

BD

Bases de Datos

UD 4

Lenguaje de definición de datos (DDL)

ÍNDICE

1. Introducción
2. Tipos de datos en SQL Server
3. Administración de tablas (modo gráfico) e inserción de datos.
4. Administración de tablas (modo SQL)
 - 4.1 Creación y modificación de tablas (CREATE TABLE)
 - 4.2 Borrado y vaciado de tablas (DROP y TRUNCATE)
 - 4.3 Creación de índices (CREATE INDEX)

1. Del diseño lógico al diseño físico

El **diseño físico** es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él.

En nuestro caso utilizaremos **Microsoft SQL Server**, y como su sintaxis se adapta en gran medida a SQL ANSI, los comandos que usaremos serán muy similares en el resto de SGBD relacionales.

Cuando llegamos a este punto disponemos de:

- descripción del sistema de información
- el esquema conceptual del modelo entidad-relación
- el esquema del diccionario de datos
- el esquema relacional del modelo relacional

Mientras que en el diseño lógico se especifica qué se guarda, en el diseño físico se especifica cómo se guarda. Para ello, el diseñador debe conocer muy bien toda la funcionalidad del SGBD concreto que se vaya a utilizar y también el sistema informático sobre el que éste va a trabajar.

El diseño físico no es una etapa aislada, ya que algunas decisiones que se tomen durante su desarrollo, por ejemplo, para mejorar las prestaciones, pueden provocar una reestructuración del esquema lógico.

El objetivo de esta etapa es producir una descripción de la implementación de la base de datos en memoria secundaria. Esta descripción incluye las estructuras de almacenamiento y los métodos de acceso que se utilizarán para conseguir un acceso eficiente a los datos.

El objetivo por tanto es:

- Conocer las **utilidades o herramientas** para la definición de información en un SGBD.
- Conocer las **órdenes SQL** para la definición de datos.
- Elegir el **tipo de dato** más adecuado para cada tipo de información.
- Definir las **estructuras físicas** de almacenamiento e implantar todas las **restricciones** reflejadas en el diseño lógico (clave primaria, clave alternativa, clave ajena...).

En esta unidad didáctica estudiaremos las instrucciones o comandos SQL del lenguaje de definición de datos (DDL) que permiten la creación de nuestra base de datos y que incluirá las tablas con sus campos e índices.

La primera fase del diseño lógico consiste en traducir el esquema lógico global en un esquema que se pueda implementar en el SGBD escogido. Para ello, es necesario conocer toda la funcionalidad que éste ofrece. Por ejemplo, el diseñador deberá saber:

- Si el sistema soporta la definición de claves primarias, claves ajenas y claves alternativas.
- Si el sistema soporta la definición de datos requeridos (es decir, si se pueden definir atributos como no nulos).
- Si el sistema soporta la definición de dominios.
- Si el sistema soporta la definición de reglas de negocio (es decir, restricciones de cardinalidad o de cálculo)
- Cómo se crean las tablas base

Para poder conectar a la BD de **SQL Server** podemos utilizar dos herramientas:

- **Por línea de comandos: [sqlcmd.exe](#)**
- **Mediante la aplicación de escritorio [SQL Server Management Studio](#)**

Las instrucciones que vamos a ejecutar funcionan en las dos herramientas, aunque **SQL Server Management Studio** proporciona mucha más información de forma gráfica.

UTILIZACIÓN DE MS SQL SERVER

Para este curso, trabajaremos mediante una **máquina virtual** con Windows 10 Pro y MS SQL Server ya instalado para trabajar con él. Podéis descargar la máquina virtual desde el siguiente enlace:



https://drive.google.com/drive/folders/1wi9eHxx1K_Dg_-_d9l79qLjOIwUd7Rj?usp=sharing

Son un total de 9 archivos de 2GB cada uno (descargadla desde casa para evitar saturar la red del instituto).

Dicha máquina virtual se abre con el programa **VMWare Player** (podéis instalar la versión gratuita desde <https://www.vmware.com/go/getplayer-win>)

2. Tipos de datos en SQL Server

En SQL Server, cada columna tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener: un número entero, una cadena de caracteres, importes, fecha/hora, etc. Los tipos de datos más utilizados son los siguientes:

Numéricos

Los datos numéricos son muy importantes de cara a almacenar datos numéricos tales como: edades, cantidades con o sin decimales, valores booleanos (0 o 1), etc.

ENTEROS	
BIT	Acepta los valores 0, 1 o NULL
TINYINT	Acepta los valores del 0 al 255 (no permite valores negativos)
SMALLINT	Acepta los valores del -32.768 al 32.767 (permite negativos)
INT	Acepta los valores del -2.147.483.648 a 2.147.483.647
<i>NOTA: Para claves principales propias (codAutor, codPersonaje, etc.) utilizaremos SMALLINT o INT.</i>	
DECIMALES Y MONETARIOS (todos incluyen valores negativos)	
DECIMAL	Precisión hasta 9 dígitos (con decimales) Ejemplo. DECIMAL (5,2): 5 dígitos de los cuales 2 son decimales. 123,00
NUMERIC	Precisión hasta 19 dígitos (con decimales y mayor precisión). Ejemplo. Numeric (10,5): 10 dígitos de los cuales 5 son decimales. 12345,12000
SMALLMONEY	Precisión de 9 dígitos. Para importes (menos precisión).
MONEY	Precisión de 19 dígitos. Para importes (más precisión).
NOTA: Muchos expertos NO recomiendan el uso de SMALLMONEY ni MONEY. Siempre tendremos más control utilizando DECIMAL y NUMERIC.	

Cadenas de caracteres

Este tipo de dato es muy importante cuando deseemos almacenar información de nombres, apellidos, direcciones, observaciones, etc. En definitiva, campo que contenga texto o números con los que no tengamos intención de operar.


Tipo de cadena	Carácter	Unicode
FIJA	CHAR	NCHAR
VARIABLE	VARCHAR	NVARCHAR

Los tipos de datos *Unicode* están diseñados para que contengan texto con símbolos de idiomas diferentes del alfabeto (japonés, chino, griego, etc.). No obstante, son menos eficientes que los tipos *carácter*, ya que utilizan dos bytes para representar cada letra a diferencia de los de tipo *carácter* que utilizan sólo una.

Del mismo modo tenemos las cadenas de longitud fija y de longitud variable. Las de longitud variable cuentan con ventaja puesto que no reservan un tamaño específico, sino que se va reservando a medida que se necesita. Las de longitud fija reservan el espacio, aunque finalmente éste no se utilice, por lo que son más ineficientes. Ej. Un tipo de dato definido como Char(20) para almacenar la cadena “hola” utilizará las 20 posiciones aunque sólo necesite 4 caracteres.

