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 2018/2019)

Sistemas de Informação II - 2.º teste (2019/07/06) Duração: 2:30 horas

Docentes: Afonso Remédios e Walter Vieira

1) Num sistema para a gestão de manutenções de veículos foram criadas as seguintes tabelas:

```
create table veiculo(matricula varchar(16) primary key,
                     kmActuais int default 0,
                     descr varchar(255) not null
                    )

create table manutencao(matricula varchar(16) references veiculo,
                        km int,
                        valorTotal numeric(4,2) default 0,
                        data date not null,
                        primary key(matricula, km)
                       )

create table manutencaoItem(matricula varchar(16),
                        km int,
                        nLinha int,
                        valor numeric(4,2) not null,
                        foreign key (matricula, km) references manutencao(matricula, km),
                        primary key(matricula, km, nLinha)
                       )
```

a. Implemente em TSQL a função **valorTotalManutencao** que, recebendo no primeiro parâmetro a matricula e no segundo parâmetro os quilómetros (km), devolve a soma do valor dos itens associados a essa manutenção.

b. Construa o procedimento armazenado **insereManutencaoItem** que recebe como parâmetros a matricula, os quilómetros, a data, o número da linha e o valor do item, e procede da seguinte forma:

   - Assumindo que o veículo respetivo já existe, cria a entrada na tabela manutencao caso ainda não exista;
   - Atualiza o valor total da manutenção usando a função implementada na alínea a).

c. Altere o código que desenvolveu na alínea b) de forma que a atualização do valor total da manutenção seja efetuada por um gatilho definido sobre a tabela manutencaoItem. Deve apresentar também o código do gatilho.

---

2) Considere a tabela seguinte criata em TSQl ,com os registos indicados:

```
create table t(i int primary key, j int)
insert into t values (1,1)
insert into t values (3,3)
insert into t values (5,5)
insert into t values (7,7)
```

Considere também a lista de instruções seguinte:

```
(1) update t set j = j+10 where i=3
(2) select * from t where i >= 3 and i < 5
(3) update t set j = j+20 where i=7
(4) select * from t
(5) update t set i = i+3 where i=1
(6) begin tran
(7) commit
```

a. Indique um escalonamento de duas transações (e os respetivos níveis de isolamento) que usem apenas as instruções (1), (3),(6) e (7) e que provoquem uma situação de *deadlock*.

b. Indique um escalonamento de duas transações (e os respetivos níveis de isolamento) que usem qualquer subconjunto das instruções indicadas e que provoquem uma situação de *non repeatable read*.

c. Indique um escalonamento de duas transações (e os respetivos níveis de isolamento) que provoquem um *phantom tuple*.

d. Se a tabela t tivesse sido criada sem *primary key* e o nível de isolamento fosse *serializable*, seria possível o escalonamento <t1.(6),t1.(4),t2.(6),t2.(1),t1.(5),t2.(2),t1.(7),t2.(7)>? Justifique a resposta.

e. Indique um escalonamento de duas transações (e os respetivos níveis de isolamento) que usem qualquer subconjunto das instruções indicadas e que façam com que o escalonamento seja recuperável, mas com conflitos entre, pelo menos, dois pares de instruções.

Notas: Caso não exista o escalonamento pedido para uma dada alínea, justifique porque é que ele não existe.
Uma transação não deve executar a mesma instrução mais do que uma vez.
No escalonamento da alínea d), t1.(6) significa a transação t1 a executar a instrução (6), t2.(1), significa t2 a executar a instrução (1), etc.

---

3) Responda às seguintes questões:
   a. Qual a necessidade que os SGBD têm de usar o conceito de *range lock*?
   b. Na norma ISO é possível haver *lost updates* como resultado da concorrência de duas transações que apenas realizem escritas? Justifique a resposta.
   c. O que distingue o controlo de concorrência otimista do controlo de concorrência pessimista?

---

4) Considere as definições de entidades seguintes:

```
public class Veiculo  {
      public string Matricula { get; set; }
      public int kmActuais { get; set; }
      public string descr  { get; set; }
 }

public class Manutencao  {
      public Veículo V { get; set; }
      public string Matricula { get; set; }
      public int km { get; set; }
      public Decimal valorTotal  { get; set; }
      public DateTime data  { get; set; }
      public List<ManutencaoItem > Itens { get; set; }
 }
```

```
public class ManutencaoItem  {
      public int km { get; set; }
      public int NLinha  { get; set; }
      public Decimal Valor  { get; set; }
}
```

Crie a classe ManutencaoMapper e implemente com base em Ado.NET o seu método insereManutencao(Manutencao m) com gestão transacional realizada com System.Transactions.TransacrionScope. Admita a existência da classe DBHelper com a propriedade DbString cujo valor é a *connection string* para acesso à base de dados. O veículo associado à manutenção a inserir já deve existir. Deve usar o procedimento armazenado criado no exercício 1) b).

---

5) Considere o modelo de dados que seria gerado para modelar as tabelas `veiculo`, `manutencao` e `manutencaoItem` do exercício 1) utilizando a Entity Framework com a abordagem DBFirst. Construa uma aplicação c# que permita:
   a) Listar o total de despesas de manutenção com os itens cujo valor unitário ultrapassa 100€ na manutenção dos 50000Km para o veículo de matrícula '00-ZZ-00'. Não pode usar código SQL.
   b) Somar 10000Km ao campo `kmActuais` do veículo de matrícula '00-ZZ-00', garantindo consistência transacional. Não pode usar código SQL.

Cotação:

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