**Licenciatura em Engenharia Informática e de Computadores**

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

1º Teste, 11 de Janeiro de 2019

**Sistemas de Informação II**

Duração: **2h30m**

**Justifique devidamente** cada resposta.

1. Considere o modelo físico criado pelo Código 1.

```
create table t_Funcionario(
  func_nif int primary key,
  nome varchar(50) not null
)
create table t_Docente(
  func_nif int not null
                references t_Funcionario,
  num_mec int not null primary key,
  departamento varchar(30)
)

create table t_NaoDocente(
  func_nif int not null
                references t_Funcionario,
  num_bi int not null unique
)
```

Código 1: Ilustração do modelo de dados de suporte à questão

   (a) [3] Garanta, através de *triggers* T-SQL sobre as tabelas `t_Docente` e `t_NaoDocente`, que as três tabelas presentes no Código 1 implementam uma generalização total e exclusiva. Garanta apenas esta restrição para as inserções.

   (b) [2] Crie em T-SQL o procedimento armazenado `InsereDocente(nif, nome, num,dept)`, que permite inserir um funcionário docente. Garanta o correcto controlo transaccional, bem como a propagação de eventuais erros.

   (c) [2.5] Pretende-se criar o tipo `Docente(nif, nome, numero, departamento)`, que permite inserções, remoções e actualizações. Descreva e implemente uma solução em T-SQL, utilizando o procedimento implementado na alínea 1b.

   (d) [1] Indique as principais diferenças entre um *trigger after* e um *trigger instead of*.

   (e) [2.5] Implemente os métodos `Read` e `Update` do *Data Mapper* ilustrado no Código 2, usando ADO.NET. Note que o *Data Mapper* permite o mapeamento de uma entidade `Docente` que represente um funcionário docente, com as seguintes propriedades: `Nif`, `Nome`, `Numero`, `Departamento`.

```
class DocenteMapper{
    private string connectionString;
    public DocenteMapper(string connectionString){ \* ...*\   }
    public Docente Create(Docente docente) {  \* ...*\  }
    public Docente Read(int nif) {  \* ...*\  }
    public void Update(Docente docente) {  \* ...*\  }
    public void Delete(Docente docente)  {  \* ...*\  }
}
```

Código 2: Código parcial de um *Data Mapper* para a entidade do `Docente`

   (f) [1] Comente a afirmação: "A *Entity Framework* usa o padrão `Virtual Proxy` para garantir o *track changes*".

2. Considere os seguintes Códigos em T-SQL:

```
create table T
(
  t_id int primary key,
  txt varchar(100) unique null
);

insert into T(t_id,txt)
  values(1,'a'),(2,'b'),(3,'c');
```

Código 3: Tabela T

```
--Instrucao I1
delete from T where t_id %2 = 0

--Instrucao I2
update T set txt = 'b' where t_id %2 <> 0

--Instrucao I3
select * from T
```

Código 4: Instruções SQL

De acordo com as listagens ilustradas nos Códigos 3 e 4, indique:

a) [1.5] Indique o resultado da execução concorrente de dois processamentos transacionais T1 e T2, admitindo que um hipotético escalonamento é:
**T1.BEGIN TRAN**, **T1.I1**, T2.BEGIN TRAN, T2.I2, **T1.I3**, T2.I3, T2.COMMIT, **T1.COMMIT** .
Considere que o nível de isolamento é READ COMMITTED em ambos os processamentos. Justifique.

b) [0.75] Modifique o escalonamento da Alínea 2a para ser **Recuperável** e **Não cascadeless**. Justifique.

c) [0.75] Modifique o escalonamento da Alínea 2a para ser **Estrito**. Justifique.

d) [1.5] Qual seria o resultado do escalonamento da Alínea 2a se o nível de isolamento fosse REPEATABLE READ? Justifique.

e) [1.5] Indique o resultado da execução concorrente de dois processamentos transacionais T1 e T2, admitindo que um hipotético escalonamento é:
**T1.BEGIN TRAN**, **T1.I3**, T2.BEGIN TRAN, T2.I3, **T1.I1**, T2.I2, **T1.COMMIT**, T2.COMMIT .
Considere que o nível de isolamento é READ UNCOMMITTED em ambos os processamentos. Justifique.

f) [1.25] Observe os Códigos 5 e 6. Admitindo que a tabela T está vazia antes da execução dos processamentos transaccionais, indique qual o nível de isolamento mais baixo que garanta acções bem formadas e elimine todas as anomalias resultantes da concorrência. Justifique.

```
BEGIN TRANSACTION
SELECT t_id FROM T
INSERT INTO T VALUES (1,'A');
SELECT txt, t_id FROM T
COMMIT TRANSACTION
```

Código 5: Processamento transacional 1

```
BEGIN TRANSACTION
INSERT INTO T VALUES
 (SELECT count(*)+1 FROM T,'B')
SELECT * FROM T
COMMIT TRANSACTION
```

Código 6: Processamento transacional 2

g) [0.75] Indique, justificando, porque na implementação do nível de isolamento SNAPSHOT, embora recorrendo a múltiplas versões, também são usadas trancas exclusivas.

11 de Janeiro de 2019, Nuno Datia, Ricardo Silva