

Material de Apoio

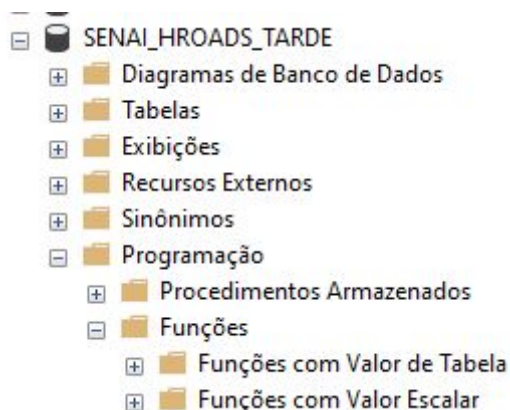
Sumário

1. Considerações Gerais	3
2. Funções Escalares	4
3. Funções de Tabela	5
Inline	5
Multi-Statement	6
Modelo Físico	7
4. Exercícios	8
5. Referências	9

1. Considerações Gerais

Aqui são tratados exclusivamente a criação de functions pelo usuário e sua aplicação:

- Existem 3 tipos de funções que podem ser criadas por um usuário, e estas estão divididas em 2 subcategorias: **Funções Escalares e Funções de Tabela**.
 - **Funções Escalares** retornam valores únicos (ex: função que soma dois números)
 - **Funções de Tabela** retornam tabelas (ex: função que te retorna uma tabela com apenas clientes que possuem mais de 18 anos)
- As funções ficam armazenadas dentro de determinado banco de dados, então não se esqueça do USE!, e ficam armazenadas no diretório - Programação - Funções.



- Sempre que for criar uma função será utilizado o comando **CREATE FUNCTION**.

2. Funções Escalares (Scalar Function)

Modelo base:

```
CREATE FUNCTION schema.Nome da Função
( @NomeParâmetro AS Tipo Parâmetro)
RETURNS Data Type do retorno da Função

AS

BEGIN

    Comandos

    RETURN Valor que a função deve retornar

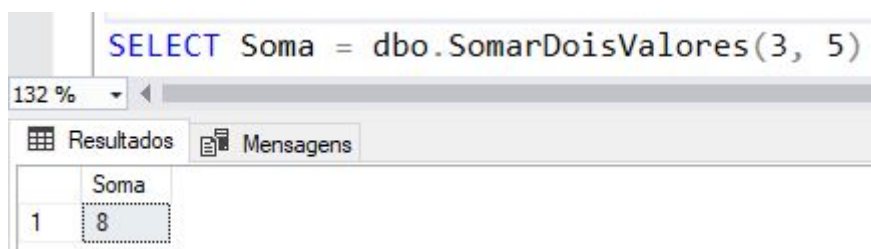
END
```

Aplicação:

```
CREATE FUNCTION SomarDoisValores(
@valor1 INT,
@valor2 INT
)
RETURNS INT
AS
BEGIN
    DECLARE @soma AS INT
    SET @soma = @valor1 + @valor2
    RETURN @soma
END
```

A função acima recebe dois parâmetros(dois valores INT enviados pelo usuário) e retorna outro INT, sendo esse a soma dos dois outros valores, importante notar que o BEGIN e END delimitam o **corpo da função**.

Selecionar função e resultado:



3. Funções de Tabela

Agora falando sobre as funções que retornam tabelas ao invés de valores individuais.

Função Inline

A função Inline retornará uma tabela, e é primariamente usada para **visualização de elementos**, por exemplo, para visualizar elementos de uma única ou diferentes tabelas baseado em determinado parâmetro.

Modelo base:

```
CREATE FUNCTION schema.Nome da Função
( @NomeParâmetro AS Tipo Parâmetro)

RETURNS Data Type do retorno da Função

AS

BEGIN

    RETURN Comando SELECT

END
```

Aplicação:

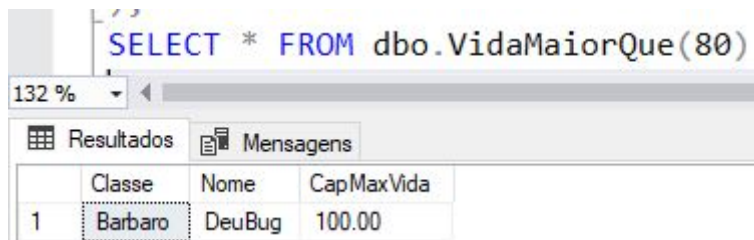
```
CREATE FUNCTION VidaMaiorQue (
    @VidaPersonagem INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT          Classes.Classe,          Personagens.Nome,
    Personagens.CapMaxVida

    FROM Classes

    INNER JOIN  Personagens  ON  Personagens.CapMaxVida  >
    @VidaPersonagem AND Personagens.idClasse = Classes.idClasse
);
```