Fecha/hora

En algunas situaciones será necesario que almacenemos a qué hora se ha realizado cierta operación (compra, venta, reserva de libro, etc.) para lo cual será necesario este tipo de dato:

DATE	Campo sólo fecha (si sólo necesitamos guardar la fecha)
DATETIME	<p>Campo fecha/hora con precisión de segundos con decimales</p> <pre> DECLARE @date DATETIME SET @date = '2016-10-09 23:58:58.991' SELECT @date 'DATETIME' </pre> 
SMALLDATETIME	<p>Campo fecha/hora con precisión de segundos SIN decimales</p> <pre> DECLARE @date SMALLDATETIME SET @date = '2016-10-09 23:58:58.991' SELECT @date 'SMALLDATETIME' </pre> 

Tipos de datos en SQL Server – Documentación oficial

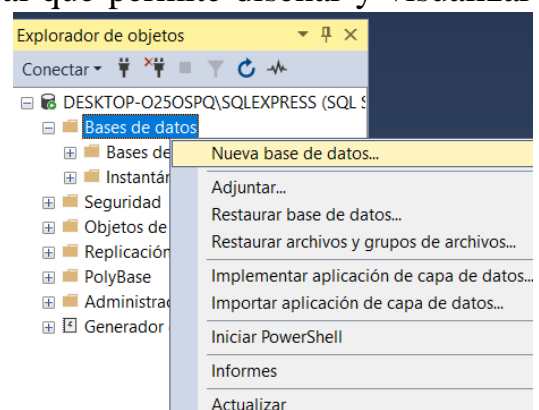
<https://docs.microsoft.com/es-es/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15>

3. Administración de tablas (modo gráfico SQL Server MS)

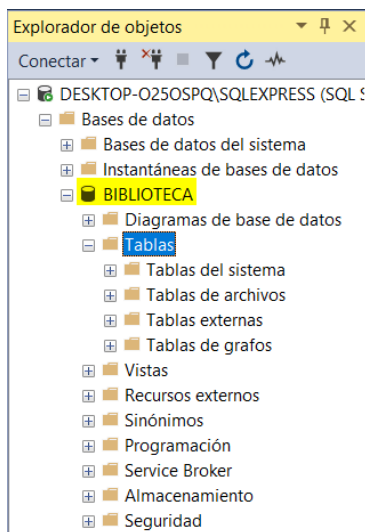
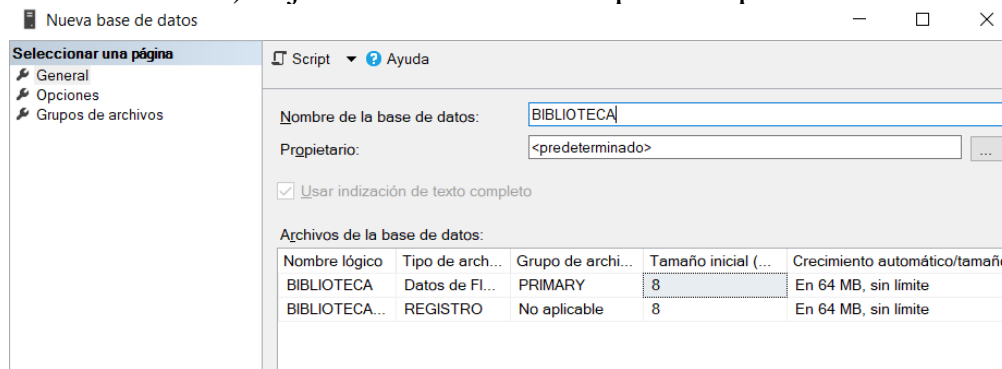
Utilizaremos la herramienta **Microsoft SQL Server Management Studio**.

El **Diseñador de Tablas** es una herramienta visual que permite diseñar y visualizar tablas de bases de datos. Podemos utilizarlo para crear, editar o eliminar tablas, columnas, claves, índices, relaciones y restricciones.

Antes de nada, deberemos **crear una base de datos**:



En la pestaña **General**, indicaremos el **nombre de la base de datos** (ej. BIBLIOTECA) dejando el resto de las opciones por defecto:

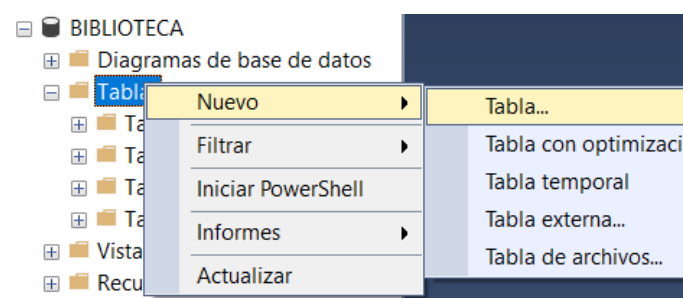


Aparecerá la nueva Base de datos **BIBLIOTECA**.

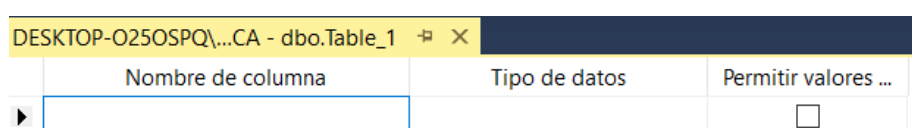
Si la desplegamos podremos acceder a las tablas (por defecto existen tablas de sistema, de archivos, externas y de grafos), pero no debemos gestionarlas nosotros mismos.

CREACIÓN DE TABLAS

Clic con el botón derecho en **Tablas** de la base de datos y selecciona **Crear > Tabla**:



Aparecerá el editor de tablas, en el que definiremos las columnas, el tipo de dato que tendrá, si permite valores nulos, claves principales y ajenas.



Ejemplo. Crearemos las tablas para el siguiente modelo relacional:

LIBRO (ISBN, título, autor, editorial)

PK: ISBN

SOCIO (idSocio, nombre)

PK: idSocio

RESERVA (ISBN, idSocio, fechaReserva)

PK: ISBN, idSocio, fechaReserva

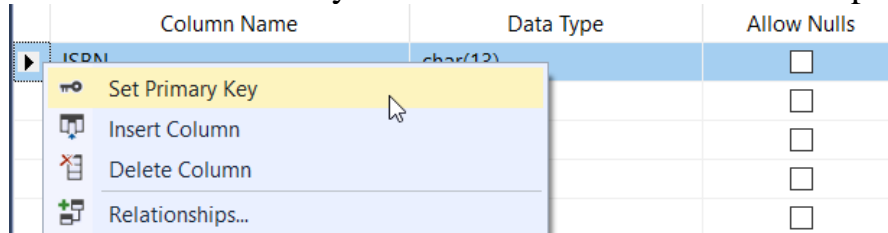
FK: ISBN → LIBRO

FK: idSocio → SOCIO

Tabla LIBRO

Column Name	Data Type	Allow Nulls
ISBN	char(13)	<input type="checkbox"/>
título	varchar(50)	<input type="checkbox"/>
autor	varchar(80)	<input type="checkbox"/>
editorial	varchar(40)	<input type="checkbox"/>

Para definir la clave principal, nos situamos en la parte derecha de la/s columna/s, clic con el botón derecho y seleccionamos: Establecer clave principal.

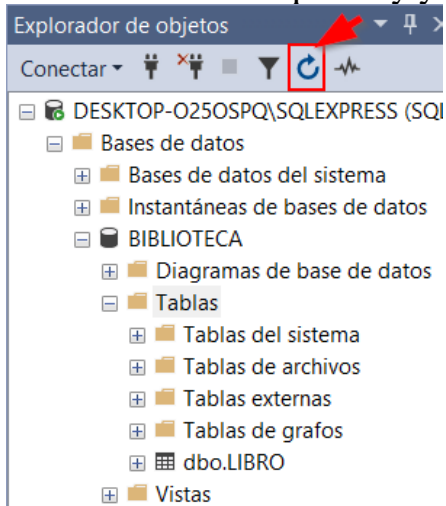


Cuando tengamos lista la tabla, hacemos [Control + S] o seleccionamos el botón guardar

Escribimos el nombre de la tabla:

Hacemos clic en Aceptar.

