DI-FCT-NOVA 6 de julho de 2017

Bases de Dados

Exame de Recurso, 2016/17

Uma proposta de resolução

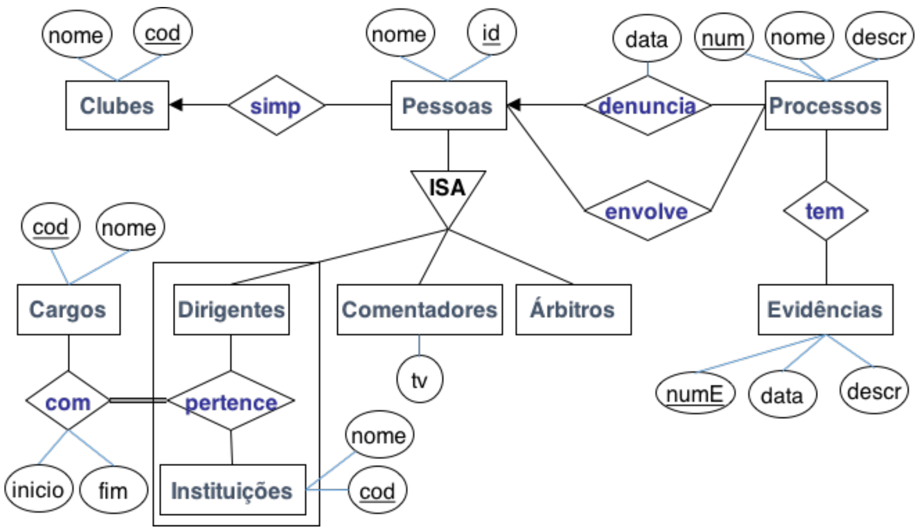
Parte I

(correspondente à matéria do 1º teste)

Grupo I.1

1.

a)



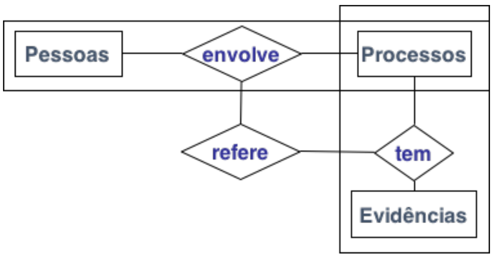
b)

```
create table clubes(
    cod varchar(3) primary key,
    nome varchar(20) not null
);

create table processos(
    num number primary key,
    nome varchar(100) not null,
    descr varchar(200),
    denuncia number,
    dataDenuncia date,
    foreign key (denuncia) references pessoas
);

create table evidencias(
    numE number primary key,
    data date not null,
    descr varchar(200),
);

create table tem(
    numE number,
    num number,
    primary key (num, numE),
    foreign key (num) references processos,
    foreign key (numE) references evidencias
);

create table pessoas(
    id number primary key,
    nome varchar(100) not null,
    simpatiza varchar(3),
    foreign key (simpatiza) references clubes
    );

create table arbitros(
    id number primary key,
    foreign key (id) references pessoas
);

create table comentadores(
    id number primary key,
    tv varchar(3),
    check tv in (‘RTP’,’SIC’,’TVI,’CM’),
    foreign key (id) references pessoas
);

create table dirigentes(
    id number primary key,
    foreign key (id) references pessoas
);
```

```
create table instituicoes(
    cod varchar(3) primary key,
    nome varchar(20) not null
);

create table com(
    cargo varchar(3),
    id number,
    inst varchar(3),
    inicio date not null,
    fim date,
    primary key (cargo, id, inst),
    foreign key (cargo) references cargos,
    foreign key (id, inst) references pertence
);

create table pertence(
    id number,
    cod varchar(2),
    primary key (id, cod),
    foreign key (id) references dirigentes,
    foreign key (cod) references instituicoes
);

create table cargos(
    cod varchar(3) primary key,
    nome varchar(20) not null
);

create table envolve(
    num number, id number,
    primary key(num, id),
    foreign key (id) references pessoas,
    foreign key (num) references processos
);
```

2.

a) Apenas se apresentam as entidades e relações envolvidas, bem como a nova relação. Os atributos encontram-se em **1b)**:



b) Para além das tabelas em **1b)**:

```
create table refere(
    id number,
    proc number,
    evid number,
    primary key (id, proc, evid),
    foreign key (id, proc) references envolve,
    foreign key (proc, evid) references tem
);
```

Grupo I.2

a) A única chave candidata é {Pessoa, Instituição}

b) Pessoa → Nome; Instituição → Descrição; Pessoa, Instituição → Cargo, DataInício

c) R1 = (Pessoa, Nome), R2 = (Instituição, Descrição), R3 = (Pessoa, Instituição,Cargo, DataInício)

d) O significado das dependências, pela ordem que aparecem, é:

- Cada pessoa só tem um nome.
- Cada instituição só tem uma descrição.
- Cada pessoa numa instituição só tem uma data de início de um cargo.
- Cada pessoa só tem um cargo em cada instituição

Parte II
(correspondente à matéria do 2º teste)

Descrição de uma base de dados de eventos de jogo, e de resultados de jogos.

