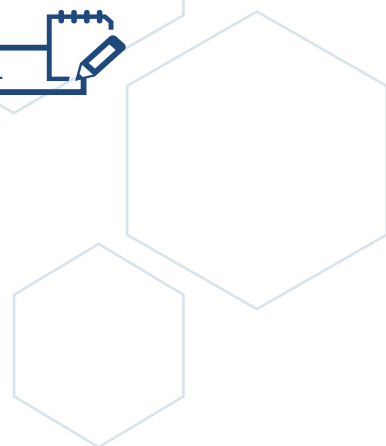
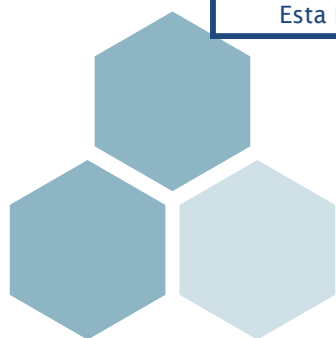




Comandos adicionais

- ◆ Funções de cadeia de caracteres;
- ◆ Função CASE;
- ◆ Manipulação de campos do tipo datetime;
- ◆ Alteração da configuração de idioma a partir do SSMS.

Esta Leitura Complementar refere-se ao conteúdo das Aulas 28 a 30.



1.1. Funções de cadeia de caracteres

Estas funções executam operações em um valor de entrada de cadeia de caracteres e retornam o mesmo dado trabalhado. Por exemplo, podemos concatenar, replicar ou inverter os dados de entrada. A seguir, vamos apresentar as funções de cadeia de caracteres principais fornecidas pelo SQL Server:

- **LEN (expressão_string)**

Esta função retorna o número de caracteres especificado no parâmetro **expressão_string**.

Veja um exemplo:

```
SELECT LEN ('Brasil');
```

Results		Messages	
		(No column name)	
1	6		

- **REPLICATE (expressão_string, mult)**

Esta função repete os caracteres do parâmetro **expressão_string** pelo número de vezes especificado no parâmetro **mult**.

Veja um exemplo:

```
SELECT REPLICATE ('Teste',4);
```

Results		Messages	
		(No column name)	
1	TesteTesteTesteTeste		

- **REVERSE (expressão_string)**

Esta função retorna a ordem inversa do parâmetro **expressão_string**.

Veja um exemplo:

```
SELECT REVERSE ('amina');
```

Results		Messages	
		(No column name)	
1	anima		

- **STR (número [, tamanho [, decimal]])**

Esta função retorna dados do tipo **string** a partir de dados numéricos.

Veja um exemplo:

```
SELECT STR (213);
```

Results		Messages	
		(No column name)	
1		213	

- **SUBSTRING (expressão, início, tamanho)**

Esta função retorna uma parte dos caracteres do parâmetro **expressão** a partir dos valores de **início** e **tamanho**.

Veja um exemplo:

```
SELECT SUBSTRING ('Paralelepípedo',3,7);
```

Results		Messages	
		(No column name)	
1		ralelep	

- **CONCAT (expr1, expr2 [, exprN])**

Esta função concatena as expressões retornando um string. As expressões podem ser de qualquer tipo.

Veja um exemplo:

```
SELECT CONCAT ('SQL ', 'módulo ', 'I');
```

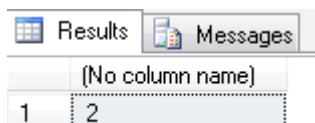
Results		Messages	
		(No column name)	
1		SQL módulo I	

- **CHARINDEX (expr1, expr2 [, exprN])**

Esta função pesquisa uma string dentro de outra, retornando a posição encontrada. Caso não encontre o valor pesquisado, o retorno será zero.

Veja um exemplo:

```
SELECT CHARINDEX ('A' , 'CASA')
```



The screenshot shows a SQL Server query results window. It has two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one column labeled '(No column name)' and one row with the value '2'.

(No column name)
2

- **FORMAT (expressão, formato)**

Formata uma **expressão** numérica ou date/time no formato definido por um string de formatação.

A seguir, temos alguns exemplos de caracteres usados na formatação de strings:

- **Caracteres para formatação de números:**

0 (zero)	Define uma posição numérica. Se não existir número na posição, aparece o zero.
#	Define uma posição numérica. Se não existir número na posição, fica vazio.
. (ponto)	Separador de decimal.
, (vírgula)	Separador de milhar.
%	Mostra o sinal e o número multiplicado por 100.

