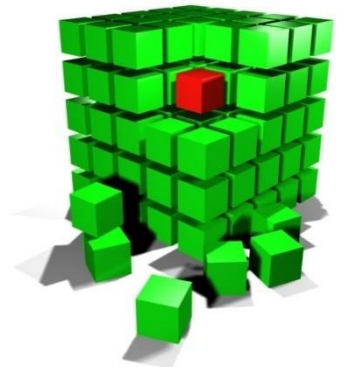
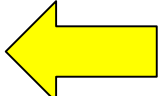


- **Aula 016 – Linguagem SQL**
 - Stored Procedures.
 - Functions.



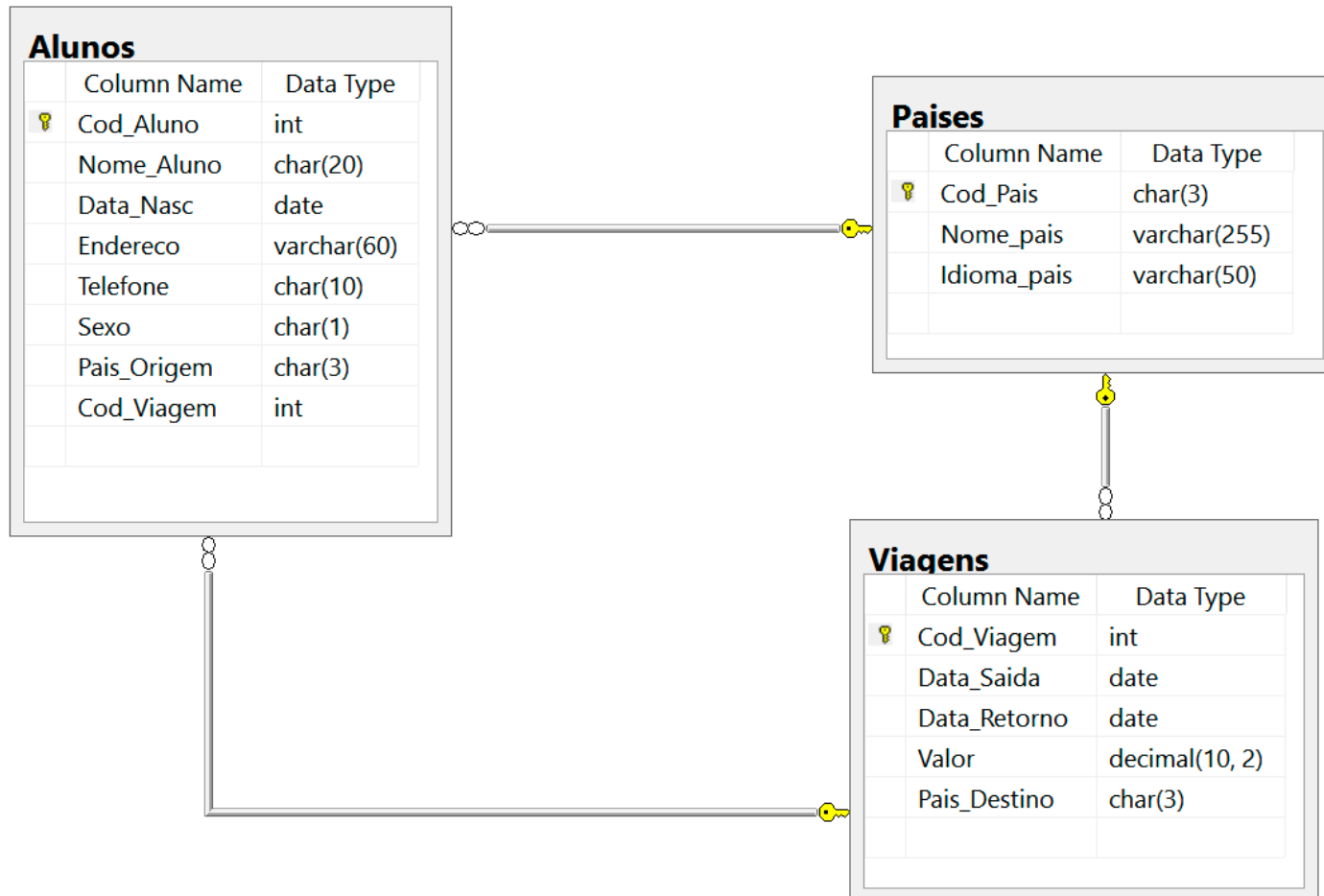
- **SQL procedural**
 - Declaração de variáveis.
 - Estruturas de decisão.
 - Estruturas de repetição.
 - Procedimentos armazenados. ←
 - Funções. ←
 - Gatilhos.
 - Cursores.



- **Banco de dados INTERCAMBIO**
 - Para esta aula, continuaremos a utilizar o banco de dados **INTERCAMBIO**. Este banco contém dados de alunos, de países e das viagens que eles realizam para esses países.
 - **01_INTERCAMBIO_Cria_Banco.sql** 



■ Banco de dados INTERCAMBIO



■ Stored Procedures

- Procedimentos armazenados, ou **stored procedures**, são blocos de comando que têm como objetivo executar uma série de comandos SQL no servidor de banco de dados.
- Eles têm como característica básica o fato de serem armazenados no servidor sendo, portanto, muito mais rápido execução das tarefas.
- A sua utilização facilita a escrita de programas diretamente no servidor de banco de dados, ao invés de se utilizar outro tipo de aplicativo.



■ Stored Procedures

- Um procedimento armazenado é um conjunto de instruções que realiza uma determinada tarefa. A definição e o funcionamento dos procedimentos são similares à programação em outras linguagens de programação e envolvem basicamente os seguintes passos:
 1. Identificação do procedimento;
 2. Definição do parâmetro;
 3. Conjunto de instruções do procedimento;
 4. Submissão do código ao SGBD.



- **Vantagens de um Stored Procedure**
 - Podem receber parâmetros de entrada e retornar resultados, através de parâmetros de saída.
 - Reduzem o tráfego de rede gerado pela aplicação.
 - Melhoram a velocidade de execução de consultas e a geração de relatórios.
 - Facilitam e centralizam o gerenciamento de permissões.
 - Ocultam a complexidade de acesso ao banco de dados.



■ Tipos de Stored Procedures em T-SQL

- **User-defined stored procedures** – são criadas nos bancos de dados do usuário. São utilizadas para a execução de tarefas repetitivas.
- **Temporary stored procedures** – são criadas e mantidas pelo SQL Server, no banco de dados **tempdb**. Normalmente, são associadas a tarefas de manutenção e gerenciamento de conexões. Podem ser locais (#) ou globais (##).
- **System stored procedures** – são criadas durante a instalação do SQL Server e ficam gravadas no banco de dados **master**. Iniciam com **sp_**. São utilizadas para uma série de tarefas de manutenção e gerenciamento de diversos objetos.
- **Extended stored procedures** – executam funções externas ao servidor SQL Server. Iniciam com **xp_**.



■ Criação de Stored Procedures

- A criação de um procedimento armazenado consiste basicamente em se definir um conjunto de comandos e salvá-lo no banco de dados, com um nome específico.
- No **Microsoft SQL Server**, o procedimento armazenado pode ser criado por meio do SQL Management Studio ou utilizando a linguagem T-SQL, através do comando **CREATE PROCEDURE**.



