# COLLATIONS

Intercalaciones – Proporcionan propiedades de distinción entre mayúsculas y minúsculas, acentos y reglas de ordenación para los datos.

* CaseSensitivity
  + CI -> Case Insensitive
  + CS -> Case Sensitive
* AccentSensitivity
  + AI -> Accent Insensitive
  + AS -> Accen Sensitive
* KanatypeSensitive
  + KS -> Kanatype Sensitive
* WidthSensitivity
  + WS -> Width Sensitive
* VariationSelectorSensitivity
  + VSS -> Variation Selector Sensitive
* UTF8
* BIN
* BIN2

SQL SCRIPT:

SELECT name, collation\_name FROM sys.databases WHERE name = 'smcdb1';

# Tipos de datos comunes

Información útil: [Learning Microsoft](https://learn.microsoft.com/es-es/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16)

1. Char vs NChar
   * CHAR (N): Almacena caracteres no unicode. Tamaño fijo. Ocupa el tamaño especificado. N define el tamaño de la cadena en bytes y debe ser un valor entre 1 y 8000.
   * NCHAR (N | max): Almacena caracteres unicode. Tamaño fijo. Ocupa el doble del tamaño especificado. N define el tamño de la cadena en bytes y debe ser un valor entre 1 y 8000 o usar maxx para indicar hasta un máximo de tamaño de 2^31-1 bytes (2Gb)
   * ¿Cuando usar?
     + CHAR cuando se trabaja con caracteres no unicode y se conoce la longitus exacta
     + NCHAR cuando se requiere soporte para caracteres unicode.
   * Observaciones
     + Char siempre reserva 10 bytes
     + Char(n) y Nchar(n), la n NO define el número de caracteres, sino que define la longitud de la cadena en bytes o de la cadena en pares de bytes como nchar
     + Si no se especifica un valor de n, el valor por defecto es 1
     + Se asignan al collation por defecto
2. Varchar vs Nvarchar
   * VARCHAR: Almacena caractreres no unicode. Tamaño variable. Ocupa el tamaño de datos más el tamaño de longitud
   * NVARCHAR: Almacena caracteres unicode. Tamaño variable. Ocupa el doble del tamaño de datos más el tamaño de longitud.
   * ¿Cuando usar?
     + Similar a CHAR vs NCHAR, pero con almacenamiento variable, ahorrando espacio en disco.
3. Datetime vs Smalldatetime
   * DATETIME: Precisión hasta milisegundos. Rango de fechas 1753-9999.
   * SMALLDATETIME: Precisión hasta minutos. Rango de fechas 1900-2079
   * ¿Cuando usar?
     + DATETIME para mayor precisión
     + SMALLDATETIME para rangos más limitados y menor almacenamiento.
4. Binary vs Varbinary
   * BINARY: Almacena datos binarios de longitud fija. Se define una longitud fija a la hora de definir la columna. BINARY(N)
   * VARBINARY: Almacena datos binarios de longitud variable. No se especifica una longitud fija al definir la columna. VARBINARY(MAX) y puede almacenar datos binarios de diferentes longitudes.
   * ¿Cuando Usar?
     + BINARY cuando se necesita almacenar datos binarios de longitud fija, como hashes o códigos de verificación.
     + VARBINARY cuando se necesita almacenar datos binarios de longitud variable como imágenes, documentos o cualquier tipo de archivo binario.
5. Int vs Bigint vs Decimal vs Money vs Smallmoney
   * INT: Números enteros de 4 bytes
   * BIGINT: Números enteros de 8 bytes
   * DECIMAL: Números ddecimales de precisión fija
   * MONEY: Números decimales con precisión fija para moneda.
   * SMALLMONEY: Versión más pequeña de MONEY
   * ¿Cuando usar?
     + INT para números enteros pequeños
     + BIGINT para números enteros grandes
     + DECIMAL/MONEY para valores monetariosque requieran precisión decimal
6. Datetime vs Smalldatetime vs Datetimeoffset
   * DATETIMEOFFSET: Similar a DATETIME pero incluye información de la zona horaria
   * ¿Cuando usar?
     + Cuando se necesita información de la zona horaria

## Comparación ‘INT’ Autonumérico vs ‘UniqueIdentifier’ GUID

* ‘INT’ Autonumérico:
  + **Ventajas:** Compacto, eficiente en términos de almacenamiento, fácil de entender.
  + **Desventajas:** Predictible, no adecuado para ambientes distribuidos.
* ‘Uniqueidentifier’ (GUID)
  + Ventajas: Globalmente único, adecuado para ambientes distribuidos
  + Desventajas: Mayor tamaño de almacenamiento, menos eficiente en términos de índices y rendimiento.

¿Cuando usar?

* Usar ‘INT’ autonumérico cuando la predictibilidad no sea un problema y se priorice la eficiencia.
* Usa ‘Uniqueidentifier’ cuando se necesite un identificador globalmente único o se trabaje en ambientes distribuidos.

# USO DE DEFAULT VALUES

El uso de valores por defecto puede ser beneficioso en ciertos casos para simplificar la lógica y garantizar la consistencia de datos.

Ejemplos de uso:

* Mejora de la inserción de datos
  + Campo booleano que indica si un artículo está activo, con valor por defecto a ‘true’

CREATE TABLE Articulos (

ArticuloID INT PRIMARY KEY,

Nombre NVARCHAR(100),

Activo BIT DEFAULT 1

);

* Evitar valores NULL en columnas opcionales
  + Tabla de eventos con columna opción que almacena ubicación

CREATE TABLE Eventos (

EventoID INT PRIMARY KEY,

Nombre NVARCHAR(100),

Ubicacion NVARCHAR(255) DEFAULT 'Ubicación No Especificada'

);