UD4. DISEÑO FÍSICO DE LA BASE DE DATOS (DDL)

1. INTRODUCCIÓN

Las siglas **SQL Structured Query Language** significan **Lenguaje Estructurado de Consultas**. Es un lenguaje que se utiliza para definir, gestionar y manipular la información contenida en una BDR (Base de Datos Relacional).

1.1. EVOLUCIÓN DEL SQL

- En **1975** surge un lenguaje para un prototipo de BDR de IBM, el System R. El nombre de este primer lenguaje de consultas es el de SEQUEL. (Este nombre con el tiempo evolucionaría hasta el actual SQL).
- En **1979** aparece **Oracle**, el primer SGBDR comercial que utiliza SQL.
- En **1986** el instituto de estándares **ANSI** (**A**merican **N**ational **S**tandard **I**nstitute) lo adopta como lenguaje estándar para las BBDDRR. La primera versión del lenguaje se conoce como SQL-86 ó SQL1.
- En **1987** lo adoptará también como estándar la organización internacional **ISO** (**I**nternational **S**tandarization **O**rganization).
- En **1992** se aprueba como norma internacional la versión **SQL-92** ó **SQL2** de SQL, versión importante pues es la más difundida y adoptada masivamente por los gestores de BBDDRR comerciales e incluso algunos SGBD NO relacionales también incluyen soporte para SQL.
- **Posteriormente** surgen **actualizaciones** como **SQL-99** ó **SQL3** (incluye, entre otras cosas, tratamiento de objetos), **SQL:2011** (añade tratamiento de datos temporales, entre otras características) y la última y más reciente **SQL:2016** (año 2016) que, entre otras novedades, permite compatibilidad con JSON.

1.2. CARACTERÍSTICAS PRINCIPALES

- La mayoría de los SGBDR comerciales usan SQL, aunque no siguen el estándar estrictamente. De hecho, en algunos SGBDR al lenguaje de consultas que utilizan se le denomina con un nombre específico. Por ejemplo, en el caso del SGBDR MS SQL Server®, el lenguaje de consultas se conoce como TRANSACT-SQL.
- Los SGBDR incorporan un **motor SQL en el servidor y Herramientas cliente** que permiten enviar comandos SQL para que sean procesados por el motor del servidor.
- Basado en el Álgebra Relacional.

1.3. TIPOS DE SENTENCIAS DDL

Las sentencias **del Lenguaje Definición de Datos**, que son las que nos ocupan en esta unidad didáctica, se utilizan para:

- **Crear** objetos en la BD (**CREATE**)
- **Eliminar** objetos en la BD (**DROP**)
- **Modificar** objetos en la BD (**ALTER**)

2. ARCHIVOS DE LAS BASES DE DATOS

Cuando se crea una base de datos se establece una **estructura de almacenamiento de datos**. Esta estructura incluye, **al menos, un archivo de datos y un archivo de registro de transacciones.**

2.1. ARCHIVOS DE LAS BASES DE DATOS DE SQL Server

Hay tres **tipos de archivos** de bases de datos:

- **Archivo de datos principal o primario:** Cada base de datos debe tener **un único archivo** de datos principal. Contiene la información de inicio para la base de datos y se utiliza **para almacenar datos**.

  Estos archivos tienen la extensión **.MDF**

- **Archivos de datos secundarios:** Estos archivos pueden contener todos los datos y objetos que no estén incluidos en el archivo de datos principal. Las bases de datos no necesitan archivos de datos secundarios si el archivo principal puede contener todos los datos de la base de datos, por lo tanto, este tipo de archivos es **opcional**.

  Los nombres de estos archivos, que pueden ser varios por base de datos, tienen la extensión **.NDF**

- **Archivos de registro o de LOG:** Estos archivos contienen toda **la información de registro de transacciones requerida para recuperar la base de datos.** Cada base de datos tiene **al menos un archivo de registro**.

  Los nombres de estos archivos tienen la extensión **.LDF**

IMPORTANTE: El término **archivo de base de datos** generalmente significa cualquiera de los tres tipos de archivos. El término **archivo de datos** se refiere bien al archivo de datos primario o al secundario. El término **archivo de registro** se refiere al registro de transacciones de la bd.

- **Los grupos de archivos:** Es posible especificar un grupo de archivos en lugar de los nombres de archivo. Estos grupos presentan la ventaja de equilibrar las cargas de trabajo sobre los diferentes discos del sistema. Los datos se escriben de forma equitativa en los distintos archivos del grupo.

