

## Tarefa Orientada 14

### **Funções**

#### Objetivos:

- Criar Funções
- Chamar Funções
- Utilizar parâmetros de entrada

Uma função definida pelo utilizador (*User-Defined Function* - UDF), abreviadamente designada função, é um tipo de objeto guardado na base de dados. As funções podem conter uma ou mais instruções SQL, tal como os procedimentos armazenados. Contudo, existem várias diferenças entre procedimentos armazenados e funções que se apresentarão mais à frente.

Existem três tipos de UDF:

- Função escalar (*scalar-valued function*) - devolve um valor de um dos tipos de dados T-SQL.
- Função com valor de tabela (*table-valued function*):
  - simples - (*simple table-valued function ou inline table-valued function*) - devolve um resultado do tipo tabela, sendo baseada numa só instrução SELECT.
  - com múltiplas instruções - (*multi-statement table-valued function*) - devolve um resultado do tipo tabela e é baseada em várias instruções SQL.

Uma UDF pode ter vários parâmetros de entrada. Porém, ao contrário de um procedimento armazenado, uma UDF não pode ter parâmetros de saída. Nas UDF a instrução RETURN tem de ser usada para devolver o resultado da função. O valor devolvido tem de ser compatível com o tipo de dados definido na cláusula RETURN.

1 Formule, analise e execute as instruções apresentadas.

1.1 Exemplo de uma UDF escalar para devolver o identificador de um fornecedor, dado o seu nome.

```
CREATE FUNCTION fnIDFornecedor (@NomeFornecedor varchar (50))
RETURNS int
BEGIN
    RETURN (SELECT IDFornecedor FROM Fornecedores WHERE
    Nome=@NomeFornecedor)
END
```

--instrução que invoca a função escalar

```
SELECT DataFactura, TotalFactura FROM Facturas
WHERE Fornecedor= dbo.fnIDFornecedor ('IBM')
```

--resultado

|   | DataFactura         | TotalFactura |
|---|---------------------|--------------|
| 1 | 2006-02-25 00:00:00 | 116,54       |
| 2 | 2006-03-14 00:00:00 | 1083,58      |

Uma UDF escalar é invocada incluindo a função numa expressão. No exemplo, a cláusula WHERE usa o valor devolvido pela função fnIDFornecedor. De notar que poderá ser necessário incluir o nome do esquema para referir uma UDF. Já uma UDF não escalar (*table-valued*) é invocada referindo-a da mesma forma que uma tabela ou vista.

Note-se que um esquema é um contentor de objetos de base de dados que pode ser definido no âmbito de uma base de dados. Um esquema só pertence a um utilizador, mas um utilizador pode ter vários esquemas. O uso de esquemas permite que os objetos não fiquem “dependentes” do utilizador que os criou. Se não for definido nenhum esquema, por omissão, será assumido o esquema dbo.

Uma UDF não pode executar ações que modifiquem o estado da base de dados. Uma UDF não pode ter instruções INSERT, UPDATE ou DELETE que afetem as tabelas ou vistas da BD. Contudo, no corpo de uma função, é possível criar uma tabela ou uma variável do tipo tabela e realizar as

instruções INSERT, UPDATE ou DELETE sobre estas tabelas. Uma função escalar pode incluir as todas as instruções necessárias no bloco BEGIN-END.

Para criar uma UDF escalar utilize a seguinte sintaxe:

```
CREATE FUNCTION [nome_do-esquema.]nome_da_função  
[declarações de parâmetros]  
RETURNS tipo_dados  
[WITH [ENCRYPTION] [,SCHEMABINDING] [, cláusula EXECUTE AS]]  
AS  
BEGIN  
    instruções_SQL  
    RETURN expressão_escalar  
END
```

Se o esquema não for especificado como parte do nome da UDF, a função será guardada no esquema associado ao utilizador corrente. A sintaxe usada para definir parâmetros é semelhante à usada na declaração dos mesmos em procedimentos armazenados. Todavia, nas funções, as declarações têm de ser efetuadas entre parêntesis.

