Tipos de dados em constantes:

'Texto' = varchar

N'Testo' = nvarchar

0x12Ef = varbinary

{D '1753-02-22'} = datetime nao nulo

{D '9999-12-31'} = datetime não nulo

{t '18:22:11'} = datetime nulo

{ts '1753-02-22 18:22:11'} = datetime não nulo

12.0 = decimal(3,1)

+0012.0000 = decimal(6,4) (zeros a esquerda não aumentam a precisão)

12.0E0 = float

$-12.0 = money(19,4)

'2022-02-02' = Data que é convertida na execução da consulta

{d'2022-02-02'} = Data que é convertida na compilação da consulta

Tipo float não é preciso:

declare @x float = 11111

declare @y float = -11111

declare @z float = 0.1

select case when (@x + @y) + @z = @x + (@y + @z) then 'Igual' else 'Diferente' end

Tipo decimal é preciso em todas as casas definidas:

declare @x decimal = 11111

declare @y decimal = -11110.1

declare @z decimal = 0.1

select case when (@x + @y) + @z = @x + (@y + @z) then 'Igual' else 'Diferente' end

-- conversão de dados e cálculos

Função para mostrar o tipo de dados:

CREATE FUNCTION [Tipo](@p AS SQL\_VARIANT) RETURNS NVARCHAR(100) AS BEGIN

 DECLARE @t AS NVARCHAR(100)

 DECLARE @i AS NVARCHAR(100)

 SET @t=CAST(SQL\_VARIANT\_PROPERTY(@p,'BaseType') AS NVARCHAR(MAX))

 SET @I=CASE

 WHEN @t LIKE '%binary%' THEN

 concat('(',CAST(SQL\_VARIANT\_PROPERTY(@p,'TotalBytes') AS NVARCHAR(MAX)),')')

 WHEN @t LIKE '%char%' THEN

 concat('(',CAST(SQL\_VARIANT\_PROPERTY(@p,'MaxLength') AS NVARCHAR(MAX)),

 ') ',CAST(SQL\_VARIANT\_PROPERTY(@p,'Collation') AS NVARCHAR(MAX)))

 ELSE

 concat('(',CAST(SQL\_VARIANT\_PROPERTY(@p,'Precision') AS NVARCHAR(MAX)),

 ',',CAST(SQL\_VARIANT\_PROPERTY(@p,'Scale') AS NVARCHAR(MAX)),')')

 end

 RETURN CONCAT(@t,@i)

END

Conversões de dados implícitas em cálculos:

Quando um operador combinar duas expressões com tipos de dados diferentes, o tipo de dados com a precedência mais baixa será convertido no tipo de dados de maior precedência. Se a conversão não for uma conversão implícita com suporte, será retornado um erro. Para um operador combinando expressões de operando que tem o mesmo tipo de dados, o resultado da operação terá esse tipo de dados.

O SQL Server usa a seguinte ordem de precedência para tipos de dados:

1. UDT (tipos de dados definidos pelo usuário) (maior)
2. **sql\_variant**
3. **xml**
4. **datetimeoffset**
5. **datetime2**
6. **datetime**
7. **smalldatetime**
8. **date**
9. **time**
10. **float**
11. **real**
12. **decimal**
13. **money**
14. **smallmoney**
15. **bigint**
16. **int**
17. **smallint**
18. **tinyint**
19. **bit**
20. **ntext**
21. **text**
22. **imagem**
23. **timestamp**
24. **uniqueidentifier**
25. **nvarchar** [incluindo **nvarchar(max)**]
26. **nchar**
27. **varchar** [incluindo **varchar(max)**]
28. **char**
29. **varbinary** [incluindo **varbinary(max)**]
30. **binary** (mais baixo)

Para tipos numeric/decimal, a escala e a precisão são:

| **Operação** | **Precisão de resultado** | **Escala de resultado 1** |
| --- | --- | --- |
| e1 + e2 | max(s1, s2) + max(p1 - s1, p2 - s2) + 1 | máx(s1, s2) |
| e1 - e2 | max(s1, s2) + max(p1 - s1, p2 - s2) + 1 | máx(s1, s2) |
| e1 \* e2 | p1 + p2 + 1 | s1 + s2 |
| e1 / e2 | p1 - s1 + s2 + máx(6, s1 + p2 + 1) | máx(6, s1 + p2 + 1) |
| e1 { UNION | EXCEPT | INTERSECT } e2 | max(s1, s2) + max(p1 - s1, p2 - s2) | máx(s1, s2) |
| e1 % e2 | min(p1 - s1, p2 - s2) + max(s1, s2) | máx(s1, s2) |

