



Fecomércio RS



Senac

**Técnico em Informática
Módulo 4**

**Aula 4 - Stores Procedures
BANCO DE DADOS II**

Stores Procedures

Disciplina de Banco de Dados II

AGENDA 30/08/2016

- Procedimentos Armazenados
- Blocos BEGIN...END
- Parâmetros IN, OUT e INOUT
- Funções

Stores Procedures

Disciplina de Banco de Dados II

Introdução aos Stored Procedures

Stores Procedures

Disciplina de Banco de Dados II

Introdução aos Stored Procedures

Quando desenvolvemos aplicações que acessam banco de dados (boa parte delas), é comum executarmos rotinas complexas de manipulação desses dados a partir da linguagem/ferramenta utilizada. Para isso, utilizamos várias instruções SQL em sequência para obter o resultado esperado.

Stores Procedures

Disciplina de Banco de Dados II

Introdução aos Stored Procedures

Dependendo da rotina a ser executada, isso pode requerer várias consultas e atualizações na base, o que acarreta um maior consumo de recursos pela aplicação. No caso de aplicações web, isso se torna ainda mais visível, devido a maior quantidade de informações que precisam trafegar pela rede e de requisições ao servidor.

Stores Procedures

Disciplina de Banco de Dados II

Introdução aos Stored Procedures

Uma boa forma de contornar ou ao menos atenuar esse consumo de recursos diretamente pela aplicação é transferir parte do processamento direto para o banco de dados. Assim, considerando que as máquinas servidoras geralmente têm configurações de hardware mais robustas, enquanto nada se pode garantir com relação às máquinas clientes, essa pode ser uma “saída” a se considerar.

Stores Procedures

Disciplina de Banco de Dados II

Introdução aos Stored Procedures

A questão em que se esbarra é: como executar várias ações no banco de dados a partir de uma única instrução?

A resposta é: **Stored Procedures (ou Procedimentos Armazenados, em português).**

Stores Procedures

Disciplina de Banco de Dados II

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 retornar valores.

Stores Procedures

Disciplina de Banco de Dados II

Usar ou Não usar Stored Procedures

Stores Procedures

Disciplina de Banco de Dados II

Para exemplificar o funcionamento dos Stored Procedures e comparar a execução de uma rotina utilizando e não utilizando essa técnica, tomemos como base o seguinte contexto de uma aplicação comercial.

Stores Procedures

Disciplina de Banco de Dados II

- O cliente faz um pedido, no qual são inseridos itens;
- O pedido (bem como os itens) permanece com status “PENDENTE” até ser confirmado;
- O operador confirma o pedido, registrando o movimento

Quando o pedido é confirmado é preciso executar as ações:

- Atualizar o status do pedido;
- Atualizar o status dos itens do pedido;
- Lançar o valor do pedido

Stores Procedures

Disciplina de Banco de Dados II

Temos então pelo menos três instruções de atualização e/ou inserção. Poderíamos representar essa situação graficamente pela Figura 1.

Stores Procedures

Disciplina de Banco de Dados II



CLIENTE

ATUALIZAR STATUS PEDIDO →
ATUALIZAR STATUS ITENS →
LANÇAR NO CAIXA →



SERVIDOR / BD

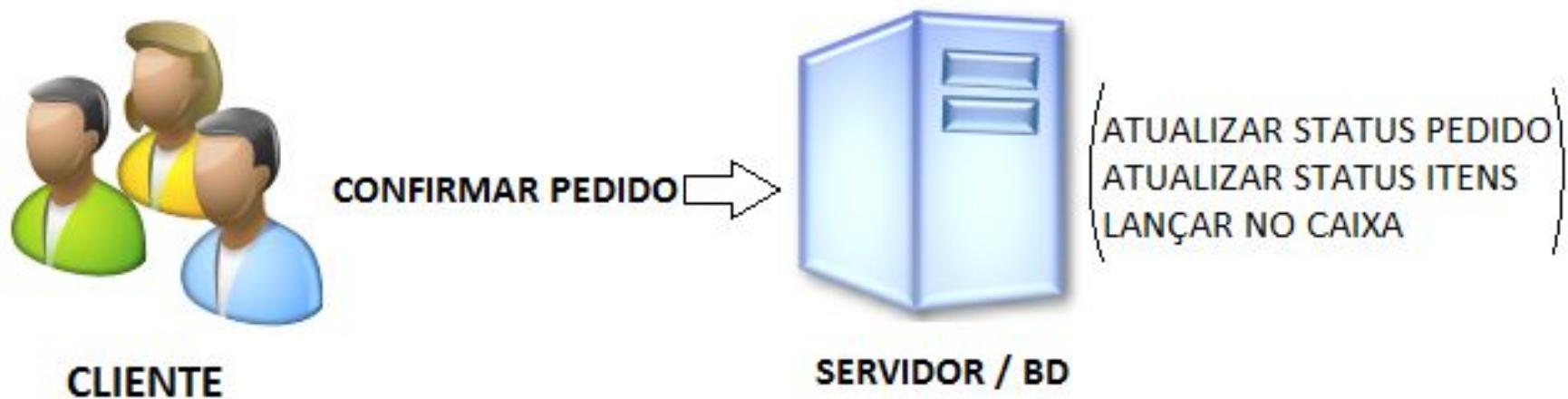
Stores Procedures

Disciplina de Banco de Dados II

Por outro lado, poderíamos agrupar essas três instruções no corpo de um procedimento e chamá-lo a partir da aplicação uma única vez. As ações de update/insert/delete, a partir daí, ficariam por conta do servidor. A representação gráfica desse modelo é mostrada na Figura 2 (considerando um procedure teoricamente chamado de “CONFIRMAR PEDIDO”).

