instituição na mesma semana';

end if;

return new;

end;

$$ language plpgsql;

create or replace function triggerConsultaOmissa() returns

trigger as $$

declare espec varchar(20);

begin

if (new.num\_cedula IS NOT NULL AND new.num\_doente IS NOT NULL AND new.data\_analise IS NOT NULL) then

select especialidade into espec from medico

where (num\_cedula = new.num\_cedula);

if espec != new.especialidade then

raise exception 'A especialidade do médico é diferente da especialidade da consulta';

end if;

end if;

return new;

end;

$$ language plpgsql;

create trigger triggerNConsultas before insert on consulta for each row execute procedure triggerCemConsultas();

create trigger triggerOmissa before insert on analise for each row execute procedure triggerConsultaOmissa();

**Índices:**

**--1--**

create index index\_consulta on consulta using hash(num\_doente)

Para o primeiro caso, é necessário criar um índice para organizar a coluna do num\_doente com uma HashTable para facilitar a comparação direta. Neste caso prefere-se a utilização de hash table visto ser uma simples igualdade.

**--2--**

create index index\_medico on medico using hash(especialidade)

Para o segundo caso, é necessário criar um índice para a especialidade como forma de otimizar a query e, neste caso, como é apenas uma comparação simples, prefere-se o uso de uma hash table.

**--3--**

create index index\_medico on medico using btree(especialidade)

Neste caso, como temos blocos de apenas 2KB é preferível usar uma btree para cada "especialidade", visto que é preferível ter ligações feitas por apontadores em vez de ter os blocos fisicamente juntos no disco.

**--4--**

create index\_medico\_1 on medico using hash(num\_cedula)

create index\_consulta\_1 on consulta using hash(num\_cedula)

create index\_consulta\_2 on consulta using btree(data\_consulta)

Criámos duas hash para tornar a comparação entre num\_cedula mais rápida e depois criamos um índice btree para tornar a comparação de datas mais rápida.

**Modelo Multidimensional e ETL de carregamento:**

DROP TABLE IF EXISTS d\_tempo CASCADE;

DROP TABLE IF EXISTS d\_instituicao CASCADE;

DROP TABLE IF EXISTS f\_presc\_venda CASCADE;

DROP TABLE IF EXISTS f\_analise CASCADE;

CREATE TABLE d\_tempo

(id\_tempo SERIAL NOT NULL,

dia INTEGER NOT NULL,

dia\_da\_semana INTEGER NOT NULL,

semana INTEGER NOT NULL,

mes INTEGER NOT NULL,

trimestre INTEGER NOT NULL,

ano INTEGER NOT NULL,

UNIQUE(dia,mes,ano),

PRIMARY KEY(id\_tempo));

CREATE TABLE d\_instituicao

(id\_inst SERIAL NOT NULL,

nome VARCHAR(40) NOT NULL,

tipo VARCHAR(40) NOT NULL,

num\_regiao INTEGER NOT NULL,

num\_concelho INTEGER NOT NULL,

FOREIGN KEY(nome) REFERENCES instituicao(nome) ON UPDATE CASCADE,

FOREIGN KEY(num\_regiao) REFERENCES regiao(num\_regiao) ON UPDATE CASCADE,

FOREIGN KEY(num\_concelho) REFERENCES concelho(num\_concelho) ON UPDATE CASCADE,

PRIMARY KEY(id\_inst));

CREATE TABLE f\_presc\_venda

(id\_presc\_venda INTEGER NOT NULL,

id\_medico INTEGER NOT NULL,

num\_doente INTEGER NOT NULL,

id\_data\_registo INTEGER NOT NULL,

id\_inst INTEGER NOT NULL,

substancia VARCHAR(20) NOT NULL,

quant INTEGER NOT NULL,

FOREIGN KEY(id\_presc\_venda) REFERENCES prescricao\_venda(num\_venda) ON UPDATE CASCADE,

FOREIGN KEY(id\_medico) REFERENCES medico(num\_cedula) ON UPDATE CASCADE,

FOREIGN KEY(id\_data\_registo) REFERENCES d\_tempo(id\_tempo) ON UPDATE CASCADE,

FOREIGN KEY(id\_inst) REFERENCES d\_instituicao(id\_inst) ON UPDATE CASCADE,

PRIMARY KEY(id\_presc\_venda));

