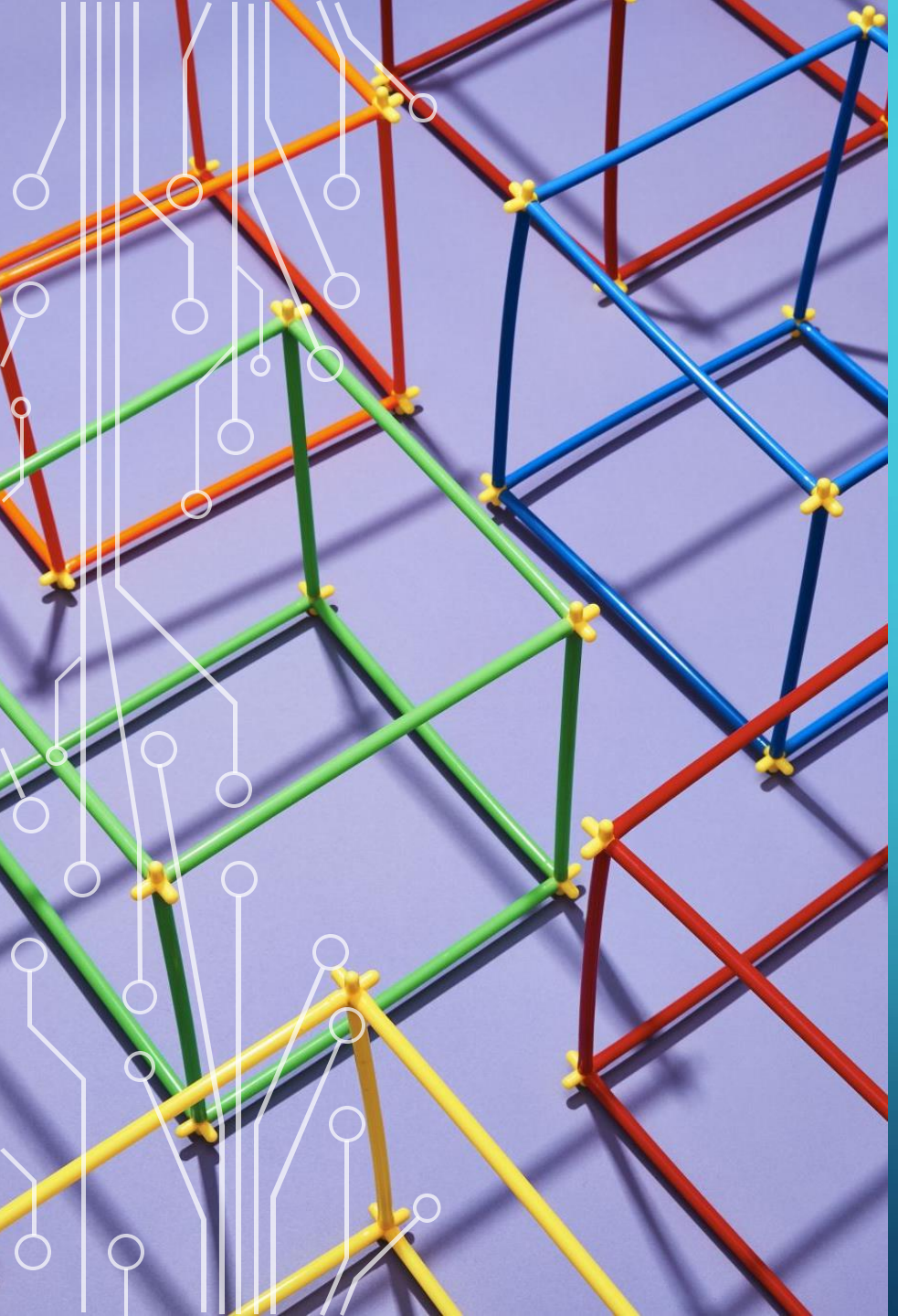
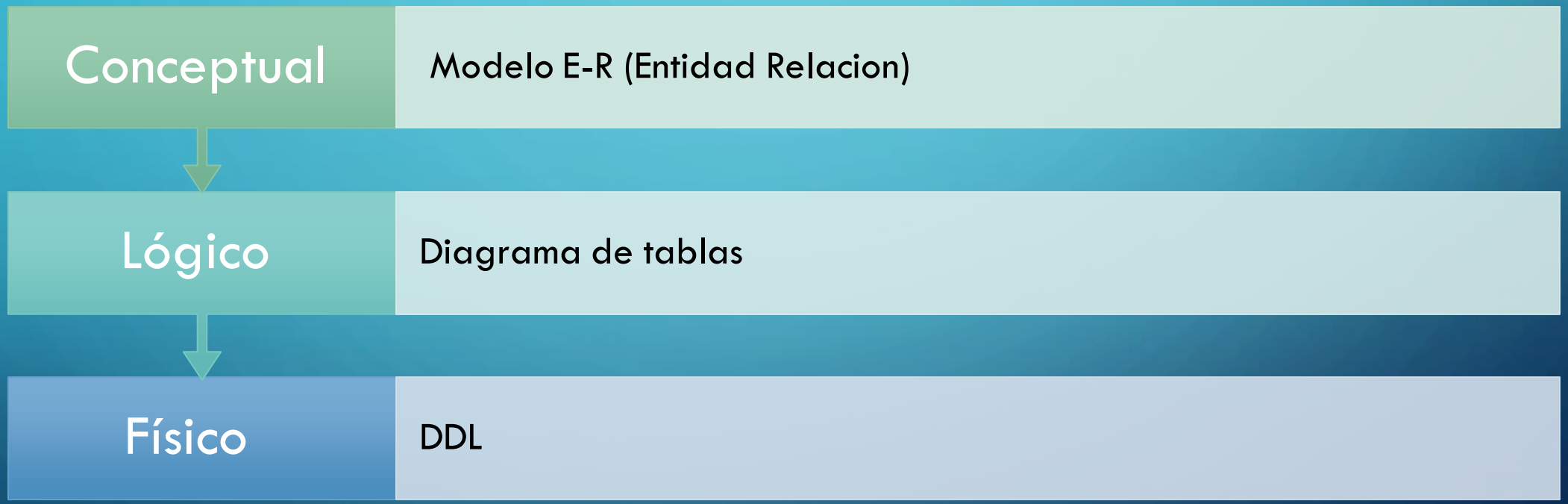