Para invocar uma função que tem parâmetros é necessário passar todos os parâmetros e por posição. Os parâmetros opcionais apenas podem ser substituídos por DEFAULT. Por esta razão, é habitual declarar primeiro os parâmetros obrigatórios

A opção SCHEMABINDING liga a função ao esquema de BD. Isto impede que se apague ou altere tabelas ou vistas usadas pela função. Esta opção é usada principalmente por funções com valor tabela.

A opção *ENCRYPTION* impede os utilizadores de verem o código de uma função.

A opção *EXECUTE AS* permite que os utilizadores executem o procedimento armazenado com as permissões especificadas nesta cláusula.

Para criar uma UDF *simple table-valued* (ou *inline table-valued function*) utilize a seguinte sintaxe:

```
CREATE FUNCTION [nome_do-esquema.]nome_da_função
[declarações de parâmetros]
RETURNS TABLE
[WITH [ENCRYPTION] [,SCHEMABINDING]]
AS
    RETURN instrução_SELECT
```

Conforme se depreende da sintaxe, o resultado da função é definido na cláusula RETURN e só pode conter uma instrução.

## 1.2 Exemplo de uma UDF com valor tabela simples.

```
CREATE FUNCTION fnTopFornecedoresDivida
(@montanteRef money=0) -- parâmetro opcional inicializado a 0
RETURNS table
RETURN
SELECT Nome, SUM(TotalFactura) AS Total
FROM Facturas JOIN Fornecedores ON Fornecedor =IDFornecedor
WHERE TotalFactura - Pagamento - Crédito >0
GROUP BY Nome
HAVING SUM(TotalFactura) >= @montanteRef
```

```
-- invocação da função
select * from dbo.fnTopFornecedoresdivida(1000)
```

```
--resultado da invocação
```

| Results |              |         |
|---------|--------------|---------|
|         | Nome         | Total   |
| 1       | Mc Graw Hill | 3179,82 |
| 2       | Patinter     | 4239,74 |

A função devolve uma tabela com o total de todas as faturas pendentes (não completamente pagas) O parâmetro de entrada @montanteRef é opcional, pois tem um valor de 0 por omissão. Este parâmetro é usado na cláusula HAVING para devolver apenas os fornecedores cujos valores dos totais das faturas sejam maiores do que o valor especificado pelo parâmetro.

Seguidamente demonstra-se que é possível efetuar a junção de uma tabela com o resultado de uma UDF com valor tabela. Note-se que o uso do pseudónimo `FTop` evita a necessidade de repetir o nome da função nas cláusulas FROM e ON.

### 1.3 Exemplo de uso da UDF numa junção.

```
select f.Nome, f.localidade, Total
from Fornecedores f join dbo.fnTopFornecedoresDivida(DEFAULT) FTop
ON f.Nome=FTop.Nome
```

--resultado

|   | Nome         | localidade | Total   |
|---|--------------|------------|---------|
| 1 | Mc Graw Hill | Lisboa     | 3179,82 |
| 2 | Patinter     | Viseu      | 4239,74 |

Uma UDF como a exemplificada atua como uma vista dinâmica. Dado que uma função pode aceitar parâmetros, o resultado devolvido pode ser modificado em função destes parâmetros. Este aspeto corresponde a uma extensão poderosa da funcionalidade da SQL.

Para criar uma UDF *multi-statemente table-valued* utilize a seguinte sintaxe:

```
CREATE FUNCTION [nome_do-esquema.]nome_da_função  
[declarações de parâmetros]  
RETURNS @variável_tabela TABLE  
(Definição de coluna da tabela [, definição de outras colunas da tabela])  
[WITH [ENCRYPTION] [,SCHEMABINDING] [, cláusula EXECUTE AS]]  
[AS ]  
BEGIN  
    Instruções_sql  
    RETURN  
END
```

Uma vez que uma UDF *multi-statemente table-valued* cria uma nova tabela, é necessário definir a estrutura dessa tabela. Para esse efeito, declara-se uma variável do tipo tabela na cláusula RETURN e definem-se as suas colunas.

