

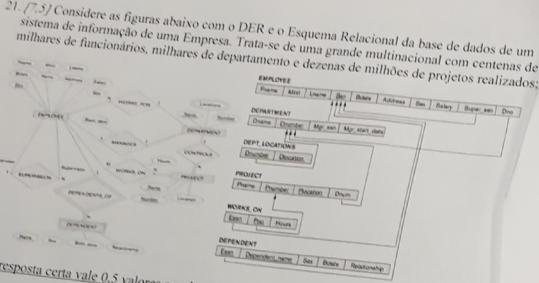
Nome: \_\_\_\_\_ N.º Mec. \_\_\_\_\_

Relativamente às perguntas 1 a 20, assinale na tabela ao lado, um X na coluna "V" que estão corretas e na "F" para as que estão incorretas.

Cada uma destas perguntas vale 0,5 valores e cada resposta errada desconta 0,25 valores /10/

1. Em SQL Server, podemos ter distintos DB users (um por cada DB) mapeados para o mesmo login;
2. Uma vista (view) com a cláusula 'WITH CHECK OPTION' garante que as condições da cláusula WHERE são verificadas numa operação de escrita (insert/update);
3. No controle de concorrência de transações, o mecanismo de locking é um método do tipo optimista;
4. Um Trigger do tipo After não deve ser utilizado quando sabemos que a operação (insert/update/delete) tem uma elevada probabilidade de ser "rolled back";
5. Num escalonamento concorrente, podemos ter situações de conflito quando temos transações simultâneas de consulta de dados;
6. Em SQL Server, um Stored Procedure pode ser utilizado como fonte de dados de uma consulta (i.e. na cláusula FROM de um SELECT);
7. Em SQL Server, o processo de page split ocorre quando se efetuam consultas (queries);
8. No modelo relacional, devemos evitar o estabelecimento de relacionamentos entre duas relações (i.e. junções) que não sejam baseados em atributos chave primária e estrangeira;
9. Em termos de SQL DCL, podemos cancelar (anular) um DENY ou um GRANT com um REVOKE;
10. Num Trigger do tipo After Update, a tabela lógica deleted não contém tuplos (i.e. está vazia);
11. Um Checkpoint é uma marca que representa o momento em que o SGBD escreve para o disco o Transaction Log;
12. Uma UDF do tipo 'Escalar' não pode ser utilizada numa restrição de integridade do tipo Check;
13. Um cursor permite percorrer sequencialmente os tuplos retornados por uma query definida para o efeito;
14. Em SQL Server, os Triggers são disparados uma vez por cada tuplo afetado pela operação (insert/update/delete);
15. Em transações concorrentes, uma leitura suja (dirty-read) ocorre quando uma segunda transação T2 acede a um elemento modificado por uma transação T1, sendo que a transação T1 acaba por ser "rolled back" entretanto;
16. Uma aplicação deve promover a utilização de SQL dinâmico contendo elementos fornecidos pelos utilizadores (na interface gráfica) de forma a agilizar o processo de interação com a base de dados;
17. Numa transação podemos ter um Rollback implícito quando ocorre um erro numa instrução SQL da transação.
18. Em SQL Server, um atributo definido como unique não aceita valores null repetidos;
19. A variável global @@rowcount do SQL Server permite saber quantas linhas de código já foram executadas até ao momento na corrente batch;
20. Em SQL Server, não podemos definir o nível de isolamento de uma transação;

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	



Nota: Cada resposta certa vale 0,5 valores e cada errada desconta 0,25 valores.

a. Imagine que queremos implementar um processo que elimine um funcionário e respetiva informação dependente (associada), de acordo com os requisitos listados abaixo. Deverá garantir a atomicidade de todas as operações envolvidas no processo:

- Eliminar os dependentes do funcionário;
- Caso seja supervisor de outros funcionários, estes devem ficar sem supervisor;
- Caso seja gestor de um departamento, deve gerar ainda uma mensagem de erro e cancelar todas as operações efetuadas;
- Caso esteja associado a projetos (works\_on), então deverá ser substituído pelo gestor do departamento para o qual trabalha;

Assinale na tabela ao lado, um X na coluna "V" para as declarações que estão corretas e na "F" para as que estão incorretas:

A	V
B	F
C	V
D	F
E	V
F	F
G	V
H	F

- A. A ferramenta a utilizar deve implementar as seguintes ações (pela ordem definida): 1. verificar se o funcionário é gestor de departamento, abortando o processo em caso afirmativo; 2. delete do funcionário de Employee; 3. remover a referência nos funcionários supervisionados (Super\_ssn passa a null); 4. delete de Dependent; 5. substituir do funcionário nos projetos associados pelo gestor do seu departamento;
- B. Sem o requisito iv, podíamos impor as restrições de integridade (associadas ao delete do funcionário) de forma declarativa no momento da criação das tabelas;
- C. Se implementarmos o processo com um Trigger Instead Of (no delete) em Employee, então o seguinte bloco de instruções T-SQL pode ser utilizado para verificar o critério iii:
- ```

DECLARE @issn as char(9);
SELECT @issn = ssn FROM inserted;
if (select count(*) from Department where Mgr_ssn=@issn) > 0
begin
    -- error code here
end
    
```
- É possível implementar o processo com uma UDF do tipo escalar retornando um inteiro (0 - sucesso; 1 - erro);

- E. Numa implementação com SP e, caso a primeira instrução seja verificar se o funcionário é gestor de departamento (abortando o processo em caso afirmativo), então não temos necessidade de utilizar uma transação para garantir a atomicidade de todas as ações do processo;
- F. A ferramenta ideal para fornecer o serviço desejado é um Stored Procedure (SP);
- G. Não faz sentido implementar o processo com um Trigger do tipo After delete na tabela Employee;

- b. Considere que todas as tabelas foram criadas (definidas), por defeito, com uma *primary key* e a base de dados tem habitualmente as seguintes consultas:
- Pesquisar os funcionários pelo Ssn ou pelo conjunto primeiro (Fname) e último nome (Lname);
  - Pesquisar departamentos pelo seu número (Dnumber) ou pelo nome (Dname);
  - Saber o nome dos projetos e número de horas para os quais determinado funcionário trabalha (Ssn);
  - Obter o salário médio dos funcionários por género (Sex) e localização do departamento (Dlocation);

Tendo como referência o SQL Server e a criação de índices para as tabelas Employee (PK - Ssn), Department (PK - Dnumber) e Works\_on (PK - Essn,Pno), assinale na tabela ao lado, um X na coluna "V" para as declarações que estão corretas e na "F" para as que estão incorretas:

- A. Devemos criar um índice *nonclustered* para o atributo Pno de Works\_on;
- B. Devemos criar um *nonclustered* para o atributo Dno de Employee;
- C. Devemos criar dois índices *nonclustered* para os atributos Fname e Lname (um para cada) de Employee;
- D. Devemos criar um índice *clustered* composto para os atributos (Fname, Lname) de Employee;
- E. Devemos criar um índice *nonclustered* para o atributo Essn de Works\_on;
- F. Por defeito, será criado um *clustered index* para cada tabela;
- G. Devemos criar um índice filtrado para os atributos Fname e Lname de Employee;
- H. Se utilizarmos *unique* na definição do atributo Dname de Department, já não necessitamos de criar explicitamente um índice *nonclustered* para este atributo;

| V | F |
|---|---|
| A |   |
| B |   |
| C |   |
| D |   |
| E |   |
| F |   |
| G |   |
| H |   |