Exemplos:

SET @t = '1' + '2' + 1

SELECT @t,dbo.Tipo(@t)

SET @t = '1' + 2 + 1

SELECT @t,dbo.Tipo(@t)

SET @t = 0.12345678 / 10000000000000000000000000000000

SELECT @t,dbo.Tipo(@t)

Arredondamento:

É feito arredondamento nas conversões implicitas de tipo (como por exemplo, colocar um valor com mais precisão em um campo com menos precisão)

Para **truncar** o valor, usar a função ROUND( xxx, xxx, 1)

DECLARE @t AS SQL\_VARIANT

SET @t = 1.00/7.00

SELECT @t,dbo.Tipo(@t)

SET @t = CAST(-1.00/7.00 AS DECIMAL(9,3))

SELECT @t,dbo.Tipo(@t)

SET @t = ROUND(-1.00/7.00 ,3)

SELECT @t,dbo.Tipo(@t)

SET @t = ROUND(-1.00/7.00 ,3 ,1)

SELECT @t,dbo.Tipo(@t)

SET @t = CAST(6.666 AS INT)

SELECT @t,dbo.Tipo(@t)

SET @t = CAST('59.234' AS NUMERIC(15,4))

SELECT @t,dbo.Tipo(@t)

Cuidado no arredondamento de somas com divisão:

DECLARE @t1 AS DECIMAL(10,2) = 10/3.0

DECLARE @t2 AS DECIMAL(10,2) = 10/3.0

DECLARE @t3 AS DECIMAL(10,2) = 10/3.0

DECLARE @t AS DECIMAL(10,2) = @t1 + @t2 + @t3

DECLARE @td AS DECIMAL(10,2) = (10/3.0) + (10/3.0) + (10/3.0)

SELECT @t,dbo.Tipo(@t)

SELECT @td,dbo.Tipo(@t)

Diferença entre soma de valores arredondados e arredondamento da soma de valores:

SELECT SUM(c/3) FROM c

SELECT CAST(SUM(c/3) as DECIMAL(5,2)) FROM c

SELECT SUM(CAST(c/3 as DECIMAL(5,2))) FROM c

Conversões de/para texto:

Funções PARSE e FORMAT tratam a linguagem definida na sessão

SET LANGUAGE Brazilian

SELECT PARSE('12.345' AS DECIMAL(10,2))

SELECT FORMAT(59876.234,'#,###.00000')

SET LANGUAGE English

SELECT PARSE('12.345' AS DECIMAL(10,2))

SELECT FORMAT(59876.234,'#,###.00000')

Conversão de DATAS:

CAST Leva em consideração a linguagem definida e o formato de data

SET DATEFORMAT mdy

SELECT CAST('1/2/2023' AS DATE)

SET DATEFORMAT dmy

SELECT CAST('1/2/2023' AS DATE)

Exceto em formatos de AAAA-MM-DD

Para conversão para texto, mais prático usar FORMAT:

SELECT FORMAT(CAST(@t AS DATE),'dd/MM/yyyy')

SELECT FORMAT(CAST(@t AS DATE),'yyyyMMdd')

Exemplo de nome de mês que dá problema:

DECLARE @yourInputDate NVARCHAR(32) = '28 listopad 2018';

SET LANGUAGE Polish;

SELECT CONVERT(DATE, @yourInputDate) AS [SL\_Polish];

SET LANGUAGE Croatian;

SELECT CONVERT(DATE, @yourInputDate) AS [SL\_Croatian];

SET LANGUAGE English;

Datas Parciais:

Mais legível:

DECLARE @d date = {D '2023-08-02'}

SELECT YEAR(@d)\*100 + MONTH(@d)

Para cálculo de diferença de meses:

DECLARE @d date = {D '2023-08-02'}

SELECT YEAR(@d)\***12** + MONTH(@d)

ATENÇÃO para mêses 13 e 14!!!!!