Grupo II.1

1. Apresente uma expressão em **álgebra relacional e uma consulta SQL** para cada uma das perguntas (com abreviatura óbvias:

a) $\Pi_{descricao}(\sigma_{nome='FJM'}(pessoas) \bowtie contas \bowtie mens \bowtie classif \bowtie tags)$

select descrição
from pessoas **natural inner join** contas **natural inner join** mensagens
natural inner join classificações **natural inner join** tags
where nome = 'Francisco J Marques'**;**

b) $$\Pi_{pes.nome}(\sigma_{dest.para=r.email}((\sigma_{clube=FCP}(pes) \bowtie mens \bowtie contas \bowtie dest) \times \rho_r(\sigma_{clube=SLB}(pes) \bowtie mens \bowtie contas)))$$

select A.nome
from pessoas A **natural inner join** contas B **natural inner join** destinatários
natural inner join mensagens C,
pessoas D **natural inner join** contas E **natural inner join** mensagens F
where A.clube = 'FCP' **and** destinatário.para = E.email **and** D.clube = 'SLB'**;**

c) $\Pi_{idMens,email}(mensagens \bowtie (\Pi_{idMens}(mensagens) - \Pi_{mensOrig}(forwards)))$

with mensagensSemForward **as**
(select idMens **from** mensagens**) except (select** mensagemOrig **from** forwards**)**
select idMens, email
from mensagensSemForward **natural inner join** mensagens**;**

2. Apresente **uma consulta SQL** para cada uma das perguntas:

a) **select** count(*)
from mensagem
where texto **like** '%General Nhaga%'**;**

b) **select** idPessoa
from pessoas **natural inner join** contas **natural inner join** mensagens
natural inner join destinatários
group by idPessoa
having count(*) > 50**;**

c) **select** clube, count(*)
from pessoas **natural inner join** mensagens **natural inner join** classificações
where tag = 'parvoíce'
group by clube**;**

Grupo II.2

1.

a) **alter table** destinatários **add constraint** existMail
foreign key para **references** contas(email)**;**
alter table destinatários **add constraint** notNullPara
modify para varchar(50) **not null**;

b) Para simplificar, comecemos por criar uma view que retorna o conjunto de emissores de mensagens que são replies, e cujo endereço não estava nos destinatários da mensagem original:

```
create view emissoresNãoDest as
select mensagens.email
from  mensagens inner join replies on (mensagemOrig = mensagens.idMens)
where email not in (select para from destinatários
                            where detinatários.idMens = mensagens.idMens);

create assertion soOrigem ckeck
not exists (select * from);
```

c) Para simplificar, comecemos por criar uma view que para cada mensagem original diga quais os destinatários da mensagem do reply:

```
create view destinatarioReply as
select replies.idMens, para
from  replies natural inner join destinatários;

create assertion soOrigem ckeck
not exists (select *
        from mensagens natural inner join replies
        where (mensagemOrig, email)  not in (select * from destinatarioReply));
```

2.

a)

```
create trigger generalNhaga on insert of mensagens
referencing new row as nrow
for each row
when nrow.texto like '%General Nhaga%'
begin
    insert into ainspecionar values (nrow.idMens);
end;
```

b)

```
create trigger pinto2vieira on insert of destinatários
referencing new row as nrow
for each row
when nrow.para = 'lfvieira@slb.pt'
begin
    select email into origem from mensagens where idMens = nrow.idMens
    if origem = 'pintodacosta@fcp.pt' then
        insert into ainspecionar values (nrow.idMens);
    end if;
end;

create trigger vieira2pinto on insert of destinatários
referencing new row as nrow
for each row
when nrow.para = 'pintodacosta@fcp.pt'
begin
    select email into origem from mensagens where idMens = nrow.idMens
    if origem = 'lfvieira@slb.pt' then
        insert into ainspecionar values (nrow.idMens);
    end if;
end;
```

