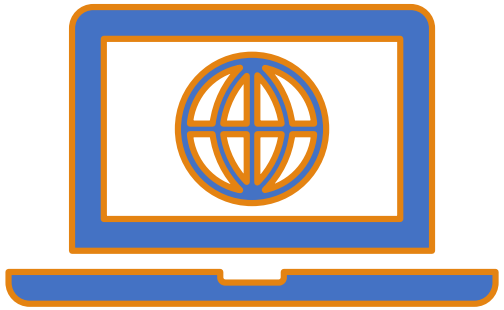


BANCO DE DADOS RELACIONAL

PROF. NILTON

AULA DE HOJE

- Storage Procedures

Storage Procedures

É comum, no desenvolvimento de um sistema, criarmos rotinas complexas de manipulação de dados. Normalmente essas rotina ficam na própria aplicação onde utilizamos várias instruções SQL em sequência para obter o resultado esperado.

Dependendo da rotina, pode ser necessário várias consultas e atualizações na base de dados, o que acarreta um alto consumo de recursos gerados pela aplicação. No caso de aplicações WEB isso se torna ainda mais visível por ser necessário uma quantidade maior de informações trafegando pela rede.

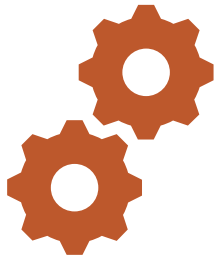
Storage Procedures

Uma boa forma de contornar ou amenizar esse consumo de recursos diretamente pela aplicação é transferir parte do processamento direto para o banco de dados. Levando em consideração que as máquinas servidoras geralmente têm configurações de hardware melhores em relação às máquinas clientes.

A questão é: como executar várias ações no banco de dados a partir de uma única instrução? A resposta para essa pergunta é: Stored Procedures (ou Procedimentos Armazenados, em português).

Stored procedures são rotinas definidas no banco de dados, identificadas por um nome pelo qual podem ser invocadas. Um procedimento desses pode executar uma série de instruções, receber parâmetros e disponibilizar valores.

Storage Procedures - Vantagens



Simplificação da execução de instruções SQL pela aplicação;

Transferência de parte da responsabilidade de processamento para o servidor.

Facilidade na manutenção, reduzindo a quantidade de alterações na aplicação.

Storage Procedures - Desvantagens

Necessidade de maior conhecimento da sintaxe do banco de dados para escrita de rotinas em SQL;

As rotinas ficam mais facilmente acessíveis. Alguém que tenha acesso ao banco poderá visualizar e alterar o código.

Storage Procedures - Sintaxe

```
DELIMITER $$
DROP PROCEDURE IF EXISTS nome_procedimento $$
CREATE PROCEDURE nome_procedimento (parâmetros)
BEGIN
   /*CORPO DO PROCEDIMENTO*/
END $$
DELIMITER ;
```

Storage Procedures - Sintaxe

Em “**nome_procedimento**”, deve-se informar o nome que identificará o procedimento armazenado. Este nome segue as mesmas regras para definição de variáveis, não podendo iniciar com número ou caracteres especiais (exceto o underline “_”).

Os “parâmetros” são opcionais e, caso não sejam necessários, devem permanecer apenas os parênteses vazios na declaração do procedure. Para que um procedimento receba parâmetros, é necessário seguir certa sintaxe (dentro dos parênteses).

Storage Procedures

Declaração de parâmetros

CREATE PROCEDURE nome_procedimento (MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)

O “**nome**” dos parâmetros também segue as mesmas regras de definição de variáveis.

O “**TIPO**” nada mais é que do tipo de dado do parâmetro (int, varchar, decimal, etc).