■ Criação de Stored Procedures

- Em seu conjunto de comandos podemos referenciar tabelas, **views**, outros stored procedures e tabelas temporárias. Podemos incluir qualquer comando T-SQL, com exceção dos seguintes: **CREATE PROCEDURE**, **CREATE DEFAULT**, **CREATE RULE**, **CREATE TRIGGER** e **CREATE VIEW**.
- Para utilizar um procedimento armazenado, utilizamos o comando **EXECUTE**, ou **EXEC**.



■ Criação de Stored Procedures

- **CREATE PROCEDURE** – cria um stored procedure.
- **EXECUTE** ou **EXEC** – executa um stored procedure.
- **ALTER PROCEDURE** – permite alterar um stored procedure criado anteriormente.
- **DROP PROCEDURE** – exclui um stored procedure.



■ Sintaxe de um Stored Procedure

```
CREATE PROCEDURE nome_stored_procedure ←  
[  
    {@nome_parâmetro1 tipo_dados_do_parâmetro1}  
    [ = valor_default] [OUTPUT]  
]  
[, ... n]  
AS  
comando 1  
...  
comando n } ← Bloco de comandos em T-SQL
```



■ Criação de Stored Procedures



A seguir, veremos como criar Stored Procedures dentro do banco de dados INTERCAMBIO, utilizando a linguagem T-SQL.



■ Criação de Stored Procedures

-- Habilita o banco de dados INTERCAMBIO

USE INTERCAMBIO ←

GO

-- Lista o nome das tabelas disponíveis

SELECT name

FROM sys.tables

GO

→ {
name

Países
Viagens
Alunos
AlunosCOPIA
AERONAVES
VEICULOS

(6 row(s) affected)



■ Criação de Stored Procedures

- Exibe informações sobre os stored
- procedures do banco de dados em uso

```
SELECT * FROM sys.procedures
```

```
GO
```



■ Criação de Stored Procedures

-- Criando um stored procedure que exibe "Alô mundo!"

CREATE PROCEDURE AloMundo ←

AS

PRINT 'Alô Mundo!'

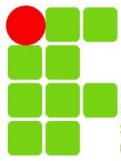
GO

-- Utiliza o stored procedure ALOMUNDO.

EXECUTE AloMundo ←

GO





■ Criação de Stored Procedures

-- Cria um stored procedure que seleciona os dados dos alunos do sexo masculino

CREATE PROCEDURE usp_AlunosMasculinos ←

AS

```
SELECT Cod_Aluno      AS 'Código do Aluno',  
       Nome_Aluno     AS 'Nome do Aluno',  
       Data_Nasc       AS 'Data de Nascimento',  
       Endereco        AS 'Endereço',  
       Telefone,  
       Sexo,  
       Pais_Origem     AS 'Nacionalidade',  
       Cod_Viagem      AS 'Código da Viagem'
```

FROM Alunos

WHERE Sexo = 'M'



■ Criação de Stored Procedures

-- Utiliza o stored procedure USP_ALUNOSMASCULINOS

EXEC usp_AlunosMasculinos

GO

Código do Aluno	Nome do Aluno	Data de Nascimento	Endereço	Telefone	Sexo	Nacionalidade	Código da Viagem
3	Carlos Renato	1979-05-10	Av. Faria Lima, 347 - São Paulo	1156121010	M	BRA	3
4	Hugo Silva	1975-10-02	Av. da Consolação, 1216 - São Paulo	1154788901	M	BRA	4
5	Marcos Antônio	1985-10-23	Rua Agripino Lopes, 100 - São Paulo	1156010201	M	BRA	5
7	Antônio Pereira	1979-06-18	Rua Joaquim Nabuco, 18 - Jacareí	1234966852	M	BRA	7
8	Jair Lopes	1982-12-14	Rua Pedro XIII, Santa Isabel	1236691857	M	BRA	8
9	Miguel Firmino	1982-05-12	Av. Colinas, 2340 - São José dos Campos	1234581212	M	BRA	9
39	Leandro Leite	1982-06-15	Av. do Povo, 3489 - Rio de Janeiro	2156894510	M	BRA	9
42	Guilherme dos Santos	1978-10-11	Av. Brasil, 1008 - São Paulo	1155661010	M	BRA	1
44	Amílcar Júnior	1976-04-10	Rua Senador Kennedy, 901 - São Paulo	1155640290	M	BRA	8
45	Alexandro Duarte	1974-05-29	Rua Maria Cíntia - Cruzeiro	1236661515	M	BRA	11
46	Maurício dos Santos	1988-04-13	Rua Marta Silva - Jacareí	1236665650	M	BRA	16
49	André César	1984-12-30	Rua Militar, 349 - Rio de Janeiro	2128002525	M	BRA	14
50	Edson Lopes	1979-01-11	Av. Pedro Silva, 3047 - Rio de Janeiro	2126900099	M	BRA	14

(28 row(s) affected)



■ Criação de Stored Procedures

- Exibe o nome, a data de criação e a data
- de modificação dos stored procedures

```
SELECT name          AS 'Procedure',  
        create_date  AS 'Data de Criação',  
        modify_date  AS 'Data de Alteração'  
  
FROM sys.procedures  
  
GO
```



■ Criação de Stored Procedures

Procedure	Data de Criação		Data de Alteração	
-----	-----		-----	
AloMundo	2018-09-09	11:27:00.507	2018-09-09	11:27:00.507
usp_AlunosMascullinos	2018-09-09	11:35:49.477	2018-09-09	11:35:49.477

(2 row(s) affected)



■ Modifica um Stored Procedure

-- Modifica o stored procedure ALOMUNDO -> ALTER

ALTER PROCEDURE AloMundo ←

AS

PRINT 'Hello World!' ←

GO

-- Utiliza o stored procedure ALOMUNDO.

EXECUTE AloMundo

GO



■ Extended Stored Procedures (T-SQL)

-- Exemplo de uso de um stored procedure do sistema

-- No caso, XP_SUBDIRS exibe o conteúdo do drive C:

```
EXEC XP_SUBDIRS 'C:\'
```

```
GO
```

-- No caso, XP_FILEEXIST verifica a existência de um arquivo

```
EXEC XP_FILEEXIST 'E:\01_INTERCAMBIO_Cria_Banco.sql'
```

```
GO
```

-- Exibe uma lista com todos os stored procedures do sistema

-- EXEC master.dbo.sp_helpextendedproc ←

```
EXEC SP_HELPEXTENDEDPROC
```

```
GO
```



■ Extended Stored Procedures (T-SQL)

name	dll
sp_AddFunctionalUnitToComponent	(server internal)
sp_audit_write	(server internal)
sp_availability_group_command_internal	(server internal)
sp_begin_parallel_nested_tran	(server internal)
sp_bindsession	(server internal)
sp_change_tracking_waitforchanges	(server internal)
.....
xp_sqlagent_notify	xpstar.dll
xp_sqlagent_param	xpstar.dll
xp_sqlmaint	xpstar.dll
xp_sscanf	xplog70.dll
xp_subdirs	xpstar.dll
xp_sysmail_activate	xpstar.dll
xp_sysmail_attachment_load	xpstar.dll
xp_sysmail_format_query	xpstar.dll

**Não exibiu o total
de linhas afetadas!**



■ Stored Procedures com parâmetros

- Cria um stored procedure que exibe uma saudação
- para o nome passado como parâmetro

```
CREATE PROCEDURE usp_saudacao
```

```
    @nome VARCHAR(200) ←
```

```
AS
```

```
    PRINT 'Olá, ' + @nome + '!' ←
```

```
GO
```