TEMA 3: CREANDO UNA BBDD CON SQL SERVER

BERNAT COSTA

BERNAT.COSTA@CESURFORMACION.COM

3 MODELOS PARA CREAR BBDD



¿QUÉ ES DDL?

Es la parte de SQL que nos crea, modifica y borra las tablas y las relaciones de nuestra BBDD.

Serán esas instrucciones SQL para manipular la estructura de nuestras tablas.

No es un diagrama como el Modelo ER o el Modelo lógico. Se escribe en texto plano.

SQL SERVER

Vamos a usar este SGBD en clase de aquí, a final de curso.

Levantaremos el motor de la BD con Docker como hemos visto y usaremos SQL Server Management Studio para lanzar nuestras instrucciones SQL contra el servidor.

RECORDAD: PARA CONECTARNOS…

- Contenedor Docker de SQL Server arrancado y en verde en docker desktop.
- Si no tengo el contenedor creado:
  - descargo el docker-compose.yml en una carpeta
  - Arranco consola o power shell y navego hasta la carpeta
  - Ejecuto docker-compose up –d
- Desde SSMS
  - Autenticación SQL
  - Servidor: localhost
  - Login: sa
  - Password: "la que tengas definida en el fichero docker-compose.yml"
- Recordad que en clase, debéis ejecutar docker desktop y la consola como Administradores

CREAR LA BBDD

- Antes de empezar a crear tablas, DEBEMOS crear una BBDD.
  - CREATE DATABASE nombreDeLaBBDD
- Una vez creada, deberemos indicarla al cliente SQL que queremos usar esa bbdd. (desplegable en SSMS)
  - USE nombreDeLaBBDD
- Si vamos a la carpeta data de nuestro ordenador vinculada al contenedor docker, veremos que han aparecido 2 ficheros nuevos para esa BBDD.