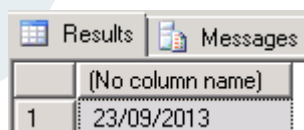
Qualquer outro caractere inserido na máscara de formatação será exibido normalmente na posição em que foi colocado.

- Caracteres para formatação de data:

d	Dia com 1 ou 2 dígitos.
dd	Dia com 2 dígitos.
ddd	Abreviação do dia da semana.
dddd	Nome do dia da semana.
M	Mês com 1 ou 2 dígitos.
MM	Mês com 2 dígitos.
MMM	Abreviação do nome do mês.
MMMM	Nome do mês.
yy	Ano com 2 dígitos.
yyyy	Ano com 4 dígitos.
hh	Hora de 1 a 12.
HH	Hora de 0 a 23.
mm	Minutos.
ss	Segundos.
fff	Milésimos de segundo.

Veja um exemplo:

```
SELECT FORMAT (GETDATE(), 'dd/MM/yyyy');
```



(No column name)	
1	23/09/2013

1.2. Função CASE

Os valores pertencentes a uma coluna podem ser testados por meio da cláusula **CASE** em conjunto com o comando **SELECT**. Dessa maneira, é possível aplicar diversas condições de validação em uma consulta.

No exemplo a seguir, **CASE** é utilizado para verificar se os funcionários da tabela **TB_EMPREGADO** são ou não sindicalizados:

```
SELECT NOME, SALARIO, CASE SINDICALIZADO
                        WHEN 'S' THEN 'Sim'
                        WHEN 'N' THEN 'Não'
                        ELSE 'N/C'
                        END AS [Sindicato?] ,
      DATA_ADMISSAO
FROM TB_EMPREGADO;
```

Já no próximo exemplo, verificamos em qual dia da semana os empregados foram admitidos:

```
SELECT NOME, SALARIO, DATA_ADMISSAO,
      CASE DATEPART(WEEKDAY, DATA_ADMISSAO)
        WHEN 1 THEN 'Domingo'
        WHEN 2 THEN 'Segunda-Feira'
        WHEN 3 THEN 'Terça-Feira'
        WHEN 4 THEN 'Quarta-Feira'
        WHEN 5 THEN 'Quinta-Feira'
        WHEN 6 THEN 'Sexta-Feira'
        WHEN 7 THEN 'Sábado'
      END AS DIA_SEMANA
FROM TB_EMPREGADO;
```

1.3. Manipulando campos do tipo datetime

O tipo de dado **datetime** é utilizado para definir valores de data e hora. Aceita valores entre 1º de janeiro de 1753 até 31 de dezembro de 9999. O formato no qual digitamos a data depende de configurações do servidor ou usuário.

Vejamos algumas funções utilizadas para retornar dados desse tipo:

- **SET DATEFORMAT**

É utilizada para determinar a forma de digitação de data/hora durante uma sessão de trabalho.

```
SET DATEFORMAT (ordem)
```

A ordem das porções de data é definida por **ordem**, que pode ser um dos valores da tabela adiante:

Valor	Ordem
mdy	Mês, dia e ano (Formato padrão americano).
dmy	Dia, mês e ano.
ymd	Ano, mês e dia.
ydm	Ano, dia e mês.
myd	Mês, ano e dia.
dym	Dia, ano e mês.

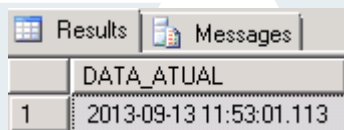
O exemplo a seguir determina o formato ano/mês/dia para o valor de data a ser retornado:

```
SET DATEFORMAT YMD;
```

- **GETDATE()**

Retorna a data e hora atual do sistema, sem considerar o intervalo de fuso-horário. O valor é derivado do sistema operacional do computador no qual o SQL Server é executado:

```
SELECT GETDATE() AS DATA_ATUAL;
```



	DATA_ATUAL
1	2013-09-13 11:53:01.113

Para sabermos qual será a data daqui a 45 dias, utilizamos o seguinte código:

```
SELECT GETDATE() + 45;
```

Já no código a seguir, **GETDATE()** é utilizado para saber há quantos dias cada funcionário da tabela **TB_EMPREGADO** foi admitido:

```
SELECT CODFUN, NOME, CAST(GETDATE() - DATA_ADMISSAO AS INT) AS DIAS_
NA_EMPRESA
FROM TB_EMPREGADO
```

- **DAY()**

Esta função retorna um valor inteiro que representa o dia da data especificada como argumento **data** na sintaxe adiante:

```
DAY(data)
```

O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna.

