# **Exercícios de Fixação**

# **(6) Inserindo e Modificando Dados**

# **(6.1) CONTROLE DE TRANSAÇÃO E INSERT**

**1)** Marque a alternativa CORRETA:

(1) Transação -> É uma "unit of work" (unidade de trabalho) que pode incluir várias consultas ou modificações no banco de dados.

(2) Podemos utilizar o BEGIN TRAN para iniciar uma transação, o ROLLBACK para confirmar os dados e o COMMIT para desfazer as alterações.

(3) As transações no SQL Server possuem as propriedades ACID (atomicidade, consistência, isolamento e durabilidade).

(4) O SQL Server On-Premises utiliza por default o nível de isolamento READ COMMITED SNAPSHOT, enquanto o Azure SQL Database utiliza o READ COMMITED.

a) V, F, V, F

b) V, V, V, V

c) V, F, V, V

d) F, F, V, V

e) F, F, V, F

**2)** Dado o script abaixo, no Exemplo 1 podemos validar a alteração antes de confirmar (COMMIT). Já no Exemplo 2, se o UPDATE for executado não conseguiremos fazer um ROLLBACK.

-- EXEMPLO 1 - UPDATE COM TRANSAÇÃO EXPLÍCITA:

BEGIN TRAN

UPDATE [dbo].[Cliente]

SET Dt\_Nascimento = '19800101'

WHERE Nm\_Cliente = 'Luiz Lima'

COMMIT

-- EXEMPLO 2 - UPDATE COM TRANSAÇÃO IMPLÍCITA:

UPDATE [dbo].[Cliente]

SET Dt\_Nascimento = '19890922'

WHERE Nm\_Cliente = 'Luiz Lima'

ROLLBACK

(V) Verdadeiro

(F) Falso

**3)** O SELECT abaixo vai ser executado com sucesso.

CREATE TABLE #TEMP\_CLIENTE (

Id\_Cliente INT IDENTITY(1,1) NOT NULL,

Nm\_Cliente VARCHAR(100) NOT NULL,

Dt\_Nascimento DATE NOT NULL

)

INSERT INTO #TEMP\_CLIENTE (Nm\_Cliente, Dt\_Nascimento)

VALUES ('Fabrício Lima', NULL)

(V) Verdadeiro

(F) Falso

**4)** O SELECT abaixo vai ser executado com sucesso.

CREATE TABLE #TEMP\_CLIENTE (

Id\_Cliente INT IDENTITY(1,1) NOT NULL,

Nm\_Cliente VARCHAR(100) NOT NULL,

Dt\_Nascimento DATE NULL

)

INSERT INTO #TEMP\_CLIENTE (Nm\_Cliente)

VALUES ('Luiz Lima')

(V) Verdadeiro

(F) Falso

**5)** Os comandos abaixo irão falhar, pois não é permitido utilizar procedures ou código sql dinâmico para fazer o INSERT em uma tabela.

-- PROCEDURE:

INSERT INTO #TEMP\_CLIENTE (Id\_Cliente, Nm\_Cliente, Dt\_Nascimento)

EXEC [dbo].[stpRetornaClientes]

-- CÓDIGO SQL DINÂMICO:

INSERT INTO #TEMP\_CLIENTE (Id\_Cliente, Nm\_Cliente, Dt\_Nascimento)

EXEC sp\_executesql N'SELECT Id\_Cliente, Nm\_Cliente, Dt\_Nascimento FROM [dbo].[Cliente]'

(V) Verdadeiro

(F) Falso

**6)** O SELECT abaixo vai ser executado com sucesso.

SELECT UPPER(Nm\_Cliente), Dt\_Nascimento

INTO #TEMP\_CLIENTE

FROM [dbo].[Cliente]

(V) Verdadeiro

(F) Falso

**7)** No SQL Server, podemos utilizar o BULK INSERT para importar dados de arquivos para dentro do banco de dados.

BULK INSERT #TEMP\_CLIENTE

FROM 'C:\SQLServer\Clientes-ANSI.txt'

WITH

(

CODEPAGE = '1252',

DATAFILETYPE = 'CHAR',

FIRSTROW = 2, -- UTILIZADO PARA IGNORAR A LINHA DO CABEÇALHO

FIELDTERMINATOR = ',', -- SEPARADOR DAS COLUNAS

ROWTERMINATOR = '\n' -- SEPARADOR DAS LINHAS

);

(V) Verdadeiro

(F) Falso

**8)** Para uma grande quantidade de dados, o SCRIPT 1 será mais rápido, pois o desempenho das transações IMPLÍCITAS é melhor do que as transações EXPLÍCITAS. Além disso, temos a mesma quantidade de transações nos dois scripts abaixo.

