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 2021/2022)

Sistemas de Informação - exame de ER (2022/07/12) Duração: 2:30 horas

Docentes: Afonso Remédios, Nuno Datia e Walter Vieira

---

1) Considere as seguintes tabelas:

```
create table condutor(
    cId serial primary key,
    cNome varchar(50)
);

create table veiculoLigeiro(
    vlId serial primary key,
    vlMatricula char(10) not null unique,
    vlCilindrada int not null,
    vlLugares int not null default 5,
    vlCondutor int references condutor
);
```

```
create table veiculoPesado(
    vlId serial primary key,
    vlMatricula char(10) not null unique,
    vlCilindrada int null,
    vlTara int not null,
    vlCondutor int references condutor
);
```

a) Apresente o código em *plpgsql* que permita criar a vista **Veiculo(id,matricula,cilindrada,lugares,tara,condutor,tipo)**, que produz a informação conjunta presente nas tabelas **veiculoLigeiro** e **veiculoPesado**. O atributo **tipo** deve tomar um dos valores **{'ligeiro', 'pesado'}**, consoante a tabela de onde o tuplo foi obtido. Coloque **null** nos atributos para os quais não é possível atribuir um valor.

b) Crie em *plpgsql* o procedimento armazenado **InsereVeiculo** (com os parâmetros necessários, incluindo um atributo discriminativo para indicar o tipo de veículo a inserir) que permita inserir um veículo, ligeiro ou pesado. Garanta que não podem existir veículos ligeiros e pesados com a mesma matrícula, e que o mesmo condutor não pode estar associado a vários veículos simultaneamente. Devem ser feitas estas verificações e gerados erros caso elas não se verifiquem.

c) Apresente o código *plpgsql* que permita executar sem erro instruções das formas:

```
insert into public.veiculo(matricula,cilindrada,lugares,condutor,tipo)
values ('xx-11-xx',1000,2,1,'ligeiro');

insert into public.veiculo(matricula,cilindrada,tara,condutor,tipo)
values ('yy-11-yy',1000,3500,2,'pesado');
```

Na solução, deve usar os objetos desenvolvidos nas alíneas a) e b). Admita que existe o condutor indicado e que ele não está associado a nenhum veículo. Justifique a solução que propôs.

d) Apresente o código *plpgsql* que permite realizar duas chamadas do procedimento armazenado desenvolvido em b), em âmbito transacional. Garanta o comportamento transacional adequado (incluindo o nível de isolamento), mesmo em caso de erros ocorridos durante a execução do procedimento armazenado.

---

2) Considerando as tabelas do exercício 1:

(a) Apresente o código das classes de modelo (entidades) para modelar as tabelas **condutor** e **veiculoLigeiro**, de modo a serem utilizadas em JPA. Inclua todas as associações. Use anotações JPA.

(b) Admita a existência da classe auxiliar **VeicLigCondDTO** com os membros públicos **string nomeCond, string matricula, int cilidranda, int lugares**. Usando JPA, Implemente o método **inserirVeicLigCondutor(VeicLigCondDTO vlc)** da classe **BusinessLogic** que permita, com base nos membros de **vlc**, inserir um condutor e um veículo ligeiro não existentes, associando o veículo ao condutor. Deve ser garantida consistência transacional. Admita a existência de um *persistence-unit* identificado por **"pu-ER"**, devidamente configurado no ficheiro *persistence.xml*. **(Ver notas seguintes).**

Notas:

(1) Dado que o objetivo desta alínea é avaliar o domínio que os alunos têm sobre o JPA, o uso de quaisquer classes que não as do JPA obriga à sua implementação.

(2) Deve implementar a solução mais simples e adequada possível para a resolução desta alínea.

(c) Usando JPA, Implemente o método **inserir2VeicLCond(VeicLigCondDTO vlc1, VeicLigCondDTO vlc2)** da classe **BusinessLogic** que use o método **inserirVeicLigCondutor** desenvolvido em b) e permita inserir, na mesma transação, dois condutores e dois veículos ligeiros diferentes e não existentes na base de dados. Se for caso disso, deve indicar quais as alterações que deveria realizar ao código que apresentou em b).

---

3) Considere a tabela criada em *postgresql*, com os registos indicados (coluna da esquerda), bem como as instruções existentes na coluna da direita

```
create table T
(
    t_k int primary key,
    t_value varchar(10) null
);

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

```
start transaction; --(1)
select * from T where t_k = 2 for share; --(2)
select * from T where t_k = 3; --(3)
insert into T values(4,'d'); --(4)
delete from T where t_k >= 2; --(5)
update T set t_value='c' where t_k < 4; --(6)
delete from T where t_k = 1; --(7)
commit; --(8)
rollback; --(9)
```

a) Considere dois processamentos transacionais T1 = <T1.1, T1.3, T1.4, T1.8> e T2 = <T2.1, T2.6, T2.8>. Indique, justificando, qual o nível de isolamento mínimo de cada transação adequado para eliminar quaisquer anomalias com qualquer dos escalonamentos entre estas duas transações.

b) Considere dois processamentos transacionais T1 = <T1.1, T1.3, T1.4, T1.3,T1.8> e T2 = <T2.1,T2.5,T2.8>. e o escalonamento <T1.1, T2.1, T1.3, T2.5, T1.4, T2.8, T1.3, T1.8>. Indique qual o nível de isolamento mínimo de cada transação adequado para eliminar quaisquer anomalias, indicando o resultado da execução do escalonamento com esses níveis de isolamento.

c) Considere dois processamentos transacionais T1 = <T1.1, T1.2, T1.5, T1.8> e T2 = <T2.1, T2.6, T2.8>. Diga, para cada um dos níveis de isolamento *{read committed, repeatable read}* qual o resultado da execução do escalonamento <T1.1, T2.1, T1.2, T1.5, T2.6, T1.8, T2.8> assumindo que as transações se executam com o mesmo nível de isolamento. Caso o escalonamento não seja possível para um dado nível de isolamento, deve manter o mais possível a parte inicial do mesmo e dizer quais as possibilidades de execução a partir dessa parte inicial.

d) Considere apenas as instruções (1), (5), (6), (7) e (8) e indique um escalonamento de duas transações que usem subconjuntos destas instruções que, para os valores colocados na tabela no código apresentado, provoque uma situação de *deadlock*. Se não for possível a existência de *deadlocks*, justifique-o. Pode usar a mesma instrução nas duas transações.

e) Considere a transação T = <T.1, T.2, T.7, T.2,T.8> que se executa com o nível de isolamento *read committed*. Indique qual a transação (incluindo o respetivo nível de isolamento) que produziria os mesmos resultados (nas mesmas situações de concorrência) num sistema que use o protocolo *two phase lock*.

Notas:

(1) Considere que as transações têm início exatamente no momento em que se executa a instrução start transaction.

(2) Por resultado da execução de um escalonamento entende-se os valores retornados pelas instruções select, o que fica na base de dados e possíveis erros que ocorram durante a excução.

---

4) Sobre outros modelos de representação de dados, responda às seguintes questões:

a) Apresente, com a devida explicação, duas diferenças entre os modelos de armazenamentos por família de colunas e orientado a documentos.

b) Que mecanismos são usados nas bases de dados NoSQL para garantir tolerância a falhas caso haja falhas num servidor?

Cotação:

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