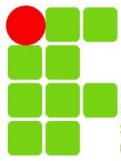
- Utiliza a stored procedure USP_SAUDACAO

```
EXEC usp_saudacao 'Paulo'
```

```
EXEC usp_saudacao 'Cris'
```

```
GO
```





■ Stored Procedures com parâmetros

- Cria um stored procedure que calcula a soma de duas
- variáveis passadas como parâmetro. Esse stored procedure
- retorna um valor através da variável 'soma', em OUTPUT.

```
CREATE PROCEDURE usp_soma
```

```
    @valor1 INT,
```

```
    @valor2 INT,
```

```
    @soma INT OUTPUT ←
```

```
AS
```

```
    SET @soma = @valor1 + @valor2 ←
```

```
GO
```



■ Stored Procedures com parâmetros

-- Utiliza a stored procedure USP_SOMA

DECLARE @saida INT

EXEC usp_soma 5, 3, @saida OUTPUT

PRINT @saida

GO



Selecione todas
as linhas

-- Utiliza a stored procedure USP_SOMA

DECLARE @saida INT

EXEC usp_soma 100, 50, @saida OUTPUT

PRINT @saida

GO



■ Stored Procedures com parâmetros

-- Cria um stored procedure para descobrir o código do país passado como
-- parâmetro. Utiliza SET NOCOUNT para esconder as linhas tornadas pela consulta.

```
CREATE PROCEDURE usp_DescobreCodigoPais
```

```
    @Pais VARCHAR(255)
```

```
AS
```

```
    SET NOCOUNT ON ←
```

```
    SELECT    Cod_Pais    AS 'Código',
```

```
            Nome_pais AS 'País'
```

```
    FROM Pais
```

```
    WHERE Nome_pais = @Pais
```

```
    SET NOCOUNT OFF ←
```

```
GO
```



■ Stored Procedures com parâmetros

-- Utiliza a stored procedure USP_DESCOBRECODIGOPAIS

EXEC usp_DescobreCodigoPaís 'Brasil'

EXEC usp_DescobreCodigoPaís 'Turquia'

EXEC usp_DescobreCodigoPaís 'Japão'

GO

Código País

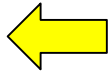
BRA Brasil

Código País

TUR Turquia

Código País

JPN Japão



■ Stored Procedures com parâmetros

- Cria um stored procedure que fornece informações
- sobre os países cujo idioma faz parte do
- parâmetro passado

```
CREATE PROCEDURE usp_InfoIdiomasPaíses  
    @idioma VARCHAR(50)
```

```
AS
```

```
    SET NOCOUNT ON
```

```
    SELECT * FROM Países
```

```
    WHERE Idioma_pais LIKE ('%' + @idioma + '%')
```

```
    SET NOCOUNT OFF
```



■ Stored Procedures com parâmetros

-- Utiliza a stored procedure USP_INFOIDIOMASPAISES

EXEC usp_InfoIdiomasPaises 'Português' ←

EXEC usp_InfoIdiomasPaises 'hin'

GO

Cod_Pais	Nome_pais	Idioma_pais
---	-----	-----
AGO	Angola	Português
BRA	Brasil	Português
CPV	Cabo Verde	Português
GNB	Guiné-Bissau	Português
MAC	Macau	Chinês e Português
MOZ	Moçambique	Português
PRT	Portugal	Português
STP	São Tomé e Príncipe	Português
TLS	Timor-Leste	Português



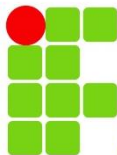
■ Stored Procedures com parâmetros

-- Cria um stored procedure que aceita um trecho do nome do aluno como parâmetro de entrada. Ele busca
-- informações de todas as tabelas e exibe as informações das viagens cadastradas para esses alunos.

```
CREATE PROCEDURE usp_BuscaDadosAlunos
    @nomeAluno AS VARCHAR(20)
AS
    SET NOCOUNT ON
    SELECT
        Viagens.Cod_Viagem AS 'Código da Viagem',
        Alunos.Nome_Aluno AS 'Nome',
        Alunos.Telefone,
        Alunos.Sexo,
        (SELECT Nome_pais FROM Paises WHERE Cod_Pais = Alunos.Pais_Origem) AS 'Origem',
        (SELECT Nome_pais FROM Paises WHERE Cod_Pais = Viagens.Pais_Destino) AS 'Destino',
        Viagens.Data_Saida AS 'Data de Saída',
        Viagens.Data_Retorno AS 'Data de Retorno',
        Viagens.Valor AS 'Preço da Viagem R$'
    FROM Alunos INNER JOIN Viagens
        ON Alunos.Cod_Viagem = Viagens.Cod_Viagem
    WHERE Alunos.Nome_Aluno LIKE '%' + @nomealuno + '%'
    ORDER BY Alunos.Nome_Aluno, Viagens.Pais_Destino
    SET NOCOUNT OFF
```

GO





■ Stored Procedures com parâmetros

- Cria um stored procedure que aceita um trecho do
- nome do aluno como parâmetro de entrada. Ele busca
- informações de todas as tabelas e exibe as informações
- das viagens cadastradas para esses alunos.

```
CREATE PROCEDURE usp_BuscaDadosAlunos
```

```
    @nomeAluno AS VARCHAR(20)
```

```
AS
```

```
    SET NOCOUNT ON
```

```
    SELECT
```

```
        Viagens.Cod_Viagem AS 'Código da Viagem',
```

```
        Alunos.Nome_Aluno AS 'Nome',
```

```
        Alunos.Telefone,
```



■ Stored Procedures com parâmetros

```
Alunos.Sexo,  
(SELECT Nome_pais FROM Países WHERE Cod_Pais =  
Alunos.Pais_Origem) AS 'Origem',  
(SELECT Nome_pais FROM Países WHERE Cod_Pais =  
Viagens.Pais_Destino) AS 'Destino',  
Viagens.Data_Saida AS 'Data de Saída',  
Viagens.Data_Retorno AS 'Data de Retorno',  
Viagens.Valor AS 'Preço da Viagem R$'  
FROM Alunos INNER JOIN Viagens  
ON Alunos.Cod_Viagem = Viagens.Cod_Viagem
```



■ Stored Procedures com parâmetros

```
WHERE Alunos.Nome_Aluno LIKE '%' + @nomealuno +  
'%'  
  
ORDER BY Alunos.Nome_Aluno,  
          Viagens.Pais_Destino  
  
SET NOCOUNT OFF  
  
GO
```



■ Stored Procedures com parâmetros

-- Utiliza a stored procedure SP_BUSCADADOSALUNOS

```
EXEC usp_BuscaDadosAlunos 'P'
```

```
GO
```

```
EXEC usp_BuscaDadosAlunos 'Ana'
```

```
GO
```

```
EXEC usp_BuscaDadosAlunos 'Ana Mara'
```

```
GO
```

```
EXEC usp_BuscaDadosAlunos 'Silva'
```

```
GO
```

```
EXEC usp_BuscaDadosAlunos 'Luís' ← Não retornou ninguém!
```

```
GO
```



■ Stored Procedures com parâmetros

EXEC usp_BuscaDadosAlunos 'Silva'

GO

Código da Viagem	Nome	Telefone	Sexo	Origem	Destino	Data de Saída	Data de Retorno	Preço da Viagem R\$
15	Ágatha Silva	1156020303	F	Brasil	Mauritânia	2010-11-04	2010-12-30	6250.50
15	Ana Mara da Silva	1236658956	F	Brasil	Mauritânia	2010-11-04	2010-12-30	6250.50
6	Gislaine Silva	1234598910	F	Brasil	Estados Unidos da América	2010-03-16	2011-03-16	14255.35
4	Hugo Silva	1154788901	M	Brasil	Turquia	2010-08-11	2011-05-21	12350.25
10	Marta da Silva	1234568989	F	Brasil	Senegal	2010-09-10	2010-10-02	6525.00
9	Mônica Silva	2189562121	F	Brasil	Rússia	2010-06-21	2010-09-05	9250.00
11	Patrícia Silva	1236620181	F	Brasil	Qatar	2010-07-27	2011-11-15	12355.00

EXEC usp_BuscaDadosAlunos 'Luís'

GO

Código da Viagem	Nome	Telefone	Sexo	Origem	Destino	Data de Saída	Data de Retorno	Preço da Viagem R\$



■ Stored Procedures com parâmetros

-- Exibe informações sobre o stored procedure USP_BUSCADADOSALUNOS.

```
EXEC SP_HELP usp_BuscaDadosAlunos
```

```
GO
```

-- Exibe os comandos do stored procedure passado como parâmetro.