3. OBJETOS DE UNA BASE DE DATOS

Una Base de Datos, para SQL Server, es una colección de datos, tablas y otros objetos. Estos permiten estructurar los datos y definir las restricciones de integridad.

El objeto **DATABASE** contiene todos los demás objetos:

- El **catálogo** de bases de datos.
- Los **objetos de usuario** (tablas, valores por defecto, vistas, reglas, desencadenadores, funciones, procedimientos almacenados).
- Los **índices**, los **tipos** de datos, las **restricciones**.
- El **registro de transacciones.**

3.1. TABLAS DEL SISTEMA

Almacenan información de configuración y definición de los objetos de BBDD. No deben modificarse. Algunas de ellas en SQL Server son: **syscolumns, sysindexes, sysforeignkeys…**

3.1.1. Catálogo del sistema

El catálogo del sistema constituye **el núcleo de todo SGBD**. Es una BD para almacenar los esquemas o descripciones de TODAS las BBDD que el SGBD mantiene.

Cada una de las BBDD se describe por los datos almacenados en el catálogo (**metadatos**).

3.1.2. Catálogo de bases de datos

Tablas del sistema que almacenan la información específica de una BD. En algunos gestores se implementa en el **Information Schema.**

El **Information Schema** son vistas que proporcionan información (metadatos) sobre todos los objetos de datos almacenados en una BD en concreto.

A pesar de ello, cada gestor lo implementa de forma propietaria (catálogo) y no todos los gestores lo implementan.

3.1.3. Catálogo e INFORMATION SCHEMA de SQL Server

- **Vistas del catálogo:**
  - En MS SQL Server® está localizado en la base de datos **master**, y es la colección de tablas de sistema que almacenan la información sobre el sistema completo y todas las bases de datos.
  - Dentro de esta información se incluyen los metadatos de todas las instancias, como las cuentas de inicio de sesión, los servidores vinculados y la configuración del sistema.
  - Asimismo, **master** es la base de datos que registra la existencia de las demás bases de datos, la ubicación de los archivos de las bases de datos y la información de inicialización de MS SQL Server®.
- **Information Schema:**
  - Las vistas **INFORMATION_SCHEMA** se incluyen en cada base de datos. Cada vista de esquema de información contiene metadatos para todos los objetos de datos almacenados en esa base de datos en concreto

3.2. OBJETOS DE USUARIO

Son aquellos que el usuario crea dentro de la base de datos:

- Tablas
- Tipos de datos definidos por el usuario
- Restricciones
- Valores por defecto
- Reglas de validación
- Índices
- Vistas
- Funciones definidas por el usuario
- Procedimientos almacenados
- Disparadores

3.3. TIPOS DE DATOS

Todos los objetos con datos tienen asociado un tipo de datos en el gestor de BBDD que define la clase de los datos del objeto (carácter, entero, binario, etc).

Los **objetos en Transact-SQL que tienen tipos de datos** son:

- las **columnas** de tablas y vistas,
- los **parámetros** de procedimientos almacenados,
- las **variables**,
- **valores devueltos por las funciones**, y,
- los procedimientos almacenados que devuelven un código, que siempre es de tipo integer.

Antes de poder **crear una tabla** deben definirse los tipos para los datos de la tabla. Los tipos de datos especifican el tipo de información (caracteres, números o fechas) que puede contener una columna, así como la forma en la que se almacenan los datos.

SQL Server proporciona:

- varios **tipos de datos de sistema**,
- también permite **tipos de datos definidos por el usuario** que están basados en tipos de datos de sistema.

3.3.1. Consideraciones previas a escoger un tipo de dato

- **Qué tipo de información se va a almacenar:** Por ejemplo, no se pueden guardar caracteres en un campo cuyo tipo de datos sea numérico.
- **El espacio de almacenamiento necesario:** Se conoce como *dimensionar el campo*.
- **Qué tipo de operaciones se van a realizar con los valores del campo:** Por ejemplo, no se podría calcular la suma de dos cadenas de caracteres.
- **Si se desea ordenar o indexar por ese campo:** Los criterios de ordenación difieren en función del tipo de dato, así, los números almacenados en un campo de texto se ordenan según el valor de su código ASCII (1, 10, 11, 2, 20...) que no coincide con la ordenación numérica.

