INSTITUTO FEDERAL
Rio Grande do Norte | Campus Parnamirim

User Defined Functions (UDFs)

Prof. Fábio Procópio

fabio.procopio@ifrn.edu.br

Tecnologia em Sistemas para Internet – Programação em Bancos de Dados

Relembrando...

- Na aula passada, abordamos o conteúdo de visões;
- Nesta aula, falaremos sobre UDFs (*User Defined Functions*) e algumas funções de sistema do SQL Server. Vamos nessa?!

Introdução

- Como o nome sugere, *User Defined Functions* (UDF) são funções que são definidas por usuários do banco de dados;
- As UDFs podem ser usadas para executar uma lógica complexa quando uma única consulta simples não é suficiente;
- As UDFs podem receber parâmetros de entrada e retornar dados;
- Elas podem ser de dois tipos:
  - Escalar – retorna um único valor
  - Conjunto de linhas – retorna um conjunto de linhas
- Além das funções escalares e de conjunto de linhas, o SQL Server oferece também as funções de agregação, analíticas e de classificação

Funções do SQL Server

- As funções escalares possuem várias categorias:
  - Funções de configuração
  - Funções de conversão
  - Funções de cursor
  - Funções de dados e de data/hora
  - Funções JSON
  - Funções lógicas, matemáticas e *strings*
  - Funções de metadados
  - Funções de segurança
  - Funções de sistema
  - Funções estatísticas do sistema
  - Funções de texto e de imagem

Para conhecer essas funções, clique aqui.

Função escalar – 1 de 8

- Sintaxe básica para escrever uma função do tipo escalar:

```
CREATE FUNCTION schema.nome_funcao(@par1 tipo, @par2 tipo, ..., @parN tipo)
RETURNS tipo_valor_retornado
AS
  BEGIN
     RETURN valor
  END
```

Função escalar – 2 de 8

- Vamos criar uma função para retornar a quantidade de alunos matriculados em um curso:

```
CREATE FUNCTION dbo.fnQtdeAlunos(@IdCurso TINYINT)
RETURNS SMALLINT
AS
  BEGIN
     DECLARE @qtde_alunos SMALLINT

     SELECT @qtde_alunos = COUNT(NuMatricula)
       FROM TbAluno
      WHERE IdCurso = @IdCurso

     RETURN @qtde_alunos
  END
```

Função escalar – 3 de 8

- Vamos modificar a função anterior para validar se o curso informado está cadastrado:

```
ALTER FUNCTION dbo.fnQtdeAlunos(@IdCurso TINYINT)
RETURNS SMALLINT
AS
  BEGIN
     DECLARE @qtde_alunos SMALLINT
     IF NOT EXISTS(SELECT IdCurso FROM TbCurso WHERE IdCurso = @IdCurso)
        BEGIN
           /*Se o curso não existir, retornar o valor -1*/
           RETURN -1
        END
     SELECT @qtde_alunos = COUNT(NuMatricula)
       FROM TbAluno
      WHERE IdCurso = @IdCurso

     RETURN @qtde_alunos
  END
```

Função escalar – 4 de 8

- Agora, vamos testar a função que verifica a quantidade de alunos cadastrados
  - para invocar uma função, utiliza-se a palavra-chave SELECT
  - é preciso informar o nome do esquema no qual a função foi criada (no nosso caso, o esquema é o **dbo**)

```
SELECT dbo.fnQtdeAlunos(1) AS 'Quantidade de alunos'

/*Agora, vamos informar um curso inexistente*/
SELECT dbo.fnQtdeAlunos(100) AS 'Quantidade de alunos'
```

Função escalar – 5 de 8

- Vamos criar uma função que receba um CPF, sem pontos e sem traço, e devolva-o formatado

```
CREATE FUNCTION dbo.fnFormataCPF(@cpf VARCHAR(11))
RETURNS VARCHAR(14)
BEGIN
   DECLARE @cpf_formatado VARCHAR(14)
   SET @cpf_formatado = SUBSTRING(@cpf, 1, 3);
   SET @cpf_formatado = @cpf_formatado + '.' + SUBSTRING(@cpf, 4, 3);
   SET @cpf_formatado = @cpf_formatado + '.' + SUBSTRING(@cpf, 7, 3);
   SET @cpf_formatado = @cpf_formatado + '-' + SUBSTRING(@cpf, 10, 2);

   RETURN @cpf_formatado;
END
GO

/*Testando...*/
SELECT dbo.fnFormataCPF('52998224725');
```