-- SCRIPT 1 - INSERT COM TRANSAÇÃO IMPLÍCITA

DECLARE @i INT

SET @i = 1

WHILE (@i <= 50000)

BEGIN

INSERT INTO Demo VALUES ('Performance test')

SET @i=@i+1

END

GO

-- SCRIPT 2 - INSERT COM TRANSAÇÃO EXPLÍCITA

DECLARE @i INT

SET @i = 1

BEGIN TRAN -- ABRINDO UMA ÚNICA TRANSAÇÃO EXPLÍCITA

WHILE (@i <= 50000)

BEGIN

INSERT INTO Demo VALUES ('Performance test')

SET @i=@i+1

END

COMMIT TRAN

(V) Verdadeiro

(F) Falso

# **(6.2) UPDATE / DELETE / TRUNCATE**

**9)** No SQL Server, não podemos utilizar o JOIN junto com o comando UPDATE.

UPDATE C

SET C.Vl\_Salario \*= 1.1

FROM [dbo].[Cliente] AS C

JOIN #TEMP\_ATUALIZAR\_CLIENTES AS T ON C.Id\_Cliente = T.Id\_Cliente

(V) Verdadeiro

(F) Falso

**10)** No SQL Server, podemos utilizar o OUTPUT junto com o comando UPDATE. Dessa forma, podemos trazer as informações dos registros antes (inserted) e após (deleted) a modificação.

UPDATE C

SET

C.Vl\_Salario \*= 1.1,

C.Ds\_Observacao = T.Ds\_Observacao

OUTPUT

deleted.Id\_Cliente,

deleted.Vl\_Salario AS Vl\_Salario\_Antigo,

inserted.Vl\_Salario AS Vl\_Salario\_Novo,

deleted.Ds\_Observacao AS Ds\_Observacao\_Antigo,

inserted.Ds\_Observacao AS Ds\_Observacao\_Novo

FROM [dbo].[Cliente] AS C

JOIN #TEMP\_ATUALIZAR\_CLIENTES AS T ON C.Id\_Cliente = T.Id\_Cliente

(V) Verdadeiro

(F) Falso

**11)** Dado a tabela abaixo, crie um script para:

DROP TABLE IF EXISTS #TEMP\_CLIENTE

CREATE TABLE #TEMP\_CLIENTE (

Id\_Cliente INT IDENTITY(1,1) NOT NULL,

Nm\_Cliente VARCHAR(100) NOT NULL,

Vl\_Salario NUMERIC(9,2) NULL

)

(1) Inserir um cliente com o nome “Luiz Vitor” e Salário R$1.000,00.

(2) Atualizar o salário do cliente “Luiz Vitor” para R$2.000,00.

(3) Excluir o cliente “Luiz Vitor”.

**12)** Marque a alternativa CORRETA:

(1) O DELETE é uma operação “fully logged”, enquanto o TRUNCATE TABLE é “minimally logged”.

(2) Uma vez que usamos o TRUNCATE TABLE, não será possível reverter essa ação, pois ele é “minimally logged”.

(3) Assim como o DELETE, o TRUNCATE TABLE também aciona as triggers.

(4) O TRUNCATE TABLE reinicia o IDENTITY da tabela.

a) V, V, F, F

b) V, F, F, V

c) V, F, V, F

d) F, V, F, F

e) F, F, F, V

# **(6.3) TRIGGERS / TASK IMPORT / EXPORT**

**13)** Marque a alternativa CORRETA:

(1) No SQL Server, podemos utilizar a TASK “Import / Export Data” para importar ou exportar dados de arquivos (TXT, CSV, Excel, etc.).

(2) Utilizar TRIGGER é uma boa prática e não impacta a performance do banco de dados.

(3) Ao utilizar uma trigger em uma tabela, quando fizermos um ROLLBACK as alterações da trigger não serão desfeitas.

(4) As triggers podem ser executadas explicitamente. Por exemplo:

EXEC TrgCliente\_Delete

a) V, F, F, V

b) V, F, V, F

c) V, F, F, F

d) F, F, V, F

e) F, V, F, F

**14)** O que a TRIGGER abaixo faz?

CREATE TRIGGER TrgCliente\_Insert

ON Cliente AFTER INSERT

AS

BEGIN

INSERT INTO Cliente\_Audit (Tp\_Alteracao, Id\_Cliente, Nm\_Cliente, Vl\_Salario)

SELECT 'I', Id\_Cliente, Nm\_Cliente, Vl\_Salario

FROM inserted

END

a) Vai fazer o INSERT na tabela “Cliente\_Audit” antes da tabela “Cliente”.

b) Não vai fazer o INSERT na tabela “Cliente”, mas faz na tabela “Cliente\_Audit”.

c) Não vai fazer o INSERT na tabela “Cliente\_Audit”, mas faz na tabela “Cliente”.

d) Vai fazer o INSERT na tabela “Cliente” e depois insere na tabela “Cliente\_Audit”.

e) Sei não Luiz, explica ai panoix.