Projeto de Bases de Dados - Parte 4

Grupo 17 - L05 - Tiago Oliveira:

- André Avelar - 92422 - 33.33% (11h)
- Francisco Giro - 92504 - 33.33% (11h)
- Guilherme Fontes - 92470 -33.33% (11h)

RI

```
/*Regra de Integridade nº1 - ri_100*/

create or replace function ri100() returns
 trigger as $$
 declare i integer;
 begin
 select count(*) into i
            from Appointment
            where certificate_num=new.certificate_num and
            institution_name=new.institution_name and
            extract(week from appo_date)=extract(week from new.appo_date);
            if i > 100 then
            raise exception 'Doctor #% exceded number of appointments this week.',new.certificate_num
            using hint = 'Nao pode dar mais consultas nesta instituicao ate ao final da semana';
            end if;
            return new;
 end;
$$ language plpgsql;

drop trigger if exists ri100_update on Appointment;
drop trigger if exists ri100_insert on Appointment;

create trigger ri100_update before update on Appointment for each row execute procedure ri100();
create trigger ri100_insert before insert on Appointment for each row execute procedure ri100();


/*Regra de Integridade nº2 - ri_análise*/


create or replace function rianalise() returns
 trigger as $$
 declare s varchar(80);
 begin
 select doc_speciality into s
 from Doctor
 where certificate_num=new.certificate_num;

 if not(s=new.analysis_speciality) then
            raise exception 'Specialitys dont match.'
```

```
            using hint = 'It must be another doctor with the right speciality';
 end if;
 return new;
 end;
$$ language plpgsql;

drop trigger if exists rianalise_update on Analysis;
drop trigger if exists rianalise_insert on Analysis;

create trigger rianalise_update before update on Analysis for each row execute procedure rianalise();
create trigger rianalise_insert before insert on Analysis for each row execute procedure rianalise();
```

Ìndices:

1.

O tipo de índice é de Dispersão (HASH) sobre o atributo num_doente, sobre a tabela consulta.

Visto que a query se trata de uma igualdade (num_doente = <valor>), a hash table é a melhor forma de obter todos os resultados

2.

O tipo de índice é de Dispersão (HASH) sobre o atributo especialidade, sobre a tabela medico.

Visto que a query se trata de uma igualdade (especialidade = <valor>), em que este valor só pode tomar um baixo número de valores diferentes(6),

logo a hash table continua a ser a melhor forma de obter todos os resultados

3.

Como cada bloco tem 2kb e cada registo ocupa 1kb então temos 2 registos por bloco.

Logo como o número de registos é muito reduzido por bloco, teremos de ler muitos blocos. O que reduz muito o benefício dos índices.

Então não é preciso especificar mais índices para além dos já existentes da chave secundária.

4.

Como esta interrogação compara a celula do medico e um intervalo de datas, então o melhor tipo de indice é o bitmap pois a interrogação é sobre vários atributos(num_celula e data).