Refrescamos el Esquema y ya podremos visualizar la tabla.



Repetimos la operación para el resto de las tablas.

Tabla SOCIO:









	Column Name	Data Type	Allow Nulls
	idSocio	int	<input type="checkbox"/>
	nombre	varchar(80)	<input type="checkbox"/>
			<input type="checkbox"/>

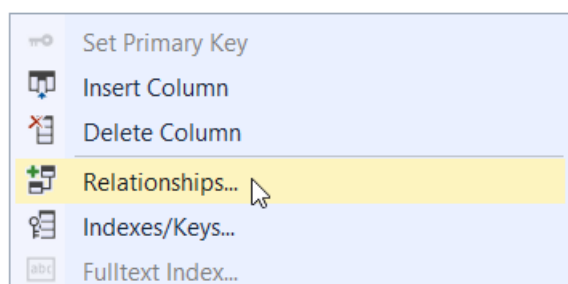
Tabla RESERVAS:

	Column Name	Data Type	Allow Nulls
	ISBN	char(13)	<input type="checkbox"/>
	idSocio	int	<input type="checkbox"/>
	fechaReserva	date	<input type="checkbox"/>

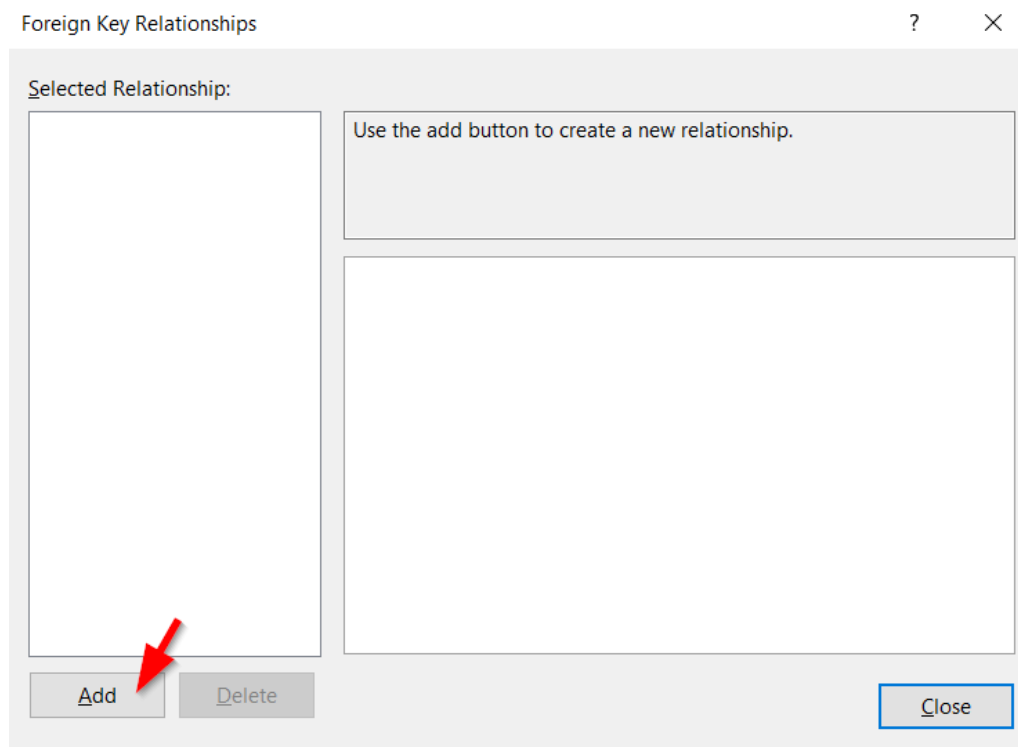
CLAVES AJENAS

Si alguna tabla tiene columnas que sean claves ajenas (ISBN e idSocio), guardaremos siempre antes la tabla y le daremos un nombre y después: haremos clic con el botón derecho en la columna en cuestión y seleccionaremos **Relaciones:**

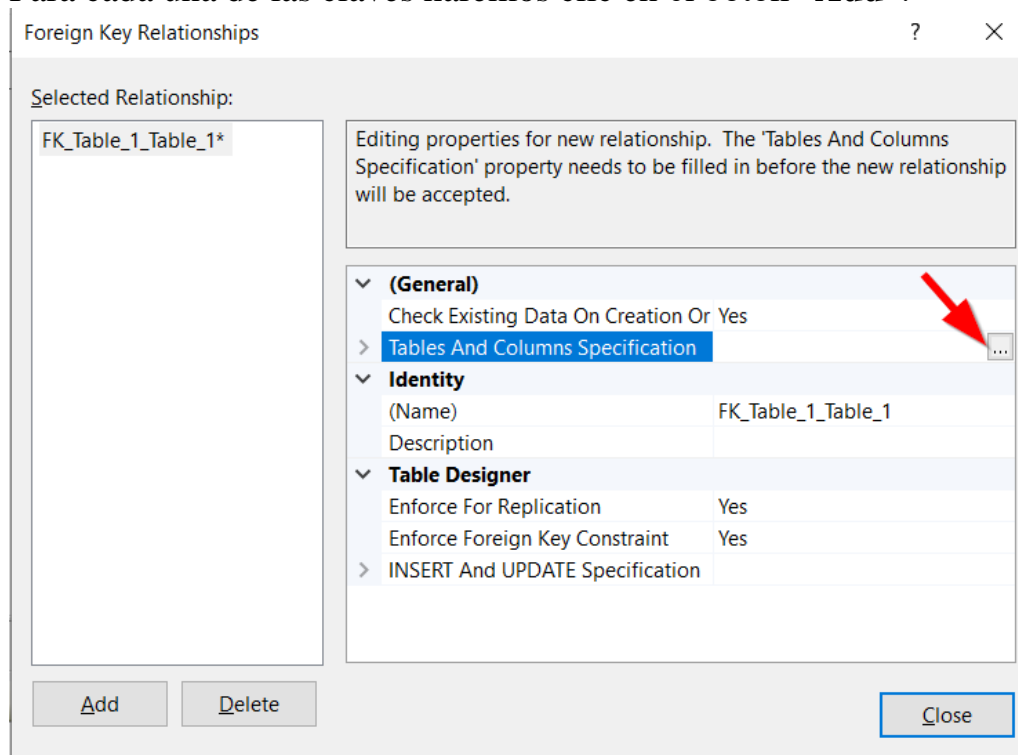
	Column Name	Data Type	Allow Nulls
	ISBN	char(13)	<input type="checkbox"/>
	idSocio	int	<input type="checkbox"/>
	fechaReserva	date	<input type="checkbox"/>
			<input type="checkbox"/>



Aparecerá el siguiente menú:



Para cada una de las claves haremos clic en el botón “Add”:



Por defecto aparecen una serie de campos:

Tables and Columns

Relationship name:
FK_Table_1_Table_1

Primary key table: Table_1 Foreign key table: Table_1

ISBN	ISBN
idSocio	idSocio
fechaReserva	fechaReserva

OK Cancel

Tendremos que seleccionar de la parte izquierda la tabla original y el campo que representa la PK y en la parte izquierda la tabla con la clave ajena y el campo con la clave ajena.

