Projeto de Base de Dados - Parte 4

Maria Beatriz Venceslau - 93734 - 18h (4ª) - 6h (continuação 3ª)
Helena Teixeira - 93720 - 18h (4ª) - 6h (continuação 3ª)
Maria Joana Lobo - 93736 - 18h (4ª) - 6h (continuação 3ª)

***Todo o esforço foi em simultâneo por zoom.**

G98 Turno: Terça-feira 8:30
Docente: Carlota Dias

Triggers das RIs

RI-100

```
create or replace function verificar_medico_consultas() returns trigger AS $$
declare num_consultas int;

BEGIN
   select count(*) into num_consultas
   from consulta
   where consulta.nome_instituicao = new.nome_instituicao
   and consulta.num_cedula = new.num_cedula
   and extract(week from cast(new.data as date)) = ( select extract(week from
cast(consulta.data as date)))
   and extract(year from cast(new.data as date)) = ( select extract(year from cast(consulta.data
as date)));

   if(num_consultas = 100) then
     raise exception 'RI-100: o médico % não pode dar mais de 100 consultas por semana na
mesma instituição', new.num_cedula;
   else
   end if;
   return new;

end
$$ Language plpgsql;

create trigger verificar_medico_consultas_trigger before insert on consulta
for each row execute procedure verificar_medico_consultas();
```

RI-análise

```
drop trigger if exists verificar_especialidade_trigger on analise;

create or replace function verificar_especialidade() returns trigger AS $$
declare esp varchar (255);


BEGIN
   select especialidade into esp from medico where num_cedula= new.num_cedula;
   if(new.num_cedula is NULL or new.num_doente is NULL or new.data is NULL ) then
      new.especialidade = NULL;
   else
      new.especialidade= esp;
   end if;
   return new;
```

```
END
$$ Language plpgsql;

create trigger verificar_especialidade_trigger before insert on analise
for each row execute procedure verificar_especialidade();
```

Índices

1) Se a condição num_doente = <um_valor> devolver muitas entradas não é muito benéfico criar um índice para esta coluna (num_doente). Caso contrário, visto que se trata de uma seleção por igualdade, será útil criar um índice do tipo hash (dispersão):

   ```
   CREATE INDEX num_doente_idx ON consulta USING HASH(num_doente);
   ```

2) Sabendo que há apenas seis especialidades: “E1” a “E6”, é benéfico criar um índice para esta coluna (especialidade), este índice terá uma estrutura B-tree

   ```
   CREATE INDEX especialidade_idx ON medico USING BTREE(especialidade);
   ```

3) Com os dados fornecidos sobre a dimensão das tabelas conseguimos aferir que:
   Tendo blocos de 2Kb e registos de 1Kb, teremos 2 registos por bloco.
   Sendo que temos 6 especialidades, e os médicos estão distribuídos uniformemente pelas mesmas, temos uma seletividade de 100/6 = 16.(6)%
   Assim, cada bloco terá em média (2*0.16) 0.32 respostas.
   Concluindo, teremos de ler 32% da tabela, pelo que será benéfico a utilização de índices, visto diminuir para ⅓ o número de leituras desta.

   Visto que se trata de uma seleção por igualdade, será útil criar um índice do tipo hash (dispersão):

   ```
   CREATE INDEX especialidade_idx ON medico USING HASH(especialidade);
   ```

4) A query apresentada verifica uma igualdade de valores entre consulta.num_cedula e medico.num_cedula e verifica se o valor consulta.date se encontra entre os valores ‘data_1’ e ‘data_2’. Assim, compensa utilizar um índice do tipo btree, uma vez que o índice do tipo hash, apesar de mais rápido, não suporta pesquisas em intervalos.

   Criámos o índice por ordem de seletividade de cada condição, neste caso, a data tem maior seletividade.

   ```
   CREATE INDEX num_cedula_data_idx ON consulta USING BTREE(data, num_cedula);
   ```

Criação do Modelo Multidimensional

```
drop table d_tempo cascade;
drop table d_instituicao cascade;
drop table f_prescricao_venda cascade;
drop table f_analise cascade;

create table d_tempo (
   id_tempo serial not null,
---auto-incrementa
   dia smallint not null,
   dia_da_semana smallint not null,
   semana smallint not null,
   mes smallint not null,
   trimestre smallint not null,
   ano integer not null,
   constraint pk_d_tempo primary
key(id_tempo));

create table d_instituicao(
   id_inst serial not null,
   nome varchar(255) not null,
   tipo varchar(255) not null,
   num_regiao integer not null,
   num_concelho integer not null,
   constraint pk_d_inst primary key(id_inst),
   constraint fk_nome_inst foreign
key(nome) references instituicao(nome),
   constraint fk_regiao_inst foreign
key(num_regiao) references
regiao(num_regiao),
   constraint fk_concelho_inst foreign
key(num_concelho,num_regiao) references
concelho(num_concelho,num_regiao));

create table f_prescricao_venda(
   id_pres_venda integer not null,
   id_medico integer not null,
   id_data_registo integer not null,
   id_inst integer not null,
   constraint pk_f_presc_venda primary
key(id_pres_venda),
   constraint fk_f_presc_venda foreign
key(id_pres_venda) references
prescricao_venda(num_venda),
   constraint fk_f_presc_id_medico foreign
key(id_medico) references
medico(num_cedula),
   constraint fk_f_presc_id_data_registo
foreign key(id_data_registo) references
d_tempo(id_tempo),
   constraint fk_f_presc_id_inst foreign
key(id_inst) references
d_instituicao(id_inst));

create table f_analise(
   id_analise serial not null,
   id_medico integer,
   num_doente integer not null,
   id_data_registo integer not null,
   id_inst integer not null,
   nome varchar(255) not null,
   quant integer not null,
   constraint pk_f_analise primary
key(id_analise),
   constraint fk_f_analise_id_analise foreign
key(id_analise) references
analise(num_analise),
   constraint fk_f_analise_id_medico foreign
key(id_medico) references
medico(num_cedula),
   constraint fk_f_analise_id_data_registo
foreign key(id_data_registo) references
d_tempo(id_tempo),
   constraint fk_f_analise_id_inst foreign
key(id_inst) references
d_instituicao(id_inst));
```