Função escalar – 6 de 8

- Vamos criar uma função para validar um CPF, informado sem formatação. Se válido, a função devolve-o formatado. Senão, retorna uma mensagem de erro.

```
CREATE FUNCTION dbo.fnValidaCPF(@CPF VARCHAR(11))
RETURNS VARCHAR(20)
AS
BEGIN
  DECLARE @cont TINYINT, @num TINYINT;
  DECLARE @soma1 SMALLINT;

  SET @cont = 1;
  SET @soma1 = 0;
  WHILE @cont <= 9
    BEGIN
      SET @num = CONVERT(TINYINT, SUBSTRING(@CPF, @cont, 1));
      SET @soma1 = @soma1 + @num * (11 - @cont);
      SET @cont = @cont + 1;
    END
```

Função escalar – 7 de 8

```
/*Continuando...*/
DECLARE @digito1 TINYINT;
SET @digito1 = (@soma1 * 10) % 11;
IF @digito1 = 10 OR @digito1 = 11
    SET @digito1 = 0;

DECLARE @soma2 SMALLINT;
SET @soma2 = 0
SET @cont = 1;
WHILE @cont <= 10
  BEGIN
    SET @num = CONVERT(TINYINT, SUBSTRING(@CPF, @cont, 1));
    SET @soma2 = @soma2 + @num * (12 - @cont);
    SET @cont = @cont + 1;
  END
```

Função escalar – 8 de 8

```
  /*Continuando...*/
  DECLARE @digito2 TINYINT;
  SET @digito2 = (@soma2 * 10) % 11;
  IF @digito2 = 10 OR @digito2 = 11
     SET @digito2 = 0;

  IF (@digito1 = CONVERT(TINYINT, SUBSTRING(@CPF, 10, 1))) AND
     (@digito2 = CONVERT(TINYINT, SUBSTRING(@CPF, 11, 1)))
     RETURN (SELECT dbo.fnFormataCPF(@CPF));

  /*Validar também CPFs como 11111111111, 22222222222, ...*/
  RETURN 'ERRO: CPF Inválido';
END
GO
/*Testando...*/
SELECT dbo.fnValidaCPF('52998224725') AS 'Válido',
       dbo.fnValidaCPF('62998224725') AS 'Inválido'
```

13

Função do conjunto de linhas – 1 de 3

- Sintaxe básica para escrever uma função do tipo tabela:

```
CREATE FUNCTION schema.nome_funcao(par1 tipo, par2 tipo, ..., parN tipo)
RETURNS TABLE
AS
   RETURN (
      instrução SELECT
   )
```

14

Função do conjunto de linhas – 2 de 3

- Vamos criar uma função para retornar o nome e o semestre de ingresso a partir de um IdCurso informado:

```
CREATE FUNCTION dbo.fnAlunosIngresso(@IdCurso TINYINT)
RETURNS TABLE
AS
  RETURN (
            SELECT NoPessoa, NoSemestre
              FROM TbPessoa AS p
             INNER JOIN TbAluno AS a
                ON p.IdPessoa = a.NuMatricula
             INNER JOIN TbSemestre AS s
                ON a.IdSemestreIngresso = s.IdSemestre
             WHERE a.IdCurso = @IdCurso
          )
```

Função do conjunto de linhas – 3 de 3

- Agora, vamos testar a função que retornar os alunos e os respectivos semestres de ingresso
  - para invocar uma função desse tipo, utiliza-se a palavra-chave SELECT como se estivesse consultando uma tabela
  - é preciso informar o nome do esquema no qual a função foi criada (no nosso caso, o esquema é o **dbo**)

```
SELECT * FROM dbo.fnAlunosIngresso(1)
```

Referências

1. MiCROSOFT. **CREATE FUNCTION (Transact-SQL)**. Disponível em: https://docs.microsoft.com/pt-br/sql/t-sql/statements/create-function-transact-sql?view=sql-server-2017. Acessado em: 12 ago. 2019.

2. MICROSOFT. **Quais são as funções do banco de dados SQL?** Disponível em: https://docs.microsoft.com/pt-br/sql/t-sql/functions/functions?view=sql-server-2017. Acessado em: 24 jul 2019.