Banco de Dados

Quiz 01 - Automação

# 01) Procedimentos armazenados (*stored procedures*) são comandos agrupados, isto é, um conjunto de instruções relacionadas que podem ser executados em uma chamada similar às funções. Assim, indique a sintaxe correta de criação de um *stored procedure*:

1. CREATE TABLE nome\_procedimento AS codigo\_sql
2. CREATE PROCEDURE codigo\_sql AS nome\_procedimento
3. CREATE SQL nome\_procedimento AS codigo\_sql
4. CREATE PROCEDURE nome\_procedimento AS codigo\_sql
5. CREATE EXEC codigo\_sql AS nome\_procedimento

**Dica**

Lembre-se de que um procedimento armazenado tem um nome e agrupa uma sequência de comandos SQL.

**Feedback**

A estrutura correta para criar uma *stored procedure* é: CREATE PROCEDURE, em seguida vem o nome do procedimento (nome\_procedimento) e, depois, a sequência de instruções SQL (AS codigo\_sql).

# 02) Procedimentos são armazenados e podem ser executados/reutilizados quantas vezes forem necessárias. Assim como nas funções, os procedimentos também podem receber parâmetros. Dessa forma, indique a sintaxe correta para a execução de um *stored procedure* por meio de parâmetros:

1. RUN nome\_procedimento @parametro = ‘valor\_atributo’
2. EXEC nome\_procedimento @parametro = ‘valor\_atributo’
3. SQL @parametro = ‘valor\_atributo’ nome\_procedimento
4. EXEC @parametro = ‘valor\_atributo’ nome\_procedimento
5. DO nome\_procedimento @parametro = ‘valor\_atributo’

**Dica**

Lembre-se de que, para ativar uma stored procedure, é necessário utilizar o comando SQL para executá-la, seguindo parâmetros se necessário.

**Feedback**

Para ativar um *stored procedure,* é necessário utilizar o comando SQL EXEC seguidos, opcionalmente, por parâmetros. Logo, a resposta correta é: EXEC nome\_procedimento @parametro = ‘valor\_atributo’

# 03) Indique se a afirmativa é verdadeira ou falsa.

# Uma das vantagens dos *stored procedures* é agrupar uma coleção de comandos SQL, fazendo com que esse conjunto possa ser reaproveitado e reutilizado em outro contexto, ocultando a complexidade da sua implementação e de suas regras de negócio.

Entre suas características principais, podem ou não receber parâmetros de entrada, porém nunca permitem retorno em sua saída.

Verdadeira Falsa

**Dica**

Lembre-se da estrutura dos *stored procedures* e dos parâmetros de entrada e saída.

**Feedback**

A afirmativa é falsa, pois os parâmetros de saída habilitam o retorno de dados de um procedimento armazenado para um procedimento chamador. O comando RETURN, por exemplo, termina incondicionalmente o procedimento e retorna um valor inteiro ao chamador e pode ser usado, também, para retornar status de sucesso ou falha do procedimento.

# 04) Indique se a afirmativa é verdadeira ou falsa.

# Os procedimentos armazenados são criados no mesmo arquivo de desenvolvimento do banco de dados, pois remetem a dados do projeto e, portanto, cada banco tem seu conjunto de procedimentos armazenados.

Verdadeira Falsa

**Dica**

Lembre-se de que, na estrutura de um banco de dados, existe uma pasta chamada Procedimentos Armazenados, que armazena *stored procedures* criados.

**Feedback**

A afirmativa é verdadeira. Uma vez criado, o procedimento é salvo automaticamente na pasta “Procedimentos armazenados”, no menu lateral esquerdo. Depois de armazenado, o procedimento pode ser executado como uma função.

# 05) Um *trigger*, ou gatilho, é um procedimento de armazenamento automático, quando ocorre um determinado evento ou ação em um banco de dados. Dessa forma, indique qual das ações a seguir pode ser utilizada como procedimento de armazenamento automático de banco de dados.

1. Login do banco de dados.
2. Atualização de software do banco de dados.
3. Conexão com banco de dados.
4. Shutdown do servidor.
5. Comandos DML (Data Manipulation Language – insert, update, delete).