Continuando con nuestro ejemplo: damos de alta la clave ajena **ISBN** a LIBRO:

Tablas y columnas

Nombre de la relación:
FK_RESERVA_LIBRO

Tabla de clave principal: LIBRO Tabla de clave externa: RESERVA

ISBN	ISBN
------	------

OK Cancel

El nombre de la clave ajena será FK_TABLA1_TABLA2 o si tenemos más de una clave ajena que proviene de la misma tabla (PK compuesta) lo haremos como FK_TABLA_NOMBRECAMPOFK.

En nuestro ejemplo:

- Tabla: RESERVA
- Campo FK: LIBRO

Y después la de **idSocio** a SOCIO:

Tablas y columnas

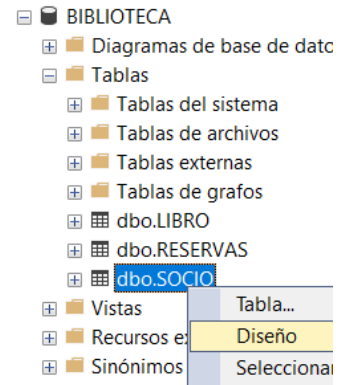
Nombre de la relación:
FK_RESERVA_SOCIO

Tabla de clave principal: SOCIO Tabla de clave externa: RESERVA

idSocio	idSocio
---------	---------

OK Cancel

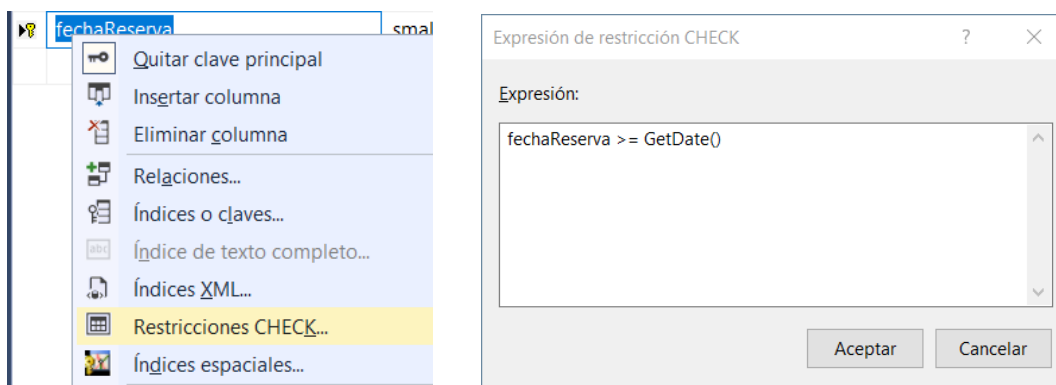
En cualquier momento podemos modificar las columnas de una tabla o modificar su estructura haciendo clic con el botón derecho sobre la tabla en cuestión y seleccionando la opción Diseño:



RESTRICCIONES CHECK

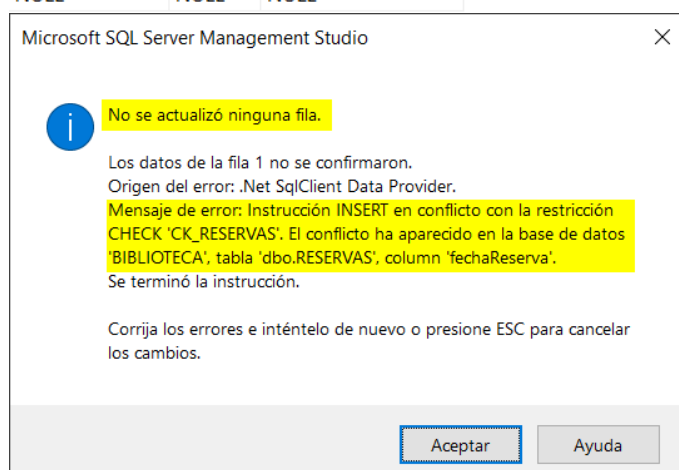
En ciertas situaciones puede interesarnos poner restricciones sobre las columnas de la tabla para controlar que los valores que se insertan son correctos.

En nuestro ejemplo vamos a registrar una restricción sobre el campo fechaReserva para que no pueda ser anterior a la fecha del día (*GetDate es una función que devuelve la fecha del día*).



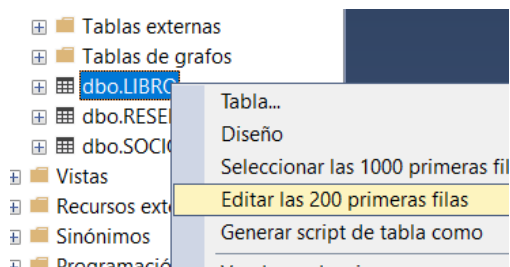
Si intentamos insertar una fecha anterior a la del día:

ISBN	idSocio	fechaReserva
978847888441	1	2015-01-09 00:00:00
NULL	NULL	NULL



INSERCIÓN DE REGISTROS

Para insertar registros seleccionamos la tabla en cuestión y hacemos clic en **Editar las 200 primeras filas**:



Insertamos dos registros en la tabla LIBRO:

DESKTOP-O25OSPQ...TECA - dbo.LIBRO				
	ISBN	titulo	autor	editorial
	9788478884452	Harry Potter y la Piedra Filosofal: 1	Rowling, J.K.	Salamandra
	9788496208568	JUEGO DE TRONOS: CANCION DE HIELO Y FUEGO, 1	GEORGE R.R. MARTIN	GIGAMESH
▶*	NULL	NULL	NULL	NULL

Insertamos tres registros en la tabla SOCIO:

DESKTOP-O25OSPQ...TECA - dbo.SOCIO		
	idSocio	nombre
	1	Carlos
	2	María
	3	Pepe
▶*	NULL	NULL

Insertamos registros en la tabla RESERVA:

CASO 1: Probamos reservar un Libro que no existe

DESKTOP-O25OSPQ...A - dbo.RESERVAS			
	ISBN	idSocio	fechaReser...
	1	1	2020-01-01
*	NULL	NULL	NULL



No se actualizó ninguna fila.

Los datos de la fila 1 no se confirmaron.

Origen del error: .Net SqlClient Data Provider.

Mensaje de error: Instrucción INSERT en conflicto con la restricción FOREIGN KEY 'FK_RESERVAS_LIBRO'. El conflicto ha aparecido en la base de datos 'BIBLIOTECA', tabla 'dbo.LIBRO', column 'ISBN'.

Se terminó la instrucción.