-- Para isso, o stored procedure não deve ter sido criado com o

-- parâmetro WITH ENCRYPTION.

```
EXEC SP_HELPTEXT usp_BuscaDadosAlunos
```

```
GO
```

-- Mostra uma listagem dos stored procedures existentes no banco de
-- dados atual.

```
EXEC SP_STORED_PROCEDURES
```

```
GO
```



■ Stored Procedures com decisão

-- Exemplo de uso de IF-ELSE: criação de um stored procedure que retorna o
-- maior de dois valores passados como parâmetros.

```
CREATE PROCEDURE usp_AchaMaior
```

```
    @valor1 FLOAT,
```

```
    @valor2 FLOAT
```

```
AS
```

```
    DECLARE @maior FLOAT
```

```
    IF (@valor1 > @valor2)
```

```
        SET @maior = @valor1
```

```
    ELSE
```

```
        SET @maior = @valor2
```

```
    PRINT 'Maior valor entre ' + CAST(@valor1 AS VARCHAR) +
```

```
        ' e ' + CAST(@valor2 AS VARCHAR) +
```

```
        ' é: ' + CAST(@maior AS VARCHAR)
```



■ Stored Procedures com decisão

-- Utiliza a stored procedure USP_ACHAMAIOR

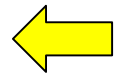
```
EXEC usp_AchaMaior 5, 8
```

```
EXEC usp_AchaMaior 1.356, 8.6352
```

```
GO
```

Maior valor entre 5 e 8 é: 8

Maior valor entre 1.356 e 8.6352 é: 8.6352



■ Stored Procedures com decisão

-- Cria um stored procedure que informa a quantidade de países que possuem o idioma passado
-- como parâmetro.

```
CREATE PROCEDURE usp_ContaIdiomas
    @idioma VARCHAR(50)
AS
    DECLARE @mensagemOk VARCHAR(100)
    DECLARE @mensagemErro VARCHAR(100)
    DECLARE @total INT

    SET @mensagemOK = 'Quantidade de registros encontrados para o idioma ' + @idioma + ': '
    SET @mensagemErro = 'Erro: nenhuma ocorrência foi encontrada para o idioma ' + @idioma +
    '!'

    SET @total = (SELECT COUNT(*) FROM Paises WHERE Idioma_pais LIKE ('%' + @idioma + '%'))

    IF (@total > 0)
        PRINT @mensagemOK + CAST(@total AS VARCHAR)
    ELSE
        PRINT @mensagemErro

GO
```



■ Stored Procedures com decisão

- Cria um stored procedure que informa a
- quantidade de países que possuem o idioma
- passado como parâmetro.

```
CREATE PROCEDURE usp_ContaIdiomas  
    @idioma VARCHAR(50)
```

```
AS
```

```
    DECLARE @mensagemOk VARCHAR(100)  
    DECLARE @mensagemErro VARCHAR(100)  
    DECLARE @total INT
```



■ Stored Procedures com decisão

```
SET @mensagemOK = 'Quantidade de registros  
encontrados para o idioma ' + @idioma + ': '
```

```
SET @mensagemErro = 'Erro: nenhuma ocorrência foi  
encontrada para o idioma ' + @idioma + '!'
```

```
SET @total = (SELECT COUNT(*) FROM Países WHERE  
Idioma_pais LIKE ('%' + @idioma + '%'))
```

```
IF (@total > 0)
```

```
    PRINT @mensagemOK + CAST(@total AS VARCHAR)
```

```
ELSE
```

```
    PRINT @mensagemErro
```



■ Stored Procedures com decisão

-- Utiliza a stored procedure USP_CONTAIDIOMAS

EXEC usp_ContaIdiomas 'Inglês'

EXEC usp_ContaIdiomas 'Japonês'

EXEC usp_ContaIdiomas 'Malgaxe'

EXEC usp_ContaIdiomas 'Americano'

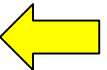
GO

Quantidade de registros encontrados para o idioma Inglês: 81

Quantidade de registros encontrados para o idioma Japonês: 2

Quantidade de registros encontrados para o idioma Malgaxe: 1

Erro: nenhuma ocorrência foi encontrada para o idioma Americano!



■ Stored Procedures com decisão

-- Cria um stored procedure que calcula a idade do aluno passado como parâmetro.

```
CREATE PROCEDURE usp_CalculaIdadeAluno
    @nome AS CHAR(20)
AS
    SET NOCOUNT ON
    DECLARE @data_nascimento DATETIME
    DECLARE @idade INT
    DECLARE @data_atual DATETIME

    SET @data_atual = (SELECT GETDATE())
    SET @data_nascimento = (SELECT Data_Nasc FROM Alunos WHERE Nome_Aluno = @nome)

    SET @idade = DATEDIFF(YEAR, @data_nascimento, @data_atual) - CASE
        WHEN @data_atual < DATEADD(YEAR, DATEDIFF(YEAR, @data_nascimento, @data_atual),
@data_nascimento)
            THEN 1
            ELSE 0
    END

    SELECT @nome AS 'Nome do Aluno',
        @idade AS 'Idade do Aluno'

    SET NOCOUNT OFF

GO
```



■ Stored Procedures com decisão

- Cria um stored procedure que calcula a idade do
- aluno passado como parâmetro.

```
CREATE PROCEDURE usp_CalculaIdadeAluno
    @nome AS CHAR(20)
AS
    SET NOCOUNT ON
    DECLARE @data_nascimento DATETIME
    DECLARE @idade INT
    DECLARE @data_atual DATETIME

    SET @data_atual = (SELECT GETDATE())
```



■ Stored Procedures com decisão

```
SET @data_nascimento = (SELECT Data_Nasc FROM  
Alunos WHERE Nome_Aluno = @nome)
```

```
SET @idade = DATEDIFF(YEAR, @data_nascimento,  
@data_atual) - CASE
```

```
    WHEN @data_atual < DATEADD(YEAR,  
DATEDIFF(YEAR, @data_nascimento, @data_atual),  
@data_nascimento)
```

```
    THEN 1
```

```
    ELSE 0
```

```
END
```



■ Stored Procedures com decisão

```
SELECT @nome AS 'Nome do Aluno',  
       @idade AS 'Idade do Aluno'
```

```
SET NOCOUNT OFF
```

```
GO
```



■ Stored Procedures com decisão

-- Utiliza a stored procedure USP_CALCULAIDADEALUNO

EXEC usp_CalculaIdadeAluno 'Maria Cristina'

