

# Módulo 3

## Transact SQL

Tecnara – Cursos de formación



## Índice de contenidos

1. SQL y Transact-SQL
2. Entity Framework



05

## SQL y Transact-SQL

# SQL y T-SQL

## El lenguaje SQL

- Acrónimo de Structured Query Language (Lenguaje de Consultas estructurado).
- Estándar ANSI/ISO
- Combina en un mismo lenguaje elementos de Definición de Datos, de Manipulación de Datos y de Consulta de Datos.

# SQL y T-SQL

- Microsoft (propietario de SQL Server) coge SQL y le añade comandos, tipos de datos y funciones que no están incluidos en el estándar. También le añade transacciones.
- Al nuevo lenguaje lo llama Transact SQL (T-SQL).
- Siempre que se trabaje con SQL Server, hay que pensar que se está trabajando con esa versión extendida de SQL.
- A continuación se revisan los tipos de datos y funciones disponibles en T-SQL que se pueden necesitar al realizar consultas. Para la creación/modificación/eliminación de bases de datos y tablas, aprenderemos a utilizar las herramientas visuales que ofrece MSSMS (Microsoft SQL Server Management Studio).

# T-SQL: Tipos de datos

## Numéricos exactos

- **Bit**: valores 0 o 1 (para cualquier valor distinto de 0)  
(realmente usa un byte: 8 bits, para guardar el valor)
- **Tinyint**: valores de 0 a 255 (tamaño en memoria: 1 byte)
- **Smallint**: valores de  $-2^{15}$  a  $2^{15} - 1$  (2 bytes)
- **Int**: valores de  $-2^{31}$  a  $2^{31} - 1$  (4 bytes)
- **Bigint**: valores de  $-2^{63}$  a  $2^{63} - 1$  (8 bytes)



# T-SQL: Tipos de datos

## Numéricos exactos

- Numeric/**Decimal**: valores de  $-10^{38} + 1$  a  $10^{38} - 1$ . (Precisión/escala) (entre 5 y 17 bytes)
  - Numeric(p,s): p entre 1 y 38 (por defecto 18). S entre 0 y p (por defecto 0)
- Smallmoney: valores de -214,748.3648 a 214,748.3648 (escala=4) (4 bytes)
- Money: valores de  $-2^{63}$  a  $2^{63} - 1$  (escala=4) (8 bytes)

# T-SQL: Tipos de datos

## Numéricos aproximados

- Real: valores de  $-3.40\text{E}+38$  a  $-1.18\text{E}-38$ , 0 y de  $1.18\text{E}-38$  a  $3.40\text{E}+38$  (4 bytes)
- Float: valores de  $-1.79\text{E}+308$  a  $-2.23\text{E}-308$ , 0 y de  $2.23\text{E}-308$  a  $1.79\text{E}+308$  (8 bytes)



# T-SQL: Tipos de datos

## Fecha/hora

- `Smalldatetime`: fechas entre 01/01/1900 y 06/06/2079, precisión: 1 min (4 bytes)
- **`Datetime`**: Fechas entre 01/01/1753 y 31/12/9999, precisión: 3,33 ms (8 bytes)

# T-SQL: Tipos de datos

## Fecha/hora

- Date: Fechas entre 01/01/01 y 31/12/9999. Solo día-mes-año. (3 bytes)
- Time: Horas entre las 00:00:00.0000000 y las 23:59:59.9999999 (5 bytes)
- Datetime2: combina los tipos Date y Time (8 bytes)
- Datetimeoffset: Datetime2, pero considerando diferencia horaria con UTC. Ej: 2019-07-15 20:44:35.1933333 +00:00 (10 bytes)

# T-SQL: Tipos de datos

## Cadenas de caracteres

### No Unicode (ASCII)

- Char: tamaño fijado, entre 0 y 8000 caracteres (por defecto 30 bytes)
- Varchar: tamaño variable, entre 0 y 8000 caracteres (1 byte/carácter)
- Varchar(max) / Text: tamaño variable, entre 0 y  $2^{31} - 1$  caracteres

### Unicode (más caracteres disponibles, pero ocupan el doble de espacio)

- Nchar: tamaño fijado, entre 0 y 4000 caracteres (por defecto 60 bytes)
- **Nvarchar**: tamaño variable, entre 0 y 4000 caracteres (2 bytes/carácter)
- Nvarchar(max) / Ntext: tamaño variable, entre 0 y  $2^{30} - 1$  caracteres