A função acima foi criada a partir do Projeto HROADS e, baseado no parâmetro VidaPersonagem ela retornará um tabela apenas com personagens que possuem vida acima da vida especificada. (importante notar que na FUNÇÃO INLINE o **Begin** e o **End** não são estritamente necessários)

Selecionar função e resultado:



The screenshot shows a SQL query window with the text: `SELECT * FROM dbo.VidaMaiorQue(80)`. Below the query, the 'Results' tab is active, displaying a table with the following data:

	Classe	Nome	CapMaxVida
1	Barbaro	DeuBug	100.00

A função retornou os personagens que possuem vida maior que o parâmetro fornecido (80 no caso).

Deve-se notar que diferente da função escalar, nas de tabela o **FROM deve ser usado pois a função retorna uma tabela!**

Função Multi-Statement

A função multi-statement também retorna uma tabela, e o seu grande diferencial é que **ela aceita condicionais IF...ELSE!** Por ser mais pesada que a função inline ela deve ser usada com cautela, reiterando que para apenas visualizar elementos a função inline deve ser usada! Também vale notar que a Multi-Statement não retorna simplesmente uma tabela, **ela retorna uma variável que armazena a tabela!**

Modelo base:

```
CREATE FUNCTION schema.Nome da Função
( @NomeParâmetro AS Tipo Parâmetro)
RETURNS @Variável que armazena a tabela AS Definição da
Tabela
AS
BEGIN
    Comandos
    RETURN @Variável que armazena a tabela
END
```

Aplicação:

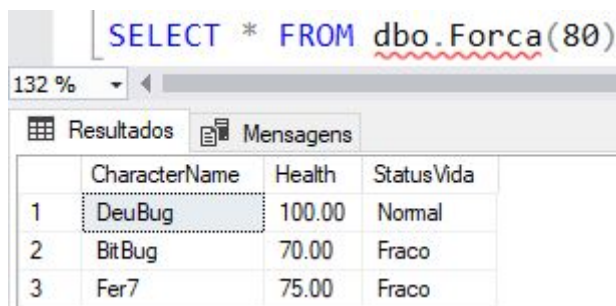
```

CREATE FUNCTION dbo.Forca
(
    @VidaBase INT
)
RETURNS @ResultTable TABLE(
    NomePersonagem VARCHAR(50), Vida DECIMAL(7,2), Classificação
    VARCHAR(50)
) AS BEGIN
    INSERT INTO @ResultTable
        SELECT Nome, CapMaxVida, NULL
        FROM dbo.Personagens
    UPDATE @ResultTable
        SET Classificação =
        CASE WHEN Vida < @VidaBase THEN 'Fraco'
        ELSE 'Normal'
        END
    RETURN
END

```

Aqui foi criada uma função que irá retornar uma nova tabela(**note que foi criado a variável @ResultTable para armazenar essa tabela**) contendo o nome, a vida e uma classificação do personagem baseado nos parâmetros descritos. A classificação do personagem é **dada por um CASE WHEN...ELSE**, ou seja, naquele caso, quando Vida é menor que a VidaBase fornecida a função vai classificar o personagem como 'Fraco', caso contrário como 'Normal'

Selecionar função e resultado:



	CharacterName	Health	StatusVida
1	DeuBug	100.00	Normal
2	BitBug	70.00	Fraco
3	Fer7	75.00	Fraco

Baseado no parâmetro '80' os personagens foram corretamente classificados como fracos ou fortes.

4. Exercícios de Fixação (Baseados no Projeto HROADS)

1. Crie uma função capaz de multiplicar dois números quaisquer.
2. Crie uma função que mostre apenas personagens com Mana acima de 70
3. Crie uma função que classifique aqueles com Mana acima de 70 como 'Poderosos', caso contrário como 'Normais'

5. Referências

<http://www.linhadecodigo.com.br/artigo/687/sql-server-funcoes-de-usuario-user-functions.aspx>

<http://db4beginners.com/blog/voce-sabe-o-que-e-uma-function/>

<https://www.devmedia.com.br/construindo-funcoes-para-sql-server/20934>