3.3.2. Tipos de datos de sistema en SQL Server

TIPOS DE DATOS COMUNES	MS SQL SERVER®	ANSI SQL	Número de bytes
Entero	**int**	*integer*	4
	bigint	----	8
	smallint	----	2
	tinyint	----	1
Numérico exacto	**decimal[(p[, s])]**	*dec*	2-17
	numeric[(p[, s])]	----	2-17

Numérico aproximado *(para datos ciéntificos)*	**float[(n)]**	*double precision, float[(n)] para n=8-15*	8
	real	*float[(n)] para n=1-7*	4
Moneda	**money**	----	----
	smallmoney	----	----
Fecha y hora	**Time**	----	3-5
	Date	----	3
	Datetime	----	8
	Smalldatetime	----	4
	Datetime2	----	6-8
	Datetimeoffset	----	8-10
Carácter	**char[(n)]**	*character*[(*n*)] n=1- 8000	1–8000
	varchar[(n|max)]	*char VARYING*[(*n*)] *character VARYING*[(*n*)] n=1- 8000	1–8000 max=2GB
	text *(obsoleto)*	----	0 a 2 GB
Caracteres Unicode	**nchar[(n)]**	*national char(n)* n=1- 4000	1–8000
	nvarchar[(n|max)]	national character varying(*n*) n=1- 4000	1–8000 max=2GB
	ntext *(obsoleto)*	*national text*	0 a 2 GB
Binario	**binary[(n)]**	---- n=1-8000	0–8000
	varbinary[(n|max)]	*binary VARYING[(n)]*	0–8000
Imagen	**image** *(obsoleto)*	----	0 a 2GB
Identificador global	**uniqueidentifier**	----	16
Especial	**bit**	---- acepta 1, 0 ó null	1
	cursor	----	0-8

	Tipo CLR: hierarchyid	----	----
	uniqueidentifier	----	----
	timestamp*(obsoleto)*	*rowversion*	8
	table	----	----
	sql_variant	----	0-8016
	xml	----	----
	sysname	----	256
	rowversion	*rowversion*	

IMPORTANTE: Aunque en alguna documentación se considera **Identity[(semilla, incremento)]** como un tipo de datos, no lo es. Es una **propiedad** que se aplica a **una única columna de la tabla** para que ésta contenga **valores secuenciales** generados automáticamente por el sistema. Por ello se suelen usar **para los valores de las claves principales**.

3.3.3. Tipos de datos definidos por el usuario en SQL Server

Los **tipos de datos definidos por el usuario** son útiles cuando varias tablas de una base de datos utilicen un campo con el mismo tipo, tamaño y posiblemente las mismas restricciones de validez o valores predeterminados de datos. **Estos tipos de datos pueden tener enlazados reglas y valores predeterminados.**

Los tipos de datos definidos por el usuario están basados en tipos de datos proporcionados de sistema.

Un tipo de datos definido por el usuario se define **para una base de datos específica**.

IMPORTANTE: Los tipos de datos definidos por el usuario que se crean en la base de datos **model** se incluyen automáticamente en todas las bases de datos que se crean a partir de ese momento. Todos los tipos de datos definidos por el usuario se agregan como filas de la tabla **systypes**.

3.4. RESTRICCIONES

Las **restricciones** son el método más adecuado para conseguir la *integridad de los datos*. Diferentes tipos de restricciones aseguran que los valores que se escriban en los datos de las columnas son válidos y que se mantienen las relaciones entre las tablas.

TIPO DE RESTRICCIÓN	
DEFAULT	**PRIMARY KEY**
CHECK	**UNIQUE**
[NOT] NULL	**FOREIGN KEY**

4. SENTENCIAS DDL

Las sentencias de definición de datos permiten crear, borrar o modificar bases de datos, así como los OBJETOS de la propia la base de datos. Es decir, permitirán:

- **crear un nuevo objeto** de la base de datos. Instrucción **CREATE**
- **modificar o alterar la estructura** del objeto. Instrucción **ALTER**
- y **borrar el objeto de la bd**, y cuando hablamos de borrar, en DDL nos referimos a eliminar el objeto **completamente de la bd, estructura y contenido**. Instrucción **DROP**.

4.1. CREACIÓN, MODIFICACIÓN Y BORRADO DE BASES DE DATOS (DATABASE)

4.1.1. Crear BBDD en SQL Server

Para crear una base de datos en SQL Server, es necesario estar conectado como administrador del sistema o tener permiso para utilizar la instrucción CREATE DATABASE y situarse en la base de datos **master**.