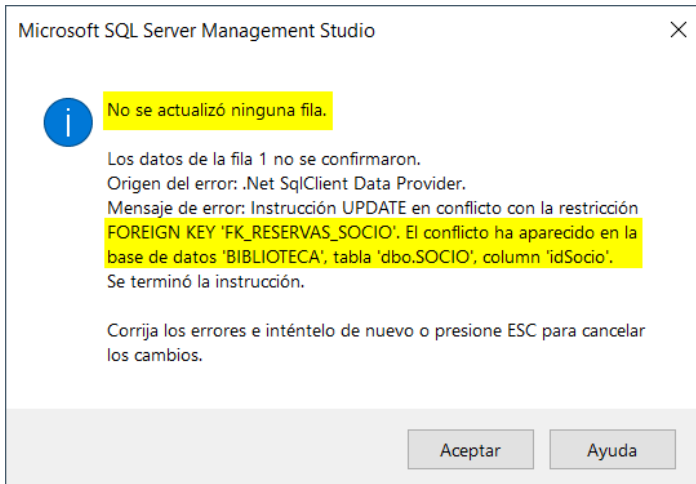
Corrija los errores e inténtelo de nuevo o presione ESC para cancelar los cambios.

Aparecerá un mensaje de error de incumplimiento de la restricción de clave ajena del atributo ISBN de la tabla RESERVAS sobre la tabla LIBRO.

NOTA: Para deshacer los cambios deberemos hacer clic sobre la tecla ESCAPE.

CASO 2: Probamos a reservar un libro que existe por un socio que no existe.

	ISBN	idSocio	fechaReser...
...	9788478884452	8	2020-10-10
*	NULL	NULL	NULL



Tampoco permite la inserción al no existir el idSocio = 8.

CASO 3: Insertamos datos cuya clave ajena existe en las tablas relacionadas.

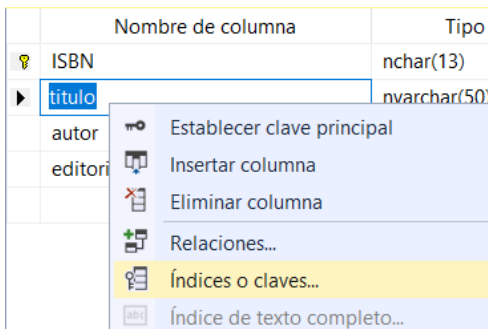
	ISBN	idSocio	fechaReserva
	9788478884452	2	2020-10-10 00:00:00
▶*	NULL	NULL	NULL

Al no contener errores de restricción de clave ajena permitirá realizar la inserción de los registros sin problemas.

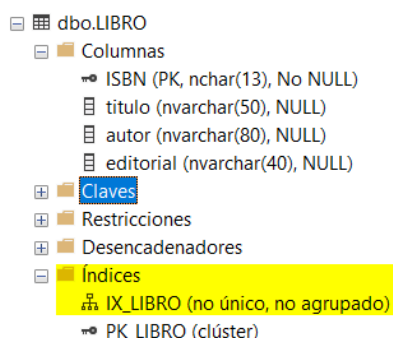
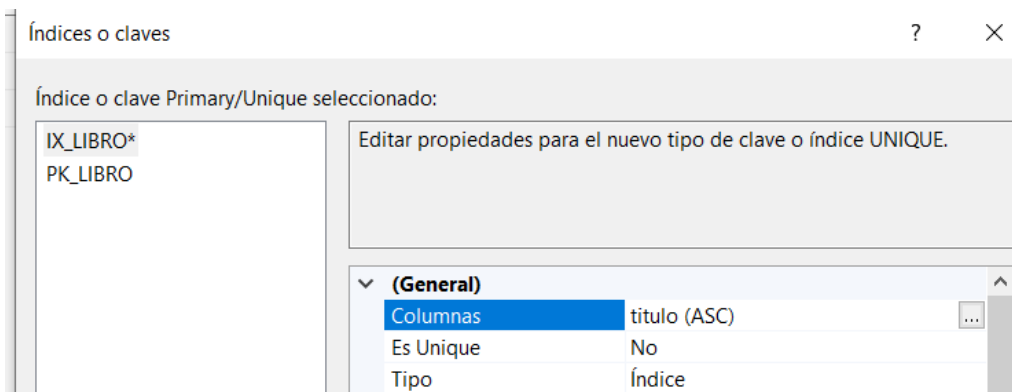
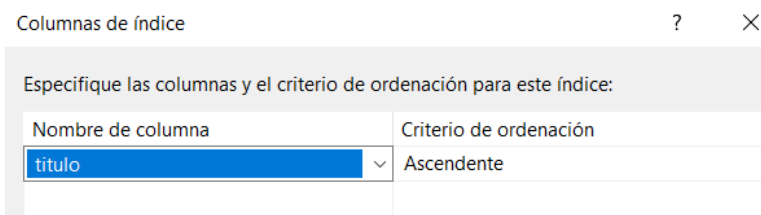
CREACIÓN DE ÍNDICES

En multitud de aplicaciones informáticas, se suelen realizar búsquedas por nombre / apellidos de una persona. En esos casos, tenemos a nuestro alcance la posibilidad de crear un índice sobre uno o varios campos de la tabla.

Ejemplo. En la tabla LIBROS nos interesa crear un índice sobre el campo título.

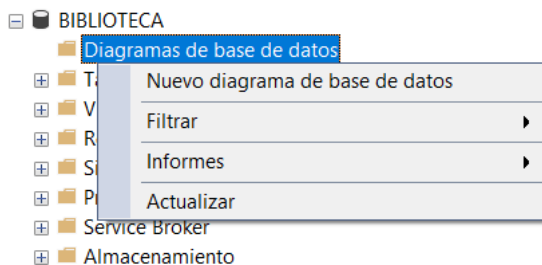


Hacemos clic en **Agregar** y después definimos qué campo actuará como índice y la ordenación de éste:

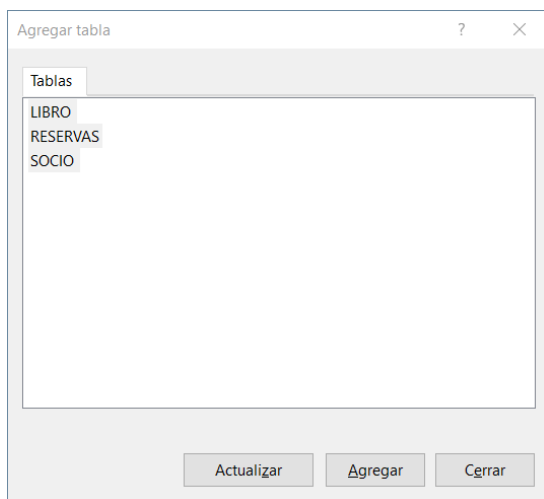


GENERACIÓN DEL ESQUEMA DE TABLAS

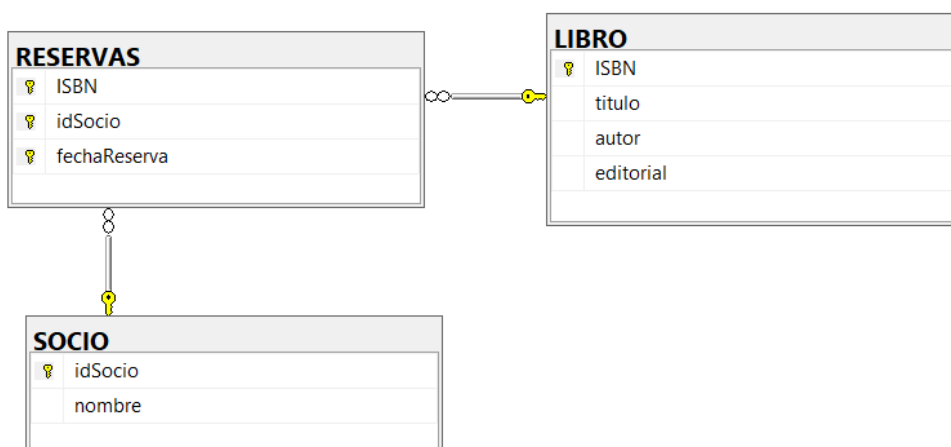
Por último, podemos utilizar una herramienta muy útil para obtener una representación gráfica de la base de datos, incluyendo las tablas y sus relaciones. Para obtenerlo, clic con el botón derecho encima de “Diagramas de base de datos” y seleccionamos “Nuevo diagrama de base de datos”.