EXEC usp_CalculaIdadeAluno 'Jair Lopes'

EXEC usp_CalculaIdadeAluno 'Miguel Firmino' ←

GO

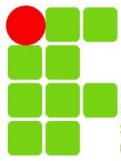
Nome do Aluno	Idade do Aluno
---------------	----------------

Maria Cristina	32
----------------	----

Nome do Aluno	Idade do Aluno
---------------	----------------

Jair Lopes	35
------------	----





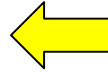
■ Stored Procedures que retornam valor

- Cria um stored procedure que informa o valor médio das
- viagens para o país passado como parâmetro

```
CREATE PROCEDURE usp_MediaViagens
```

```
    @pais VARCHAR(50),
```

```
    @media MONEY OUTPUT
```



```
AS
```

```
    SET @media = (SELECT AVG( Valor) AS 'Média dos  
Valores'
```

```
    FROM Viagens
```

```
    WHERE Pais_Destino = (SELECT Cod_Pais
```

```
        FROM Países
```

```
        WHERE Nome_pais = @pais))
```



■ Stored Procedures que retornam valor

```
-- Utiliza a stored procedure USP_MEDIAVIAGENS  
DECLARE @pais VARCHAR(50) = 'Estados Unidos da América'  
DECLARE @resultado MONEY ←  
EXEC usp_MediaViagens @pais, @resultado OUTPUT ←  
PRINT 'Custo médio das viagens realizadas para o ' +  
@pais + ': R$ ' + CAST(@resultado AS VARCHAR)  
GO
```

Custo médio das viagens realizadas para o Estados Unidos da América: R\$ 16016.13



■ Removendo um Stored Procedure

-- Exclui o stored procedure USP_SAUDACAO -> DROP

DROP PROCEDURE usp_saudacao ←

GO

-- Tenta utilizar um Stored Procedure que não existe

EXEC usp_saudacao

GO

Could not find stored procedure 'usp_saudacao'.



■ Removendo um Stored Procedure

- Exibe o nome, a data de criação e a data
- de modificação dos stored procedures

```
SELECT name          AS 'Procedure',  
        create_date  AS 'Data de Criação',  
        modify_date  AS 'Data de Alteração'  
  
FROM sys.procedures  
  
GO
```



■ Removendo um Stored Procedure

Procedure	Data de Criação		Data de Alteração	
-----	-----		-----	
AloMundo	2018-09-09	11:27:00.507	2018-09-09	17:30:40.280
usp_AlunosMascullinos	2018-09-09	11:35:49.477	2018-09-09	11:35:49.477
usp_soma	2018-09-09	13:31:26.690	2018-09-09	13:31:26.690
usp_DescobreCodigoPais	2018-09-09	13:47:31.257	2018-09-09	13:47:31.257
usp_InfoIdiomasPaises	2018-09-09	14:03:38.213	2018-09-09	14:03:38.213
usp_BuscaDadosAlunos	2018-09-09	14:12:56.460	2018-09-09	14:12:56.460
usp_AchaMaior	2018-09-09	16:22:18.367	2018-09-09	16:25:20.193
usp_ContaIdiomas	2018-09-09	16:31:48.250	2018-09-09	16:41:36.220
usp_CalculaIdadeAluno	2018-09-09	16:45:56.120	2018-09-09	16:47:03.180
usp_MediaViagens	2018-09-09	17:03:59.953	2018-09-09	17:03:59.953

(10 row(s) affected)



■ Exibe informações dos Stored Procedures

-- Exibe informações sobre os Stored Procedures
-- do banco de dados em uso

```
SELECT name          AS 'Stored Procedure',  
       definition    AS 'Definição',  
       type_desc     AS 'Tipo'  
FROM sys.sql_modules M INNER JOIN sys.objects O  
     ON M.object_id = O.object_id  
WHERE type_desc LIKE '%procedure%'  
GO
```



■ User Defined Functions

- Funções, ou **functions** (UDF), são blocos de comando que têm como objetivo executar uma série de comandos SQL no servidor de banco de dados.
- As funções são semelhantes aos stored procedures, porém, com a diferença de que é necessário o retorno de um valor por meio da cláusula **RETURN**.
- Os stored procedures são chamadas através do comando **EXECUTE**, enquanto que as funções são chamadas em geral no comando **SELECT** ou em comandos de atribuição, e é justamente daí que vem o seu poder.



■ User Defined Functions

- As UDFs formam um recurso muito poderoso dentro do SQL Server, mas deve-se tomar bastante cuidado com elas.
- A utilização de funções escalares em cláusulas **WHERE** ou **ORDER BY** provavelmente reduzirá o desempenho do comando **SELECT**.
- Além disso, as funções de tabela de muitos comandos normalmente não são indexadas de forma nenhuma, o que também pode reduzir o desempenho.



■ User Defined Functions

- Existem três tipos de UDFs no SQL Server:
 - **Scalar functions**: funções escalares.
 - **Inline table-valued functions**: funções de tabela com um comando.
 - **Multi-statement table-valued functions**: funções de tabela com vários comandos.



■ UDF Escalar

- Uma **UDF escalar** é semelhante ao que chamamos de função, na maioria das linguagens de programação.
- Os dados retornados devem ser do tipo escalar (string, número ou data).
- Os dados retornados não podem ser uma tabela ou um cursor.
- Uma UDF escalar pode ser utilizada em qualquer lugar em que uma função predefinida do SQL Server é permitida, e também deve ser **determinística**.



■ UDF Escalar

- Uma função determinística não pode chamar funções que retornam valores diferentes toda vez que uma é chamada (**DATE** ou **TIME**, por exemplo). Também não pode alterar o banco de dados.
- Pode ser utilizada com mais liberdade em uma consulta T-SQL.
- Todas as UDFs de T-SQL devem ser determinísticas.
- A maioria de funções internas do SQL Server são funções escalares, tais como **CONVERT**, **LEN**, **REPLICATE**, etc.



■ Sintaxe de uma UDF Escalar

```
CREATE FUNCTION [proprietario.]nome_função  
    ([@nome_parâmetro tipo_escalar_do_parâmetro  
    [ = default] [, ...])  
RETURNS tipo_escalar_de_retorno ←  
[WITH <opções_da_função>]  
[AS]  
    BEGIN  
        corpo_função  
    RETURN expressão_escalar ←  
END
```



■ UDF Inline

- Uma UDF inline funciona de forma muito parecida com uma visão (**VIEW**).
- Elas retornam o resultado de um único comando **SELECT**, com a diferença que podem aceitar parâmetros.
- Contém algumas restrições, como por exemplo, o fato de não pode alterar o estado do banco de dados.



■ Sintaxe de uma UDF Inline

```
CREATE FUNCTION [proprietario.]nome_função  
    ([@nome_parâmetro  
tipo_escalar_do_parâmetro [ = default] [,  
...])  
RETURNS TABLE ←  
[WITH <opções_da_função>]  
[AS]  
    RETURN expressão_de_consulta ←
```



■ UDF Multi-Statement

- Permitem a criação de códigos mais complexos, retornando tabelas temporárias (um conjunto de valores).
- Uma UDF que retorna uma tabela é uma alternativa excelente a uma visão porque aceita parâmetros e pode conter várias instruções complexas, enquanto uma visão só pode conter uma única instrução SQL Server.
- A UDF que retorna uma tabela pode ser utilizada como uma tabela.