ETL de carregamento

```
--populate d_tempo 1 em 1 dia de 2017 a
2020
drop function populate_tempo();

create or replace function populate_tempo()
returns void as $$
declare initial_date timestamp;
begin
initial_date = '2017-01-01 00:00:00.00';
while initial_date < '2020-12-31 00:00:00.00'
   loop
   insert into d_tempo(dia,dia_da_semana,
semana, mes, trimestre, ano)
       values(extract(day from initial_date),
extract(dow from initial_date), --somar 1 cuz
domingo???
       extract(week from initial_date),
extract(month from initial_date),
       extract(quarter from
initial_date),extract(year from initial_date));
       initial_date = initial_date + interval '1
day';
   end loop;
return;
end
$$ language plpgsql;

select populate_tempo();

-- populate d_instituicao from instituicao da
E3
insert into
d_instituicao(nome,tipo,num_regiao,num_c
oncelho)
   select distinct nome, tipo, num_regiao,
num_concelho from instituicao;

--populate f_pres_venda
insert into
f_prescricao_venda(id_pres_venda,
id_medico, id_data_registo, id_inst)
          select distinct
new_presc.num_venda as id_pres_venda,
new_presc.num_cedula as id_medico ,
id_tempo as id_data_registo, id_inst
          from(select distinct num_venda
,venda_farmacia.inst as nome,
num_cedula,extract(day from
prescricao_venda.data) as dia,
           extract(month from
prescricao_venda.data) as mes,extract(year
from prescricao_venda.data) as ano
          from prescricao_venda natural join
venda_farmacia
          where
prescricao_venda.num_venda =
venda_farmacia.num_venda) as new_presc
          natural join d_tempo
          natural join d_instituicao;

--populate f_analise

insert into f_analise (id_analise, id_medico,
num_doente, id_data_registo,id_inst, nome,
quant)
   select distinct new_analise.num_analise
as id_analise, new_analise.num_cedula as
id_medico, new_analise.num_doente as
num_doente, id_tempo as id_data_registo,
          id_inst, new_analise.nome_analise
as nome, new_analise.quant as quant
          from (select distinct num_analise,
num_cedula, num_doente, nome as
nome_analise,inst as nome, quant,
          extract(day from
analise.data_registo) as dia,extract(month
from analise.data_registo) as mes,
          extract(year from
analise.data_registo) as ano from analise)
as new_analise
          natural join d_tempo
          natural join d_instituicao;
```

Queries OLAP

```
---query 1
select ano, mes, especialidade,count(*) as num_glicemia
from (select especialidade, num_cedula as id_medico
from medico) as new_medico
natural join (select id_data_registo as id_tempo, nome, id_medico from f_analise) as
new_f_analise
natural join d_tempo
where new_f_analise.nome = 'glicemia' and d_tempo.ano between '2017' and '2020'
group by
        cube((especialidade),(mes),(ano));


--query 2
with substancias as (select substancia, quant,d_instituicao.num_concelho as num_concelho,
d_tempo.mes as mes, d_tempo.dia_da_semana as dia_da_semana, d_tempo.id_tempo as
id_tempo
        from prescricao_venda natural join f_prescricao_venda
        natural join venda_farmacia
        natural join d_instituicao
        natural join d_tempo
        where f_prescricao_venda.id_pres_venda = prescricao_venda.num_venda and
                prescricao_venda.num_venda = venda_farmacia.num_venda and
d_instituicao.nome = venda_farmacia.inst
                and d_tempo.id_tempo = f_prescricao_venda.id_data_registo and
d_tempo.trimestre = '1' and d_tempo.ano = '2020'
                and d_instituicao.num_regiao = '2'),
                prescricoes_por_dia as (select id_tempo,substancia, count(*)as num from
substancias group by substancia,id_tempo)

select num_concelho, mes, dia_da_semana, sum(quant) as quant_substancia,null as
substancia, null as media
        from substancias
        group by rollup(num_concelho, mes, dia_da_semana)
union
select null,null, null, null, substancia, avg(num) as media
        from prescricoes_por_dia
        group by substancia;
```

Nota: Schema.sql da entrega 3 teve de ser alterado (enviado junto dos restantes ficheiros). Foi necessário popular a entrega 3 com valores de glicémia.