Instituto Superior de Engenharia de Lisboa
Área Departamental
de Engenharia de Electrónica e Telecomunicações e de Computadores
Licenciatura Engenharia Informática e de Computadores (2º Sem 2016/2017)

Sistemas de Informação II - 1.º teste Duração: 2:30 horas

Docentes: Nuno Datia e Walter Vieira

---

1) Considere o seguinte código TSQL:

```
create table sensor(
        pk int primary key,
        nome nvarchar(30) not null,
        lat decimal not null,
        lon decimal not null,
        limiteMinimo decimal not null,
        limiteMaximo decimal not null,
);
create table sensorQueue(
        sensorId int not null,
        data datetime not null,
        valorLeitura decimal,
        primary key(sensorID,data)
);
```

```
    create table leitura(
        sensorId int not null references
    sensor,
        data datetime not null default
    getdate(),
        valorLeitura decimal,
        primary key(sensorID,data)
    );
    create table alarme(
        sensorId int not null references
    Sensor,
        data datetime not null,
        causa nchar(18) not null,
        primary key(sensorID,data),
        foreign key(sensorID,data)
                references leitura,
        check(
                causa in
                ('Superior ao Maximo',
                        'Inferior ao Minimo'))
);
```

a) Implemente a função **LeituraValida** que valida a leitura de um sensor. Recebe como parâmetros o identificador de um sensor e um valor de leitura, devolvendo os seguintes valores: 2 se estiver tudo válido; 1 se o identificador do sensor não existir na tabela; 0 se a leitura estiver acima do valor máximo admitido para o sensor; -1 se a leitura estiver abaixo do limite mínimo admitido para o sensor.

b) Implemente o procedimento armazenado **InsereLeitura**, que permite inserir uma leitura de um sensor. Considere que o procedimento recebe os parâmetros necessários para uma inserção com sucesso na tabela leitura. Se o valor da leitura do sensor, passado como parâmetro, estiver fora da gama de valores admissíveis, tem de ser adicionalmente inserido na tabela alarme um tuplo, indicando no campo causa o motivo do alarme. O procedimento armazenado deve gerar uma mensagem de aviso sempre que o identificador do sensor não for válido, com a mensagem "Identificador de sensor inválido" e estado 1. O procedimento devolve 1 se houve um erro, 0 em caso contrário. Garanta a consistência das operações, devendo propagar todas as exceções geradas pela execução do código.

c) Sabendo que todas as leituras de sensores são inicialmente colocadas na tabela **sensorQueue**, crie um gatilho sobre essa tabela que, reutilizando o código das alíneas a) e b), insira as leituras válidas. Note que esta tabela apenas guarda as leituras que ainda não foram processadas e inseridas na tabela **leitura**. Assim, ficam na tabela **sensorQueue** apenas as leituras com identificadores de sensor desconhecidos. Considere que sempre que o gatilho correr (após inserts), todas os registos da tabela **sensorQueue** (novos e antigos) são processados. Garanta que cada processamento de um registo na tabela **sensorQueue** é independente, não devendo ser anuladas as inserções anteriores pela existência de uma leitura de um sensor desconhecido.

---

2)

a) O que entende por controlo de concorrência otimista e pessimista?

b) Quais as suas vantagens e desvantagens relativas?

c) Como é que ambas as formas podem ser usadas com Entity Framework?

3) Considere o seguinte código TSQL:

```
create table fatura(id numeric(6), ano numeric(4), nifcli numeric(8), data datetime,
                                primary key(id, ano))

create function novoId(@ano numeric(4))
returns numeric(6)
as
begin
   declare @m numeric(6)
   select @m = max(id)+1 from fatura where ano = @ano
   if @m is null
      set @m = 1
   return @m
end

create proc insFat(@nifCli numeric(8))
as
begin
    declare @id numeric(6)
  declare @dt datetime = getdate()
  set transaction isolation level repeatable read
  begin tran
  begin try
   select @id = dbo.novoid(datepart(year,@dt))
   insert into fatura values(@id,datepart(year,getdate()),
                    @nifCli, @dt)
    commit
  end try
  begin catch
      select ERROR_MESSAGE()
      if @@TRANCOUNT > 0
            rollback
  end catch
end
```

Detetou-se que quando duas transações executam o procedimento armazenado **insFat**, concorrentemente, por vezes, uma delas aborta com uma exceção a que corresponde a mensagem "Violation of PRIMARY KEY ..."

a) Explique porque isso acontece.
b) Se o nível de isolamento fosse SERIALIZABLE observar-se-ia o mesmo ou outro efeito? Justifique.

4) Considere o seguinte código TSQL:

```
create  table tx(i int primary key, j int)

insert into tx values(1,1)
insert into tx values(2,2)
insert into tx values(3,3)
insert into tx values(4,4)

--Transação T1
set tran isolation level repeatable read
begin tran --1a
update tx set j = j+20 where i > 2 --1b
select * from tx where i < 3 --1c
rollback
commit --1d

--Transação T2
set tran isolation level repeatable read
begin tran --2a
update tx set j = j+20 where i > 2 --2b
select * from tx where i < 3 --2c
commit --2d
```

```
--Transação T3
set tran isolation level repeatable read
begin tran --3a
update tx set j = j+20 where i < 3 --3b
select * from tx where i > 2 --3c
commit --3d
```

a) É possível gerar situações de ***deadlock*** com a execução concorrente de T1 e T2? Se sim indique o escalonamento respetivo; se não, justifique.
b) Idem entre T1 e T3.
c) Repita a alínea a), mas admitindo que o nível de isolamento de ambas as transações é ***read committed***.
d) Repita a alínea b), mas admitindo que o nível de isolamento de ambas as transações é ***read committed***.

---

5) Considere o código XML seguinte:

```
<?xml version="1.0" encoding="UTF-8" ?>
    <a>
        <b id='1' desc='xyz'>
            <c>ABC</c>
            <d>123</d>
        </b>
        <x>
            <c>abc</c>
            <d>123</d>
            <e nome='abc'/>
        </x>
    </a>
```

a) Apresente o código *XSchema* para definir o tipo do elemento **a**, garantindo que: (i) as definições são criadas no *namespace* http://si2.1ep.pt; (ii) o elemento **e** é opcional.
b) Apresente as alterações necessárias ao código anterior, para que ele seja associado ao código *XSchema* implementado na alínea anterior.

Cotação:

alínea	1.a	1.b	1.c	2.a	2.b	2.c	3.a	3.b	4.a	4.b	4.c	4.d	5.a	5.b	**Total**
cotação	2	2,5	3,5	0,5	0,5	0,5	1,5	1,5	1,5	1,5	1,5	1,5	1	0,5	**20**