



Function / Stored Procedure

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Function



***Function* - Funções**

As funções no SQL Server permitem encapsular lógicas de processamento que podem ser reutilizadas em consultas. Diferente das *Stored Procedures*, **elas retornam obrigatoriamente um valor** (escalar ou tabela) e podem ser usadas diretamente em instruções **SELECT**, **WHERE**, etc.



Funções Escalares

```
CREATE FUNCTION fn_Dobro(@Numero INT)
```

```
RETURNS INT
```

```
AS
```

```
BEGIN
```

```
    RETURN @Numero * 2;
```

```
END;
```



Funções Escalares

Como usar

Diretamente no **SELECT**

```
SELECT dbo.fn_Dobro(5) AS Resultado;
```

Em conjunto com uma tabela de exemplo

```
SELECT Pnome, Unome, Salario, dbo.fn_Dobro(Salario) AS Salario_Dobrado  
FROM FUNCIONARIO;
```



***Function* - Funções**

Tipos de Funções

Funções escalares - retornam um único valor (ex: número, string, data).

Funções de tabela (*inline*) - retornam uma tabela como resultado de uma query.

Funções de tabela (*multi-statement*) - retornam uma tabela construída dentro da função com múltiplas instruções.



Funções Escalares

```
CREATE FUNCTION fn_Nome(@parametro TIPO)
```

```
RETURNS TIPO -- INT
```

```
AS
```

```
BEGIN
```

Bloco de comandos

```
RETURN @valor;
```

```
END;
```



Funções Escalares

CREATE FUNCTION fn_Nome(@parametro **TIPO**)

RETURNS TIPO

AS

BEGIN

Bloco de comandos

RETURN @valor;

END;

Criar uma função que calcula a idade de um funcionário com base na data de nascimento



Funções *Inline* (Retornando Tabela)

```
CREATE FUNCTION fn_Nome(@parametro TIPO)
RETURNS TABLE
AS
RETURN
(
    SELECT colunas
    FROM tabelas
    WHERE condições
);
```

Retornar todos os funcionários de um determinado departamento.



Funções *Multi-Statement* (Tabela com Lógica)

```
CREATE FUNCTION fn_Nome(@parametro TIPO)
RETURNS @Tabela TABLE
(
    atributo TIPO,
    atributo TIPO,
    atributo TIPO
)
AS
BEGIN

    INSERT INTO @Tabela
    SELECT colunas
    FROM tabelas
    WHERE condições;
    RETURN;
END;
```

Criar uma função que retorna nome completo dos funcionários e o valor do salário anual, com férias e décimo terceiro;



Função com Mais de Um Parâmetro

```
CREATE FUNCTION fn_Nome(  
    @parametro TIPO,  
    @parametro TIPO  
)  
RETURNS TIPO  
AS  
BEGIN  
    Bloco de comandos;  
    RETURN @valor;  
END
```

Queremos calcular o salário anual de um funcionário (12 meses), mas também considerar um bônus variável (%), passado como parâmetro.



Exercícios

Desenvolver os exercícios da lista.

Stored Procedure

Criação e Execução no SQL Server



***Stored Procedure* - Procedimento Armazenado**

Definição

São lotes (*batches*) de declarações SQL que podem ser executados como uma subrotina.

Permitem centralizar a lógica de acesso aos dados em único local, facilitando a manutenção e otimização de código.

Também é possível ajustar permissões de acesso aos usuários, definindo quem pode ou não executá-las



Criar um Procedimento Armazenado

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```



Criar um Procedimento Armazenado

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie um procedimento que exiba seu nome.



Executar um Procedimento Armazenado

`EXEC(UTE) nome_procedimento @DepartamentoID = valor`

Obs. Se o procedimento armazenado for o primeiro comando de um batch não é necessário usar a palavra EXEC.



Exemplo

```
CREATE PROCEDURE teste
```

```
AS
```

```
SELECT 'Herysson Figueiredo' AS Nome
```

Para executar:

```
EXEC(UTE) teste
```



Exemplo

```
CREATE PROCEDURE sp_FuncionarioDepartamento(@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie um procedure que liste o nome completo dos funcionários e o nome dos seus respectivos departamentos.



Visualizar conteúdo de SP

Use o procedimento armazenado `sp_helptext` para extrair o conteúdo de texto de um *stored procedure*.

EXEC `sp_helptext` nome_procedimento



Criptografar *Stored Procedure*

```
CREATE PROCEDURE sp_Funcionario
```

```
WITH ENCRYPTION
```

```
AS
```

```
SELECT *
```

```
FROM FUNCIONARIO
```



Criptografar *Stored Procedure*

```
CREATE PROCEDURE sp_Funcionario
```

```
WITH ENCRYPTION
```

```
AS
```

```
SELECT *
```

```
FROM FUNCIONARIO
```

Agora tente visualizar seu
conteúdo com sp_helptext:
Exec sp_helptext p_LivroISBN



Modificar Stored Procedure