**Dica**

Triggers estão associados à manipulação de dados de um banco de dados.

**Feedback**

Os comandos DML (Data Manipulation Language – insert, update, delete) podem ser utilizados como procedimentos de armazenamento automático de banco de dados, já que os *triggers* estão associados à manipulação de dados desses bancos, ou seja, aos comandos DML do SQL.

# 06) Para entender o funcionamento de um *trigger*, siga o exemplo a seguir, no cenário de uma livraria: após a inserção de um novo livro no banco de dados, o *trigger* vai calcular o tempo de leitura em horas, considerando que uma pessoa leva dois minutos para ler uma página.

O cálculo será: **(QuantidadePaginas \* 2 minutos) / 60 minutos**

No código a seguir, você verá que primeiro criou-se o *trigger* e foi definido em qual tabela estará a ação que vai disparar o gatilho. No exemplo, a inserção de dados na tabela **Livros** é que vai disparar o gatilho chamado **CalculaPaginas**. O comando **BEGIN** inicia um bloco de comandos e o **END** fecha. Dentro do **BEGIN**, declarou-se duas variáveis: uma para guardar o **Id**, e a outra para a quantidade de páginas do registro inserido. Então, definimos que essas duas variáveis terão o mesmo valor do **Id** e da quantidade de páginas do registro inserido.

Foi definida a fórmula do tempo de leitura em horas para o registro em que **Id** seja igual ao inserido (ou seja, o **Id** do próprio registro inserido).

Por último, foi inserido o evento que dispara o *trigger*, ou seja, a inserção de um registro na tabela **Livros**.

Com base em todo esse contexto, indique o número de horas que será necessário para a leitura do livro, considerando os dados de entrada do comando **insert**:

**CREATE TRIGGER CalculaPaginas -- nome do trigger**

**ON Livros -- tabela que a ação ocorrerá**

**FOR INSERT -- operação irá disparar o gatilho**

**AS**

**BEGIN -- abre bloco de comando executado em conjunto**

**DECLARE -- declara duas variáveis**

**@Id INT,**

**@QtdPaginas INT**

**SELECT @Id = Id, @QtdPaginas = QuantidadePaginas**

**FROM INSERTED -- seleciona os dados da inserção**

**SELECT (QuantidadePaginas \* 2) / 60 as 'Tempo de Leitura**

**em Horas' FROM Livros -- cálculo do tempo de leitura em horas**

**WHERE Id = @Id**

**END -- final do bloco**

**-- evento que dispara o trigger**

**INSERT INTO Livros (Titulo, DataPublicacao, QuantidadePaginas, EditoraId)**

**VALUES ('Livro BD - 2 edição', '11-02-2021', 300, 1)**

1. 0
2. 5
3. 10
4. 15
5. 20

**Dica**

Encontre a quantidade de páginas dentro dos parâmetros do comando **insert.**

**Feedback**

O tempo de leitura é de 10 horas. Tomando como base a declaração do comando insert, observamos que o livro tem 600 páginas. Substituindo o valor dentro do cálculo (QuantidadePaginas \* 2) / 60, temos: (300 \* 2) / 60 = 600 / 60 = 10 horas.

# 07) No contexto de banco de dados, sobre o recurso de gatilhos (*triggers*), indique se a afirmativa a seguir é verdadeira ou falsa.

# Um gatilho é um procedimento armazenado que é ativado por um evento e executa uma ou mais ações em um banco de dados.

Verdadeira Falsa

**Dica**

Lembre-se de que trigger é um procedimento armazenado automático.

**Feedback**

A afirmativa é verdadeira. Sendo trigger um procedimento armazenado automático, ele é disparado a partir de um evento ou ação no banco de dados.

# 08) Procedimentos armazenados (*stored procedures*) são comandos agrupados, isto é, um conjunto de instruções relacionadas que podem ser executados em uma chamada, similar às funções. Considerando o comando de criação de uma *stored procedure* a seguir, indique a resposta correta após sua criação e execução.