¿COMO CREAMOS UNA TABLA?

CREATE TABLE *nombre_tabla*

(

nombre_columna1 *tipo_dato* (*size*) *restricciones*,

nombre_columna2 *tipo_dato*(*size*) *restricciones*,

nombre_columna3 *tipo_dato*(*size*) *restricciones*,

....

)

RESTRICCIONES DISPONIBLES

NOT NULL la columna no puede almacenar un valor NULL.

UNIQUE :la columna debe tener un valor único.

PRIMARY KEY : (implica NO NULO y único). Identificador único y clave principal.

FOREIGN KEY : los datos deben coincidir con los valores de integridad referencial en otra tabla.

DEFAULT : valor predeterminado de ese atributo. (por ejemplo, predefinir en una tabla Animales de un veterionario la especie a Perro.)

TIPOS DE DOMINIO DISPONIBLES EN SQL SERVER

https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

Grupo	Tipo de dato	Intervalo	Tamaño
Numéricos exactos	bigint		8 bytes
	int		4 bytes
	smallint		2 bytes
	tinyint	De 0 a 255	1 byte
	bit	Tipo de datos entero que puede aceptar los valores 1, 0 ó NULL	2 bytes
	decimal, numeric, decimal (p, s)	*p* (precisión): el número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima de 38. La precisión predeterminada es 18. *s* (escala): el número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. La escala debe ser un valor comprendido entre 0 y *p*. Sólo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0.	Precisión 1 - 9: 5 bytes
	money	Tipos de datos que representan valores monetarios o de moneda:	8 bytes
	smallmoney		4 bytes
Numéricos aproximados	float (n)		Depende del valor de *n*
	real		4 Bytes
Fecha y hora	**datetime**	Del 1 de enero de 1753 hasta el 31 de diciembre de 9999	
	smalldatetime	Del 1 de enero de 1900 hasta el 6 de junio de 2079	
Cadenas de caracteres	char (n)	Caracteres no Unicode de longitud fija, con una longitud de *n* bytes. *n* debe ser un valor entre 1 y 8.000	n bytes
	varchar (n)	Caracteres no Unicode de longitud variable. *n* indica que el tamaño de almacenamiento máximo es de 2^{31} - 1 bytes	n bytes (aprox.)
	text	En desuso, sustituido por *varchar*.	max bytes (aprox.)
Cadenas de caracteres unicode	nchar (n)	Datos de carácter Unicode de longitud fija, con *n* caracteres. *n* debe estar comprendido entre 1 y 4.000	2 * n bytes
	nvarchar (n)	Datos de carácter Unicode de longitud variable. *n* indica que el tamaño máximo de almacenamiento es 2^{31} - 1 bytes	2 * n bytes + 2 bytes
	ntext (n)	En desuso, sustituido por *nvarchar*.	2 * n bytes
Cadenas binarias	binary (n)	Datos binarios de longitud fija con una longitud de *n* bytes, donde *n* es un valor que oscila entre 1 y 8.000	*n* bytes
	varbinary (n)	Datos binarios de longitud variable. n indica que el tamaño de almacenamiento máximo es de 2^{31} - 1 bytes	n bytes

TIPOS DE CAMPO MÁS USADOS

INT para enteros

BIT para boleanos

VARCHAR(n de caracteres) para TEXTO

- IMPORTANTE NO OLVIDARSE DE INDICAR LA N. Si no ponemos N, por defecto, será un VARCHAR(1)
- Si debemos guardar caracteres no latinos usaremos NVARCHAR (caracteres chinos, rusos, árabes...)