```
ALTER PROCEDURE nome_procedimento
```

```
AS
```

```
BEGIN
```

```
    bloco
```

```
END
```



Modificar Stored Procedure

```
ALTER PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS  
BEGIN
```

```
    bloco
```

```
END
```

Altere o procedimento `sp_FuncionarioDepartamento` para que receba o nome do departamento e liste todos os funcionários que fazem parte do mesmo.



Modificar Stored Procedure

```
EXEC sp_FuncionarioDepartamento @NomeDepartamento= 'Nome';
```



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie uma procedure que atualiza o salário de um funcionário baseado no CPF, se não encontrar nenhum funcionário com o CPF passado exiba uma mensagem.



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie um procedure que insira um novo funcionário no banco



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie um procedure que insira um novo funcionário e um novo departamento ao qual ele faz parte e o gerencia.



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie um procedure que insira um novo funcionário mas antes verifique se já não existe um funcionário com o mesmo nome (nome completo).



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro Tipo_dados)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

Crie uma procedure lista os funcionários cujo salário está entre dois valores fornecidos.



Exemplo

```
CREATE PROCEDURE nome_procedimento (@Parâmetro1 Tipo, @Parâmetro2 Tipo,)
```

```
AS
```

```
BEGIN
```

Bloco de códigos

```
END
```

```
EXEC nome_procedimento 25000,35000
```

```
EXEC nome_procedimento @Parâmetro2=35000, @Parâmetro1=25000
```



Exemplo: Parâmetro com valor padrão

```
CREATE PROCEDURE p_teste_valor_padrao ( @param1 INT, @param2  
VARCHAR(20) = 'Valor Padrão')
```

```
AS
```

```
SELECT 'Valor do parâmetro 1: ' + CAST(@param1 AS VARCHAR)
```

```
SELECT 'Valor do parâmetro 2 ' + @param2
```

Execução:

```
EXEC p_teste_valor_padrao 30
```

```
EXEC p_teste_valor_padrao @param1 = 40, @param2='Valor Modificado'
```




Exemplo: Parâmetro com valor padrão

```
CREATE PROCEDURE p_teste_valor_padrao ( @param1 INT, @param2  
VARCHAR(20) = 'Valor Padrão')
```

```
AS
```

```
SELECT 'Valor do parâmetro 1: ' + CAST(@param1 AS VARCHAR)
```

```
SELECT 'Valor do parâmetro 2 ' + @param2
```

Crie uma procedure que faz uma listagem dos funcionários por departamento, mas se o departamento não for especificado, ela lista todos os funcionários.



Parâmetros de Saída

Os parâmetros de saída habilitam um procedimento armazenado a retornar dados ao procedimento chamado.

Usamos a palavra-chave **OUTPUT** quando o procedimento é criado, também quando é chamado.

No procedimento armazenado, o procedimento de saída aparece como uma variável local; No procedimento chamador, uma variável deve ser criada para receber o parâmetro de saída;



Parâmetros de Saída

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```



Parâmetros de Saída

Executar passando um parâmetro:

```
DECLARE @valor AS INT = 15
```

```
EXEC teste @valor OUTPUT
```

```
PRINT @valor
```



Comando RETURN

O comando **RETURN** termina incondicionalmente o procedimento e retorna um valor inteiro ao chamador

Pode ser usado para retornar status de sucesso ou falha de procedimento.



Exemplo

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```

Crie uma procedure que retorna o nome completo de um funcionário com base no CPF passado como parâmetro de entrada.



Exemplo

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```

Crie um procedure para calcular o salário total de todos os funcionários de um determinado departamento e retorna o valor por meio de um parâmetro de saída.



Exemplo

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```

Crie uma procedure que retorna a idade de um funcionário com base no CPF fornecido.



Exemplo

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```

Crie uma procedure que calcula um aumento salarial com base em uma porcentagem fornecida e retorna o novo salário via um parâmetro de saída. Ela também deve verificar se o salário resultante excede um valor máximo predefinido (60000)



Exemplo

```
ALTER PROCEDURE teste (@par1 as INT OUTPUT)
```

```
AS
```

```
SELECT @par1*2
```

```
RETURN
```

Crie uma procedure que calcula o tempo de serviço de um funcionário a partir de sua data de admissão e retorna os anos, meses e dias de serviço como parâmetros de saída.



Referências

DATE, C. J. Introdução a sistemas de bancos de dados. 8. ed. Rio de Janeiro, RJ: Campus, 1998. 674 p.

GARCIA-MOLINA, Hector; WIDOM, Jennifer; ULLMAN, Jeffrey D. Implementação de sistemas de banco de dados. Rio de Janeiro, RJ: Campus, 2001. 685 p.

Elmasri, Ramez; Navathe, Shamkant B.. Sistemas de banco de dados, 7^a ed., 2018. (Biblioteca Digital)