**OPCIONES DE INTERCALACIÓN**: Pueden establecerse a nivel de servidor, de base de datos, de columna o de expresión:

(https://learn.microsoft.com/es-es/sql/relational-databases/collations/collation-and-unicode-support?view=sql-server-ver16#SQL-collations)

|  |  |
| --- | --- |
| BIN | Orden binario. |
| BIN2 | Ordenación punto de código binario. |
| CI | No distingue entre mayúsculas y minúsculas. |
| CS | Sí distingue entre mayúsculas y minúsculas. |
| AI | No distingue acentos. |
| AS | Sí distingue acentos. |
| KS | Sí distingue Kana. |
| WS | Sí distingue ancho. |
|  | En el caso de KS y WS, su ausencia indica que no distingue Kana ni ancho respectivamente. |

**DIFERENTES TIPOS DE DATOS EN SQL SERVER**

**Alfanuméricos:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TAMAÑO FIJO. SE DEBE UTILIZAR CUANDO SEPAMOS QUE LOS DATOS QUE ALBERGARÁ SERÁN SIEMPRE DE LA MISMA LONGITUD | TAMAÑO VARIABLE.  SE DEBE UTILIZAR CUANDO DESCONOCEMOS LA LONGITUD DE LOS DATOS QUE SE ALMACENARÁN | ARGUMENTO DEFINE EL TAMAÑO DE LA CADENA EN BYTES | ARGUMENTO DEFINE EL TAMAÑO DE LA CADENA EN PARES DE BYTES | MÁXIMO VALOR DEL ARGUMENTO | UNICODE |
| CHAR | ✔ |  | ✔ |  | 8000 |  |
| VARCHAR |  | ✔ | ✔ |  | 8000 - MAX |  |
| NCHAR | ✔ |  |  | ✔ | 4000 | ✔ |
| NVARCHAR |  | ✔ |  | ✔ | 4000 - MAX | ✔ |
| TEXT | SERÁ ELIMINADO Y SE RECOMIENDA SUSTITUIR POR VARCHAR(MAX) | | | | | |
| NTEXT | SERÁ ELIMINADO Y SE RECOMIENDA SUSTITUIR POR NVARCHAR(MAX) | | | | | |

CHAR o NCHAR se deben utilizar en campos cuyo contenido sepamos que será siempre de la misma longitud o con variaciones mínimas, por ejemplo, la matrícula de un coche.

Por el contrario, VARCHAR o NVARCHAR los utilizaremos en campos cuya longitud puede ser muy diferente de unos registros a otros, por ejemplo, una dirección postal.

NCHAR y NVARCHAR sólo deberían utilizarse en aplicaciones multilenguaje en las que se vayan a utilizar caracteres Unicode.

**Fechas:**

DATE: Sólo almacena la fecha, es decir: año, mes y día

TIME(n): Sólo almacena la hora, es decir: horas, minutos, segundos y hasta 7 dígitos de nanosegundos. Donde n es el número de dígitos que se usarán.

SMALLDATETIME: almacena fecha y hora sin nanosegundos. Además, los segundos siempre son cero. Por lo tanto, la precisión es de un minuto.

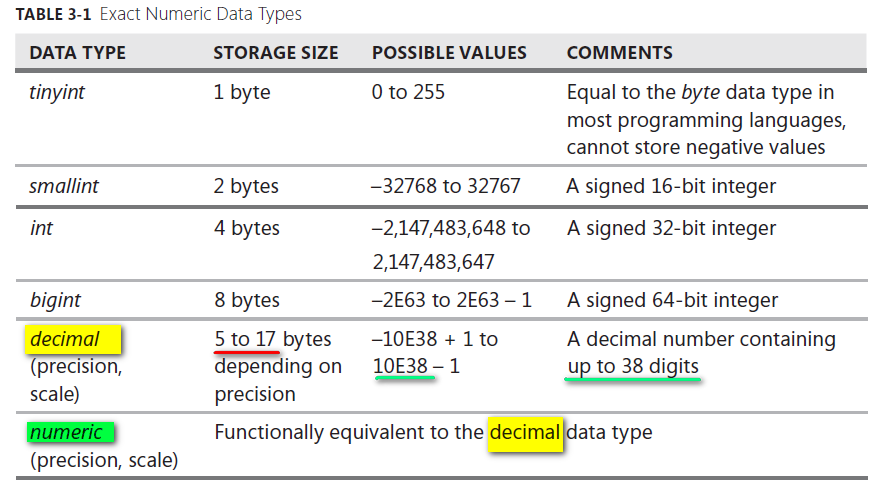
DATETIME: Almacena fecha y hora con milisegundos. Rango de fechas: 01/01/1753 a 31/12/9999. Rango de horas: 00:00:00 a 23:59:59.997

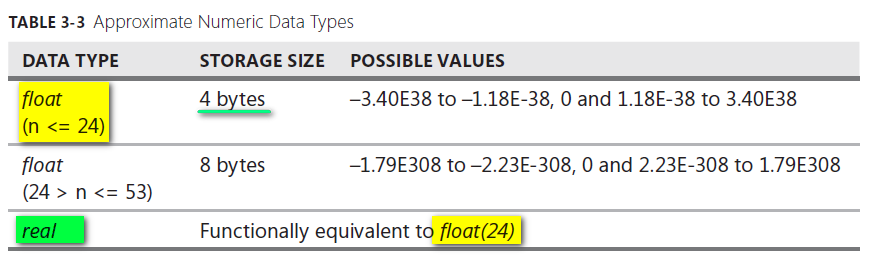
DATETIME2(n): Almacena fecha y hora con milisegundos. Rango de fechas: 01/01/0001 a 31/12/9999. Rango de horas: 00:00:00 a 23:59:59.9999999. n es el número de dígitos que se usarán.