Stores Procedures

Disciplina de Banco de Dados II



Stores Procedures

Disciplina de Banco de Dados II

**Principais vantagens e desvantagens de
seu uso: Pontos positivos:**

Stores Procedures

Disciplina de Banco de Dados II

- 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.

Stores Procedures

Disciplina de Banco de Dados II

**Principais vantagens e desvantagens de
seu uso: Pontos negativos:**

Stores Procedures

Disciplina de Banco de Dados II

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

Stores Procedures

Disciplina de Banco de Dados II

Criando e invocando Stored Procedures no MySQL

Stores Procedures

Disciplina de Banco de Dados II

Entrando no real foco deste artigo, será explicado a seguir como trabalhar com procedures no banco de dados MySQL, iniciando pela sintaxe utilizada para criação desse tipo de objeto, que pode ser vista na Listagem 1.

Stores Procedures

Disciplina de Banco de Dados II

Listagem 1: Sintaxe para criação de stored procedures no MySQL

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

Stores Procedures

Disciplina de Banco de Dados II

DELIMITER \$\$

Por padrão o MySQL utiliza o sinal de ponto e vírgula como delimitador de comandos.

No entanto, dentro do corpo do stored procedure será necessário separar algumas instruções internamente utilizando esse mesmo sinal, por isso é preciso inicialmente alterar o delimitador padrão do MySQL (neste caso, para \$\$) e ao fim da criação do procedimento, restaurar seu valor padrão.

Stores Procedures

Disciplina de Banco de Dados II

CREATE PROCEDURE nome_procedimento (parâmetros)

Onde consta “**nome_procedimento**”, deve-se informar o nome que identificará o procedimento armazenado.

Nome que será invocado pelo método **CALL**

Stores Procedures

Disciplina de Banco de Dados II

`CREATE PROCEDURE nome_procedimento (parâmetros)`

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), apresentada abaixo (Listagem 2).

Stores Procedures

Disciplina de Banco de Dados II

Listagem 2: Sintaxe de declaração de parâmetros em stored procedures

(MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)

Stores Procedures

Disciplina de Banco de Dados II

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).

Stores Procedures

Disciplina de Banco de Dados II

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:

Stores Procedures

Disciplina de Banco de Dados II

- **IN:** parâmetro é entrada/recebimento de dados, não podendo ser usado para retorno;
- **OUT:** 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:** 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.

Stores Procedures

Disciplina de Banco de Dados II

Tendo criado o procedure, chamá-lo é bastante simples. Para isso fazemos uso da palavra reservada CALL, como mostra o código da Listagem 3.

Listagem 3: Sintaxe para chamar um stored procedure

```
CALL nome_procedimento(parâmetros);
```

Stores Procedures

Disciplina de Banco de Dados II

A seguir temos um exemplo de uso de cada tipo de parâmetro.

Stores Procedures

Disciplina de Banco de Dados II

Usando parâmetro de **entrada**

Stores Procedures

Disciplina de Banco de Dados II

Listagem 4: Usando parâmetro de entrada

```
DELIMITER $$
```

```
CREATE PROCEDURE Selecionar_Produtos(IN quantidade INT)
```

```
BEGIN
```

```
    SELECT * FROM PRODUTOS
```

```
    LIMIT quantidade;
```

```
END $$
```

```
DELIMITER ;
```

Stores Procedures

Disciplina de Banco de Dados II

Assim, caso desejássemos selecionar dois registros dessa tabela, poderíamos usar o procedure como mostra a Listagem 5.

Stores Procedures

Disciplina de Banco de Dados II

Esse procedimento tem por função fazer um select na tabela PRODUTOS, limitando a quantidade de registros pela quantidade recebida como parâmetro.

Listagem 5: Chamando procedure com parâmetro de entrada

```
CALL Selecionar_Produtos(2);
```

Stores Procedures

Disciplina de Banco de Dados II

Usando parâmetro de saída.

Stores Procedures

Disciplina de Banco de Dados II

Listagem 6: Usando parâmetro de saída

```
DELIMITER $$
```

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

```
DELIMITER ;
```

Stores Procedures

Disciplina de Banco de Dados II

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.

Stores Procedures

Disciplina de Banco de Dados II

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, como vemos na Listagem 7.

Stores Procedures

Disciplina de Banco de Dados II

Listagem 7: Chamando procedure com parâmetro de saída

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

Ao executar a segunda linha, teremos como retorno o valor da variável @total, que será preenchida no procedure.

Stores Procedures

Disciplina de Banco de Dados II

Usando parâmetro de **entra e saída**.

O terceiro exemplo mostra um stored procedure chamado **Elevar_Ao_Quadrado**, que recebe uma variável e a altera, definindo-a como o seu próprio valor elevado à segunda potência.

Stores Procedures

Disciplina de Banco de Dados II

Listagem 8: Usando parâmetro de entra e saída

```
DELIMITER $$
```

```
CREATE PROCEDURE Elevar_Ao_Quadrado(INOUT numero INT)
```

```
BEGIN
```

```
    SET numero = numero * numero;
```

```
END $$
```

```
DELIMITER :
```

Stores Procedures

Disciplina de Banco de Dados II

Listagem 9: Chamando procedure com parâmetro de entrada e saída

```
SET @valor = 5;
CALL Elevar_Ao_Quadrado(@valor);
SELECT @valor;
```

Nesse caso, a mesma variável é usada como entrada e saída.

Revisão Banco de Dados

Disciplina de Banco de Dados II



alan.correa.sul@gmail.com



[alanalvescorrea](https://www.linkedin.com/in/alanalvescorrea)



github.com/alanalvescorrea/