Vejamos os exemplos a seguir:

```
-- Número do dia correspondente à data de hoje
SELECT DAY(GETDATE());

-- Todos os funcionários admitidos no dia primeiro de
-- qualquer mês e qualquer ano
SELECT * FROM TB_EMPREGADO
WHERE DAY(DATA_ADMISSAO) = 1;
```

- **MONTH()**

Esta função retorna um valor inteiro que representa o mês da data especificada como argumento **data** da sintaxe adiante:

```
MONTH(data)
```

O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna.

Vejamos os exemplos:

```
-- Número do mês correspondente à data de hoje
SELECT MONTH(GETDATE())

-- Empregados admitidos em dezembro
SELECT * FROM TB_EMPREGADO
WHERE MONTH(DATA_ADMISSAO) = 12
```

- **YEAR()**

Esta função retorna um valor inteiro que representa o ano da data especificada como argumento **data** na sintaxe adiante:

```
YEAR(data)
```

O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna.

O exemplo a seguir retorna os empregados admitidos no ano 2006:

```
SELECT * FROM TB_EMPREGADO  
WHERE YEAR(DATA_ADMISSAO) = 2006;
```

Já o exemplo a seguir retorna os empregados admitidos no mês de janeiro de 2011:

```
SELECT * FROM TB_EMPREGADO  
WHERE YEAR(DATA_ADMISSAO) = 2011 AND  
      MONTH(DATA_ADMISSAO) = 1;
```

- **DATEPART()**

Esta função retorna um valor inteiro que representa uma porção (especificada no argumento **parte**) de data ou hora definida no argumento **data** da sintaxe adiante:

```
DATEPART (parte, data)
```

O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna, enquanto **parte** pode ser um dos valores descritos na tabela a seguir:

Valor	Parte retornada	Abreviação
year	Ano	yy, yyyy
quarter	Trimestre (1/4 de ano)	qq, q
month	Mês	mm, m
dayofyear	Dia do ano	dy, y
day	Dia	dd, d
week	Semana	wk, ww
weekday	Dia da semana	dw
hour	Hora	hh
minute	Minuto	mi, n
second	Segundo	ss, s
millisecond	Milissegundo	ms
microsecond	Microsegundo	mcs
nanosecond	Nanossegundo	ns
TZoffset	Diferença de fuso-horário	tz
ISO_WEEK	Retorna a numeração da semana associada a um ano	isowk, isoww

Vale dizer que o resultado retornado será o mesmo, independentemente de termos especificado um valor ou a respectiva abreviação.

O exemplo a seguir utiliza **DATEPART** para retornar os empregados admitidos em dezembro de 1996. Para isso, são especificadas as partes de ano (**YEAR**) e mês (**MONTH**):

```
SELECT * FROM TB_EMPREGADO
WHERE DATEPART(YEAR, DATA_ADMISSAO) = 1996 AND
      DATEPART(MONTH, DATA_ADMISSAO) = 12;
```

- **DATENAME()**

Esta função retorna como resultado uma string de caracteres que representa uma porção da data ou hora definida no argumento **data** da sintaxe adiante:

```
DATENAME(parte, data)
```

O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna, enquanto **parte**, que representa a referida porção, pode ser um dos valores descritos na tabela anterior.

O exemplo a seguir é utilizado para obter os funcionários que foram admitidos em uma sexta-feira:

```
-- Funcionários admitidos em uma sexta-feira
SELECT
    CODFUN, NOME, DATA_ADMISSAO,
    DATENAME(WEEKDAY, DATA_ADMISSAO) AS DIA_SEMANA,
    DATENAME(MONTH, DATA_ADMISSAO) AS MES
FROM TB_EMPREGADO
WHERE DATEPART(WEEKDAY, DATA_ADMISSAO) = 6;
```

O resultado retornado por **DATENAME()** dependerá do idioma configurado no servidor SQL.