# T-SQL: Tipos de datos

## Cadenas binarias (mostradas en Hexadecimal)

- Binary: tamaño fijado, entre 0 y 8000 bytes (por defecto 30 bytes)
- Varbinary: tamaño variable, entre 0 y 8000 bytes (1 byte/carácter)
- Varbinary(max) / Image: tamaño variable, entre 0 y  $2^{31} - 1$  bytes



# T-SQL: Tipos de datos

## Otros

- `Sql_variant`: contenedor de tamaño fijo para guardar distintos tipos.
- `Timestamp` / `Rowversion`: valor único a nivel de BD que se actualiza cada vez que se actualiza un registro.
- `Uniqueidentifier`: GUID (Global Unique Identifier).
- `Xml`: para guardar información estructurada en base a un esquema.
- `Hierarchyid`: para guardar paths dentro de un árbol de jerarquías
- `Geometry`: datos espaciales en 3 coordenadas: (X, Y, Z)
- `Geography`: datos espaciales en 2 coordenadas: (latitud, longitud)

# T-SQL: Tipos de datos

## Cambio de tipo

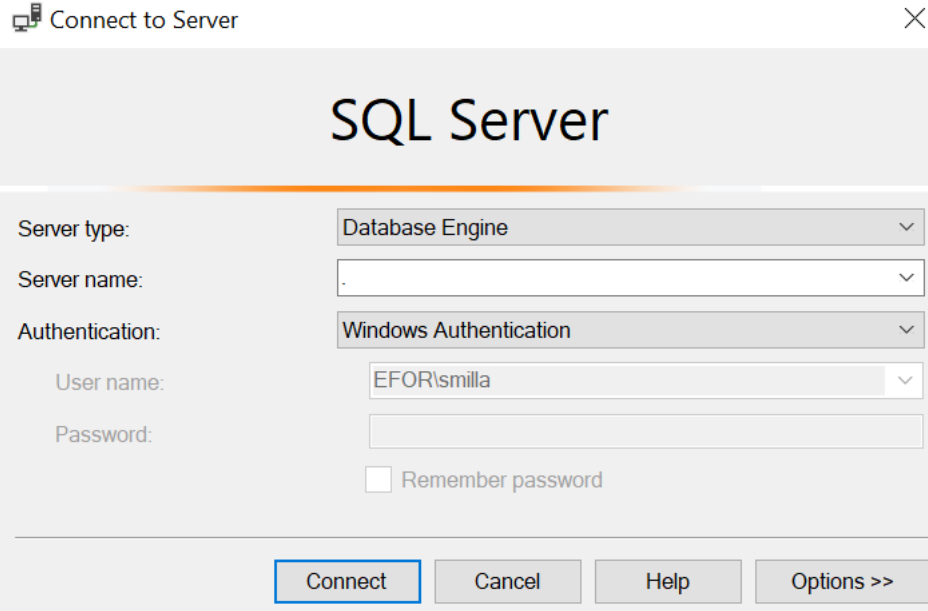
- `CAST('1' as float) / CONVERT(float, '1')`
- `CONVERT(varchar, cast('2017-08-25' as date), 100)`

<https://www.mssqltips.com/sqlservertip/1145/date-and-time-conversions-using-sql-server/>

# Ejecutando Management Studio

Para acceder a la BD local desde el programa Management Studio, hay que introducir los datos de acceso a la misma:

- Tipo de servidor: Motor de Base de datos.
- Nombre de servidor: el nombre de la máquina local
  - El nombre de la máquina local (EFOR para la imagen) se puede sustituir por el carácter punto '.' o por '.' para acortar y simplificar
- Autenticación: Autenticación Windows, con el usuario actual (marcado por defecto)



Connect to Server

## SQL Server

Server type: Database Engine

Server name: .

