Projeto de Bases de Dados - Parte 3

Grupo: 90, Turno: L04, Docente: Tiago Oliveira

Nome	**Nº Aluno**	**Esforço**
Mafalda Serafim	92512	15h00 (50%)
Rafael Gonçalves	92544	15h00 (50%)

1. Comandos de criação da base de dados

```
create table regiao
        (num_regiao         serial          not null unique,
        nome                varchar(8)      not null,
        num_habitantes      integer         not null,
        constraint pk_regiao primary key(num_regiao),
        constraint ri_1 check(nome in ('Norte', 'Centro', 'Lisboa', 'Alentejo', 'Algarve')));

create table concelho
        (num_concelho       serial          not null,
        num_regiao          serial          not null,
        nome                varchar(50)     not null,
        num_habitantes      integer         not null,
        constraint pk_concelho primary key(num_concelho, num_regiao),
        constraint fk_concelho_regiao foreign key(num_regiao) references
regiao(num_regiao));
```

NOTA: RI-concelho-1:nome = {concelhos de portugal continental}
Pressupondo que as tabelas de região e concelho são preenchidas uma única vez pelo administrador do programa, optámos por não especificar esta restrição de integridade uma vez que seria da responsabilidade do administrador garantir o seu cumprimento.

```
create table instituicao
        (nome               varchar(50)     not null unique,
        tipo                varchar(11)     not null,
        num_regiao          serial          not null,
        num_concelho        serial          not null,
        constraint pk_instituicao primary key(nome),
        constraint fk_instituicao_concelho foreign key(num_concelho, num_regiao)
references concelho(num_concelho, num_regiao),
        constraint ri_instituicao_1 check(tipo in ('farmacia', 'laboratorio', 'clinica', 'hospital')));

create table medico
        (num_cedula         serial          not null unique,
        nome                varchar(50)     not null,
        especialidade       varchar(50)     not null,
        constraint pk_medico primary key(num_cedula));

 create table consulta
        (num_cedula         serial          not null,
        num_doente          serial          not null,
        data                date            not null,
        nome_instituicao    varchar(50)     not null,
        constraint pk_consulta primary key(num_cedula, num_doente, data),
```

```
    constraint fk_consulta_medico foreign key(num_cedula) references medico(num_cedula),
    constraint fk_consulta_instituicao foreign key(nome_instituicao) references instituicao(nome),
    constraint ri_consulta_1 check(extract(isodow from data) not in (6, 7)),
    constraint ri_consulta_2 unique(num_doente, data, nome_instituicao));

create table prescricao
    (num_cedula  serial       not null,
    num_doente   serial       not null,
    data         date         not null,
    substancia   varchar(50)  not null,
    quant        integer      not null,
    constraint pk_prescricao primary key(num_cedula, num_doente, data, substancia),
    constraint fk_prescricao_consulta foreign key(num_cedula, num_doente, data) references consulta(num_cedula, num_doente, data));

create table analise
    (num_analise  serial       not null unique,
    especialidade varchar(50)  not null,
    num_cedula    integer,
    num_doente    integer,
    data          date,
    data_registo  date         not null,
    nome          varchar(50)  not null,
    quant         integer      not null,
    inst          varchar(50)  not null,
    constraint pk_analise primary key(num_analise),
    constraint fk_analise_consulta foreign key(num_cedula, num_doente, data) references consulta(num_cedula, num_doente, data),
    constraint fk_analise_instituicao foreign key(inst) references instituicao(nome),
    constraint ri_analise_aux check((num_cedula is not null and num_doente is not null and data is not null) or (num_cedula is null and num_doente is null and data is null)));
```

NOTA: RI-analise: a consulta associada pode estar omissa; não estando, a especialidade da consulta tem de ser igual à do médico.
Esta restrição de integridade não foi implementada completamente devido à sua complexidade. No entanto, garantimos que a consulta associada é foreign key ou que os parâmetros relativos a esta se encontram todos a null.

```
create table venda_farmacia
    (num_venda   serial       not null unique,
    data_registo  date         not null,
    substancia    varchar(50)  not null,
    quant         integer      not null,
    preco         numeric(5,2) not null,
    inst          varchar(50)  not null,
    constraint pk_venda_farmacia primary key(num_venda),
```

```
    constraint fk_venda_farmacia_instituicao foreign key(inst) references instituicao(nome));

create table prescricao_venda
    (num_cedula  serial       not null,
    num_doente   serial       not null,
    data         date         not null,
    substancia   varchar(50)  not null,
    num_venda    serial       not null,
    constraint pk_prescricao_venda primary key(num_cedula, num_doente, data, substancia, num_venda),
    constraint fk_prescricao_venda_venda_farmacia foreign key(num_venda) references venda_farmacia(num_venda),
    constraint fk_prescricao_venda_prescricao foreign key(num_cedula, num_doente, data, substancia) references prescricao(num_cedula, num_doente, data, substancia));
```