CREATE TABLE f\_analise

(id\_analise INTEGER NOT NULL,

id\_medico INTEGER NOT NULL,

num\_doente INTEGER NOT NULL,

id\_data\_registo INTEGER NOT NULL,

id\_inst INTEGER NOT NULL,

nome VARCHAR(30) NOT NULL,

quant INTEGER NOT NULL,

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

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),

PRIMARY KEY(id\_analise));

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

SELECT distinct EXTRACT(DAY FROM t.data\_prescricao\_venda) AS dia,

EXTRACT(DOW FROM t.data\_prescricao\_venda) AS dia\_da\_semana,

EXTRACT(WEEK FROM t.data\_prescricao\_venda) AS semana,

EXTRACT(MONTH FROM t.data\_prescricao\_venda) AS mes,

EXTRACT(QUARTER FROM t.data\_prescricao\_venda) AS trimestre,

EXTRACT(YEAR FROM t.data\_prescricao\_venda) AS ano

FROM ((select data\_prescricao\_venda from prescricao\_venda)

UNION

(select data\_registo from analise)) as t

ORDER BY dia,dia\_da\_semana,semana,mes,trimestre,ano;

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

SELECT nome,tipo,num\_regiao,num\_concelho FROM instituicao;

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

SELECT P.num\_venda as id\_presc\_venda, P.num\_cedula as id\_medico, num\_doente, id\_tempo as id\_data\_registo, id\_inst, P.substancia, quant

FROM prescricao\_venda P NATURAL JOIN medico M

INNER JOIN venda\_farmacia V ON(P.num\_venda = V.num\_venda)

INNER JOIN d\_tempo T ON (DATE\_PART('day',P.data\_prescricao\_venda) = T.dia AND DATE\_PART('month',P.data\_prescricao\_venda) = T.mes AND DATE\_PART('year',P.data\_prescricao\_venda) = T.ano)

INNER JOIN d\_instituicao I ON (V.inst = I.nome)

ORDER BY P.num\_venda, P.num\_cedula, num\_doente, id\_tempo, id\_inst, P.substancia, quant;

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

SELECT num\_analise as id\_analise, A.num\_cedula as id\_medico, A.num\_doente, id\_tempo as id\_data\_registo, id\_inst, A.nome, quant

FROM analise A INNER JOIN medico M ON(A.num\_cedula = M.num\_cedula)

INNER JOIN d\_tempo T ON (DATE\_PART('day',A.data\_registo) = T.dia AND DATE\_PART('month',A.data\_registo) = T.mes AND DATE\_PART('year',A.data\_registo) = T.ano)

INNER JOIN d\_instituicao I ON (A.inst = I.nome)

ORDER BY num\_analise, A.num\_cedula, A.num\_doente, id\_tempo, id\_inst, I.nome, quant;

**Queries OLAP:**

**1)** SELECT a.especialidade, t.mes, t.ano, COUNT(\*) AS analises\_glicemia

FROM f\_analise AS f\_a

INNER JOIN analise AS a ON (f\_a.id\_analise = a.num\_analise)

INNER JOIN d\_tempo AS t ON (f\_a.id\_data\_registo = t.id\_tempo)

WHERE t.ano >= '2017' AND t.ano <= '2020'

GROUP BY (t.ano, t.mes), ROLLUP (a.especialidade);

**2)** SELECT c.nome AS concelho, p.substancia, t.mes, t.dia\_da\_semana, COUNT(\*) as quantidade\_total, COUNT(\*)/91::float as nr\_medio\_prescricoes\_diario

FROM f\_presc\_venda AS p

INNER JOIN d\_tempo AS t ON p.id\_data\_registo=t.id\_tempo

INNER JOIN d\_instituicao AS i ON p.id\_inst=i.id\_inst

INNER JOIN concelho AS c ON i.num\_concelho=c.num\_concelho

INNER JOIN regiao AS r ON i.num\_regiao=r.num\_regiao

WHERE t.trimestre='1' AND t.ano='2020' AND r.nome='Lisboa'

GROUP BY GROUPING SETS((p.substancia, c.nome, t.mes, t.dia\_da\_semana), (c.nome), (t.mes, t.dia\_da\_semana));

**PS** – Tivemos de fazer algumas alterações ao schema.sql da entrega 3 pelo que também está incluído no zip.