Seleccionamos las tablas que queramos mostrar (todas) y hacemos clic en Agregar:

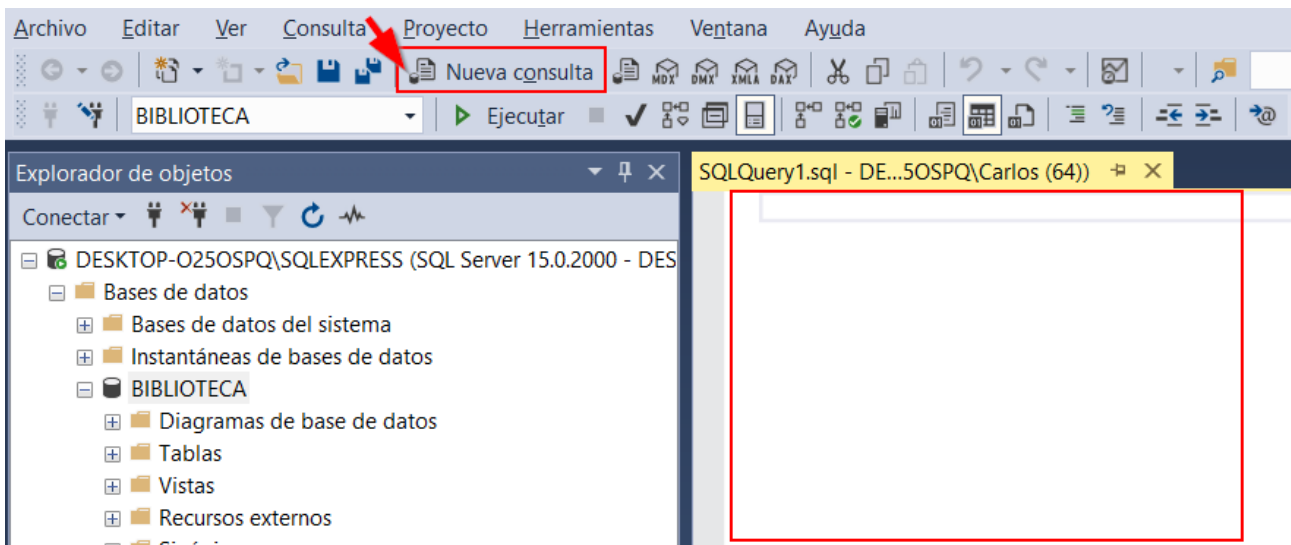


Mostrará el esquema de tablas de la base de datos:



4. Administración de tablas (modo SQL)

Para poder trabajar con el modo SQL deberemos hacer clic en el botón **Nueva consulta**, se abrirá el editor de código a la derecha:



Para crear una base de datos nueva utilizaremos el comando:

CREATE DATABASE NombreBD;

Para eliminar una base de datos nueva utilizaremos el comando:

DROP DATABASE NombreBD;

Si tenemos varias bases de datos, es importante definir cuál queremos utilizar. Las tablas se crearán en aquella que se esté “usando” en ese momento, por tanto:

USE NombreBD;

Para saber qué bases de datos hay en el sistema ejecutaremos la siguiente consulta:

```
SELECT [name] AS DATABASE_NAME, database_id, create_date
FROM sys.databases
WHERE name NOT IN ('master', 'tempdb', 'model', 'msdb')
ORDER BY name;
```

Ejercicio 4.1 (no se entrega)

Crea una base de datos y después comprueba que existe ejecutando la consulta. Bórrala después y comprueba que se ya no aparece.

NOTA: Si tenemos varias sentencias en el mismo editor de código SQL, deberemos seleccionar la consulta y presionar la tecla F5 para ejecutarla. Si no lo hacemos así, se ejecutará en orden todas las sentencias que haya en el editor.

4.1 Creación y modificación de tablas (CREATE /ALTER TABLE)

Para crear una tabla utilizaremos el comando CREATE TABLE:

```
CREATE TABLE NOMBRETABLA (  
    Campo1 <tipoDato>,  
    Campo2 <tipoDato>, ...  
    CampoN <tipoDato>  
);
```

La sintaxis para definir un “campo” en la instrucción anterior es:

```
nombre_col <tipoDato>  
    [NOT NULL | NULL]  
    [DEFAULT valor_por_defecto]  
[IDENTITY (inicio, incremento)]Autonumérico  
[PRIMARY KEY]  
[UNIQUE]
```

CREATE TABLE en SQL Server – Documentación oficial

<https://docs.microsoft.com/es-es/sql/t-sql/statements/create-table-transact-sql?view=sql-server-ver15>

Para crear una tabla en el diseño físico partiremos del modelo relacional, el diccionario de datos y la documentación de las restricciones.

Herramienta para probar las sentencias DDL

SQLFiddle → <http://sqlfiddle.com/>

RESTRICCIONES DE INTEGRIDAD

4.1.1 Restricción de clave primaria

La necesidad de identificar un registro unívocamente nos obliga a crear una restricción por el campo clave, ya que este no podrá ser nulo ni tener valores duplicados.

De hecho, no se pueden editar los datos de una tabla que no tenga clave primaria, aunque sí se pueden insertar registros.

CLAVES PRIMARIAS DE UNA SOLA COLUMNA

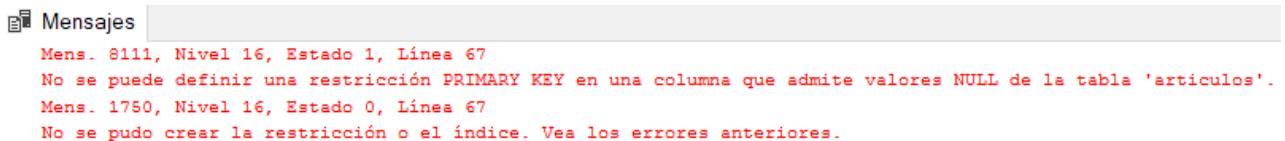
Crear clave primaria en el momento de creación de la tabla

```
CREATE TABLE ARTICULOS (  
    codigo      CHAR(4),  
    descripcion  VARCHAR(100),  
    precio      DECIMAL(10,2)  
    CONSTRAINT PK_articulos PRIMARY KEY (codigo)  
);
```

Añadir la clave primaria a una tabla ya existente

```
ALTER TABLE ARTICULOS  
ADD CONSTRAINT PK_articulos PRIMARY KEY (codigo);
```

IMPORTANTE: El campo que decidamos como clave primaria debe tener la restricción NOT NULL, sino SQL Server devolverá el siguiente error:



Mensajes

Mens. 8111, Nivel 16, Estado 1, Línea 67
No se puede definir una restricción PRIMARY KEY en una columna que admite valores NULL de la tabla 'articulos'.
Mens. 1750, Nivel 16, Estado 0, Línea 67
No se pudo crear la restricción o el índice. Vea los errores anteriores.