2. SQL

1.

```
SELECT concelho.num_concelho AS concelho, concelho.num_regiao AS regiao
FROM ((concelho
INNER JOIN instituicao ON concelho.num_concelho = instituicao.num_concelho AND concelho.num_regiao = instituicao.num_regiao)
INNER JOIN venda_farmacia ON instituicao.nome = venda_farmacia.inst)
WHERE venda_farmacia.data_registo = CURRENT_DATE
GROUP BY concelho.num_concelho, concelho.num_regiao
HAVING COUNT(*) >= ALL (
    SELECT COUNT(*) AS count
    FROM ((concelho
    INNER JOIN instituicao ON concelho.num_concelho = instituicao.num_concelho AND concelho.num_regiao = instituicao.num_regiao)
    INNER JOIN venda_farmacia ON instituicao.nome = venda_farmacia.inst)
    WHERE venda_farmacia.data_registo = CURRENT_DATE
    GROUP BY concelho.num_concelho, concelho.num_regiao);
```

NOTA: Caso dois ou mais concelhos apresentem o mesmo número de vendas, equivalente ao máximo, esta query devolve o ambos os concelhos.

2.

```
SELECT regiao.nome AS regiao, medico.num_cedula AS cedula
FROM ((((regiao
```

```
INNER JOIN instituicao ON regiao.num_regiao =  instituicao.num_regiao)
INNER JOIN consulta ON instituicao.nome = consulta.nome_instituicao)
INNER JOIN prescricao ON consulta.num_cedula = prescricao.num_cedula AND consulta.num_doente = prescricao.num_doente AND consulta.data = prescricao.data)
INNER JOIN medico ON prescricao.num_cedula = medico.num_cedula)
WHERE prescricao.data >= '2019-01-01' AND prescricao.data <= '2019-06-30'
GROUP BY regiao.nome, medico.num_cedula
HAVING (regiao.nome, COUNT(*)) IN (
        SELECT regiao, MAX(num_prescricoes) AS max_prescricoes
        FROM (
                SELECT regiao.nome AS regiao, medico.nome AS medico, COUNT(*) AS num_prescricoes
                FROM ((((regiao
                INNER JOIN instituicao ON regiao.num_regiao =  instituicao.num_regiao)
                INNER JOIN consulta ON instituicao.nome = consulta.nome_instituicao)
                INNER JOIN prescricao ON consulta.num_cedula = prescricao.num_cedula AND consulta.num_doente = prescricao.num_doente AND consulta.data = prescricao.data)
                INNER JOIN medico ON prescricao.num_cedula = medico.num_cedula)
                WHERE prescricao.data >= '2019-01-01' AND prescricao.data <= '2019-06-30'
                GROUP BY regiao.nome, medico.num_cedula
                HAVING COUNT(*) > 0) child
        GROUP BY regiao);
```

NOTA: Caso dois ou mais médicos apresentem o mesmo número de prescrições por região, apresentem o mesmo número de vendas, esta query devolve ambas as cédulas.

3.

```
SELECT medico.num_cedula AS cedula
FROM (((((((regiao
INNER JOIN concelho ON regiao.num_regiao = concelho.num_regiao)
INNER JOIN instituicao ON concelho.num_regiao =  instituicao.num_regiao AND concelho.num_concelho = instituicao.num_concelho)
INNER JOIN consulta ON instituicao.nome = consulta.nome_instituicao)
INNER JOIN prescricao ON consulta.num_cedula = prescricao.num_cedula AND consulta.num_doente = prescricao.num_doente AND consulta.data = prescricao.data)
INNER JOIN prescricao_venda ON prescricao.num_cedula = prescricao_venda.num_cedula AND prescricao.num_doente = prescricao_venda.num_doente AND prescricao.data = prescricao_venda.data)
INNER JOIN medico ON prescricao_venda.num_cedula = medico.num_cedula)
INNER JOIN venda_farmacia ON prescricao_venda.num_venda = venda_farmacia.num_venda)
WHERE prescricao_venda.substancia = 'Aspirina' AND concelho.nome = 'Arouca'
GROUP BY medico.num_cedula
HAVING COUNT(DISTINCT venda_farmacia.inst) = ALL (
```

```
SELECT COUNT(*)
FROM (concelho
INNER JOIN instituicao ON concelho.num_regiao =  instituicao.num_regiao AND concelho.num_concelho = instituicao.num_concelho)
WHERE concelho.nome = 'Arouca' AND instituicao.tipo = 'farmacia');
```

4.

```
SELECT DISTINCT analise.num_doente AS doente
FROM (analise
INNER JOIN prescricao ON analise.num_cedula = prescricao.num_cedula AND analise.num_doente = prescricao.num_doente AND analise.data = prescricao.data)
WHERE (prescricao.num_cedula, prescricao.num_doente, prescricao.data, prescricao.substancia) NOT IN (
SELECT num_cedula, num_doente, data, substancia
FROM prescricao_venda) AND EXTRACT(MONTH from prescricao.data) = EXTRACT(MONTH from CURRENT_DATE);
```