DATETIME para fechas

Decimal(18,2) para números decimales.

- El segundo digito, indica el numero de numeros detrás de la coma. El primer digito, el num de digitos que guardamos a la derecha e izquierda de la coma.
- 9.999.999.999.999.999,99 -> valor máximo de un decimal (18,2)

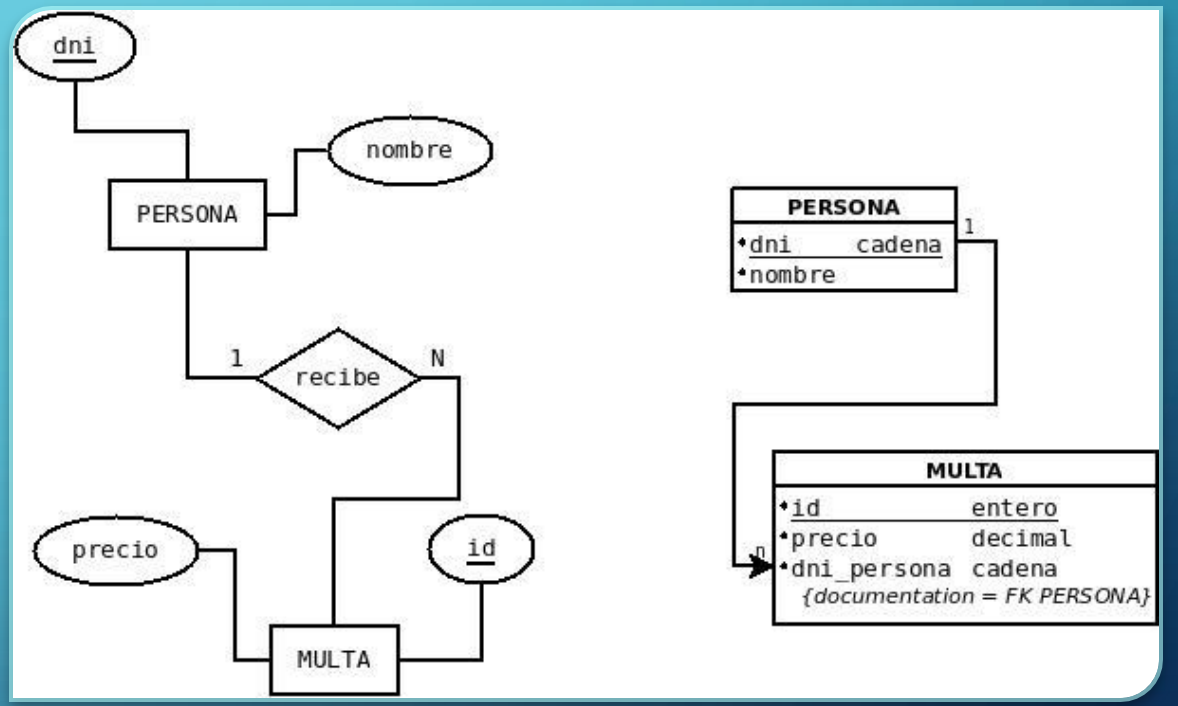
¿CÓMO CREAMOS NUESTRA PRIMERA TABLA?

```
CREATE DATABASE DGT

CREATE TABLE Persona(
        dni VARCHAR (9) PRIMARY KEY,
        nombre NVARCHAR (250) NOT NUL
)

CREATE TABLE Multa(
        id INT  PRIMARY KEY,
        precio DECIMAL(18,2) NOT NULL,
        dni_Persona VARCHAR (9) NOT NULL,

        CONSTRAINT fkDNIPersona FOREIGN KEY
        (DNI_Persona) REFERENCES Persona(DNI)
)
```