As instruções requeridas para criação do conteúdo da tabela são incluídas no bloco BEGIN-END. Este bloco acaba com a instrução RETURN que termina a função e devolve a variável tabela.

## 1.4 Exemplo de UDF multi-statemente table-valued.

```
CREATE FUNCTION fhAjustarCrédito (@MontanteREF money)
    RETURNS @TabelaRes table
        (IDFactura int,Fornecedor int,NúmeroFactura varchar(50),datafactura
        smalldatetime,
        TotalFactura money, Pagamento money, Crédito money)
BEGIN
    INSERT @TabelaRes
        SELECT IDFactura,Fornecedor,NúmeroFactura,datafactura , TotalFactura ,
        Pagamento , Crédito
        FROM Facturas
        WHERE TotalFactura - Pagamento - Crédito > 0
    WHILE (SELECT SUM(TotalFactura - Pagamento - Crédito)
        FROM @TabelaRes) >= @MontanteREF
        BEGIN
            UPDATE @TabelaRes
            SET Crédito = Crédito * .01
            WHERE TotalFactura - Pagamento - Crédito > 0
        END
    RETURN
END
```

--invocação

```
SELECT Nome, SUM(Crédito) CréditoPedido
FROM fornecedores f JOIN dbo.fhAjustarCrédito (50000) AS TabelaCréditos
ON f.IDfornecedor=TabelaCréditos.Fornecedor
GROUP BY Nome
```

--resultado

|   | Nome         | Crédito Pedido |
|---|--------------|----------------|
| 1 | Mc Graw Hill | 800,00         |
| 2 | Patinter     | 400,00         |

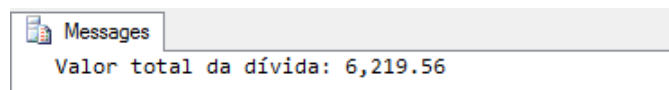
No exemplo de UDF multi-statemente table-valued, começa-se por definir a tabela a devolver. Esta tabela vai conter as faturas ainda por saldar, obtidas através do comando INSERT. Depois, a expressão de teste do ciclo WHILE inclui uma instrução SELECT para devolver o montante total em dívida das faturas por saldar. Caso esse montante seja superior ou igual ao valor do parâmetro @montanteRef, as restantes instruções são executadas. Caso contrário, o ciclo acaba. A instrução UPDATE englobada no ciclo atualiza o valor do campo crédito da tabela temporária em 1%, para as faturas ainda por saldar. Atenção que esta função pode ser demorada a executar, pois inclui muitas iterações.

## QUESTÕES

2 Implemente as seguintes instruções.

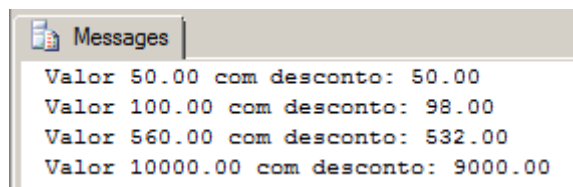
2.1 Crie uma UDF para determinar o montante total em dívida de todas as faturas pendentes.

Resultado



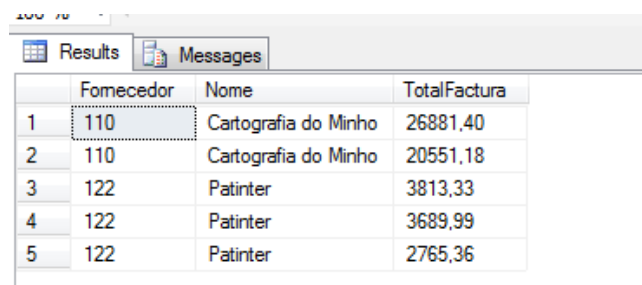
2.2 Crie uma UDF, designada por *fn\_AplicaDesconto*, que aplique um desconto sobre um dado valor monetário recebido por parâmetro. O valor de desconto a atribuir deve ser 2%, 5% e 10% para valores superiores ou iguais a 100€, 500€ e 5.000€ respetivamente.