Authentication: Windows Authentication

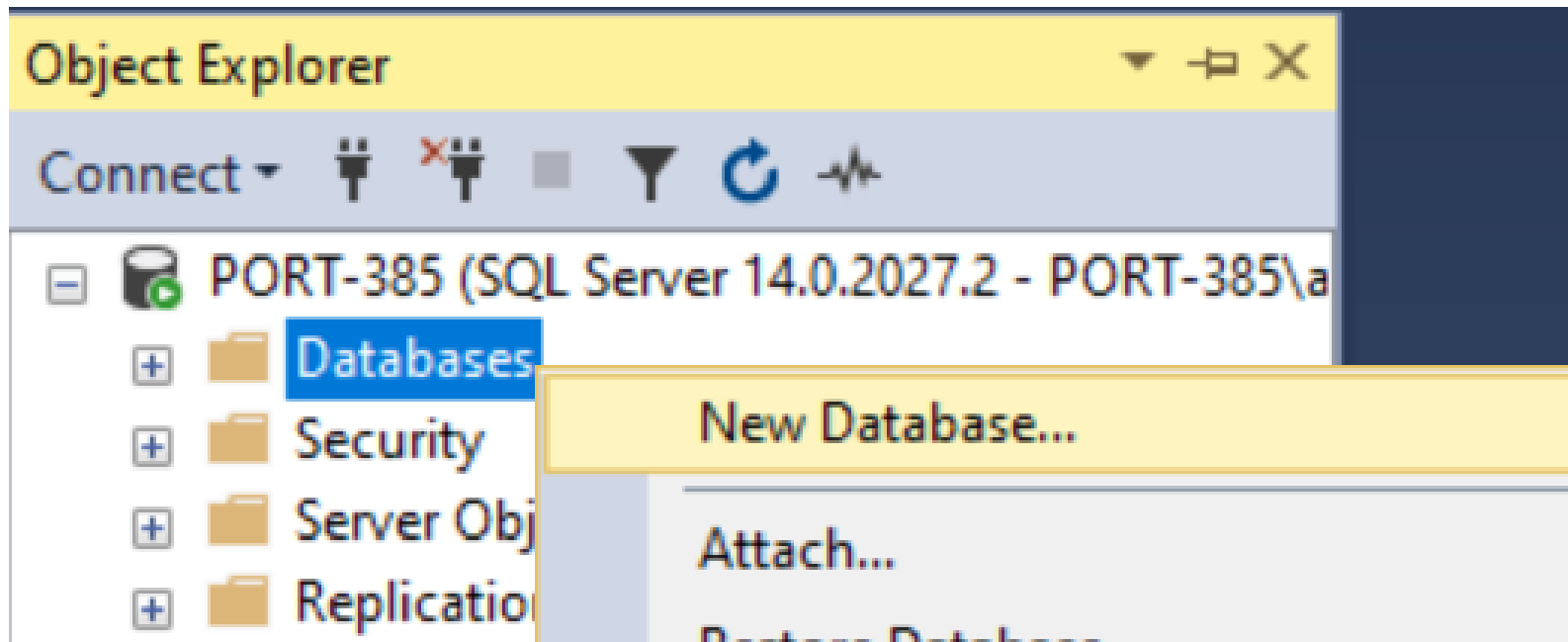
User name: EFOR\smilla

Password:

☐ Remember password

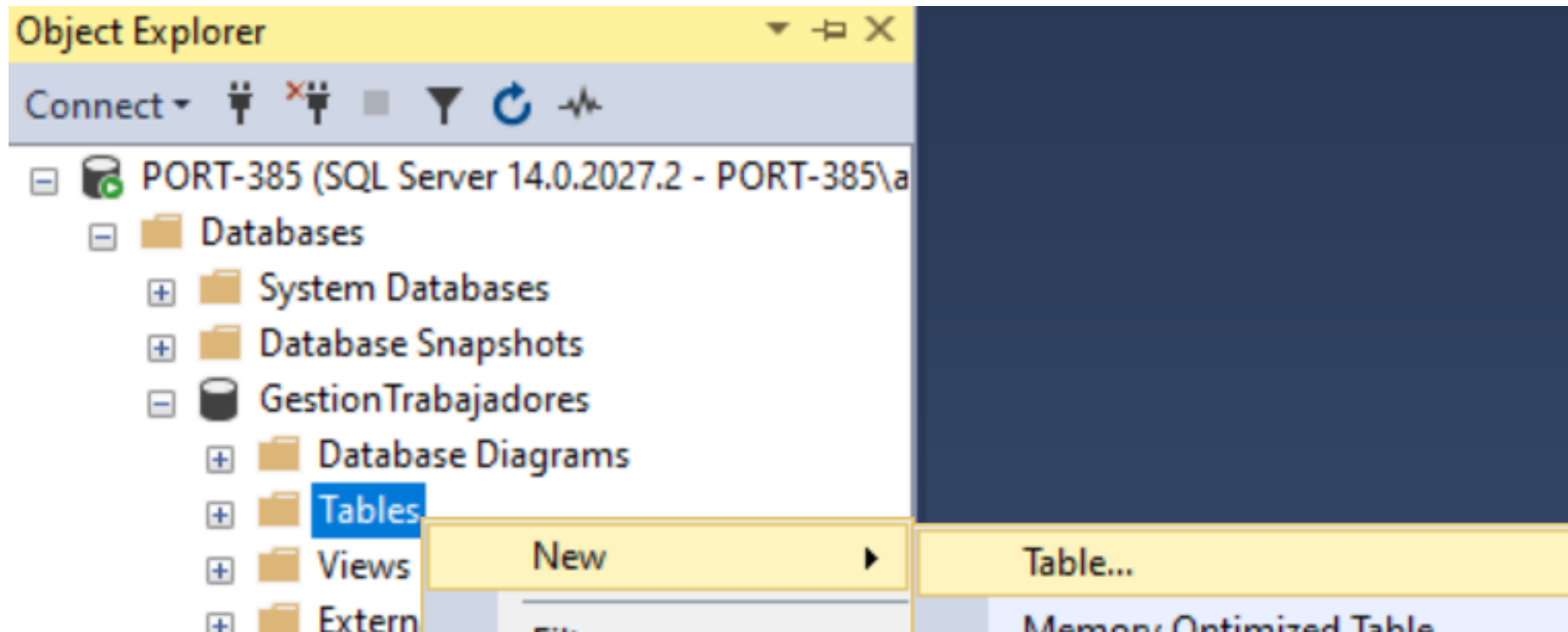
Connect Cancel Help Options >>

# T-SQL: Crear BD con SSMS





# T-SQL: Crear tabla con SSMS



# T-SQL: consultas a tablas

Select \* from personas

SELECT \* FROM PERSONAS

sElEcT \* FrOm pErSONaS

Select nombre, apellidos from personas

Select nombre as Nom, apellidos as Ape from personas

Select nombre Nom, apellidos Ape from personas

¡¡Si hay un texto seleccionado y se pulsa botón ejecutar consulta, se intenta ejecutar solo el texto seleccionado!!

# T-SQL: consultas a tablas

Select nombre Nom, apellidos Ape, Nom+' '+Ape NomApe from personas

Select 'Soy ' + trim(nombre) + ', con email ' + trim(mail) AS Description from personas

# T-SQL: consultas a tablas

Select 'hola mundo' saludo

Select 'hola mundo' saludo, 3 número, 5.7 decimal

Select cast('hola mundo' as varchar(20)) saludo, 3 número, 5.7 decimal

Select cast('hola mundo' as varchar(3)) saludo, 3 número, 5.7 decimal

Select cast('hola mundo' as varchar) saludo, 3 número, 5.7 decimal



# T-SQL: consultas a tablas

```
Select *, 5.7 t2_decimal from  
(Select cast('hola mundo' as varchar) t1_saludo, 3 t1_número) a
```

¡¡Esta estructura de consulta facilita probar el funcionamiento de operadores y funciones!!

- T1 tiene los parámetros que usa la función o el operador
- T2 ejecuta la función o el operador con los parámetros de T1

```
Select p1, p2, {p1 operador p2/función(p1, p2)} resultado from (Select {p1} p1, {p2} p2) a
```

# T-SQL: consultas a tablas

```
Select *, 5.7 t2_decimal from  
(Select cast('hola mundo' as varchar) t1_saludo, 3 t1_número) a
```

```
Select *, 5.7 t2_decimal from  
(Select cast('hola mundo' as varchar) t1_saludo, 3 t1_número  
UNION ALL  
Select 'segunda fila', 5  
) a
```

# T-SQL: consultas a tablas

Acabamos de ver cómo SIMULAR tablas y cómo hacer consultas sobre esas tablas simuladas

- Útil para aprender. Útil para resolver dudas sobre el resultado que devolverá una determinada función u operador para un determinado valor o conjunto de valores.
- Normalmente inútil para el trabajo diario de hacer consultas, ya que se suelen hacer sobre tablas reales, no simuladas.
- Puede haber una condición que no se cumple con ninguno de los datos actuales de una tabla, pero se prevé que se pueda cumplir en el futuro. Para probar esa condición, puede venir bien simular el dato.