El nombre de una base de datos debe ser único en una instancia de SQL Server. Este nombre está limitado a 128 caracteres.

Se puede crear una base de datos utilizando:

- el **Asistente** de creación de bases de datos,
- el **Management Studio** de SQL Server, o,
- la instrucción SQL **CREATE DATABASE**

Sintaxis:

```
CREATE DATABASE nombreBD [ ON [PRIMARY]
[( [ NAME = nombreLógico, ]
FILENAME = 'nombreFísico'
[, SIZE = tamaño]
[, MAXSIZE = { tamMax | UNLIMITED } ]
[ LOG ON { archivo } ]
[ COLLATE nombre_intercalación ]
[ FOR ATTACH | FOR ATTACH_REBUILD_LOG ]
```

- **NAME:** es el nombre lógico del archivo.
- **FILENAME:** ubicación y nombre físico del archivo.
- **SIZE:** tamaño inicial del archivo en MB o KB. El tamaño mínimo es de 8MB, porque una BD nueva debe tener al menos el tamaño de la BD model (versión 2016 – v13.x o posteriores).
- **MAXSIZE:** tamaño máximo del archivo indicado en KB o MB (por defecto MB). Si no se indica ningún valor, el tamaño del archivo estará limitado por el espacio libre en disco.
- **UNLIMITED:** el límite del tamaño es el espacio libre en disco.
- **LOG ON:** ubicación del registro de transacciones.
- **COLLATE:** indica el mapa de caracteres que se va a usar (*intercalación predeterminada*).
- **FOR ATTACH:** se usa para crear una BD con archivos ya creados. *Equivale a Adjuntar una base de datos de SSMS*
- **FOR ATTACH_REBUILD_LOG:** con esta opción se puede crear la base de datos adjuntando a ella los archivos de datos (mdf y ldf), pero no necesariamente los de registro (ldf).

Ejemplo:

```
CREATE DATABASE compras
```

```
ON PRIMARY (
    NAME = compras,
    FILENAME = 'C:\data\compras_data.MDF',
    SIZE = 5MB,
    MAXSIZE = 3000MB)
LOG ON (
    NAME = compras_log,
    FILENAME = 'C:\data\compras_log.LDF',
    SIZE = 1MB)
     COLLATE Modern_Spanish_CI_AS;
```

4.1.2. Modificar BBDD en SQL Server

Después de haber creado una base de datos, pueden definirse opciones de base de datos con:

- el **Management Studio** de SQL Server, o,
- por medio de la instrucción **ALTER DATABASE**.

Pueden configurarse varias opciones de base de datos, pero sólo se puede hacer en una única base de datos al mismo tiempo. Para que las opciones afecten a todas las bases de datos nuevas, es necesario cambiar la base de datos **model**, que es la que usa MS SQLServer® como plantilla para crear las bases de datos.

La instrucción **ALTER DATABASE** permite:

- Cambiar el **nombre** y la **intercalación** de una base de datos:

Sintaxis:

```
ALTER DATABASE nombreBD
{
  MODIFY NAME = nuevo_nombreBD
  COLLATE nombre_de_intercalación
  }
```

Ejemplos:

```
USE master;
GO
ALTER DATABASE SOCIOS
    MODIFY NAME = Sociedad ;
GO
```

```
USE master;
CREATE DATABASE  ejemplo_intercalacion
   COLLATE   SQL_Latin1_General_CP1_CI_AS ;
GO

ALTER DATABASE ejemplo_intercalacion
    COLLATE   French_CI_AI ;
GO
```

- Agregar y eliminar **archivos de una base de datos**, así como cambiar los atributos de los mismos.

Sintaxis:	**Ejemplos:**

ALTER DATABASE nombreBD
ADD FILE (
  NAME = nombreLógico,
  FILENAME = 'nombreFísico'
  [, SIZE = tamaño]
  [, MAXSIZE = { tamMax | UNLIMITED }]
 );
GO
	USE master;
	GO
	ALTER DATABASE pruebas
	ADD FILE (
	  NAME = pruebas2,
	  FILENAME = 'C:\data\pruebas_data2.ndf' ,
	  SIZE=3MB);
	GO

ALTER DATABASE nombreBD REMOVE FILE nombreLógico; GO	USE master; GO ALTER DATABASE pruebas REMOVE FILE pruebas2; GO

- **Cambiar atributos** de una base de datos **usando la opción SET**.

