

Tarefa 25 - Stored Procedures – MySQL - Solução

Prof. Dr. Aparecido Freitas

1. Introdução

Para a execução desta atividade iremos considerar um Banco de Dados chamado **produtodb**, com as seguintes definições de tabelas:

CREATE TABLE Fabricante (

```
idFabricante INT(11) NOT NULL,  
Nome VARCHAR(60) NOT NULL,  
PRIMARY KEY (idFabricante)
```

);

CREATE TABLE Categoria (

```
idCategoria INT(11) NOT NULL,  
Descricao VARCHAR(60) NOT NULL,  
PRIMARY KEY (idCategoria)
```

);

CREATE TABLE Produto (

```
idProduto INT(11) NOT NULL,  
Descricao VARCHAR(45) NOT NULL,  
idCategoria INT(11) NULL DEFAULT NULL,  
idFabricante INT(11) NOT NULL,  
PRIMARY KEY (idProduto),  
INDEX fk_Produto_Categoria_idx (idCategoria ASC),  
INDEX fk_Produto_Fabricante1_idx (idFabricante ASC),  
CONSTRAINT fk_Produto_Categoria FOREIGN KEY (idCategoria) REFERENCES Categoria  
(idCategoria),  
CONSTRAINT fk_Produto_Fabricante1 FOREIGN KEY (idFabricante) REFERENCES Fabricante  
(idFabricante)
```

);

CREATE TABLE Filial (

```
idFilial INT(11) NOT NULL,  
idFabricante INT(11) NOT NULL,  
Nome VARCHAR(45) NOT NULL,  
Contato VARCHAR(45) NULL DEFAULT NULL,  
PRIMARY KEY (idFilial, idFabricante),  
INDEX fk_Filial_Fabricante1_idx (idFabricante ASC),  
CONSTRAINT fk_Filial_Fabricante1 FOREIGN KEY (idFabricante) REFERENCES Fabricante  
(idFabricante)
```

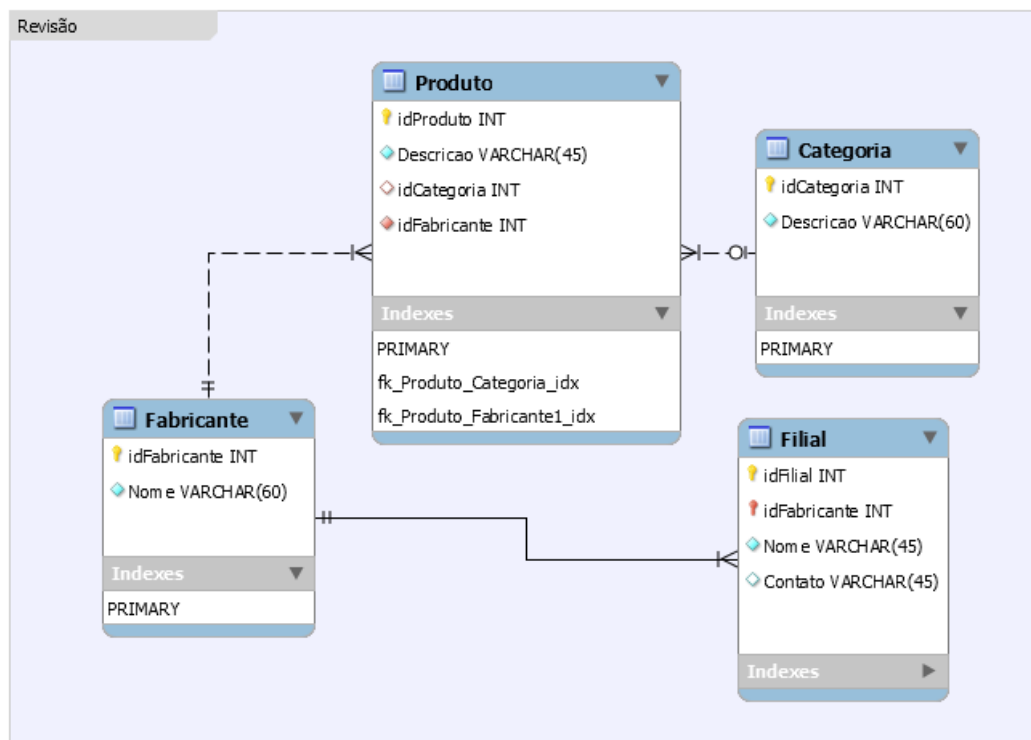
);

```

INSERT INTO fabricante (idFabricante, Nome) VALUES ('1', 'Nestlé');
INSERT INTO fabricante (idFabricante, Nome) VALUES ('2', 'Parmalat');
INSERT INTO fabricante (idFabricante, Nome) VALUES ('3', 'Kelloggs');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('1', 'Leite');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('2', 'Cereais Matinais');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('3', 'Achocolatado');
INSERT INTO produto (idProduto, Descricao, idCategoria, idFabricante) VALUES (1, 'Leite Integral', '1', '1');
INSERT INTO produto (idProduto, Descricao, idCategoria, idFabricante) VALUES (2, 'Nescau', '3', '1');
INSERT INTO produto (idProduto, Descricao, idCategoria, idFabricante) VALUES (3, 'Sucrilhos', '2', '3');