# T-SQL: consultas a tablas

`SELECT [0][1] FROM [2] WHERE [3] [4];`

[0]: TOP / DISTINCT

[1]: Columna(s) que se mostrará(n) en la tabla resultado.

[2]: Tabla(s) donde se va a buscar la información a devolver  
(pueden aplicarse filtros de relación con sentencias JOIN)

[3]: Filtros que se aplicarán a cada registro devuelto

[4]: agrupaciones /ordenaciones / paginación



# T-SQL: consultas a tablas

Consultas de información de una sola tabla, filtros generales:

`SELECT * from Personas //` sin filtros

`SELECT DISTINCT nombre from Personas //` quitar resultados iguales

`SELECT TOP 3 nombre from personas //` solo los tres primeros resultados

`SELECT TOP 25 PERCENT nombre from personas order by nombre //` solo los resultados pertenecientes al 1º cuarto del total

Consulta paginada:

`SELECT * from personas order by Nombre OFFSET 10 FETCH NEXT 10 ROWS ONLY`

# T-SQL: consultas a tablas

Generación de estadísticas a partir de los datos. Totalizadores:

`SELECT [TOTALIZADOR(nombre_columna)] from table`

- `AVG()` // media aritmética de todos los elementos
- `COUNT()` // nº total de elementos
- `MAX()` // máximo valor de entre todos los elementos
- `MIN()` // mínimo valor de entre todos los elementos
- `SUM()` // suma de todos los elementos
- `VAR()` // varianza de todos los elementos
- `STDEV()` // desviación típica de todos los elementos

`SELECT SUM(salario) from Trabajadores`

# T-SQL: consultas a tablas

Generación de estadísticas por grupos. Agrupaciones:

```
SELECT Nombre, Count(Nombre) from Personas GROUP BY Nombre
```

```
SELECT Nombre, Count(Nombre) from Personas GROUP BY Nombre ORDER BY Name Desc
```

```
¿SELECT Nombre, Apellidos, Count(Nombre) from Personas GROUP BY Nombre?
```

Concatenación de listas de registros del mismo tipo. Uniones:

```
SELECT Nombre from Personas UNION SELECT Nombre from Empresas e // Select distinct
```

```
SELECT Nombre from Personas UNION ALL SELECT Nombre from Empresas e // no distinct
```

# T-SQL: consultas a tablas

Consultas que involucran varias tablas, relacionadas o no (son la base de las BDRs):

`SELECT p.Nombre, e.Nombre from Personas p, Empresas e // combinación todos con todos`

`SELECT p.Nombre, e.Nombre from Personas p INNER JOIN Empresas e ON p.IDEmpresa = e.ID`

`SELECT p.Nombre, e.Nombre from Personas p LEFT JOIN Empresas e ON p.IDEmpresa = e.ID`

`SELECT p.Nombre, e.Nombre from Personas p RIGHT JOIN Empresas e ON p.ID = e.IDPersonaDirectorRRHH`

`SELECT p.Nombre, e.Nombre from Personas p FULL JOIN Empresas e ON p.IDEmpresa = e.ID AND p.ID = e.IDPersonaDirectorRRHH`

# T-SQL: consultas a tablas

Consulta de tipo Join sobre tablas simuladas:

```
SELECT t1.nombre, t2.nombre from
    (select 'alex' nombre, 1 idpuesto
    union select 'brian', 2
    union select 'charles', 1) t1
inner join
    (select 'granjero' nombre, 1 id
    union select 'policía', 2
    union select 'cajero', 3) t2
ON t2.id = t1.idpuesto
```

# T-SQL: consultas a tablas

Consultas con filtros personalizados:

```
SELECT IdEmpresa from Personas WHERE Nombre = 'Jack'
```

```
SELECT * from Trabajadores WHERE Salario BETWEEN 100 AND 200
```

```
SELECT * from Personas WHERE IdEmpresa IS NOT NULL
```

```
SELECT nombre from Personas WHERE apellidos like 'G%'
```

```
SELECT Nombre, Count(Nombre) from Personas GROUP BY Nombre HAVING Nombre like 'G%'
```

```
SELECT * from Personas p where EXISTS(select * from Trabajadores t where p.ID = t.IDPersona AND t.Salario BETWEEN 100 AND 200)
```

