Procedures (com Parâmteros e de Entrada e Saída)

Banco de Dados II

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

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

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

Sintaxe para criação de stored Procedures (com Parâmteros e de Entrada e Saída) no MySQL

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)
Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

Onde consta “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), apresentada abaixo.

```
MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)
```

Sintaxe de declaração de parâmetros em stored Procedures (com Parâmteros e de Entrada e Saída)

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)
Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

Onde consta “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), apresentada abaixo.

```
MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)
```

Sintaxe de declaração de parâmetros em stored Procedures (com Parâmteros e de Entrada e Saída)

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)
Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

```
MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)
```

Sintaxe de declaração de parâmetros em stored Procedures (com Parâmteros e de Entrada e Saída)

Aqui, é válido iniciar a explicação pelos itens mais simples.

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).

Sintaxe de declaração de parâmetros em stored Procedures (com Parâmteros e de Entrada e Saída)

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)
Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

```
MODO nome TIPO, MODO nome TIPO, MODO nome TIPO)
```

Sintaxe de declaração de parâmetros em stored Procedures (com Parâmteros e de Entrada e Saída)

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.

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

Outro ponto que merece destaque é o uso do comando **DELIMITER**. Por padrão o MySQL utiliza o sinal de ponto e vírgula como delimitador de comandos, separando as instruções a serem executadas. 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.

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

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

Sintaxe para chamar um stored procedure

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

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

```
DELIMITER $$

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

Usando parâmetro de entrada

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)
Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

Esse procedimento tem por função fazer um select na tabela PRODUTOS, limitando a quantidade de registros pela quantidade recebida como parâmetro. Assim, caso desejássemos selecionar dois registros dessa tabela, poderíamos usar o procedure como mostra a Listagem abaixo:

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

Chamando procedure com parâmetro de entrada

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

O próximo código mostra um exemplo de recebimento e retorno de parâmetro de saída.

```
DELIMITER $$

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

Usando parâmetro de saída

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e 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, como vemos na Listagem abaixo:

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

Chamando procedure com parâmetro de saída

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

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada 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.

```
DELIMITER $$

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

Usando parâmetro de entra e saída

Banco de Dados II

Procedures (com Parâmteros e de Entrada e Saída)

Criando e invocando Stored Procedures (com Parâmteros e de Entrada e Saída)

Nesse caso, a mesma variável é usada como entrada e saída, como vemos na chamada da listagem abaixo:

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

Chamando procedure com parâmetro de entrada e saída

Referências

- **Dev Media**

  www.devmedia.com.br/stored-Procedures (com Parâmteros e de Entrada e Saída)-no-mysql/29030