O “**MODO**” indica a forma como o parâmetro será tratado no procedimento, se será apenas um dado de entrada, apenas de saída ou se terá ambas as funções. Os valores possíveis para o modo são:

- **IN**: indica que o parâmetro é apenas para entrada/recebimento de dados, não podendo ser usado para retorno;
- **OUT**: usado para parâmetros de saída. Para esse tipo não pode ser informado um valor direto (como ‘teste’, 1 ou 2.3), deve ser passada uma variável “por referência”;
- **INOUT**: como é possível imaginar, este tipo de parâmetro pode ser usado para os dois fins (entrada e saída de dados). Nesse caso também deve ser informada uma variável e não um valor direto.

Storage Procedures
Realizando a chamada

Chamar um procedimento é bastante simples. Para isso fazemos uso da palavra reservada **CALL**

CALL nome_procedimento (parâmetros);

Storage Procedures

Parâmetro de entrada

```
DELIMITER $$
 CREATE PROCEDURE Selecionar_Produtos(IN quantidade INT)
BEGIN
    SELECT * FROM PRODUTOS
    LIMIT quantidade;
END $$
DELIMITER ;
```

Storage Procedure

Parâmetro de Saída

```
DELIMITER $$


CREATE PROCEDURE Verificar_Quantidade_Produtos(OUT quantidade INT)
BEGIN
    SELECT COUNT(*) INTO quantidade FROM PRODUTOS;
END $$
DELIMITER ;
```

Storage Procedure

Parâmetro de Saída

A função desse procedimento é retornar a quantidade de registros da tabela PRODUTOS, passando esse valor para a variável de saída “quantidade”. Para isso foi utilizada a palavra reservada INTO.

Para chamá-lo, usamos um símbolo de arroba (@) seguido do nome da variável que receberá o valor de saída. Feito isso, a variável poderá ser usada posteriormente.

```
CALL Verificar_Quantidade_Produtos(@total);
```

```
SELECT @total;
```

Storage Procedure

Parâmetro de Entrada e Saída

```
DELIMITER $$

CREATE PROCEDURE Elevar_Ao_Quadrado(INOUT numero INT)
BEGIN
    SET numero = numero * numero;
END $$
DELIMITER ;
```

O mesmo parâmetro é utilizado para entrada e saída.

Storage Procedure

Parâmetro de Entrada e Saída

```
SET @valor = 5;

CALL Elevar_Ao_Quadrado(@valor);

SELECT @valor;
```

Storage Procedure – Utilizando variáveis

DECLARE nome_variável TIPO DEFAULT valor_padrao;

A declaração das variáveis deve ser feita logo no início do corpo do procedure, para aquelas que serão utilizadas em todo o procedimento, ou dentro de um bloco BEGIN-END específico que limite seu escopo.

Para definir um valor para uma variável, usamos as palavras reservadas SET (no caso de passagem direta de valor) ou INTO (no caso de associação de valores dentro de consultas).

Outro ponto importante de se citar é o ESCOPO das variáveis, que define em que pontos elas são reconhecidas. Uma variável definida dentro de um bloco BEGIN/END é válida somente dentro dele, ou seja, após o END ela já não é mais reconhecida. Assim, é possível definir várias variáveis com o mesmo nome, mas dentro de blocos BEGIN/END distintos.

Exercícios

Crie os seguintes stored procedures:

1. Exiba a quantidade total vendida dos produtos (group by por código do Produto) (exibir id, nome do produto e quantidade)

2. Exiba todas as vendas efetuadas (número do pedido, cliente, total da venda)

3. Exiba todas as vendas realizadas de acordo com o mês/ano (mês e ano serão parâmetros IN)

4. Atualize o preço de custo de todos os produtos de acordo com a porcentagem informada.

5. Calcule a comissão para todos os vendedores com base nas vendas do mês/ano (mês e ano serão parâmetros IN) e na porcentagem de comissão informada (IN)

6. Atualize o estoque dos produtos vendidos.

Obrigado!

Duvidas?

Referências

https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

https://www.devmedia.com.br/stored-procedures-no-mysql/29030

https://www.devmedia.com.br/advanced-series-mysql-stored-procedures/7301