Parte III

(correspondente à matéria do 3º teste)

Grupo III.1

1.

a) resultado1(Descr) :-
pessoas(ID, 'Francisco J Marques', _), contas(E,ID),
mensagens(M, E, _, _, _), classificações(M,T), tags(T,Descr).

b) mensagemAtinge(M, M).
mensagemAtinge(M1, M2) :- forward(M3, M1), mensagemAtinge(M3, M2).
resultado2(Nome) :-
mensagens(M, 'pedroguerra@tvi.pt', _, _, _), mensagemAtinge(M,MA),
destinatários(MA,Para), contas(Para,ID), pessoas(ID, Nome, _).

c) pedroEnviou(E) :- mensagens(M, 'pedroguerra@tvi.pt', _, _, _), destinatários(M,E).
resultado3(Clube) :-
conta(E, P), not pedroEnviou(E), pessoas(P, _, Clube).

Grupo III.2

1. O problema é que só se podem colocar tuplos na tabela de destinatários se existir previamente uma linha na tabela de mensagens com esse idMens. Ora isto obriga a que as linhas na tabela de mensagens tenham que ser introduzidas antes das linhas dos destinatários dessas mensagens. Mas ao fazer isso, quando se introduz uma nova mensagem, essa mensagem ainda não tem qualquer destinatário. Qualquer restrição que se impusesse na base de dados para garantir que cada mensagem deveria ter pelo menos um destinatário falharia no momento em que se colocasse a mensagem na tabela. Havendo transações é possível não verificar a consistência da base de dados nesse momento, fazendo-o apenas após se introduzir a mensagem e pelo menos um destinatário.

2.

a) As duas serializações possíveis são: executar toda a transação da direita antes da da esquerda; ou executar toda a transação da direita depois da da esquerda. No primeiro caso o select devolve um único tuplo com o valor 0, e no segundo um único tuplo com valor 2

b) Os números antes das operações indicam a ordem temporal pela qual estas são executadas no escalonamento. Neste escalonamento, a operação de select devolve um único tuplo, com valor 1.

1. insert into destinatários values ('m1','a@pt');	
	2. select count(*) from destinatários where idMens = 'm1';
3. insert into destinatários values ('m1','b@pt');	

Grupo III.3

1.

```
<?xml version="1.0" encoding="UTF-8"?>
<mensagens>
    <conta email = "joao@pt">
        <nomePessoa>João</nomePessoa >
    </conta>
    <conta email = "maria@pt">
        <nomePessoa>Maria</nomePessoa >
    </conta>
```

```
        <conta email = “pedro@pt”>
            <nomePessoa>Pedro</nomePessoa >
        </conta>
        <mensagem de = “maria@pt”>
            <destinatario para = “pedro@pt”/>
            <destinatario para = “joao@pt”/>
            <subject>Olá</subject>
            <texto>Cá estou</texto>
            <data>05.07.17</data>
        </mensagem>
    </mensagens>
```

2.

- **a)** //destinatario@para
- **b)** //mensagem[data = “06.07.17”]/id(@de)/nomePessoa
- **c)** //mensagem[destinatarios@para = “joao@pt”]/data

3.

```
<!DOCTYPE mensagens[
        <!Element mensagens(tags*,conta*, mensagem*)>
        <!Element tags(#PCDATA)>
                <!ATTLIST tags tag ID #REQUIRED>
        <!Element conta(nomePessoa)>
                <!ATTLIST conta email ID #REQUIRED>
        <!Element mensagem(destinatário*,subject,texto,data)>
                <!ATTLIST mensagem de IDREF #REQUIRED>
                <!ATTLIST mensagem classificações IDREFS #REQUIRED>
        <!Element destinatario(#PCDATA)>
                <!ATTLIST destinatario para IDREF #REQUIRED>
        <!Element nomePessoa(#PCDATA)>
        <!Element subject(#PCDATA)>
        <!Element texto(#PCDATA)>
        <!Element data(#PCDATA)>
]>
```

Grupo III.4

1.

```
create type classificacao as (tag varchar(30));
create table mensagensClass(
        idMens number,
        email varchar(50),
        subject varchar(200),
        texto varchar(50000),
        classifs classificacao multiset);
```

2.

```
select distinct pessoas.nome
from mensagensClass R, unnest(R.classifs) as S, contas, pessoas
where S.tag = ‘parvoíce’ and R.email = contas.email and contas.idPessoa = pessoas.idPessoa;
```