- **DATEADD()**

Esta função retorna um novo valor de **datetime** ao adicionar um intervalo a uma porção da data ou hora definida no argumento **data** da sintaxe adiante:

```
DATEADD(parte, numero, data)
```

O argumento **parte** é a porção de data ou hora que receberá o acréscimo definido em **numero**. O argumento **data** pode ser uma expressão, literal de string, variável definida pelo usuário ou expressão de coluna, enquanto **parte** pode ser um dos valores descritos na tabela anterior, com exceção de **TZoffset**.

O código a seguir retorna o dia de hoje mais 45 dias:

```
SELECT DATEADD( DAY, 45, GETDATE());
```

Este código retorna o dia de hoje mais 6 meses:

```
SELECT DATEADD( MONTH, 6, GETDATE());
```

Já o código a seguir retorna o dia de hoje mais 2 anos:

```
SELECT DATEADD( YEAR, 2, GETDATE());
```

- **DATEDIFF()**

Esta função obtém como resultado um número de data ou hora referente aos limites de uma porção de data ou hora, cruzados entre duas datas especificadas nos argumentos **data_inicio** e **data_final** da seguinte sintaxe:

```
DATEDIFF(parte, data_inicio, data_final)
```

O argumento **parte** representa a porção de **data_inicio** e **data_final** que especificará o limite cruzado.

O exemplo a seguir retorna a quantidade de dias vividos até hoje por uma pessoa nascida em 12 de novembro de 1959:

```
SELECT DATEDIFF( DAY, '1959.11.12', GETDATE());
```

O exemplo a seguir retorna a quantidade de meses vividos até hoje pela pessoa citada no exemplo anterior:

```
SELECT DATEDIFF( MONTH, '1959.11.12', GETDATE());
```

O exemplo a seguir retorna a quantidade de anos vividos até hoje por essa mesma pessoa:

```
SELECT DATEDIFF( YEAR, '1959.11.12', GETDATE());
```

Na utilização de **DATEDIFF()** para obter a diferença em anos ou meses, o valor retornado não é exato. O código a seguir retorna **1**. No entanto, a diferença somente seria **1** no dia 20 de fevereiro de 2009. Vejamos:

```
SELECT DATEDIFF( MONTH, '2013.1.20', '2013.2.15');
```

No próximo exemplo, o resultado retornado também é **1**, mas a diferença somente seria **1** no dia 20 de junho de 2009:

```
SELECT DATEDIFF( YEAR, '2012.12.20', '2013.1.15');
```

- **DATEFROMPARTS()**

Esta função retorna uma data (DATE) a partir dos parâmetros ano, mês e dia especificados nos argumentos da seguinte sintaxe:

```
DATEFROMPARTS (ano, mês, dia)
```

No exemplo a seguir, vamos retornar a data 25 de dezembro de 2013 a partir dos seguintes parâmetros:

```
SELECT DATEFROMPARTS (2013,12,25);
```

O resultado é mostrado a seguir:

Results		Messages	
		(No column name)	
1		2013-12-25	

- **TIMEFROMPARTS()**

Esta função retorna um horário (TIME) a partir dos parâmetros hora, minuto, segundo, milissegundo e precisão dos milissegundos especificados nos argumentos da seguinte sintaxe:

```
TIMEFROMPARTS (hora, minuto, segundo, milissegundo, precisão)
```

No exemplo a seguir, vamos retornar o horário 10:25:15 a partir dos seguintes parâmetros:

```
SELECT TIMEFROMPARTS (10,25,15,0,0);
```

O resultado é mostrado a seguir:

Results		Messages	
		(No column name)	
1		10:25:15	

- **DATETIMEFROMPARTS()**

Esta função retorna um valor de data e hora (DATETIME) a partir dos parâmetros ano, mês, dia, hora, minuto, segundo e milissegundo da seguinte sintaxe:

```
DATETIMEFROMPARTS (ano, mês, dia, hora, minuto, segundo, milissegundo);
```

Caso algum valor esteja incorreto, a função retornará um erro. Agora, se o parâmetro for nulo, a função retornará valor nulo. No exemplo a seguir, vamos retornar o valor 15 de setembro de 2013 às 14:00:15.0000:

```
SELECT DATETIMEFROMPARTS (2013,9,15,14,0,15,0);
```

O resultado é mostrado a seguir:

Results		Messages	
		(No column name)	
1		2013-09-15 14:00:15.000	

- **DATETIME2FROMPARTS()**

Retorna um valor do tipo **datetime2** a partir dos parâmetros ano, mês, dia, hora, minuto, segundo, fração de segundo e a precisão da fração de segundo da seguinte sintaxe:

```
DATETIME2FROMPARTS (ano, mês, dia, hora, minuto, segundo, fração, precisão);
```