Resultado



2.3 Crie uma UDF para determinar os n melhores fornecedores (Nome), i.e. os n fornecedores cujos montantes das faturas são maiores.

Resultado (para n=5)



|   | Fornecedor | Nome                 | TotalFactura |
|---|------------|----------------------|--------------|
| 1 | 110        | Cartografia do Minho | 26881,40     |
| 2 | 110        | Cartografia do Minho | 20551,18     |
| 3 | 122        | Patinter             | 3813,33      |
| 4 | 122        | Patinter             | 3689,99      |
| 5 | 122        | Patinter             | 2765,36      |



- 2.4 Crie uma UDF que apresente, para cada fornecedor registrado na base de dados, o seu nome, bem como a quantidade e o montante total de todas as suas faturas, mas apenas para fornecedores com uma quantidade de faturas maior ou igual a n.

Resultado (para n=1)

|   | nome                 | cont | soma     |
|---|----------------------|------|----------|
| 1 | Bell                 | 1    | 936,93   |
| 2 | Cartografia do Minho | 2    | 47432,58 |
| 3 | FCA                  | 1    | 600,00   |
| 4 | IBM                  | 2    | 1200,12  |
| 5 | Mc Graw Hill         | 3    | 3179,82  |
| 6 | Patinter             | 8    | 21126,37 |

Resultado (para n=3)

|   | nome         | cont | soma     |
|---|--------------|------|----------|
| 1 | Mc Graw Hill | 3    | 3179,82  |
| 2 | Patinter     | 8    | 21126,37 |

- 2.5 Crie uma UDF que determine o identificador do fornecedor e a média do montante das respetivas faturas, apenas para os fornecedores que têm os N maiores montantes médios das faturas. A função deve devolver o nome de cada um desses fornecedores, bem como a data da fatura da sua mais recente.

Resultado (para n=5)

|   | Fom | NomeF                | datamax             | med       |
|---|-----|----------------------|---------------------|-----------|
| 1 | 110 | Cartografia do Minho | 2006-04-16 00:00:00 | 23716,29  |
| 2 | 122 | Patinter             | 2006-04-25 00:00:00 | 2640,7962 |
| 3 | 121 | Mc Graw Hill         | 2006-04-30 00:00:00 | 1059,94   |
| 4 | 81  | Bell                 | 2006-04-16 00:00:00 | 936,93    |
| 5 | 34  | IBM                  | 2006-03-14 00:00:00 | 600,06    |

2.6 Por vezes surge a necessidade de separar informação que nos é disponibilizada somente como uma linha de texto. Neste sentido, implemente uma função capaz de receber uma frase (conjunto de palavras) e dividi-la através de um delimitador. O resultado (palavras) deve ser disponibilizado no formato tabela.

## Resultados

Frase: '91,12,65,78,56,789'

Delimitador: , (vírgula)

| Results |         |
|---------|---------|
|         | Palavra |
| 1       | 91      |
| 2       | 12      |
| 3       | 65      |
| 4       | 78      |
| 5       | 56      |
| 6       | 789     |