DATETIMEOFFSET(n): es exactamente igual que DateTime2 pero añade la zona horaria.

En resumen, considero que estamos utilizando el tipo DateTime siempre por defecto sin tener en cuenta los demás y sin tener en cuenta el tipo de dato que se desea almacenar. Muchos de los campos que definimos como DateTime deberían ser Date y ahorraríamos 5 bytes en cada uno de ellos. Sólo con los campos fecha\_creacion, hora\_creacion, fecha\_actua y hora\_actua podríamos ahorrar como mínimo 16 bytes por fila.

**Numéricos:**





Fuente: https://stackoverflow.com/questions/1056323/difference-between-numeric-float-and-decimal-in-sql-server

BIT: Entero que solo admite valores 0 y 1.

MONEY y SMALLMONEY: Tienen una precisión de 4 decimales. La diferencia entre ambos es que el primero ocupa 8 bytes y el segundo 4. El intervalo es de De -922.337.203.685.477,5808 a 922.337.203.685.477,5807 (de -922.337.203.685.477,58  
a 922.337.203.685.477,58 para MONEY y de -214.748,3648 a 214.748,3647 para el segundo.

**Cadenas binarias:**

Binary(n): Son datos binarios de longitud fija que se puede especificar entre 1 y 8000. Si no se indica, será 1.

Varbinary: Son datos binarios de longitud variable que se puede especificar entre 1 y 8000. Si no se indica, será 1. También se puede indicar MAX como argumento y en este caso la capacidad será de 231 - 1.

IMAGE: SERÁ ELIMINADO Y SE RECOMIENDA SUSTITUIR POR VARBINARY(MAX)

**Otros tipos de datos:**

Hierachyid: tipo de datos de longitud variable para representar la posición de un elemento dentro de una jerarquía, con un máximo de 892 bytes.

Uniqueidentifier: GUID de 16 bytes.

Xml: almacena datos xml. Es posible especificar que se restrinja la instancia de xml a un fragmento correcto o a un documento correcto. También se puede indicar el nombre de una colección de esquemas xml. No puede superar los 2 Gb de tamaño.

**VENTAJAS Y DESVENTAJAS DE USAR UN INT AUTONUMÉRICO VS UNIQUEIDENTIFIER**

Uniqueidentifier

* VENTAJAS
  + Es muy difícil de recordar por lo que es un buen candidato cuando va asociado a datos sensibles, por ejemplo, protegidos por el RGPD.
  + Es imprescindible su uso si el identificador va a formar parte de una url puesto que, cambiando ese dato de la url será muy difícil que alguien pueda acceder a otro registro existente al que no debería poder acceder. Sin embargo, con un entero autonumérico sería muy fácil de acceder a información a la que no se debería tener acceso.
  + Es único de forma global. Con un entero, aunque sea autonumérico siempre vamos a tener un pedido con id = 1, un artículo con id = 1, un cliente con id = 1. Sin embargo, no vamos a tener dos uniqueidentifier iguales en toda la base de datos.
* DESVENTAJAS
  + Ocupa 16 bytes
  + no se pueden ordenar
  + Dependiendo del lenguaje utilizado, es posible que no exista un tipo para almacenar este tipo de dato y debas utilizar variables de tipo alfanumérico para manejarlos.

Int

* VENTAJAS
  + Ocupa 4 bytes
  + Se puede ordenar
  + Fácil de utilizar y recordar en consultas para resolver algún problema o realizar búsquedas de datos de forma manual.
* DESVENTAJAS
  + Las ventajas de uniqueidentifier son desventajas en este caso.

**CUANDO USAR DEFAULT VALUES**

Los valores por defecto se deben utilizar en aquellos campos de la BBDD que SIEMPRE deban tener un valor. Por ejemplo, si se desea que el stock de la tabla de artículos siempre tenga un valor, se puede establecer un cero como valor por defecto.

También, se deben utilizar en aquellos campos que sean susceptibles de ser utilizados para realizar operaciones matemáticas. De esta forma evitamos que el resultado de dichas operaciones sea NULL.

ALTER TABLE test ADD valor\_defecto\_entero INT DEFAULT 35

ALTER TABLE test ADD valor\_defecto\_varchar VARCHAR(10) DEFAULT 'DEFECTO'

ALTER TABLE test ADD valor\_defecto\_tinyint TINYINT DEFAULT 0