# T-SQL: consultas a tablas

Consultas con filtros personalizados que aplican operaciones de conjuntos:

```
SELECT IdEmpresa from Personas WHERE Nombre = ANY(select 'Jim' union select 'Jack')
```

```
SELECT IdEmpresa from Personas WHERE Nombre = ALL(select 'Jack' union select 'Jack')
```

```
SELECT * from Trabajadores where IDPersona IN (select ID from Personas where  
Left(Nombre, 1) between 'm' and 'r') and Salario > 200
```

```
SELECT Nombre from Personas where Nombre NOT IN (SELECT TOP 25 percent Nombre  
FROM Personas order by Nombre) order by Nombre
```



# T-SQL: consultas a tablas

Operadores matemáticos que se pueden aplicar en consultas (tanto en definición de columnas calculadas a devolver como en condiciones de filtros a aplicar)

- $5+3 = 8$ 
  - `Select p1, p2, p1+p2 resultado from (Select 5 p1, 3 p2) a`
- $5-3 = 2$
- $5*3 = 15$
- $5/3 = 1$
- $5.0/3 = 5/3.0 = 5.0/3.0 = 1.666666...$
- $5\%3 = 2$

# T-SQL: consultas a tablas

## Operadores lógicos a nivel de bit

- $4 \& 3 = 0$  --  $0100 \text{ AND } 0011 = 000 = 0$  // AND lógico
- $4 | 3 = 7$  --  $0100 \text{ OR } 0011 = 0111 = 7$  // OR lógico
- $12 \wedge 5 = 9$  --  $1100 \text{ XOR } 0101 = 1001 = 9$  // XOR lógico
- $\text{CAST}(6 \text{ as varbinary}) = 0x06 = 00000110$
- $\text{CAST}(\sim 6 \text{ as varbinary}) = 0xF9 = 11111001 = -7$  (Complemento a 2) // NOT lógico

# T-SQL: consultas a tablas

## Funciones para tratar con cadenas de caracteres

- `ASCII('a') = 97`
- `CHAR(97) = 'a'`
- `CHARINDEX('t', 'Customer') = 4`
- `CONCAT('hola', ' ', 'mundo') = 'hola mundo'`
- `'hola' + ' ' + 'mundo' = 'hola mundo'`
- `CONCAT_WS('/', '12', '03', '2019') = 12/03/2019`
- `DATALENGTH('holaquetal') = 10` -- considera espacios al principio y al final
- `DATALENGTH(3) = 4`
- `DIFFERENCE` -- compara dos valores Soundex, no se verá aquí
- `SELECT FORMAT (cast('12/01/2018' as datetime), 'MM/dd-yy') = 12/01-18`
- `SELECT FORMAT (123456789, '<# #:##-###_##>') = <1 2:34-567_89>`

# T-SQL: consultas a tablas

## Funciones para tratar con cadenas de caracteres

- `LEFT('Hola', 3) = 'Hol'`
- `LEN('Hola') = 4` -- considera espacios solo al principio: `LEN('_Hola _')` = 5
- `LOWER('SQL Server') = 'sql server'`
- `LTRIM(' Hola ') = 'Hola '`
- `NCHAR(190) = '¾'` -- tabla caracteres Unicode: <https://unicode-table.com/es>
- `PATINDEX('%a m%', 'Hola Mundo') = 4`
- `QUOTENAME('hola', '<') = '<hola>'` -- delimitadores: <>, ", ` , "" , [], {}, ()
- `REPLACE('HOLA mundo', 'O', 'oh') = 'HohLA mundoh'`
- `REPLICATE('hola mundo', 3) = 'hola mundohola mundohola mundo'`
- `REVERSE('palíndromo') = 'omordnÍlap'`