PRIMARY KEY DOBLE Y TABLA NN

- Si por ejemplo queremos crear una PK Doble, no podemos definirla al definir el campo y se define al final del create table.

```
USE DGT

CREATE TABLE Coche
(
Matricula VARCHAR(7) PRIMARY KEY,
Es_Gasolina BIT NOT NULL,
Color Varchar(10) NOT NULL,
Cilindrada INT NOT NULL
)

CREATE TABLE PersonaCoche
(
  DNI VARCHAR (9),
  Matricula VARCHAR(7),
  CONSTRAINT fkDNI FOREIGN KEY (DNI) REFERENCES Persona(DNI),
  CONSTRAINT fkCocke FOREIGN KEY (Matricula) REFERENCES Coche(Matricula),
  PRIMARY KEY (DNI, Matricula)
)
```

EJERCICIO PROBEMOS LAS RESTRICCIONES

- En el menú de la izquierda del SSMS, veremos las bbdd y tablas creadas.
  (darle a actualizar si acabamos de crear una tabla o una BBDD).
- Botón derecho sobre una tabla, y podemos indicarle edit table. Nos abrirá la tabla en modo edición y podremos insertar y modificar datos.
- Con la BBDD de la DGT:
  - Intenta introducir dos personas con el mismo DNI
  - Intenta introducir una multa con un DNI de una persona que no exista en la tabla personas
  - Intenta introducir una multa con el precio a null
  - Introduce 3 personas. 1 con 0 multas, otra con dos multas con importes distintos y otra con 1 multa.

BORRAR TABLAS

- **DROP TABLE NombreTabla**
  - CUIDADO CON ESO, borramos la tabla Y TODO SU CONTENIDO.
- **DROP TABLE IF EXIST nombreTabla**
- Para borrar una bbdd entera, podemos usar **DROP DATABASE nombreBBDD**.
  - BORRAMOS ABSOLUTAMENTE TODO LO QUE CONTENGA ESA BBDD.

CAMPOS AUTOINCREMENTALES

- Para los id's es buena idea tener un campo que se autoincremente solo. En SQL Server, se crea de la siguiente manera:
- CREATE TABLE Peliculas(
  Id int IDENTITY(1,1) PRIMARY KEY,
  nombre Varchar(100) NOT NULL
)

MODIFICAR TABLAS

- Comando ALTER TABLE. (Seguimos en DDL, modificamos los campos, añadir un campo, eliminarlo, añadir una fk, modificar el tipo de datos...

ALTER TABLE <nombre_tabla>
[ADD <definicion_columna>]
[DROP COLUMN <nombre_columna>]
[ADD CONSTRAINT <restriccion>]

```
CREATE TABLE multa
(
   Id int IDENTITY(1,1) PRIMARY KEY,
   precio decimal(18,2) not null,
   idCliente int
)

ALTER TABLE multa DROP COLUMN idCliente

ALTER TABLE multa add dni varchar(10)

ALTER TABLE multa ADD CONSTRAINT
fkmultapersona FOREIGN key (dni)
REFERENCES Persona(dni)
```

MODIFICAR UNA TABLA

ALTER SOBRE COLUMNAS

Añadir nueva columna (cuidado, va sin column!)

- **alter table jugador add email varchar (150) not null default 'asdsdaf@asda.es'**
- Acordaros! Si la tabla está rellena, no podemos añadir campo not null sin default

Modificar columna

- **alter table jugador alter column nombre varchar(30)**

Borrar columna

- **alter table jugador drop column apellido**

ALTER SOBRE CONSTRAINTS

Una constraint es tanto una FK como una PK

Borrar Constraint

- **alter table categoria drop constraint pkcategoria**
- **Alter table tarjeta drop constraint fkcategoria**

Añadir constraint

- **alter table categoria add constraint pkcategoria1 primary key (id)**
- **ALTER TABLE Tarjeta ADD CONSTRAINT fkcategoria FOREIGN KEY (idcategoria) REFERENCES Categorias(id)**


That's all Folks!