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.