■ Sintaxe de uma UDF Multi-Statement

```
CREATE FUNCTION [proprietario.]nome_função  
    ([@nome_parâmetro tipo_escalar_do_parâmetro [ =  
default] [, ...]])  
RETURNS @nome_da_variável TABLE ←  
    (definição_da_tabela)  
[WITH <opções_da_função>]  
[AS]  
    BEGIN  
        Corpo_da_função } ←  
    RETURN  
END
```



■ Diferenças entre Procedures e Functions

Stored Procedures

Não podem ser utilizadas como uma função dentro de uma consulta.

Podem alterar o banco de dados.

Não têm um tipo de retorno explícito, embora possam retornar valores e tabelas.

Chamadas sem parênteses.

Podem chamar qualquer função.

Functions

Podem ser utilizadas como uma função dentro de uma consulta.

Não podem alterar o banco de dados.

Têm um tipo de retorno explícito pela cláusula **RETURN**.

Chamadas com parênteses.

Devem ser determinísticas, sempre retornar o mesmo valor para um dado conjunto de parâmetros, e não devem ter efeitos colaterais.

■ Criação de User Defined Functions



A seguir, veremos como criar User Defined Functions dentro do banco de dados INTERCAMBIO, utilizando a linguagem T-SQL.



■ Criação de UDF escalar

-- Cálculo de fatorial (SQL Server Express Edition - Augusto Manzano)

```
CREATE FUNCTION fatorial (@N INT)
RETURNS BIGINT
AS
BEGIN
    DECLARE @fator BIGINT,
            @i INT
    SET @fator = 1
    SET @i = 1
    IF (@N <= 1)
        RETURN @fator
    ELSE
        WHILE (@i <= @N)
            BEGIN
                SET @fator = @fator * @i
                SET @i = @i + 1
            END
        RETURN @fator
    END
```

GO

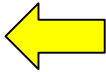


■ Criação de UDF escalar

-- Cálculo de fatorial (SQL Server Express
-- Edition - Augusto Manzano)

```
CREATE FUNCTION fatorial (@N INT)
```

```
    RETURNS BIGINT
```



```
AS
```

```
BEGIN
```

```
    DECLARE @fator BIGINT,
```

```
           @i INT
```

```
    SET @fator = 1
```

```
    SET @i = 1
```



■ Criação de UDF escalar


```
IF (@N <= 1)
    RETURN @fator
ELSE
    WHILE (@I <= @N)
        BEGIN
            SET @fator = @fator * @i
            SET @i = @i + 1
        END
    RETURN @fator ←
END
```



■ Criação de UDF escalar

-- Utilização da função FATORIAL

```
SELECT dbo.fatorial(20) AS 'Fatorial de 20',  
       dbo.fatorial(5)  AS 'Fatorial de 5'
```



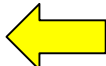
GO

-- Outro modo de chamar a função FATORIAL

```
PRINT dbo.fatorial(10)
```

GO

Fatorial de 20	Fatorial de 5
2432902008176640000	120



(1 row(s) affected)

3628800



■ Criação de UDF escalar

-- Exibe os dados das viagens dos alunos

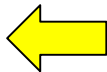
```
SELECT A.Cod_Aluno      AS 'Código do Aluno',  
       A.Nome_Aluno    AS 'Nome do Aluno',  
       A.Sexo,  
       A.Endereco      AS 'Endereço',  
       A.Pais_Origem   AS 'Origem',  
       V.Pais_Destino  AS 'Destino',  
       V.Valor         AS 'Custo R$'  
  
FROM Alunos A INNER JOIN Viagens V  
ON A.Cod_Viagem = V.Cod_Viagem  
  
GO
```

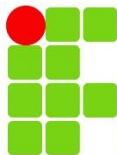


■ Criação de UDF escalar

Código do Aluno	Nome do Aluno	Sexo	Endereço	Origem	Destino	Custo R\$
1	Maria Cristina	F	Rua João XXIII, 15 - São Paulo	BRA	VGB	5350.00
2	Ana Paula Lima	F	Rua Mauro Silva, 1908 - São Paulo	BRA	VNM	8900.00
3	Carlos Renato	M	Av. Faria Lima, 347 - São Paulo	BRA	UKR	18525.35
4	Hugo Silva	M	Av. da Consolação, 1216 - São Paulo	BRA	TUR	12350.25
5	Marcos Antônio	M	Rua Agripino Lopes, 100 - São Paulo	BRA	TKL	7520.00
6	Gislaine Silva	F	Av. Nelson Dávila, 2345 - São José dos Campos	BRA	USA	14255.35
7	Antônio Pereira	M	Rua Joaquim Nabuco, 18 - Jacareí	BRA	TLS	3250.00
8	Jair Lopes	M	Rua Pedro XIII, Santa Isabel	BRA	THA	9500.00
9	Miguel Firmino	M	Av. Colinas, 2340 - São José dos Campos	BRA	RUS	9250.00
38	Jaime Augusto	M	Rua São Leopoldo, 11 - Rio de Janeiro	BRA	UKR	18525.35
39	Leandro Leite	M	Av. do Povo, 3489 - Rio de Janeiro	BRA	RUS	9250.00
40	Mônica Silva	F	Rua Americana, 5200 - Rio de Janeiro	BRA	RUS	9250.00
41	Marília dos Santos	F		BRA	TUR	12350.25
42	Guilherme dos Santos	M	Av. Brasil, 1008 - São Paulo	BRA	VGB	5350.00
43	Heidi Lima	F	Rua César Conceição, 12 - Santo Antônio do Pinhal	BRA	VNM	8900.00
44	Amílcar Júnior	M	Rua Senador Kennedy, 901 - São Paulo	BRA	THA	9500.00
45	Alexandro Duarte	M	Rua Maria Cíntia - Cruzeiro	BRA	QAT	12355.00
46	Maurício dos Santos	M	Rua Marta Silva - Jacareí	BRA	USA	22850.25
47	Mary Ann Duarte	F	Av. Brasil, 6320 - São Paulo	BRA	MEX	18600.10
48	Gabriela Pereira	F	Rua Franco Silva, 1599 - São Paulo	BRA	DNK	19000.50
49	André César	M	Rua Militar, 349 - Rio de Janeiro	BRA	NZL	15300.00
50	Edson Lopes	M	Av. Pedro Silva, 3047 - Rio de Janeiro	BRA	NZL	15300.00

(50 row(s) affected)





■ Criação de UDF escalar

```
-- Criando uma função para alinhar o campo 'Valor'  
-- @texto = campo cujos valores serão alinhados  
-- @tamanho = quantidade de vezes que o caractere será replicado  
-- @caractere = caractere que será replicado
```

```
CREATE FUNCTION sf_alinhaCampo(@texto VARCHAR(255), @tamanho  
TINYINT, @caractere CHAR(1))
```

```
    RETURNS VARCHAR(255)
```

```
AS
```

```
    BEGIN
```

```
        DECLARE @resultado VARCHAR(255)
```

```
        SET @resultado = REPLICATE(@caractere, @tamanho -  
LEN(@texto)) + @texto
```

```
        RETURN @resultado
```

```
    END
```



■ Criação de UDF escalar

-- Exibe os dados das viagens dos alunos, utilizando a função
-- SF_ALINHACAMPO, para formatar o resultado da consulta