# T-SQL: consultas a tablas

## Funciones para tratar con cadenas de caracteres

- `RIGHT('Hola', 3) = 'ola'`
- `RTRIM(' hola ') = ' hola'`
- `SOUNDEX` genera el valor Soundex de un texto. No se verá aquí.
- `'a'+SPACE(10)+'b' = 'a _ _ _ _ _ _ _ _ _ _ b'`
- `STR(185) = '_ _ _ _ _ _ _ 185'` -- por defecto, la longitud de la cadena es 10
- `STR(185.46, 8, 3) = '_185.460'` -- `LTRIM(STR())` elimina los espacios que haya
- `STUFF('Hola', 2, 2, 'amac') = 'Hamaca'`
- `SUBSTRING('Hola', 2, 2) = 'ol'`
- `TRANSLATE('Arquitecto', 'aeiou', '@310V') = '@rqV1t3ct0'`

# T-SQL: consultas a tablas

## Funciones para tratar con cadenas de caracteres

- `TRANSLATE('hAla', 'aA', '12') = 'h1l1'`
- `TRANSLATE('hAla' COLLATE Latin1_General_CS_AS, 'aA', '12') = 'h2l1'`
- `'-'+TRIM('      Hola      ')+'-' = '-Hola-'`
- `TRIM('.*-+' FROM '..**--++Ho.la++--**..') = 'Ho.la'`
- `UNICODE('¾') = 190` -- tabla caracteres Unicode: <https://unicode-table.com/es>
- `UPPER('SQL Server') = 'SQL SERVER'`

# T-SQL: consultas a tablas

## Funciones para tratar con números individuales

- $\text{PI}() = 3.14169265\dots$  (número pi)
- $\text{EXP}(1) = 2.7182818\dots$  (número e)
- $\text{ABS}(-35) = 35$
- $\text{ACOS}(1) = 0.0$  -- el parámetro debe estar entre -1 y 1
- $\text{ASIN}(0) = 0.0$
- $\text{ATAN}(0) = 0.0$
- $\text{ATN2}(0, -\text{PI}()) = \text{PI}$
- $\text{CEILING}(2.4) = 3$
- $\text{COS}(0) = 1.0$



# T-SQL: consultas a tablas

## Funciones para tratar con números individuales

- $\text{COT}(\text{PI}()/4) = 1.0$
- $\text{DEGREES}(\text{PI}()) = 180.0$
- $\text{FLOOR}(25.8) = 25$
- $\text{LOG}(\text{EXP}(2)) = 2.0$  -- logaritmo natural o neperiano (base e)
- $\text{LOG10}(100000) = 5.0$
- $\text{POWER}(2, 3) = 8.0$
- $\text{RADIANS}(90,0) = \text{PI}/2$
- $\text{RAND}() = (\text{número aleatorio entre } 0 \text{ y } 1)$
- $\text{ROUND}(1.123, 2) = 1.12$

# T-SQL: consultas a tablas

## Funciones para tratar con números individuales

- $\text{SIGN}(12) = 1.0$
- $\text{SIGN}(-12) = -1.0$
- $\text{SIN}(0) = 0.0$
- $\text{SQRT}(441) = 21$
- $\text{SQUARE}(21) = 441$
- $\text{TAN}(0) = 0.0$

# T-SQL: consultas a tablas

## Funciones para tratar con fechas

- `CURRENT_TIMESTAMP/GETDATE()` = 2019-07-15 13:25:03.730
- `GETUTCDATE()` = 2019-07-15 12:25:14.952
- `SYSDATETIME()` = 2019-07-15 13:25:27.1822487
- `DATEADD(year, 1, cast('2019/07/15' as date))` = 2020-07-15
- `DATEDIFF(month, '2019/08/25', '2018/08/25')` = -12
- `DATEFROMPARTS(2019, 7, 15)` = 2019-07-15
- `DATENAME(month, '2019/07/15')` = Julio
- `DATEPART(month, '2019/07/15')` = 7
- `DAY('2019/07/25')` = 25

# T-SQL: consultas a tablas

## Funciones para tratar con fechas

- `ISDATE('2019/25/07') = 1`
- `MONTH('2019/07/25') = 7`
- `YEAR('2019/07/25') = 2019`

# T-SQL: consultas a tablas

## Otras funciones

- `CAST('1' as float) / CONVERT(float, '1')`
- `CONVERT(varchar, cast('2017-08-25' as date), 100)`
- `DATALENGTH(CAST('1' as bigint)) = 8`
- `COALESCE(null, 0.5, 1, null, '2019/07/25', null) = 0.5`
- `CURRENT_USER`
- `SESSION_USER`
- `SYSTEM_USER`
- `USER_NAME()`