Star Schema:

```
create table d_time(
    id_time integer not null unique,
    day integer not null,
    week_day integer not null,
    week integer not null,
    month integer not null,
    trimester integer not null,
    year integer not null);


create table d_institution(
    id_inst integer not null unique,
    inst_name varchar(80) not null,
    inst_kind varchar(80) not null,
    region_num integer not null,
    county_num integer not null,
    constraint fk_dInstitution_institution
        foreign key(inst_name)
            references Institution(inst_name) on delete cascade on update cascade,
    constraint fk_dInstitution_region
        foreign key(region_num)
            references Region(region_num) on delete cascade on update cascade,
    constraint fk_dInstitution_county
        foreign key(county_num)
            references County(county_num) on delete cascade on update cascade);


create table f_presc_sale(
    id_presc_sale integer not null unique,
    id_doctor integer not null,
    patient_num integer not null,
    id_reg_date integer not null,
    id_inst integer not null,
    substance varchar(80) not null,
    subs_quant integer not null,
    constraint pk_f_presc_sale
        primary key(id_presc_sale),
```

```
  constraint fk_fpresc_sale_prescription_sale
      foreign key(id_presc_sale)
        references PrescriptionSale(sale_num) on delete cascade on update cascade,
  constraint fk_fpresc_sale_doctor
      foreign key(id_doctor)
        references Doctor(certificate_num) on delete cascade on update cascade,
  constraint fk_time
      foreign key(id_reg_date)
        references d_time(id_time) on delete cascade on update cascade,
  constraint fk_dInstitution
     foreign key(id_inst)
       references d_institution(id_inst) on delete cascade on update cascade);


create table f_analysis(
  id_analysis integer not null unique,
  id_doctor integer not null,
  patient_num integer not null,
  id_reg_date integer not null,
  id_inst integer not null,
  analysis_name varchar(80) not null,
  subs_quant integer not null,
  constraint pk_d_analise
      primary key(id_analysis),
  constraint fk_analysis
      foreign key(id_analysis)
        references Analysis(analysis_num) on delete cascade on update cascade,
  constraint fk_fanalysis_doctor
      foreign key(id_doctor)
        references Doctor(certificate_num) on delete cascade on update cascade,
  constraint fk_fanalysis_time
      foreign key(id_reg_date)
        references d_time(id_time) on delete cascade on update cascade,
  constraint fk_dInstitution
     foreign key(id_inst)
       references d_institution(id_inst) on delete cascade on update cascade);
```

Etl:

```
insert into d_time(id_time, day, week_day, week, month, trimester, year)
```

```
select distinct row_number() over (order by _date) as id_time, extract (day from _date) as day, extract(isodow from _date) as week_day, extract(week from _date) as week, extract (month from _date) as month, extract (quarter from _date) as trimester,extract (year from _date) as year from (
    (select distinct presc_date as _date from PrescriptionSale)
    union
    (select distinct reg_date as _date from Analysis)) table1;

insert into d_institution(id_inst, inst_name, inst_kind, region_num, county_num)
  select row_number() over (order by inst_name, inst_kind, region_num, county_num) id_inst, * from Institution;

insert into f_presc_sale(id_presc_sale, id_doctor, patient_num, id_reg_date, id_inst, substance, subs_quant)
  select distinct sale_num as id_presc_sale, certificate_num as id_doctor, patient_num, id_time as id_reg_date, id_inst, substance, subs_quant from (
    with a as (
      with c as (
        select a.sale_num, certificate_num, a.patient_num, presc_date, inst_name, a.substance, subs_quant from PrescriptionSale a inner join PharmacySale b
        on a.sale_num=b.sale_num
      )
      select d.id_inst, c.sale_num, c.certificate_num, c.patient_num, c.presc_date, c.substance, c.subs_quant from c inner join d_institution d on c.inst_name=d.inst_name
    )
    select * from a inner join d_time t
      on extract(year from a.presc_date)=t.year
        and extract(month from a.presc_date)=t.month
        and extract(day from a.presc_date)=t.day
  ) z;
```

OLAP:

1.

```
select doc_speciality, month, year, count(analysis_name)
from (
  f_analysis f inner join Doctor d on f.id_doctor=d.certificate_num
) c inner join d_time t on c.id_reg_date=t.id_time
where year between 2017 and 2020 and analysis_name = 'glicémia'
GROUP BY GROUPING SETS(doc_speciality, month, year);
```

2.

```
select week_day, month, county_num, sum(subs_quant), avg(subs_quant)
from (
 ((f_presc_sale s inner join d_institution i on s.id_inst=i.id_inst) c inner join Region r on c.region_num=r.region_num))
 d inner join d_time t on d.id_reg_date=t.id_time
where region_name = 'Lisboa' and month between 1 and 4 and year = '2020'
group by rollup(county_num, week_day, month);
```