# Funções internas

Funções internas foram criadas para melhorar a produtividade e facilitar

Funções podem ser determinísticas ou não-determinísticas.

Funções determinísticas sempre retornam o mesmo resultado quando recebem conjunto de parâmetros, funções determinísticas dependem muito do valor inserido como parâmetro.

Exemplo: A Função **COUNT** em uma tabela com 6 registros, irá sempre retornar o valor 6

Funções não-determinísticas retornam resultados diferentes cada vez que são chamadas.

Exemplo: A função **GETDATE** retorna a data atual, então, cada vez que essa função for chamada, o valor retornado será retornado

# Funções de agregação

Uma função de agregação *executa um cálculo em um conjunto de valores e retorna um único valor*. Sempre ignoram valores nulos (com exceção do **COUNT**)

Todas as funções de agregação são determinísticas.

Exemplo de algumas funções de agregação:

## **SUM**([Nome do grupo])

Retorna a soma de todos os elementos no grupo selecionado.

## **COUNT**([Nome do grupo])

Retorna a quantidade de registros no grupo selecionado **MAX**([Nome do grupo]) : Retorna o valor máximo no grupo selecionado

## **MIN**([Nome do grupo])

Retorna o valor mínimo no grupo selecionado

## **AVG**([Nome do grupo])

Retorna o valor médio do grupo selecionado

-- SUM , MAX, MIN e AVG não podem ter \* como parâmetro

# Funções matemáticas

As funções escalares a seguir executam um cálculo, normalmente *com base em valores de entrada fornecidos como argumentos* e retornam um valor numérico.

Exemplo de algumas funções matemáticas:

## **CEILING**([Expressão numérica])

Retorna o menor valor integral menor ou igual ao número especificado.

Exemplo: **CEILING** (95.5) = 95

## **FLOOR**([Expressão numérica])

Retorna o menor valor integral menor ou igual ao número especificado .

Exemplo: **FLOOR** (95.5) = 96

## **ROUND**([Expressão numérica],[precisão])

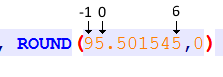
Arredonda o valor precisamente de uma casa decimal

*Precisão* : para a qual expressão deve ser arredondada.

A precisão deve ser uma valor do tipo **tinyint**, **smallint** ou **int**.

Quando a precisão é um **número positivo**, a expressão numérica é arredondada para o número de posições decimais especificado.

Exemplo:



Então se inserir **ROUND**(95.501545,5) , ele arredondará a 5ª casa decimal deixando o resultado 95.5015**50**

Se o valor inserido for negativo (no caso acima daria um erro pois ele iria precisar de mais uma casa)

Mas , suponha que o valo fosse 955.501. Se inserir **ROUND**(955.501,-1) o resultado seria 960.000 , ele voltaria uma casa.

## **PI**()

Retorna o valor 3.14159265358979

## **RAND**([Seed])

Retorna um valor float aleatório entre 0.0 e 1.0

*Seed :* A seed (ou semente) é um parâmetro opcional (já que se não for passado nenhum , o SQL trata de gerar a semente) que serve pra atribuir o retorno da função a esse numero   
Exemplo :

O Retorno da função **RAND**(12345) É 0.0139254973340301 , então toda vez que esse método for chamado com o parâmetro 12345 será retornado o valor acima.

# Funções de conversão

Funções de conversão oferecem suporte à conversão de tipo de dados.

Exemplos :

## **CAST**([Valor] AS [Tipo])

Converte o parâmetro inserido para o tipo inserido depois do AS.

Exemplo : **CAST**(‘95.0’ AS INT) = 95

## **CONVERT**([Tipo],[Valor])

Converte o segundo parâmetro inserido para o tipo inserido no primeiro parâmetro.

Exemplo : **CONVERT** (varchar(4),95.0) = ‘95.0’

-- A Diferença entra **CAST** e **CONVERT** é que **CONVERT** é uma função específica do **SQL Server** e **CAST** é padrão ANSI.

## **PARSE**([string] AS [Tipo] USING [Cultura])

Converte uma string para um dado do tipo de dados solicitado no SQL Server.