# T-SQL: consultas a tablas

## Otras funciones

- `ISNUMERIC(158) = 1`
- `ISNUMERIC('158') = 1`
- `ISNUMERIC('hola') = 0`
- `ISNUMERIC(2*3) = 1`
- `ISNUMERIC('2019-07-08') = 0`

# T-SQL: consultas a tablas

## Funciones condicionales

- `ISNULL(158, 0) = 158`
- `ISNULL(null, 0) = 0`
- `NULLIF('hola', 'hola') = null` -- debería llamarse `NULLIFEQUAL...`
- `NULLIF('hola', 'adiós') = 'hola'`
- `IIF(1=1, 'hola', 'adiós') = 'hola'`
- `IIF(1=0, 'hola', 'adiós') = 'adiós'`



# T-SQL: consultas a tablas

## Funciones condicionales

- `CASE WHEN 1=1 THEN 'hola' WHEN 1=0 THEN 'adiós' ELSE 'nada' END`
- `CHOOSE(1, 'UNO', 'DOS') = 'UNO'`

# T-SQL: consultas a tablas

## Operadores de comparación

- `IIF(4=3, 'true', 'false') = 'false'`
- `IIF(4=4, 'true', 'false') = 'true'`
- `IIF(4>3, 'hola', 'adiós') = 'hola'`
- `IIF(3<4, 1, 0) = 1`
- `IIF(3<=4, 3.14, 2.7) = 3.14`
- `IIF(3>=4, 'igualsí', 'lodudo') = 'lodudo'`
- `IIF(3<>4, 'igualsí', 'lodudo') = 'igualsí'`

# T-SQL: consultas a tablas

## Operadores de comparación para cadenas de texto

- `IIF('hola'='HOLA', 'true', 'false') = 'true'`
- `IIF('hola' LIKE '%o_a%', 'true', 'false') = 'true'`

## Operadores lógicos

- `IIF(4=4 AND ISNULL('hola'), 'true', 'false') = 'false'`
- `IIF(4=4 OR ISNULL('hola'), 'true', 'false') = 'true'`
- `IIF(4=4 AND '' NOT LIKE '%a%'), 'true', 'false') = 'true'`
- `IIF(4 BETWEEN 1 AND 5, 'hola', 'adiós') = 'hola'`
- `IIF(4 IN (1, 2, 3, 4), 1, 0) = 1`

# T-SQL: consultas a tablas

Referencia del lenguaje SQL:

[www.w3schools.com/sql/](http://www.w3schools.com/sql/)

Formatos predefinidos para la función CONVERT:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql>

06

## Entity Framework



# Entity Framework

- Añadir en carpeta Models nuevo elemento: ADO.NET Entity Data Model
- Conexión con base de datos local: '.', '(local)' o nombre del equipo
- Seleccionar tablas (si hubiera vistas u otros, seleccionarlos también)
- Clase {nombreModelo}Entities: punto de acceso a las tablas de BD
- Añadir referencia a System.Linq para EXPANDIR la funcionalidad disponible para trabajar con las clases que SIMULAN las tablas de BD.
- Enjoy!

# Entity Framework

3 estrategias para elegir cuándo cargar los datos desde la BD y cuándo volcar la info a BD:

- Cargar al principio de la ejecución, volcar al final de la ejecución  
La carga inicial puede ser muy lenta, se pueden estar cargando elementos que no se vayan a consultar ni modificar en toda la ejecución, el volcado final puede ser muy lento, y se pueden estar volcando (sobreescribiendo) elementos que no han cambiado su valor en toda la ejecución. Además, si hay errores de ejecución, se va la luz o sucede algo que finalice el programa sin que se ejecute el código del volcado, se pierden los cambios realizados desde la carga
- Cargar al principio de la ejecución, volcar cuando cambie, lo que cambie  
La carga inicial puede ser muy lenta, y se pueden estar cargando elementos que no se vayan a consultar ni modificar en toda la ejecución
- Cargar cuando se necesite, lo que se necesite, y volcar cuando cambie, lo que cambie  
Puede ser tedioso repetir una y otra vez el código de carga y volcado de cada elemento por separado, pero es la forma más óptima en cuanto a tiempo de ejecución y la más segura para no perder cambios

# MUCHAS GRACIAS



Pasión por la innovación

[www.integratecnologia.es](http://www.integratecnologia.es)