Caso a precisão receba o valor 0, é necessário que a indicação de milissegundos também seja 0, senão a função retornará um erro. No exemplo a seguir, vamos retornar o valor 10 de outubro de 2013 às 21:00:59.0000:

```
SELECT DATETIME2FROMPARTS (2013,10,10,21,0,59,0,0);
```

O resultado é mostrado a seguir:

Results		Messages	
		(No column name)	
1		2013-10-10 21:00:59	

- **SMALLDATETIMEFROMPARTS()**

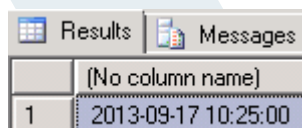
Esta função retorna um valor do tipo **smalldatetime** a partir dos parâmetros ano, mês, dia, hora e minuto da seguinte sintaxe:

```
SMALLDATETIMEFROMPARTS (ano, mês, dia, hora, minuto);
```

No exemplo a seguir, vamos retornar a data 17 de setembro de 2013 às 10:25:00:

```
SELECT SMALLDATETIMEFROMPARTS (2013,9,17,10,25);
```

O resultado é mostrado a seguir:



Results		Messages
		(No column name)
1	2013-09-17 10:25:00	

- **DATETIMEOFFSETFROMPARTS()**

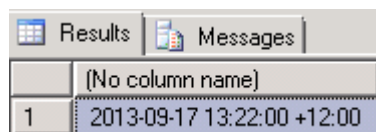
Esta função retorna um valor do tipo **datetimeoffset** representando um deslocamento de fuso horário a partir dos parâmetros ano, mês, dia, hora, minuto, segundo, fração, deslocamento de hora, deslocamento de minuto e a precisão decimal dos segundos:

```
DATETIMEOFFSETFROMPARTS (ano, mês, dia, hora, minuto, segundo, fração, desloc_hora, desloc_minuto, precisão);
```

No exemplo a seguir, vamos representar o deslocamento de 12 horas de fuso sem frações de segundos na data 17 de setembro de 2013 às 13:22:00:

```
SELECT DATETIMEOFFSETFROMPARTS (2013,9,17,13,22,0,0,12,0,0);
```

O resultado é mostrado a seguir:



Results		Messages
		(No column name)
1	2013-09-17 13:22:00 +12:00	

- **EOMonth()**

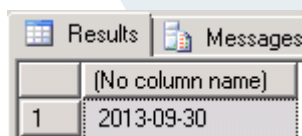
Esta função retorna o último dia do mês a partir dos parâmetros **data_início** e **adicionar_mês** (Opcional) da seguinte sintaxe:

```
EOMONTH (data_início [, adicionar_mês]);
```

O valor retornado é do tipo data (DATE). No exemplo a seguir, vamos retornar o último dia do mês atual:

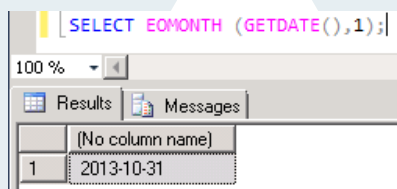
```
SELECT EOMONTH (GETDATE());
```

O resultado desse exemplo é mostrado a seguir:



Results		Messages
		(No column name)
1		2013-09-30

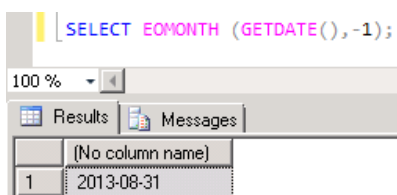
Para avançar 1 mês, adicione o valor 1 no parâmetro **adicionar_mês**:



```
SELECT EOMONTH (GETDATE(),1);
```

Results		Messages
		(No column name)
1		2013-10-31

Para voltar 1 mês adicione o valor -1 no parâmetro **adicionar_mês**:



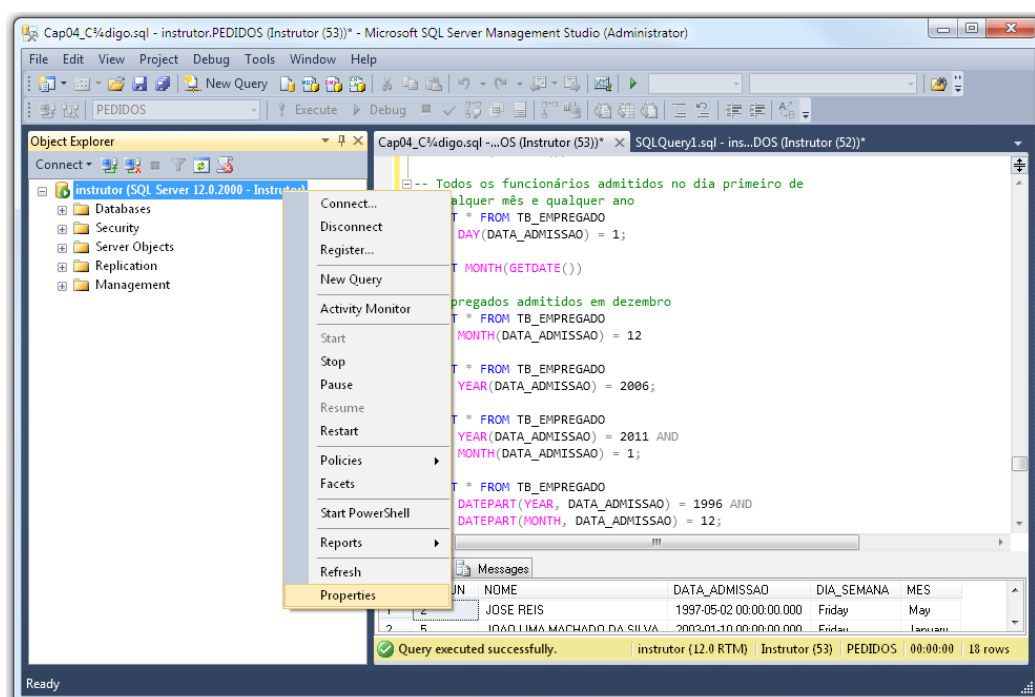
```
SELECT EOMONTH (GETDATE(),-1);
```

Results		Messages
		(No column name)
1		2013-08-31

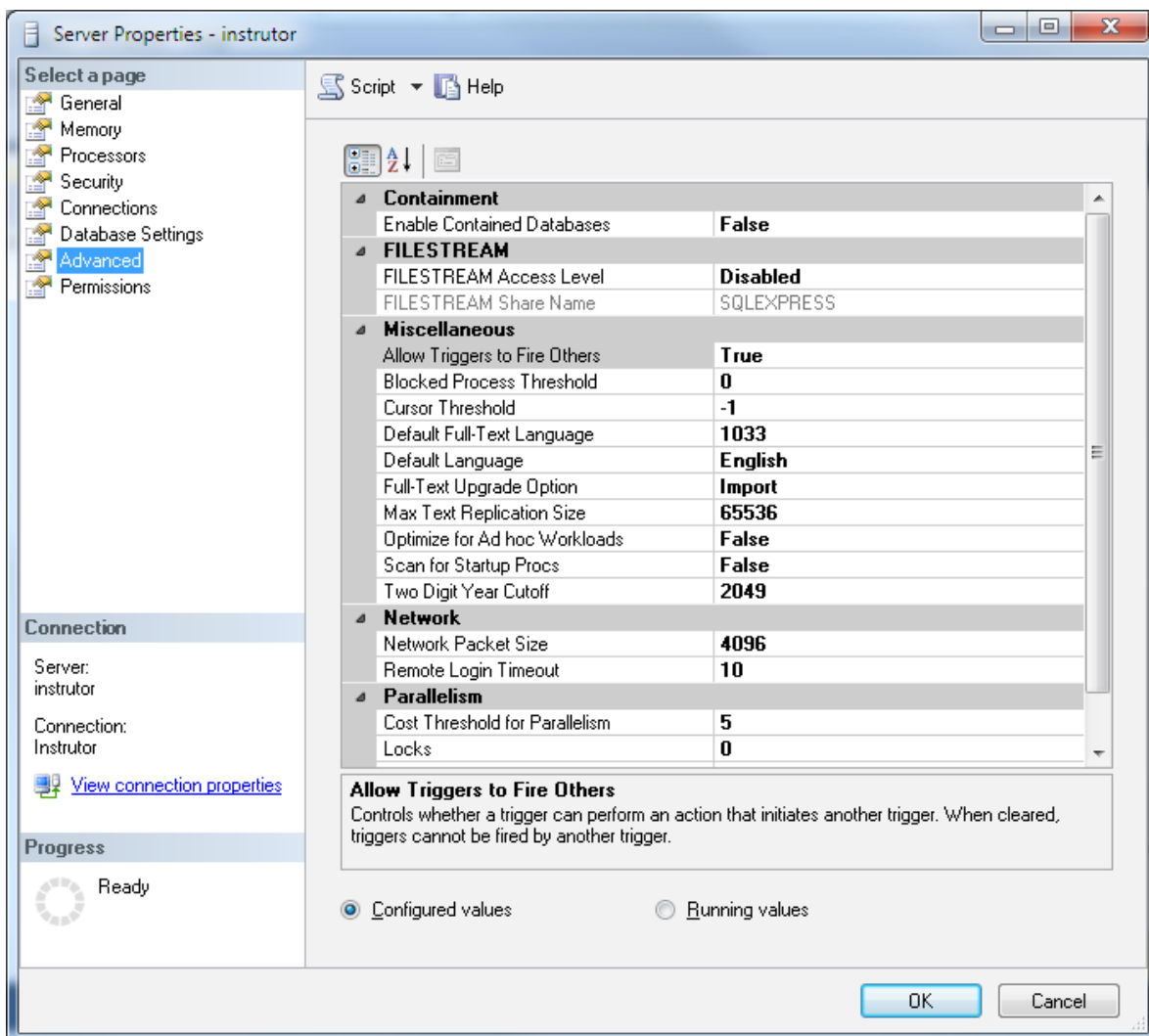
1.4. Alterando a configuração de idioma a partir do SSMS