AnoMes para data:

DECLARE @AnoMes INT = 202308

SELECT DATEFROMPARTS( @AnoMes / 100 , @AnoMes % 100 , 1 ) AS Primeiro\_Dia

SELECT EOMONTH( DATEFROMPARTS( @AnoMes / 100 , @AnoMes % 100 , 1 )) AS Ultimo\_Dia

CHAR x VARCHAR

DECLARE @T1 AS CHAR(12) = 'A'

DECLARE @t2 AS VARCHAR(12) = 'A'

SELECT @T1,@t2,CASE WHEN @T1=@t2 THEN 'Igual' ELSE 'não' END

DECLARE @T3 AS VARCHAR(200) = @T1 + 'X'

DECLARE @T4 AS VARCHAR(200) = @T2 + 'X'

SELECT @T3,@T4,CASE WHEN @T3=@t4 THEN 'Igual' ELSE 'não' END

Uso de NULL em comparações:

NULL nas comparações gera um resultado do tipo UNKNOWN

SET ANSI\_NULLS OFF;

SELECT CASE WHEN NULL=NULL THEN 'Igual' ELSE 'não' END

SET ANSI\_NULLS ON;

SELECT CASE WHEN NULL=NULL THEN 'Igual' ELSE 'não' END

with tt as (select \* from (values (1),(2),(NULL)) AS X(v)),

ta as (

 select t1.v as t1,t2.v as t2

 from tt as t1 CROSS join tt as t2

)

SELECT ta.t1,ta.t2,c1.[=],c5.[NOT =],c2.[!=],c7.[IS DISTINCT FROM],c8.[IS NOT DISTINCT FROM]

 ,c3.[= CASE],c4.[= CASE ELSE],c6.[= COALESCE] FROM ta

 LEFT JOIN (SELECT t1,t2,'=' AS [=] from ta WHERE

 t1 = t2 ) c1 ON c1.t1 IS NOT DISTINCT FROM ta.t1 AND c1.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'NOT =' AS [NOT =] from ta WHERE

 NOT t1 = t2 ) c5 ON c5.t1 IS NOT DISTINCT FROM ta.t1 AND c5.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'!=' AS [!=] from ta where

 t1 != t2 ) c2 ON c2.t1 IS NOT DISTINCT FROM ta.t1 AND c2.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'= CASE' AS [= CASE] from ta WHERE

 CASE WHEN t1 = t2 THEN 1 ELSE 0 END =1 ) c3 ON c3.t1 IS NOT DISTINCT FROM ta.t1 AND c3.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'= CASE ELSE' AS [= CASE ELSE] from ta WHERE

 CASE WHEN t1 = t2 THEN 0 ELSE 1 END =1 ) c4 ON c4.t1 IS NOT DISTINCT FROM ta.t1 AND c4.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'= COALESCE' AS [= COALESCE] from ta WHERE

 COALESCE(t1,-9999)=COALESCE(t2,-9999) ) c6 ON c6.t1 IS NOT DISTINCT FROM ta.t1 AND c6.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'IS DISTINCT FROM' AS [IS DISTINCT FROM] from ta WHERE

 t1 IS DISTINCT FROM t2 ) c7 ON c7.t1 IS NOT DISTINCT FROM ta.t1 AND c7.t2 IS NOT DISTINCT FROM ta.t2

 LEFT JOIN (SELECT t1,t2,'IS NOT DISTINCT FROM' AS [IS NOT DISTINCT FROM] from ta WHERE

 t1 IS NOT DISTINCT FROM t2 ) c8 ON c8.t1 IS NOT DISTINCT FROM ta.t1 AND c8.t2 IS NOT DISTINCT FROM ta.t2

ORDER BY 1,2

TRY\_CAST / TRY\_CONVERT / TRY\_PARSE

COLLATION

SELECT CASE WHEN 'A' = 'á' THEN 'Igual' ELSE 'não' END

SELECT CASE WHEN 'A' COLLATE Latin1\_General\_100\_CI\_AI = 'á' THEN 'Igual' ELSE 'não' END

SELECT 'Teste de acentuação'

SELECT 'Teste de acentuação' Collate SQL\_Latin1\_General\_CP1253\_CI\_AI

CREATE FUNCTION [g].[Remove\_Acentos]( @Texto nvarchar(max)) RETURNS nvarchar(max) WITH RETURNS NULL ON NULL INPUT AS BEGIN

RETURN cast(@Texto as varchar(max)) Collate SQL\_Latin1\_General\_CP1253\_CI\_AI

END