Sintaxis: ALTER DATABASE nombreBD **SET** opcion

La tabla siguiente enumera algunas de las opciones de base de datos de uso más frecuente:

CATEGORÍA DE LAS OPCIONES DE BASE DE DATOS	OPCIÓN DE BASE DE DATOS	DESCRIPCIÓN
Estado de la base de datos	**ONLINE**	Permite hacer visible de nuevo la base de datos.
	OFFLINE	Permite hacer inaccesible la base de datos, que queda detenida y cerrada correctamente. No es posible realizar operaciones de mantenimiento en una base de datos offline.
	EMERGENCY	La base de datos está en modo de sólo lectura, los registros están deshabilitados y su acceso queda limitado a los administradores del servidor.
Acceso	**SINGLE_USER**	Acceso limitado a un solo usuario.
	RESTRICTED_USER	Solo los miembros con rol db_owner, dbcreator o sysadmin pueden conectarse a la base de datos.
	MULTI_USER	Es el modo predeterminado, que permite que todos los usuarios con privilegios suficientes accedan a la información.
	DBO use only	Sólo el propietario podrá acceder a la base.

Operaciones posibles	**READ_ONLY**	La base de datos es accesible solamente para operaciones de lectura.
	READ_WRITE	La base de datos es accesible para operaciones de lectura/escritura.
Configuración	**ANSI_NULL_DEFAULT**	Define el valor por defecto de la restricción de nulidad de columna. Según la norma ANSI, una columna puede ser NULL por defecto.
	RECURSIVE_TRIGGERS	Permite la recursividad de los desencadenadores.
	AUTO_CLOSE	La base se detiene y los recursos se liberan tras la desconexión del último usuario.
	AUTO_CREATE_STATISTICS	Todas las estadísticas que falten en la optimización de una consulta se crean. Esta opción está habilitada (en ON) de forma predeterminada.
	QUOTED_IDENTIFIERS	Los identificadores delimitados (los que incluyen caracteres especiales como espacios en blanco) pueden encerrarse entre comillas dobles (por defecto se encierran entre corchetes).
	ANSI_NULLS	Si el parámetro es verdadero (true), todas las comparaciones con un valor NULL se evalúan como NULL. Si el parámetro es falso (false), todas las comparaciones con los valores NULL y los valores no unicode se evalúan como TRUE si ambos valores son NULL.
Gestión de transacciones	**ROLLBACK AFTER número**	La anulación de transacciones es efectiva después de número segundos de espera.
	ROLLBACK INMEDIATE	La anulación de la transacción es inmediata.
	NO_WAIT	Si la transacción no accede inmediatamente a los recursos que necesita, queda anulada.

Ejemplos:

```
USE master;
GO
ALTER DATABASE nombreBD  SET SINGLE_USER
WITH ROLLBACK IMMEDIATE; -- Todas las instrucciones incompletas (sin commit) se revierten y el resto de
                         -- conexiones se desconectan de inmediato.
ALTER DATABASE nombreBD SET READ_ONLY;

ALTER DATABASE nombreBD SET MULTI_USER;
```

4.1.3. Borrar BBDD en SQL Server

- **Métodos de** eliminación **de una base de datos:**
  - **Management Studio** de SQL Server, o,
  - instrucción **DROP DATABASE**.
- **Restricciones de la eliminación de bases de datos:**
  - Mientras se está restaurando.
  - Cuando se conecta a ella un usuario.
  - Si es una base de datos del sistema.

Sintaxis: DROP DATABASE *BaseDeDatos* [,…*n*]	**Ejemplo:** Borramos las BBDD de nombres socios y empleados. DROP DATABASE socios, empleados;
IMPORTANTE: Al intentar borrar la BD puede darnos un error del tipo ***No se puede quitar la base de datos 'nombreBD'; está en uso.*** En ese caso antes de la instrucción de borrado debemos poner la BD en modo SINGLE_USER con la instrucción ALTER DATABASE nombreBD set SINGLE_USER;	

4.2. Creación y borrado de tipos (TYPE)

Se pueden definir tipos de datos propios a través de:

- el **Management Studio** de SQL Server, o,
- por medio de la instrucción **ALTER DATABASE**.