Eliminar la clave primaria en una tabla existente

```
ALTER TABLE ARTICULOS  
DROP CONSTRAINT PK_articulos;
```

CLAVES PRIMARIAS COMPUESTAS POR MÁS DE UNA COLUMNA

En ocasiones la clave está compuesta por varios campos. Por ejemplo, en una tabla donde se guardan las revisiones de ITV de vehículos la clave está formada por la fecha y la matrícula del vehículo. Igual que en las claves primarias de un valor, tenemos dos modos de realizarlo:

Crear clave primaria en el momento de creación de la tabla

```
CREATE TABLE REVISION_ITV (  
    matricula    CHAR(10),  
    fecha        DATE,  
    estado       VARCHAR(100)  
    CONSTRAINT PK_mat_fecha PRIMARY KEY (matricula, fecha)  
);
```

Añadir la clave primaria a una tabla ya existente

```
CREATE TABLE REVISION_ITV (  
    matricula    CHAR (10) NOT NULL,  
    fecha        DATE NOT NULL,  
    estado       VARCHAR(100)  
);  
  
ALTER TABLE REVISION_ITV  
ADD CONSTRAINT PK_mat_fecha PRIMARY KEY (matricula, fecha);
```

Eliminar la clave primaria de una tabla ya existente

```
ALTER TABLE REVISION_ITV  
DROP CONSTRAINT PK_mat_fecha;
```

4.1.2 Restricción de campo obligatorio

Cuando un campo no puede tener valores nulos, decimos que es un campo obligatorio.

En nuestro ejemplo indicaremos la descripción como campo obligatorio para que ningún artículo tenga la descripción vacía.

Crear campo obligatorio al crear la tabla

```
CREATE TABLE ARTICULOS (  
    codigo      VARCHAR(4) PRIMARY KEY,  
    descripcion  VARCHAR(100) NOT NULL,  
    precio      DECIMAL(10,2)  
    CONSTRAINT PK_articulos PRIMARY KEY (codigo)  
);
```

Eliminar la restricción de campo obligatorio

```
ALTER TABLE articulos  
ALTER COLUMN descripcion VARCHAR(100) NULL;
```

Añadir la restricción de campo obligatorio en una tabla que ya existe

```
ALTER TABLE ARTICULOS  
ALTER COLUMN descripcion VARCHAR(100) NOT NULL;
```

4.1.3 Restricción de campo con valores únicos

Cuando un campo no puede tener valores repetidos, decimos que es un campo único.

Lo que hace realmente la base de datos es crear un índice con valores únicos.

Crear campo obligatorio al crear la tabla

```
CREATE TABLE ARTICULOS (  
    codigo      CHAR(4),  
    descripcion  VARCHAR(100) NOT NULL UNIQUE,  
    precio      DECIMAL(10,2)  
    CONSTRAINT PK_articulos PRIMARY KEY (codigo)  
);
```

Eliminar la restricción de campo obligatorio

```
ALTER TABLE ARTICULOS  
ALTER COLUMN descripcion VARCHAR(100) NOT NULL;
```

Añadir la restricción de campo obligatorio en una tabla que ya existe

```
ALTER TABLE ARTICULOS  
ADD UNIQUE (descripcion);
```

4.1.4 Restricción de campo por clave ajena

Como ya hemos visto en los esquemas lógicos relacionales obtenidos de los esquemas conceptuales, existen claves ajenas o extranjeras que contienen valores de la clave primaria de otra tabla con la que se relacionan.

En un esquema que tiene localidades y provincias, donde indicamos en cada localidad de qué provincia es, podríamos tener las siguientes tablas:

```
CREATE TABLE PROVINCIAS (  
    cod_prov INT,  
    provincia VARCHAR(40)  
    CONSTRAINT PK_provincias PRIMARY KEY (cod_prov)  
);
```

Imaginemos que tenemos una tabla MUNICIPIOS que está relacionada con la tabla PROVINCIAS (un municipio pertenece a una provincia y una provincia agrupa muchos municipios).

Crear clave ajena al crear la tabla

```
CREATE TABLE MUNICIPIOS (  
    cod_municipio INT,  
    cod_prov INT,  
    nombre_municipio VARCHAR(100)  
    CONSTRAINT PK_municipios PRIMARY KEY (cod_prov, cod_municipio)  
    CONSTRAINT FK_prov_muni FOREIGN KEY (cod_prov)  
        REFERENCES provincias (cod_prov)  
);
```

Añadir la restricción de clave ajena a una tabla existente

```
CREATE TABLE MUNICIPIOS (  
    cod_municipio INT,  
    cod_prov INT,  
    nombre_municipio VARCHAR(100)  
    CONSTRAINT PK_municip PRIMARY KEY (cod_prov, cod_municipio)  
);
```

```
ALTER TABLE MUNICIPIOS  
ADD CONSTRAINT FK_prov_muni FOREIGN KEY (cod_prov)  
REFERENCES provincias (cod_prov);
```

Eliminar la restricción de clave ajena

```
ALTER TABLE MUNICIPIOS  
DROP CONSTRAINT FK_prov_muni;
```

Ejercicio 4.2 (no se entrega)

Partiendo del esquema lógico, crea la tabla de proveedores de nuestra comunidad autónoma siguiente:

Proveedor (codProveedor, localidad, nombre, provincia)
CP: codProveedor

El diccionario de datos (sólo con restricciones de valor) es:

CAMPO	TIPO	LONGITUD	CARACTERÍSTICAS
cod_prov	Cadena	3	Clave Primaria No Nulo
localidad	Cadena	100	Nulo
nombre	Cadena	100	No nulo Único Clave Alternativa
provincia	Cadena	50	No nulo

Solución:

```
CREATE TABLE PROVEEDOR (  
    cod_prov CHAR(3) NOT NULL,  
    localidad VARCHAR(100) DEFAULT NULL,  
    nombre VARCHAR(100) NOT NULL UNIQUE,  
    provincia VARCHAR(100) NOT NULL  
    CONSTRAINT PK_proveedor PRIMARY KEY (cod_prov)  
);
```

4.1.5 Restricción CHECK

Las restricciones CHECK pueden introducirse tanto a la hora de crear la tabla con CREATE TABLE como tras haberla creado mediante la instrucción ALTER TABLE.

Ejemplo.

```
CREATE TABLE ARTICULOS (  
    codigo        CHAR(4),  
    descripcion   VARCHAR(100) NOT NULL UNIQUE,  
    precio        DECIMAL(10,2)  
    CONSTRAINT PK_articulos PRIMARY KEY (codigo),  
    CONSTRAINT CK_precio CHECK (precio > 0)  
);  
ALTER TABLE ARTÍCULOS  
ADD CONSTRAINT CK_precio CHECK (precio > 0);
```

Si necesitamos poner más condiciones, podemos utilizar los operadores AND y OR.

