Reglas de formato

Es fundamental para la buena marcha del curso que sepas interpretar el formato utilizado a la hora de describir la sintaxis de las instrucciones SQL.

En programación, cuando escribimos código, es imprescindible escribir las instrucciones siguiendo la sintaxis definida en el lenguaje, cualquier error hace que la instrucción no se entienda y no se pueda ejecutar. Esta sintaxis suele venir detallada utilizando unas reglas que debemos conocer para poder aprenderla o consultarla en cualquier momento de duda.

Las reglas de formato que utilizaremos en los diagramas de sintaxis de las sentencias SQL son muy comunes y son las utilizadas por Microsoft en toda su documentación.

Convención	Se usa para
Mayúsculas	Palabras clave (Keywords) de Transact-SQL. Son palabras del lenguaje que se tienen que escribir tal cual y que tienen un significado determinado. Aunque en la sintaxis aparecen en mayúsculas (para indicar que son palabras reservadas) se pueden escribir en minúsculas aunque recomiendo utilizar las mayúsculas, es una buena costumbre para aprender a diferenciar las palabras reservadas de lo demás y hace la instrucción más legible.
Cursiva	Parámetros proporcionados por el usuario.
negrita	Nombres de bases de datos, tablas, columnas e índices, procedimientos almacenados, utilidades, nombres de tipos de datos y texto que debe escribirse exactamente como se muestra.
subrayado	Indica el valor predeterminado que se aplica cuando la cláusula que contiene el valor subrayado se omite en la instrucción.
| (tubo)	Separa los elementos de sintaxis escritos entre corchetes o llaves. Indica alternativa O, sólo se puede utilizar uno de los elementos.
[ ] (corchetes)	Elementos opcionales de sintaxis.
{} (llaves)	Elementos obligatorios de sintaxis.
[**,**...*n*]	Indica que el elemento anterior puede repetirse *n* veces y cada repetición se separa del siguiente con una coma.
[...*n*]	Indica que el elemento anterior puede repetirse *n* veces y cada repetición se separa del siguiente con un espacio en blanco.
;	Terminador de instrucciones Transact-SQL opcional.
<etiqueta> ::=	Nombre de un bloque de sintaxis. Esta convención sirve para agrupar y etiquetar secciones de sintaxis extensas o una unidad de sintaxis que se puede usar en varias ubicaciones dentro de una instrucción. Cada ubicación en que se puede utilizar el bloque de sintaxis se indica con la etiqueta situada entre comillas angulares: <etiqueta>. ::= indica que a continuación se detalla la etiqueta.

Los caracteres que aparecen en la lista anterior (excepto el ;) no se escriben luego en la sentencia SQL.

Nombres de varias partes:

A menos que se especifique lo contrario, todas las referencias de Transact-SQL al nombre de un objeto de base de datos pueden ser un nombre de cuatro partes con el formato siguiente:

*nombre_servidor***.**[nombre_basedatos]**.**[nombre_esquema]**.**nombre_objeto

| nombre_basedatos**.**[nombre_esquema]**.**nombre_objeto

| nombre_esquema**.**nombre_objeto

| nombre_objeto

Cuando se hace referencia a un objeto específico, no siempre hay que especificar el servidor, la base de datos y el esquema para identificar el objeto, por defecto se asume la base de datos activa y el esquema dbo. No obstante, si no se encuentra el objeto, se muestra un error.

Veamos un ejemplo de sintaxis y cómo se debería de interpretar (solo nos fijamos en el formato no en el significado de las palabras reservadas:

```
SELECT [ALL|DISTINCT]
    [TOP expresion [PERCENT] [WITH TIES]]
    <lista_seleccion>
    FROM <origen>
    [WHERE <condicion_busqueda> ]
    [ORDER BY {{expression_columna|posicion_columna} [ASC|DESC]} [ ,...n ]][;]

<origen>::=
nb_tabla | nb_vista [[ AS ] alias_tabla ]
```

- La instrucción empieza por la palabra SELECT.
- A continuación, de forma opcional (por los []) podemos escribir la palabra ALL **o bien** (por |) la palabra DISTINCT (solo una de las dos), si no escribimos nada, se asume como valor por defecto ALL (porque está subrayado).
- Opcionalmente podemos continuar con la palabra TOP y una expresión, opcionalmente la palabra PERCENT y también opcionalmente WITH TIES.
- A continuación de manera obligada (no va entre []) tenemos que escribir la lista_selección, como va entre <>, esto me indica que lista_selección representa un bloque de código cuya sintaxis se detallará más abajo, en nuestro ejemplo no lo incluimos.
- A continuación, obligatoriamente se escribe FROM y un <origen>.
- <origen> consiste en un nombre de tabla o (|) un nombre de vista y opcionalmente la palabra AS (que se puede poner o no, va entre [], y no afectará al resultado, va subrayada) y un alias de tabla.
- A continuación también opcional podemos escribir WHERE y una condición de búsqueda (que vendrá definida más abajo).
- A continuación también opcional podemos escribir ORDER BY y { {una expresion_columna o una posición_columna} seguida opcionalmente por ASC o DESC (si no ponemos nada se asume ASC (viene subrayado)}, y ese bloque (delimitado por los {} más externos que empieza en una expresion_columna y acaba en DESC) se puede repetir opcionalmente n veces añadiendo una coma antes de la repetición (porque viene [,...]).
- Finalmente se puede escribir ;

Ejemplo válidos:

```
SELECT *      (nota: en la definición detallada de <lista_selección> está *)
FROM tabla1
```

Esta sería la instrucción SELECT más corta posible.

```
SELECT *
FROM tabla1; es equivalente a la anterior porque ; no añade nada a la instrucción

SELECT ALL *
FROM tabla1;  es equivalente a la anterior  porque ALL es el valor por defecto (subrayado)

SELECT TOP 2 *
FROM tabla1;

SELECT TOP 2 PERCENT *
FROM tabla1;

SELECT TOP 2 PERCENT WITH TIES *
FROM tabla1;

SELECT TOP 2 WITH TIES *
FROM tabla1;

SELECT *
FROM tabla1
WHERE col1=0;    (nota: col1=0 es una condición_busqueda válida)
```

```
SELECT *                    SELECT *
FROM tabla1                 FROM tabla1
ORDER BY 1 ASC;             ORDER BY 1;   Las dos son equivalentes

SELECT *                          Aquí hemos aplicado la repetitiva [,...]
FROM tabla1                       1 y 2 son posición_columna
ORDER BY 1 ASC, 2, col1 DESC;     col1 es una expresión_columna
```

Todas estas instrucciones serían válidas sintácticamente.

No sería válido por ejemplo escribir la cláusula ORDER BY (y lo que le sigue) delante de la cláusula FROM.