4.2.1. *Crear tipos de datos en SQL Server*

	Creación de un tipo de dato para las columnas en las que se guardan **segundos apellidos**: CREATE TYPE tipo_apellido2 FROM VARCHAR(30) NULL;
CREATE TYPE nombreTipo {FROM tipoDeBase [(longitud [, precisión])] [NULL | NOT NULL] } [;]	Creación de un tipo de dato para las columnas en las que se guardan *primeros apellidos y nombres de pila*: **CREATE TYPE** udtNombreApe1 FROM varchar(30) NOT NULL; Creación de un tipo de dato para las columnas en las que se guardan *segundos apellidos*: **CREATE TYPE** udtApe2 FROM varchar(30) NULL; -- En el nombre indicamos udt de *user defined type* -- Cuando creo la tabla **persona** puedo utilizar ese -- tipo de dato CREATE TABLE persona(nombre **udtNombreApe1**, ape1 **udtNombreApe1**, ape2 **udtApe2**);

4.2.2. Borrar tipos de datos en SQL Server

IMPORTANTE: Es imposible borrar un tipo si éste se usa en una tabla de la BD en la que ha sido creado.

Sintaxis:	**Ejemplos:**
DROP TYPE [nombreEsquema.] nombreTipo [;]	DROP TYPE **udtNombreApe1**;
	DROP TYPE **udtApe2**;

4.2.3. Crear tipos de datos TABLE (también llamados cuadro)

En Transact SQL, la instrucción CREATE TYPE permite también crear **tipos compuestos por varios campos**. Cada columna que participa en la definición de este nuevo tipo se define por el mismo principio que una columna de una tabla. De esta manera es posible definir las restricciones de integridad de clave primaria (PRIMARY KEY), de unicidad (UNIQUE), de validación (CHECK) y de no nulidad. Estas restricciones pueden ser definidas en el nivel de la columna o de la tabla. También es posible definir una columna de tipo identidad.

La instrucción CREATE TYPE permite así crear tipos denominados muy característicos, porque las restricciones de integridad permiten una definición más precisa del formato posible de los datos.

La introducción de este nuevo tipo **permitirá definir parámetros de funciones o procedimientos de tipo cuadro**. Hablamos entonces de un *table value parameter*.

Sintaxis:	
CREATE TYPE nombreTipo AS TABLE (Columna tipoColumna[restriccionColumna], ...)	■ **nombreTipo:** Nombre del tipo creado. ■ **Columna:** Nombre de la columna que participa en la definición de este nuevo tipo. Es posible definir varias columnas. ■ **tipoColumna:** Tipo de datos Transact SQL sobre el cual está definida la columna. No todas las columnas del tipo creado tienen que estar definidas sobre el mismo tipo ni con la misma precisión. ■ **restriccionColumna:** Definición de la restricción de integridad asociada a la columna.
Ejemplo: Creación de un tipo que representa a una persona. Este tipo está compuesto por los campos *sexo*, *nombre*, *apellido1* y *apellido2*: CREATE TYPE tipo_persona AS TABLE (genero char(1) check (genero in ('H', 'M')), nombre varchar(30) not null, apellido1 varchar(30) not null, apellido2 varchar(30) null);	

4.3. CREACIÓN, MODIFICACIÓN Y BORRADO DE TABLAS (TABLE)

4.3.1. Crear tablas en SQL Server

Sintaxis:

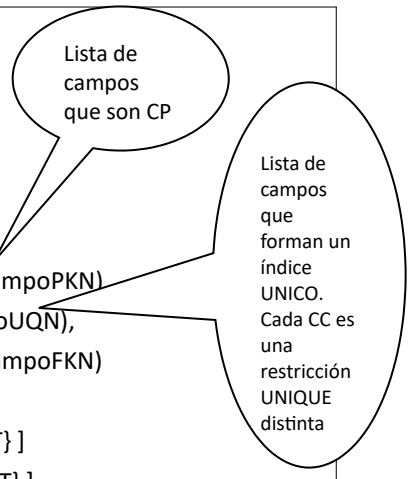
```
CREATE TABLE  nombre_tabla_nueva
( nombre_campo1   tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto] [RESTRICCIÓN PK, UQ ó FK] ,
  nombre_campo2   tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto] [RESTRICCIÓN PK, UQ ó FK] ,

  .....
  nombre_campoN   tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto] [RESTRICCIÓN PK, UQ ó FK] );
```

IMPORTANTE: Las restricciones no hay porqué indicarlas en cada campo, sino que una vez definidos todos los campos se pueden establecer todas seguidas. Observa la siguiente sintaxis (*la palabra CONSTRAINT es opcional en el caso de que no queramos dar un nombre a la restricción. En ese caso es el gestor de BD el que le asigna un nombre).*