O resultado retornado pela função **DATENAME()** dependerá do idioma do servidor SQL. Então, para que essa função retorne o nome do mês ou do dia da semana da maneira desejada, pode ser necessário alterar o idioma do SQL Server. Isso pode ser feito a partir do SQL Server Management Studio, como descrito no passo-a-passo a seguir:

1. Clique com o botão direito do mouse no nome do servidor, selecionando **Properties** no menu de contexto, como ilustrado a seguir:



2. Na janela seguinte, selecione a opção **Advanced**, procure o item **Default Language** e altere-o para o idioma que desejar:



O formato da data pode variar de acordo com o idioma escolhido:

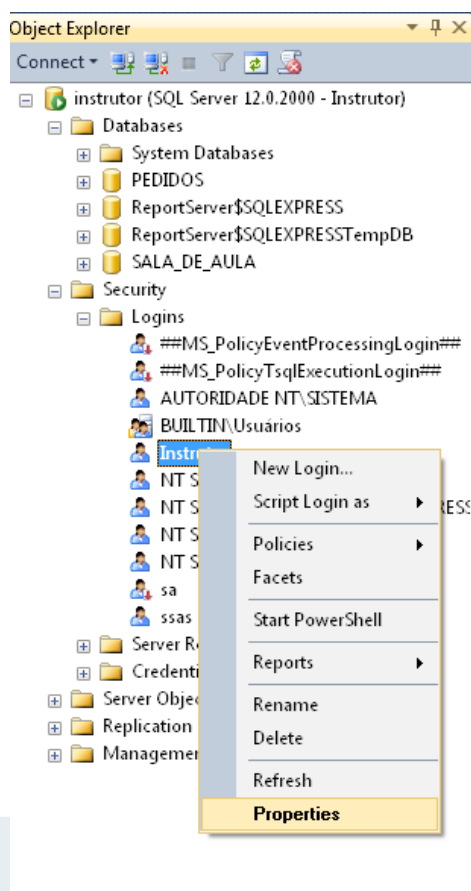
- **Brazilian:** Pode utilizar os formatos dd/mm/yy ou dd/mm/yyyy;
- **English:** Pode utilizar os formatos mm/dd/yy, mm/dd/yyyy ou yyyy.mm.dd.

A configuração escolhida será aplicada a todos os logins criados posteriormente. Logins já existentes deverão ser configurados novamente para que possuam o novo formato.