EJERCICIOS PARA PRACTICAR

Actividad 1. Escribe las sentencias de creación de tablas utilizando el lenguaje SQL visto en la teoría de la unidad. Deberás decidir el tipo de dato a utilizar y su precisión en cuanto a números enteros y cadenas de caracteres.

DUEÑO (DNI, nombre, dirección)

PK: DNI

COCHE (numBastidor, marca, modelo, DNI_dueño)

PK: numBastidor

FK: DNI_dueño → DUEÑO

Actividad 2. Escribe las sentencias de creación de tablas utilizando el lenguaje SQL visto en la teoría de la unidad. Deberás decidir el tipo de dato a utilizar y su precisión en cuanto a números enteros y cadenas de caracteres.

PROVEEDOR (codProv, nombre, dirección)

PK: codProv

ARTÍCULO (codArticulo, descripcion, precio, existencias)

PK: codArticulo

SUMINISTRO (codProv, codArticulo, unidades, fechaRecepcion)

PK: codProv, codArticulo

FK: codProv → PROVEEDOR

FK: codArticulo → ARTÍCULO

NOTA: Los identificadores codProv y codArtículo deben ser autonuméricos (deben empezar en 1 e incrementarse automáticamente).

Actividad 3. Escribe las sentencias de creación de tablas utilizando el lenguaje SQL visto en la teoría de la unidad. Deberás decidir el tipo de dato a utilizar y su precisión en cuanto a números enteros y cadenas de caracteres.

DEPARTAMENTO (codDpto, nombre, localidad)

PK: codDpto

EMPLEADO (codEmpleado, nombre, salario, codDpto, codEmpleadoJefe)

PK: codEmpleado

FK: codDpto → DEPARTAMENTO

FK: codEmpleadoJefe → EMPLEADO

Actividad 4. Escribe las sentencias de creación de tablas utilizando el lenguaje SQL visto en la teoría de la unidad. Deberás decidir el tipo de dato a utilizar y su precisión en cuanto a números enteros y cadenas de caracteres.

TEMA (codTema, nombre)
PK: codTema

LIBRO (ISBN, título, autor, numEjemplares, codTema)
PK: ISBN
FK: codTema → TEMA

SOCIO (DNI, nombre, dirección, fechaAlta, fechaBaja)
PK: DNI

TELEFONO_SOCIO (DNI, telefono)
PK: DNI, telefono
FK: DNI → SOCIO

PRESTAMO (DNI, ISBN, fechaPrestamo, fechaDevolucion)
PK: DNI, ISBN, fechaPrestamo
FK: DNI → SOCIO
FK: ISBN → LIBRO

NOTA: Ponemos fechaPréstamo como parte de la clave principal para que un socio pueda sacar el mismo libro más de una vez.

Actividad 5. Escribe las sentencias de creación de tablas utilizando el lenguaje SQL visto en la teoría de la unidad. Deberás decidir el tipo de dato a utilizar y su precisión en cuanto a números enteros y cadenas de caracteres.

CLIENTE (idCliente, ...)
PK: idCliente

PRODUCTO (idProducto, ...)
PK: idProducto

CENTRO_COMERCIAL (idCentro, ...)
PK: idCentro

COMPRA (idCliente, idProducto, idCentro, fecha)
PK: idCliente, idProducto, idCentro
FK: idCliente → CLIENTE
FK: idProducto → PRODUCTO
FK: idCentro → CENTRO_COMERCIAL

4.2 Borrado y truncado de tablas (DROP TABLE)

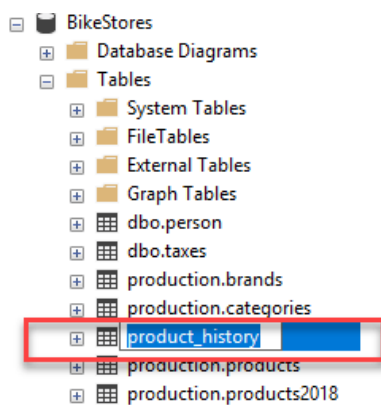
Con el comando DROP TABLE podemos eliminar una tabla, tanto su estructura como su contenido:

DROP TABLE PROVEEDOR;

Si sólo queremos eliminar el contenido, pero no la estructura utilizaremos:

TRUNCATE TABLE PROVEEDOR;

Podemos renombrar el nombre de una tabla editando el nombre desde el Explorador de objetos (clic con el botón derecho y seleccionando **Renombrar**)



CREACIÓN DE RESTRICCIONES SOBRE LAS COLUMNAS

Comenzaremos por crear una tabla de artículos con los siguientes campos:

```
CREATE TABLE ARTICULOS (  
    codigo      CHAR(4),  
    descripcion  VARCHAR(100),  
    precio       DECIMAL(10,2)  
);
```

Añadir un campo a una tabla existente

Una vez creada una tabla, es posible que necesitemos añadir, modificar o eliminar algún campo existente. Para ello utilizaremos las siguientes instrucciones SQL.

```
ALTER TABLE ARTICULOS  
    ADD descuento INT NOT NULL;
```

Modificar el tipo y las restricciones de un existente campo

Para cambiar un campo utilizaremos la siguiente sentencia:

```
ALTER TABLE ARTICULOS  
    ALTER COLUMN descuento DECIMAL(5,2) NOT NULL;
```

Eliminar un campo de una tabla existente

```
ALTER TABLE ARTICULOS  
    DROP COLUMN descuento;
```

4.3 Creación de índices

Los índices son estructuras que se crean en las Bases de Datos para poder controlar la integridad y realizar búsquedas de manera más eficiente.

Añadir índices

Si deseáramos crear un índice con valores repetidos cuando la tabla ya existe, utilizaríamos el comando siguiente:

```
CREATE INDEX INDEX_NAME  
    ON nombreTabla (col_name1,...);
```

Si deseáramos crear un índice cuando con valores únicos cuando la tabla ya existe, utilizaríamos el comando siguiente:

```
CREATE UNIQUE INDEX INDEX_NAME  
ON nombreTabla (col_name1,...);
```

Eliminar índices

Para eliminar un índice deberemos saber su nombre:

```
DROP INDEX INDEX_NAME  
ON nombreTabla;
```

Ejemplo de uso de índice en una base de datos sobre MySQL:

```
mysql> SELECT count(*) FROM `users` WHERE last_name LIKE 'a%' /*sql_no_cache*/;  
+-----+  
| count(*) |  
+-----+  
|      63285 |  
+-----+  
1 row in set (0.25 sec) <- Antes del índice  
  
mysql> ALTER TABLE `users` ADD INDEX ( `last_name` );  
Query OK, 1009024 rows affected (17.57 sec)  
Records: 1009024 Duplicates: 0 Warnings: 0  
  
mysql> SELECT count(*) FROM `users` WHERE last_name LIKE 'a%' /*sql_no_cache*/;  
+-----+  
| count(*) |  
+-----+  
|      63285 |  
+-----+  
1 row in set (0.06 sec) <- Después del índice  
  
mysql>
```