Sintaxis:

```
CREATE TABLE  nombre_tabla_nueva
        ( nombre_campo1 tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto],
          nombre_campo2 tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto],
          .....
          nombre_campoN tipo_dato [NOT] NULL  [DEFAULT valor_por_defecto],
          [CONSTRAINT nombre_restricción] PRIMARY KEY (campoPK1, campoPK2,..., campoPKN)
          [CONSTRAINT nombre_restricción] UNIQUE (campoUQ1, campoUQ2,..., campoUQN),
          [CONSTRAINT nombre_restricción] FOREIGN KEY  (campoFK1, campoFK2,..., campoFKN)
                        REFERENCES tabla_padre(campo_clave)
                        [ ON DELETE {CASCADE | NO ACTION | SET NULL | SET DEFAULT} ]
                        [ ON UPDATE {CASCADE | NO ACTION | SET NULL | SET DEFAULT} ] ,
         [CONSTRAINT nombre_restricción] CHECK  condición_a_cumplir_la_columna,
         [CONSTRAINT nombre_restricción] DEFAULT valor_por_defecto FOR nombre_campo
        );
```



IMPORTANTE: Para seguir un mismo estilo a la hora de dar nombre a nuestras restricciones, usaremos la siguiente nomenclatura:

- **Claves Primarias:** PK_nombre_tabla
- **Claves Candidatas (con restricción UNIQUE):** UQ_nombre_campo
- **Claves Foráneas:** FK_tabla_hija_tabla_padre_nombre_relación
- **Verificaciones (CHECK):** CHK_nombre_descriptivo, por ejemplo CHK_salario_max

4.3.2. Creación de campos calculados

En ocasiones tenemos que crear campos en nuestras tablas cuyo valor se obtiene a partir de otros de la misma tabla. Estos campos calculados en la tabla se identifican con la siguiente sintaxis:

nombre_campo_calculado AS expresión.

Si estamos creando una tabla con código, cantidad, precio y un campo total que sea el resultado de multiplicar cantidad*precio, en la instrucción CREATE TABLE pondremos para el campo total:

total AS (cantidad*precio)

sin especificar ni tipo de dato ni nada más: el servidor le asigna el tipo adecuado.

4.3.3. Creación de tablas a partir de otras ya existentes

Para crear una tabla a partir de otra u otras ya existentes, de tal manera que se crea la estructura de la tabla y además se le añaden los datos de las tablas a partir de la cual se ha creado; deberemos usar la sintaxis siguiente.

Sintaxis:

SELECT { lista_de_campos| * } **INTO** [#][#]nombre_tabla_nueva
FROM .....

Si antes del nombre de la nueva tabla ponemos **#**, estamos indicando que la tabla es **TEMPORAL LOCAL** (*sólo visible en la sesión actual*). Si ponemos **##** la tabla será **TEMPORAL GLOBAL** *(visible en todas las sesiones).*

IMPORTANTE:

La cláusula SELECT devuelve la información con la que nosotros queremos crear nuestra nueva tabla, tanto en lo que se refiere a campos como a filas, es decir ESTRUCTURA y CONTENIDO.

4.3.4. Alterar tablas (modificar la estructura) en SQL Server

Con este tipo de sentencias podemos modificar la estructura de una tabla añadiendo un campo nuevo, modificando uno existente, eliminándolo, añadiendo una nueva restricción…

Agregar campos a una tabla existente

Sintaxis:

```
ALTER TABLE nombre_tabla_a_modificar ADD
      nombre_campo1_nuevo  tipo_dato  [NOT] NULL  [DEFAULT valor_por_defecto]  [RESTRICCIÓN PK, UQ ó FK]
   [, nombre_campo2_nuevo  tipo_dato  [NOT] NULL  [DEFAULT valor_por_defecto]  [RESTRICCIÓN PK, UQ ó FK],
      ...,
      nombre_campoN_nuevo  tipo_dato  [NOT] NULL  [DEFAULT valor_por_defecto]  [RESTRICCIÓN PK, UQ ó FK]
   ] ;
```

IMPORTANTE:

- En SQL Server **no es necesario indicar la palabra COLUMN** después de ADD.
- Si la tabla ya tiene datos hay que tener en cuenta que los campos nuevos o bien permiten NULOS, o SI NO PERMITEN NULOS se debe especificar un valor por defecto.