**create procedure SP\_01 @codigo int, @indice float**

**as**

**begin**

**update funcionarios set salario = salario + salario \* @indice**

**where matricula = @codigo;**

**end**

**execute SP\_01 12345, 0.15;**

1. Todos os funcionários receberão aumento de salário de 15%.
2. Apenas os funcionários do departamento 12345 receberão aumento de salário de 15%.
3. Os funcionários da empresa terão desconto de 15% no salário.
4. Somente o funcionário de matrícula 12345 receberá aumento de 15% no salário.
5. O funcionário de matrícula 12345 terá desconto de 15% em seu salário.

**Dica**

Analise os valores informados para a execução do **stored procedure SP\_01** e o cálculo que está sendo realizado pelo comando **update**.

**Feedback**

Somente o funcionário de matrícula 12345 receberá um aumento de 15% no salário. Dentro do comando **update**, podemos notar que **salario = salario + salario \* @indice**, ou seja, haverá aumento de 15% no salário. Na criação do *stored procedure*, temos o parâmetro **@codigo** e, dentro do comando **update**, observamos a linha **where matricula = @codigo**, que indica que somente o funcionário de matrícula 12345 receberá o aumento.

# 09) Procedimentos armazenados (*stored procedures*) são comandos agrupados, isto é, um conjunto de instruções relacionadas que podem ser executados em uma chamada, similar às funções. Considerando o comando de criação de um *stored procedure* a seguir, indique a resposta correta após sua criação e execução.

**create procedure SP\_02 @depto1 varchar(20), @depto2 varchar(20)**

**as**

**begin**

**update funcionarios set departamento = @depto2 where departamento = @depto1;**

**end**

**execute SP\_02 ‘Administrativo’, ‘Comercial’;**

1. Todos os funcionários serão transferidos para o departamento Comercial.
2. Todos os funcionários serão transferidos para o departamento Administrativo.
3. Somente os funcionários do departamento Administrativo serão transferidos para o departamento Comercial.
4. Alguns funcionários serão transferidos para o departamento Comercial e outros serão transferidos para o departamento Administrativo.
5. Somente os funcionários do departamento Comercial serão transferidos para o departamento Administrativo.

**Dica**

Observe os valores informados para a execução da **stored procedure SP\_02** e a atualização que está sendo realizada pelo comando **update**.

**Feedback**

Somente os funcionários do departamento Administrativo serão transferidos para o departamento Comercial. No comando update, está definido que **update funcionarios set departamento = @depto2 where departamento = @depto1;** ou seja, transferir os funcionários do **@depto1** (Administrativo) para o **@depto2** (Comercial).

# 10) O *trigger* a seguir é utilizado para atualizar o saldo de um determinado produto da tabela de estoque a partir de uma venda realizada. Considerando que o produto de código 45678 tem um saldo de 150 unidades em estoque antes da execução do *trigger*, indique qual será o saldo do produto após a criação e execução do trigger.

**CREATE TRIGGER Atualiza\_Saldo\_Estoque -- nome do trigger**

**ON vendas -- tabela que a ação ocorrerá**

**FOR INSERT -- operação que irá disparar o gatilho**

**AS**

**BEGIN -- abre bloco de comando executado em conjunto**

**DECLARE -- declara duas variáveis**

**@cod\_prod int,**

**@qtde float**

**SELECT @cod\_prod = codigo\_produto, @qtde = qtde\_venda**

**FROM INSERTED -- seleciona os dados da inserção**

**UPDATE estoque SET saldo = saldo - @qtde WHERE codigo\_produto = @cod\_prod**

**END -- final do bloco**

**-- evento que dispara o trigger**

**INSERT INTO vendas (codigo\_produto, data\_venda, qtde\_venda, valor) VALUES (45678, '15-10-2021', 5.00, 2350.00)**

1. 155
2. 2355
3. 140
4. 2340
5. 145

**Dica**

Observe os valores informados para a execução do *stored procedure* **Atualiza\_Saldo\_Estoque** e a atualização que está sendo realizada pelo comando **update**.

**Feedback**

O saldo do produto após a execução do trigger será 145. Observando o comando **UPDATE estoque SET saldo = saldo - @qtde WHERE codigo\_produto = @cod\_pro** e os parâmetros do*stored procedure* **Atualiza\_Saldo\_Estoque**, notamos que o valor das vendas é 5 e, portanto, devemos subtrair esse valor do saldo inicial: 150 – 5 = 145.