```

Empregaremos o seguinte modelo de dados:



Em **Stored Procedures** pode-se também trabalhar com variáveis de usuário.

Uma variável que é definida pelo usuário, também conhecida como user variables, é escrita precedida pelo símbolo **@** (arroba) e pode receber, através da declaração **SET**, valores do tipo inteiro (**INT**), real (**FLOAT**) ou **string**.

Variáveis do usuário são diferentes de variáveis locais.

```
SET @totalProdutos = 32;

SELECT @totalProdutos;
```

A principal diferença é que seu escopo é global ou seja ela existirá em memória por toda a seção do usuário, estando disponível a qualquer momento.

Pode-se atribuir um valor a uma variável definida pelo usuário utilizando-se as instruções **SET** ou **INTO**.

Exemplos:

```
DELIMITER $$
CREATE PROCEDURE sp_VariavelUsuario()
BEGIN
    SELECT CONCAT('Total de Produtos: ', @totalProdutos);
END$$
DELIMITER ;
```

```
SET @totalProdutos = 32;

CALL sp_VariavelUsuario();
```

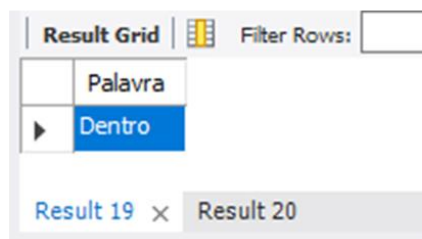
Assim como ocorre em outras linguagens, a linguagem de script utilizada para o desenvolvimento de Stored Procedures opera com escopo de variáveis. O escopo de uma variável está relacionado à sua visibilidade dentro do código.

Por exemplo, as variáveis iniciadas com o caractere arroba (@) são externas à sub-rotina, ou seja, são variáveis com escopo global. Já as definidas dentro da sub-rotina com o comando **DECLARE** são internas, ou seja, com escopo local.

Entendendo o escopo de variáveis:

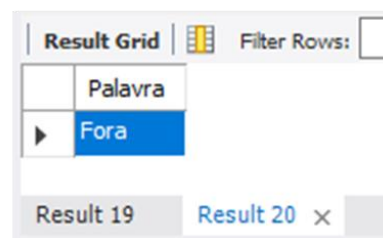
```
DELIMITER $$
CREATE PROCEDURE sp_escopo()
BEGIN
    DECLARE Palavra VARCHAR(6) DEFAULT 'Fora';
    BEGIN
        DECLARE Palavra VARCHAR(6) DEFAULT 'Dentro';
        SELECT Palavra;
    END;
    SELECT Palavra;
END$$
DELIMITER ;

CALL sp_escopo();
```



Palavra
Dentro

Result 19 x Result 20



Palavra
Fora

Result 19 Result 20 x

Desenvolva duas **Stored Procedures** onde:

A primeira deve ter o nome igual a **sp_VariavelUsuario001** e a segunda como **sp_VariavelUsuario002**.

A primeira é responsável por calcular o quadrado do valor da variável de usuário definida como **@valor**. Após atribuir o novo valor a variável, sua Stored Procedure deve invocar a segunda Stored Procedure.

A segunda Stored Procedure deve somar o valor **15** na variável de usuário **@valor** e exibir seu novo valor.

Antes de invocar a primeira **Stored Procedure** atribua o valor 5 a variável **@valor** e verifique se após a execução da primeira o valor exibido é igual a **40**.

Resposta:

```
DELIMITER $$
CREATE procedure sp_VariavelUsuario001()
BEGIN
    SET @valor = @valor * @valor;
    CALL sp_VariavelUsuario002();
END$$
DELIMITER ;

DELIMITER $$
CREATE procedure sp_VariavelUsuario002()
BEGIN
    SET @valor = @valor + 15;
    SELECT @valor;
END$$
DELIMITER ;

SET @valor = 5;
CALL sp_VariavelUsuario001();
```