*Cultura*  seria uma cadeia de caracteres opcional que identifica a cultura na qual string é formatado. Se o argumento não for fornecido, o idioma da sessão atual será usado. Esse idioma é definido implícita ou explicitamente com o uso da instrução **SET LANGUAGE**. *Cultura* aceita qualquer cultura compatível com o .NET Framework.

Exemplo : **PARSE**('13-10-2010' AS DATE USING 'en-US') se tornará 201-10-13

Por causa da cultura AAAA/MM/DD

## **TRY\_CAST** , **TRY\_CONVERT** & **TRY\_PARSE**

Não há grande alteração no algoritmo , apenas que se a conversão não for bem sucedida , este método retorna NULL

# Funções de cadeia de caracteres

As funções de cadeia de caracteres executam uma operação em um valor de entrada de cadeia de caracteres e retornam uma cadeia de caracteres ou valor numérico:

Exemplo de algumas funções de cadeia de caracteres:

## **LOWER()**

Retorna uma expressão de caractere depois de converter para minúsculas os dados de caracteres em maiúsculas.

* **UPPER**()

Retorna uma expressão de caractere depois de converter para maiúsculas os dados de caracteres em minúsculas.

* **SPACE**([Número Inteiro])

Retorna uma cadeia de espaços do tamanho do numero inserido

## **SUBSTRING**([string],[inicio],[tamanho])

Retorna parte de uma expressão de caractere, binária, de texto ou de imagem no SQL Server.

Exemplo :

**SELECT** name, **SUBSTRING**(name, 1, 1) AS ‘Letra Inicial’ ,

**SUBSTRING**(name, 3, 2) AS ‘Terceiro e Quarto caracteres’

FROM sys.databases;

-- Este método funciona igual ao string.Substring do C#

## **CONCAT**([string\_1],[string\_2],...,[string\_N])

retorna uma cadeia de caracteres resultante de concatenação ou junção de dois ou mais valores de cadeia de caracteres.

Para esse método funcionar ele deverá ter no mínimo 2 parâmetros e no máximo 254.

## **SPLIT**([String],[Separador])

Divide uma cadeia de caracteres em linhas de subcadeias de caracteres com base em um caractere separador especificado

Exemplo : (Separa divide o texto usando o espaço como separador)

**SELECT** value FROM **STRING\_SPLIT**('Lorem ipsum dolor sit amet.', ' ');

O Resultado sera :

|  |
| --- |
| **value** |
| Lorem |
| Ipsum |
| Dolor |
| Sit |
| Amet |

-- Este método funciona igual ao string.Split do C#

## **REVERSE**([String])

Retorna a ordem inversa de um valor da cadeia de caracteres.

Aceita apenas varchar ou nvarchar

## **TRIM**([Caractere] FROM [String])

Remove o caractere de espaço ou outros caracteres especificados do início ou final de uma cadeia de caracteres.

## **LEN**([String])

Retorna o número de caracteres da expressão da cadeia de caracteres especificada, ignorando espaços em branco à direita.

## **REPLACE**( [String] , [Valor a ser procurado] , [Valor que será colocado])

Substitui todas as ocorrências de um valor da cadeia de caracteres especificado por outro valor de cadeia de caracteres.

Exemplo :

**SELECT REPLACE**(‘Toni’,'T','Ant') as ‘Nome’; Irá mostrar

|  |
| --- |
| Nome |
| Antoni |

# Funções de data

São funções que lidam e manipulam com datas

Exemplos :

## **DAY , MONTH e YEAR** :

Os 3 métodos funcionam da mesma maneira, todos recebem uma data e retornam uma parte dela (**DAY** retorna o dia , **MONTH** o mês e **YEAR** o ano)

## **GETDATE**()

Retorna a data atual do sistema do banco de dados.

## **GETUTCDATE**()

Retorna a data atual do sistema operacional do computador no qual a instância do SQL Server está sendo executada

## **DATEADD([parte],[valor],[data em formato string])**

Essa função adiciona um valor número inteiro específico em uma parte especificada de um valor date de entrada e retorna esse valor modificado.

*Parte* qual parte será modificada da data (day,month,year,hour,minute,etc..)

Exemplo :

**SELECT DATEADD**(day, 31, ‘2000-01-01’);

Irá mostrar : ‘2000-02-01 00:00:00’

-- O retorno por algum motivo é sempre com horário (mesmo quando você não adicionar )