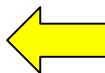
```
SELECT dbo.sf_alinhaCampo(A.Cod_Aluno, 2, 0) AS 'Código do Aluno',  
       A.Nome_Aluno    AS 'Nome do Aluno',  
       A.Sexo,  
       A.Endereco      AS 'Endereço',  
       A.Pais_Origem   AS 'Origem',  
       V.Pais_Destino  AS 'Destino',  
       dbo.sf_alinhaCampo(CAST(V.Valor AS VARCHAR), 10, 0) AS  
       'Custo R$'  
FROM Alunos A INNER JOIN Viagens V  
ON A.Cod_Viagem = V.Cod_Viagem  
GO
```



■ Criação de UDF escalar

Código do Aluno	Nome do Aluno	Sexo	Endereço	Origem	Destino	Custo R\$
01	Maria Cristina	F	Rua João XXIII, 15 - São Paulo	BRA	VGB	0005350.00
02	Ana Paula Lima	F	Rua Mauro Silva, 1908 - São Paulo	BRA	VNM	0008900.00
03	Carlos Renato	M	Av. Faria Lima, 347 - São Paulo	BRA	UKR	0018525.35
04	Hugo Silva	M	Av. da Consolação, 1216 - São Paulo	BRA	TUR	0012350.25
05	Marcos Antônio	M	Rua Agripino Lopes, 100 - São Paulo	BRA	TKL	0007520.00
06	Gislaine Silva	F	Av. Nelson Dávila, 2345 - São José dos Campos	BRA	USA	0014255.35
07	Antônio Pereira	M	Rua Joaquim Nabuco, 18 - Jacareí	BRA	TLS	0003250.00
08	Jair Lopes	M	Rua Pedro XIII, Santa Isabel	BRA	THA	0009500.00
09	Miguel Firmino	M	Av. Colinas, 2340 - São José dos Campos	BRA	RUS	0009250.00
.....						
38	Jaime Augusto	M	Rua São Leopoldo, 11 - Rio de Janeiro	BRA	UKR	0018525.35
39	Leandro Leite	M	Av. do Povo, 3489 - Rio de Janeiro	BRA	RUS	0009250.00
40	Mônica Silva	F	Rua Americana, 5200 - Rio de Janeiro	BRA	RUS	0009250.00
41	Marília dos Santos	F		BRA	TUR	0012350.25
42	Guilherme dos Santos	M	Av. Brasil, 1008 - São Paulo	BRA	VGB	0005350.00
43	Heidi Lima	F	Rua César Conceição, 12 - Santo Antônio do Pinhal	BRA	VNM	0008900.00
44	Amílcar Júnior	M	Rua Senador Kennedy, 901 - São Paulo	BRA	THA	0009500.00
45	Alexandro Duarte	M	Rua Maria Cíntia - Cruzeiro	BRA	QAT	0012355.00
46	Maurício dos Santos	M	Rua Marta Silva - Jacareí	BRA	USA	0022850.25
47	Mary Ann Duarte	F	Av. Brasil, 6320 - São Paulo	BRA	MEX	0018600.10
48	Gabriela Pereira	F	Rua Franco Silva, 1599 - São Paulo	BRA	DNK	0019000.50
49	André César	M	Rua Militar, 349 - Rio de Janeiro	BRA	NZL	0015300.00
50	Edson Lopes	M	Av. Pedro Silva, 3047 - Rio de Janeiro	BRA	NZL	0015300.00

(50 row(s) affected)



■ Criação de UDF escalar

-- Cria uma função que recebe a sigla de um país e
-- retorna seu nome

```
CREATE FUNCTION sf_descobreNomePais (@sigla CHAR(3))  
    RETURNS VARCHAR(50)  
  
AS  
  
    BEGIN  
  
        DECLARE @resposta VARCHAR(50)  
        SET @resposta = (SELECT Nome_Pais FROM Paises  
WHERE cod_Pais = @sigla)  
        RETURN @resposta  
  
    END  
  
GO
```



■ Criação de UDF escalar

- Utiliza a função para descobrir o nome do país
- passado como parâmetro

```
SELECT dbo.sf_descobreNomePaís('BRA') AS 'País'  
GO
```

País

Brasil

(1 row(s) affected) ←



■ Criação de UDF escalar

-- Exibe os dados das viagens dos alunos, utilizando a função
-- SF_ALINHACAMPO, para formatar o resultado da consulta

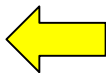
```
SELECT A.Cod_Aluno      AS 'Código do Aluno',  
       A.Nome_Aluno    AS 'Nome do Aluno',  
       A.Sexo,  
       A.Endereco      AS 'Endereço',  
       A.Pais_Origem   AS 'Código da Origem',  
       dbo.sf_descobreNomePais(A.Pais_Origem) AS 'Origem',  
       V.Pais_Destino  AS 'Código do Destino',  
       dbo.sf_descobreNomePais(V.Pais_Destino) AS 'Destino',  
       V.Valor         AS 'Custo R$'  
  
FROM Alunos A INNER JOIN Viagens V  
ON A.Cod_Viagem = V.Cod_Viagem  
  
GO
```



■ Criação de UDF escalar

Código do Aluno	Nome do Aluno	Sexo	Endereço	Código da Origem	Origem	Código do Destino	Destino	Custo R\$
1	Maria Cristina	F	Rua João XXIII, 15 - São Paulo	BRA	Brasil	VGB	Virgens Britânicas, Ilhas	5350.00
2	Ana Paula Lima	F	Rua Mauro Silva, 1908 - São Paulo	BRA	Brasil	VNM	Vietnã	8900.00
3	Carlos Renato	M	Av. Faria Lima, 347 - São Paulo	BRA	Brasil	UKR	Ucrânia	18525.35
4	Hugo Silva	M	Av. da Consolação, 1216 - São Paulo	BRA	Brasil	TUR	Turquia	12350.25
5	Marcos Antônio	M	Rua Agripino Lopes, 100 - São Paulo	BRA	Brasil	TKL	Toquelau	7520.00
6	Gislaine Silva	F	Av. Nelson Dávila, 2345 - São José dos Campos	BRA	Brasil	USA	Estados Unidos da América	14255.35
7	Antônio Pereira	M	Rua Joaquim Nabuco, 18 - Jacareí	BRA	Brasil	TLS	Timor-Leste	3250.00
8	Jair Lopes	M	Rua Pedro XIII, Santa Isabel	BRA	Brasil	THA	Tailândia	9500.00
9	Miguel Firmino	M	Av. Colinas, 2340 - São José dos Campos	BRA	Brasil	RUS	Rússia	9250.00
29	Mariana Lopes	F	Rua Califórnia, 1009 - São Paulo	BRA	Brasil	MRT	Mauritânia	6250.50
30	César Augusto	M	Rua Augusta, 235 - São Paulo	BRA	Brasil	GTM	Guatemala	8200.00
31	Melissa Pereira	F	Rua Américo Vespúcio, 56 - São Paulo	BRA	Brasil	CHE	Suíça	14200.00
32	Jéssica Patrícia	F	Rua Brigadeiro Jordão, 199 - Campos do Jordão	BRA	Brasil	QAT	Qatar	12355.00
33	Patrícia Silva	F	Rua Periquito, 24 - Campos do Jordão	BRA	Brasil	QAT	Qatar	12355.00
34	Maria Duarte	F	Rua F, 341 - Campos do Jordão	BRA	Brasil	QAT	Qatar	12355.00
35	Marcelo Augusto	M	Av. Frei Orestes Girardi, 899 - Campos do Jordão	BRA	Brasil	SYR	Síria	16100.00
36	Cristiano Leite	M	Rua Gumerindo Lopes, 566 - São Paulo	BRA	Brasil	USA	Estados Unidos da América	22850.25
37	Alberto Carlos	M	Av. Jorge Aparecido, 641 - Rio de Janeiro	BRA	Brasil	DNK	Dinamarca	19000.50
38	Jaime Augusto	M	Rua São Leopoldo, 11 - Rio de Janeiro	BRA	Brasil	UKR	Ucrânia	18525.35
39	Leandro Leite	M	Av. do Povo, 3489 - Rio de Janeiro	BRA	Brasil	RUS	Rússia	9250.00
40	Mônica Silva	F	Rua Americana, 5200 - Rio de Janeiro	BRA	Brasil	RUS	Rússia	9250.00
41	Marília dos Santos	F		BRA	Brasil	TUR	Turquia	12350.25
42	Guilherme dos Santos	M	Av. Brasil, 1008 - São Paulo	BRA	Brasil	VGB	Virgens Britânicas, Ilhas	5350.00
43	Heidi Lima	F	Rua César Conceição, 12 - Santo Antônio do Pinhal	BRA	Brasil	VNM	Vietnã	8900.00
44	Amílcar Júnior	M	Rua Senador Kennedy, 901 - São Paulo	BRA	Brasil	THA	Tailândia	9500.00
45	Alexandro Duarte	M	Rua Maria Cíntia - Cruzeiro	BRA	Brasil	QAT	Qatar	12355.00
46	Maurício dos Santos	M	Rua Marta Silva - Jacareí	BRA	Brasil	USA	Estados Unidos da América	22850.25
47	Mary Ann Duarte	F	Av. Brasil, 6320 - São Paulo	BRA	Brasil	MEX	México	18600.10
48	Gabriela Pereira	F	Rua Franco Silva, 1599 - São Paulo	BRA	Brasil	DNK	Dinamarca	19000.50
49	André César	M	Rua Militar, 349 - Rio de Janeiro	BRA	Brasil	NZL	Nova Zelândia	15300.00
50	Edson Lopes	M	Av. Pedro Silva, 3047 - Rio de Janeiro	BRA	Brasil	NZL	Nova Zelândia	15300.00