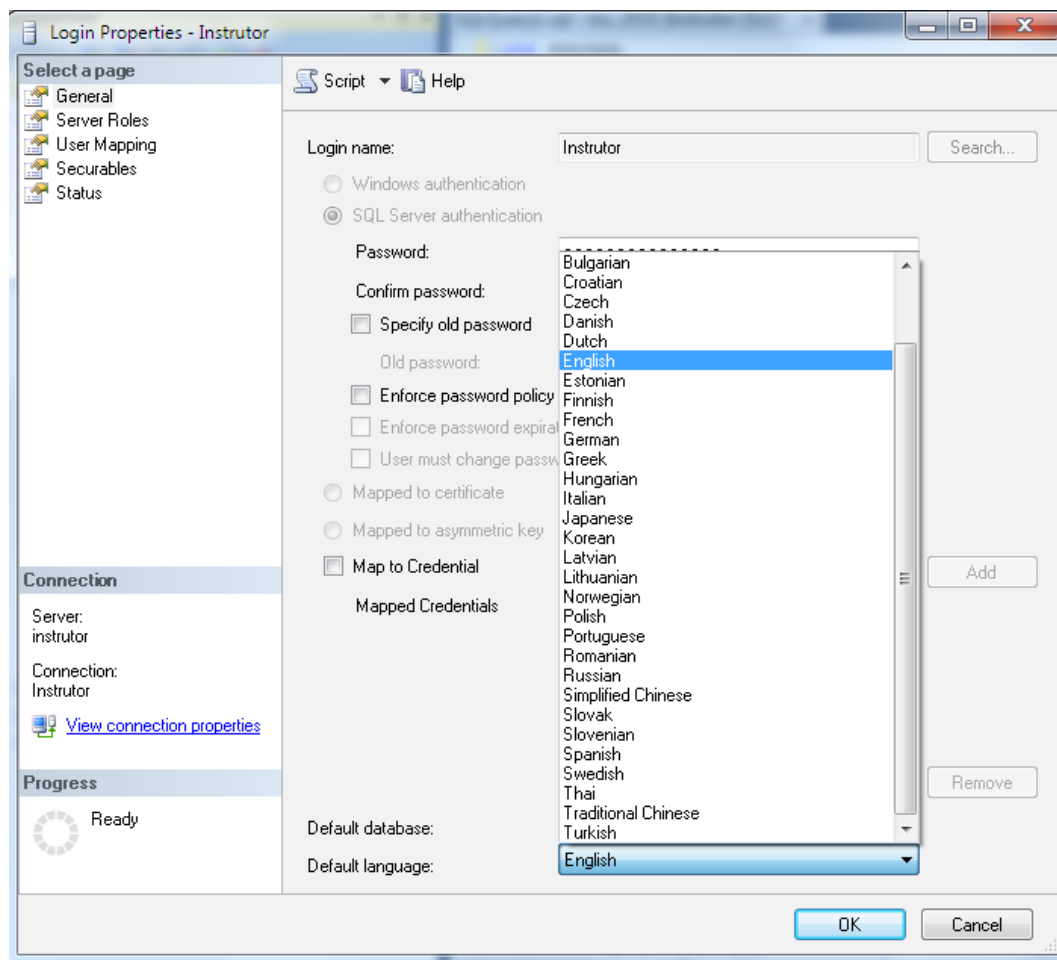
Para alterar a configuração de logins existentes, siga os passos adiante:

1. No navegador do SQL Server Management Studio, abra a pasta **Security** e, em seguida, **Logins**;

2. Clique com o botão direito no login a ser alterado e selecione **Properties** no menu de contexto, conforme ilustrado a seguir:



Será exibida a seguinte janela:



3. Na caixa de seleção **Default Language**, escolha o idioma desejado e clique em **OK**.

Alterar a configuração de idioma do servidor SQL afetará não apenas o valor retornado por **DATENAME()**, mas todos os aspectos associados ao idioma, como mensagens de erro exibidas no SQL Server Management Studio e o formato de digitação de datas.

Pontos principais

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes da leitura.

- Existem várias funções que auxiliam com valores de entrada de cadeia de caracteres, como LEN, REVERSE, CONCAT etc.;
- A função CASE permite que selecionemos valores conforme uma ou mais cláusulas específicas;
- O tipo de dado **datetime** é utilizado para definir valores de data e hora. Esse tipo é baseado no padrão norte-americano, ou seja, os valores devem atender ao modelo **mm/dd/aa** (mês, dia e ano, respectivamente). Dispomos de diversas funções para retornar dados desse tipo, tais como **GETDATE()**, **DAY()**, **MONTH()** e **YEAR()**.