4.3.5. Modificar campos de una tabla existente

Sintaxis:

```
ALTER TABLE nombre_tabla_a_modificar ALTER COLUMN
        nombre_campo1_a_mod   tipo_dato   [NOT] NULL    [DEFAULT valor_por_defecto]   [RESTRICCIÓN PK, UQ ó FK]
      [, nombre_campo2_ a_mod   tipo_dato   [NOT] NULL   [DEFAULT valor_por_defecto]   [RESTRICCIÓN PK, UQ ó FK],
        ...,
        nombre_campoN_ a_mod   tipo_dato   [NOT] NULL  [DEFAULT valor_por_defecto]  [RESTRICCIÓN PK, UQ ó FK]] ;
```

IMPORTANTE: Se indicarán todas las propiedades que deseamos modificar a cada uno de los campos indicados en la instrucción, y también las que no se modificarán.

4.3.6. Eliminar campos de una tabla existente

Sintaxis:

```
ALTER TABLE nombre_tabla_a_modificar
DROP COLUMN   nombre_campo1_a_borrar
[,nombre_campo2_a_borrar, ..., nombre_campoN_a_borrar
  ];
```

4.3.7. Borrar tablas (eliminar la estructura y el contenido) en SQL Server

Sintaxis:

DROP TABLE *nombre_tabla_a_borrar;*

IMPORTANTE: No confundir DROP TABLE, *que ELIMINA EL OBJETO de la bd*, con las cláusulas DELETE o TRUNCATE, *que eliminan el contenido de la tabla, pero el objeto sigue existiendo en la bd, aunque vacío.*

No se puede usar DROP TABLE para quitar una tabla a la que se haga referencia con una restricción FOREIGN KEY, primero es necesario borrar la restricción o bien borrar la tabla de referencia.

4.4. CREACIÓN, MODIFICACIÓN Y BORRADO DE RESTRICCIONES (CONSTRAINTS)

En la creación de tablas hemos visto como se indica que un campo tiene una determinada restricción. En este apartado vamos a ver como agregar, modificar o borrar restricciones de tablas ya creadas.

4.4.1. Agregar restricciones a una tabla en SQL Server

Sintaxis:

```
ALTER TABLE  nombre_tabla_a_modificar
        ADD  CONSTRAINT nombre_restricción  PRIMARY KEY (campoPK1, campoPK2,..., campoPKN),
        [CONSTRAINT nombre_restricción] UNIQUE (campoUQ1, campoUQ2,..., campoUQN),
        [CONSTRAINT nombre_restricción] FOREIGN KEY  (campoFK1, campoFK2,..., campoFKN)
          REFERENCES tabla_padre(campo_clave)
           [ ON DELETE {CASCADE | NO ACTION | SET NULL | SET DEFAULT} ]
           [ ON UPDATE {CASCADE | NO ACTION | SET NULL | SET DEFAULT } ],
        [CONSTRAINT nombre_restricción] CHECK  condición_a_cumplir_la_columna;
```

NOTA: *En la misma instrucción se pueden agregar 1 ó más restricciones del mismo o de distinto tipo.*

4.4.2. Modificar restricciones de una tabla en SQL Server

La instrucción ALTER TABLE nos permite, como ya hemos visto:

- Agregar, modificar o eliminar columnas
- Agregar o eliminar restricciones PK, FK, UNIQUE y CHECK

La instrucción ALTER TABLE **NO** nos permite

- Modificar restricciones, entonces **para cambiar una restricción hay que borrarla y volver a agregarla**.

4.4.3. Eliminar restricciones de una tabla en SQL Server

Sintaxis:

ALTER TABLE nombre_tabla_a_modificar
DROP CONSTRAINT *nombre_restricción_a_borrar1*
[,nombre_restricción_a_borrar2, ...,
nombre_restricción_a_borrarN] ;

NOTA: *En la misma instrucción se pueden eliminar 1 ó más restricciones del mismo o de distinto tipo.*

4.4.4. Habilitar/deshabilitar restricciones

Sintaxis:

ALTER TABLE nombre_tabla_a_modificar
{ **CHECK | NOCHECK** } CONSTRAINT *nombre_restricción_a_habilitar_o_deshabilitar*;

IMPORTANTE: Sólo se pueden deshabilitar las restricciones **CHECK** y **FOREIGN KEY**