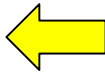
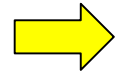
(50 row(s) affected)



■ Criação de UDF inline

- Cria uma função que recebe como parâmetro o código do país.
- Ela retorna uma consulta com os dados das viagens para esse país.

```
CREATE FUNCTION sf_exibeViagensPais (@sigla CHAR(3))  
    RETURNS TABLE  
AS  
RETURN  
    SELECT V.Cod_Viagem    AS 'Código da Viagem',  
           P.Nome_pais + ' (' + V.Pais_Destino + ')' AS 'Destino',  
           P.Idioma_pais   AS 'Idioma',  
           V.Data_Saida    AS 'Saída',  
           V.Data_Retorno  AS 'Retorno',  
           V.Valor         AS 'Valor R$'  
    FROM Países P INNER JOIN Viagens V  
        ON P.Cod_Pais = V.Pais_Destino  
    WHERE V.Pais_Destino = @sigla
```



■ Criação de UDF inline

```
-- Tenta utilizar a função SF_EXIBEVIAGENSPAIS  
-- Essa sintaxe não funciona, pois uma UDF inline  
-- funciona como se fosse uma tabela
```

```
SELECT dbo.sf_exibeViagensPais('USA')  
GO
```

```
-- Utilizando a função SF_EXIBEVIAGENSPAIS  
SELECT * FROM dbo.sf_exibeViagensPais('USA')  
GO
```



■ Criação de UDF inline

-- Utilizando a função SF_EXIBEVIAGENSPAIS

```
SELECT * FROM dbo.sf_exibeViagensPais('MEX')
```

```
GO
```

-- Utilizando a função SF_EXIBEVIAGENSPAIS

-- No caso, seleciona apenas campos específicos

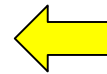
```
SELECT [Código da Viagem],
```

```
        Destino,
```

```
        [Valor R$]
```

```
FROM dbo.sf_exibeViagensPais('MEX')
```

```
GO
```



■ Criação de UDF inline

Código da Viagem	Destino	Idioma	Saída	Retorno	Valor R\$
6	Estados Unidos da América (USA)	Inglês	2010-03-16	2011-03-16	14255.35
16	Estados Unidos da América (USA)	Inglês	2010-11-05	2012-11-05	22850.25
23	Estados Unidos da América (USA)	Inglês	2011-01-08	2012-01-08	11850.95
24	Estados Unidos da América (USA)	Inglês	2011-01-10	2012-01-10	15850.95
25	Estados Unidos da América (USA)	Inglês	2010-08-02	2011-08-02	16550.20
26	Estados Unidos da América (USA)	Inglês	2010-09-05	2011-03-20	14750.25
28	Estados Unidos da América (USA)	Inglês	2010-08-15	2012-10-20	19500.80
29	Estados Unidos da América (USA)	Inglês	2011-01-01	2011-12-01	12520.25

(8 row(s) affected)

Código da Viagem	Destino	Idioma	Saída	Retorno	Valor R\$
17	México (MEX)	Espanhol ou Castelhan	2010-12-02	2012-12-30	18600.10
22	México (MEX)	Espanhol ou Castelhan	2011-01-05	2012-01-05	15650.25

(2 row(s) affected)

Código da Viagem	Destino	Valor R\$
17	México (MEX)	18600.10
22	México (MEX)	15650.25

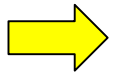

(2 row(s) affected)



■ Criação de UDF multi-statement

-- Cria uma função que retorna somente os dados das viagens cuja
-- data de saída é posterior a data passada como parâmetro.

```
CREATE FUNCTION sf_ViagensFuturas (@data DATE)
    RETURNS @viagens_futuras TABLE (Codigo INT, Saida DATE, Retorno
DATE, Destino VARCHAR(50))
AS
    BEGIN
        INSERT INTO @viagens_futuras
        SELECT V.Cod_Viagem, V.Data_Saida, V.Data_Retorno, P.Nome_pais
        FROM Viagens V INNER JOIN Países P
            ON V.Pais_Destino = P.Cod_Pais
        WHERE V.Data_Saida > @data
        RETURN
    END
```



GO



■ Criação de UDF multi-statement

-- Altera o formato de data e hora -> Brasil

SET DATEFORMAT DMY 

GO

-- Utilizando a função SF_VIAGENSFUTURAS

-- Retornando todas as viagens que serão realizadas
-- a partir de janeiro de 2011.

SELECT * FROM sf_ViagensFuturas('01-01-2011')

GO

SELECT * FROM sf_ViagensFuturas('01-02-2011')

GO



■ Exibe informações sobre as UDFs

-- Exibe informações sobre as UDFs do banco
-- de dados em uso

```
SELECT name          AS 'Nome da Função',  
       definition    AS 'Definição',  
       type_desc     AS 'Tipo'  
FROM sys.sql_modules M INNER JOIN sys.objects O  
     ON M.object_id = O.object_id  
WHERE type_desc LIKE '%function%'  
GO
```



■ Excluindo uma UDF

-- Exclui a função FATORIAL

DROP FUNCTION fatorial

GO

-- Tenta utilizar a função FATORIAL

SELECT dbo.fatorial(5) AS 'Fatorial de 5'

GO

Cannot find either column "dbo" or the user-defined function or aggregate "dbo.fatorial", or the name is ambiguous.



- Na próxima aula veremos
 - Triggers.

