DI-FCT-NOVA **11 de maio de 2016**

Bases de Dados

3º teste, 2015/16

Uma resolução

Grupo 1 [Datalog]

1.

a) refugSirios(Nome) :- refugiados(_, 'Siria', Nome, _).

b) nuncaAcolhidos(Nome) :- refugiados(Pass,_,Nome,_), entradas(_,_,Pass ,_),
not acolhido(Pass).

acolhido(Pass) :- acolhimentos(Pass ,_,_,_).

c) antepassado(Pass,PassAnt) :- progenitor(Pass,PassAnt).
antepassado(Pass,PassAnt) :- progenitor(Pass,PassX), antepassado(PassX,PassAnt).

Grupo 2 [Transações]

1.

a) O escalonamento é serializável. O único escalonamento serializado equivalente é:

```
update postosAcolhimento
set capacidade = capacidade - 50
where postoAcolh = 'A';
update postosAcolhimento
set capacidade = capacidade + 50
where postoAcolh = 'B';
```

```
update postosAcolhimento
set capacidade = capacidade * 1,1
where postoAcolh = 'A';
update postosAcolhimento
set capacidade = capacidade * 1,1
where postoAcolh = 'B'
```

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. Em ambos os casos o select devolve um único tuplo, com o valor 200.

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 150.

1. update postosAcolhimento
   set capacidade = capacidade - 50
   where postoAcolh = 'A';

2. select sum(capacidade)
   from postosAcolimento
   where postoAcolh in ('A','B');

3. update postosAcolhimento
   set capacidade = capacidade + 50
   where postoAcolh = 'B';

Grupo 3 [Objeto-Relacional]

1.

```
create type entrada as (data date, posto ref Postos scope postosfronteira);
create table refugiados(
        passRef varchar(20) primary key,
        paisRef varchar(20),
        nome varchar(100),
        dataNasc date,
        entradas entrada multiset
);
```

2. **select** R.posto-> pais
from refugiados R, **unnest**(R.entradas) **as** R
where passRef = '123456-A' **and** paisRef = 'Síria';

Grupo 4 [XML]

1.

```
<?xml version="1.0" encoding="UTF-8"?>
<refugiados nomePostoFronteira = "Lesbos">
    <refugiado pass = "12345-A">
        <nome>Adnan</nome>
        <pais>Síria</pais>
        <idade>25</>
    </refugiado>
    <refugiado pass = "23456-B">
        <nome>Bana</nome>
        <pais>Síria</pais>
        <idade>32</>
    </refugiado>
    <entrada refugiado="12345-A ">
        <data>7/05/2016</data>
    </entrada>
        <entrada refugiado="23456-B ">
        <data>8/05/2016</data>
    </entrada>
</refugiados>
```

2. Apresente expressões XPATH sobre um ficheiro de acordo com o DTD acima que devolvam os resultados das seguintes perguntas:

a) //data

b) //refugiado[pais="Siria" and idade < 6]/nome

c) //entrada[data="11/5/2016"]/id(@refugiado)/nome

3. Existem outras soluções compatíveis com o enunciado, por exemplo colocando alguma informação como elementos em vez de em atributos (e.g. a data das entradas, ou o passaporte dos refugiados). O enunciado não é explícito quanto a isso, e ambas as hipóteses são consideradas corretas.

```
<!DOCTYPE refugiados2[
    <!Element refugiados(dias*)>
        <!ATTLIST refugiados nomePostoFronteira #REQUIRED>
    <!Element dias(refugiado*)>
        <!ATTLIST refugiados data #REQUIRED>
    <!Element refugiado(nome,pais,idade)>
        <!ATTLIST refugiado pass ID #REQUIRED>
    <!Element nome(#PCDATA)>
    <!Element pais(#PCDATA)>
    <!Element idade(#PCDATA)>
]>
```

4. A pergunta devolve os nomes de refugiados que entraram em que datas. Mais concretamente, devolve um elemento com *tag* <entr> contendo um nome de refugiado e uma data, sempre que esse refugiado entrou nessa data. Por exemplo, para o ficheiro XML da pergunta 1 devolveria:

```
<entr>
        <nome>Adnan</nome>
        <data>7/5/2016</data>
</entr>
<entr>
        <nome>Bana</nome>
        <data>8/5/2016</data>
</entr>
```