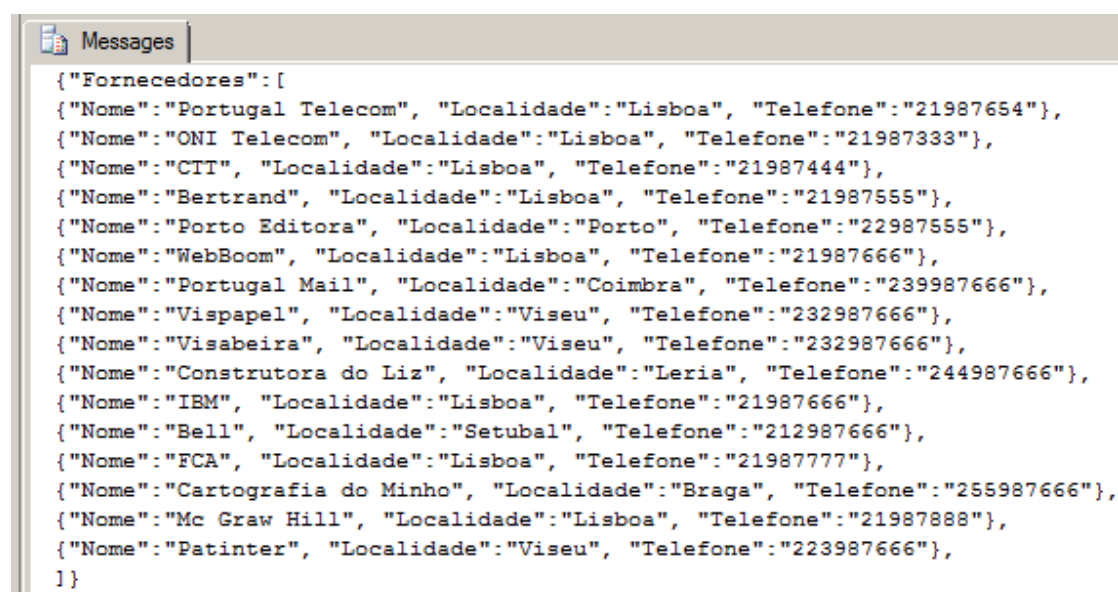
Frase: 'Olá, hoje está um dia de verão.'

Delimitador: (espaço)

| Results |         |
|---------|---------|
|         | Palavra |
| 1       | Olá,    |
| 2       | hoje    |
| 3       | está    |
| 4       | um      |
| 5       | dia     |
| 6       | de      |
| 7       | verão.  |

2.7 A diversidade de métodos e opções para a transferência de informação entre os Sistemas de Informação é enorme. Neste exercício, pretende-se criar uma UDF que, recorrendo à base de dados pagamentos, consiga gerar um resultado no formato JSON (<http://json.org>). Este formato é bastante utilizado em plataformas web, recorrendo à linguagem de programação JavaScript. A sua estrutura é simplista (“chave” : “valor”) como pode verificar na página em cima referenciada. Com este princípio, crie uma UDF que devolve o Nome, a Localidade e o Telefone de todos os fornecedores. Verifique o resultado abaixo apresentado.

## Resultado



```
{ "Fornecedores": [
  { "Nome": "Portugal Telecom", "Localidade": "Lisboa", "Telefone": "21987654" },
  { "Nome": "ONI Telecom", "Localidade": "Lisboa", "Telefone": "21987333" },
  { "Nome": "CTT", "Localidade": "Lisboa", "Telefone": "21987444" },
  { "Nome": "Bertrand", "Localidade": "Lisboa", "Telefone": "21987555" },
  { "Nome": "Porto Editora", "Localidade": "Porto", "Telefone": "22987555" },
  { "Nome": "WebBoom", "Localidade": "Lisboa", "Telefone": "21987666" },
  { "Nome": "Portugal Mail", "Localidade": "Coimbra", "Telefone": "239987666" },
  { "Nome": "Vispapel", "Localidade": "Viseu", "Telefone": "232987666" },
  { "Nome": "Visabeira", "Localidade": "Viseu", "Telefone": "232987666" },
  { "Nome": "Construtora do Liz", "Localidade": "Leria", "Telefone": "244987666" },
  { "Nome": "IBM", "Localidade": "Lisboa", "Telefone": "21987666" },
  { "Nome": "Bell", "Localidade": "Setubal", "Telefone": "212987666" },
  { "Nome": "FCA", "Localidade": "Lisboa", "Telefone": "21987777" },
  { "Nome": "Cartografia do Minho", "Localidade": "Braga", "Telefone": "255987666" },
  { "Nome": "Mc Graw Hill", "Localidade": "Lisboa", "Telefone": "21987888" },
  { "Nome": "Patinter", "Localidade": "Viseu", "Telefone": "223987666" },
] }
```