

# Lenguaje DDL

Para almacenar y tratar la información esquematizada en los modelos E/R y relacional por medio de un Sistema Gestor de Bases de Datos se utiliza el **lenguaje SQL**. SQL son las siglas de Structured Query Language (Lenguaje de Consulta Estructurado).

Es un lenguaje que permite interactuar con los SGBD Relacionales para especificar las operaciones que se desean realizar sobre los datos y su estructura. Es un lenguaje declarativo, lo cual quiere decir que en él se especifica al Sistema qué se quiere obtener pero no cómo conseguirlo.

## Lenguaje DDL

Para la creación de las tablas de la base de datos, se utiliza el lenguaje de descripción de datos (DDL, Data Definition Language).

A continuación se explica la sintaxis de cada sentencia para llegar a crear una base de datos SQL.

Hay que tener en cuenta que en este curso nos centramos en el lenguaje SQL del SGBD MySQL, y cada uno tiene sus particularidades y sus diferencias, pero en líneas generales todos deben seguir el estándar SQL y, por lo tanto, tener todos más o menos la misma sintaxis.

Toda la sintaxis detallada del lenguaje MySQL la podéis encontrar en la web de documentación de MySQL, [aquí](#). A continuación se dan las nociones básicas de las sentencias básicas del lenguaje DDL.

## Creación de tablas

```
CREATE TABLE nombreTabla (  
    columna1 tipoDato,  
    columna2 tipoDato,  
    ...  
    columnaN tipoDato)
```

Para crear la tabla Alumno:



Alumno

<u>DNI</u>
Nombre
Apellidos
Fecha_nacimiento
Email

```
CREATE TABLE Alumno (
    DNI CHAR(9),
    Nombre VARCHAR(20),
    Apellidos VARCHAR(50),
    Fecha_nacimiento DATE,
    Email VARCHAR(50)
);
```

## Claves primarias

Siguiendo con el ejemplo de la tabla Alumno, la clave primaria está formada por un solo campo, por lo que se puede poner directamente en la columna que es la clave primaria la palabra clave **PRIMARY KEY**:

```
CREATE TABLE Alumno (
    DNI CHAR(9) primary key,
    Nombre VARCHAR(20),
    Apellidos VARCHAR(50),
    Fecha_nacimiento DATE,
    Email VARCHAR(50)
);
```

Si la clave primaria está formada por varias columnas, se debe poner al final, después de la declaración de todas las columnas:

Matricula

<u>Codigo</u> (FK)
<u>DNI</u> (FK)
Nota

```
CREATE TABLE Matricula (
    Nota FLOAT,
    Codigo CHAR(5),
    DNI CHAR(9),
```

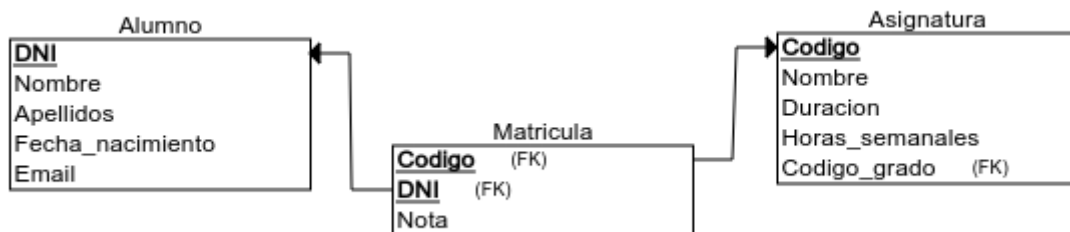
PRIMARY KEY (Codigo, DNI)

);

## Claves ajenas

La tabla Matricula tiene dos claves ajenas, en la columna Codigo y en la columna DNI.

El diagrama de las tablas Alumno, Matricula y Asignatura:



Se crea con las sentencias SQL siguientes:

```
CREATE TABLE Alumno (  
    DNI CHAR(9),  
    Nombre VARCHAR(20),  
    Apellidos VARCHAR(50),  
    Fecha_nacimiento DATE,  
    Email VARCHAR(50),  
    PRIMARY KEY (DNI)  
);  
  
CREATE TABLE Asignatura (  
    Codigo CHAR(5),  
    Nombre VARCHAR(30),  
    Duracion VARCHAR(9),  
    Horas_semanales INT,  
    Codigo_grado CHAR(5),  
    PRIMARY KEY (Codigo)  
);  
  
CREATE TABLE Matricula (  
    Nota FLOAT,
```

```
Codigo CHAR(5),
DNI CHAR(9),
PRIMARY KEY (Codigo, DNI),
FOREIGN KEY (Codigo) REFERENCES Asignatura(Codigo),
FOREIGN KEY (DNI) REFERENCES Alumno(DNI)
);
```

La clave ajena se indica después de todas las columnas, se pone FOREIGN KEY, a continuación entre paréntesis el nombre de la columna o columnas que forman parte de la clave ajena, luego la palabra REFERENCES y el nombre de la tabla con el de la columna a la que hace referencia como clave primaria, entre paréntesis.

## Algunas restricciones de integridad

- **Prohibir nulos.** Esto obliga a que la columna tenga que tener obligatoriamente un valor para que sea almacenado el registro. Para esto, basta con añadir en la columna que no se permite dejar vacía en ninguna fila, después del tipo de dato, las palabras claves NOT NULL.

```
CREATE TABLE Alumno(
  DNI CHAR(9) NOT NULL,
  Nombre VARCHAR(20) NOT NULL,
  Apellidos VARCHAR(50) NOT NULL,
  Fecha_nacimiento DATE NOT NULL,
  Email VARCHAR(50) NOT NULL,
  PRIMARY KEY (DNI)
);
```

- **Valores no repetidos.** Esto obliga a que el contenido de una o más columnas no puedan repetir valores en la tabla. Basta con añadir al final de la columna la palabra clave UNIQUE.

Si no permitimos que dos grados tengan el mismo nombre:

```
CREATE TABLE Grado(
  Codigo CHAR(5),
  Nombre VARCHAR(50) UNIQUE,
  PRIMARY KEY (Codigo)
);
```



Puede ser que los valores que no se permiten repetir sea la combinación de varios campos. En ese caso se debe poner al final de la declaración de todas las columnas, igual que se indicaba la clave primaria.

## Eliminación de tablas

La orden **DROP TABLE** seguida del nombre de una tabla, permite eliminar la tabla en cuestión.

```
DROP TABLE Grado;
```

Normalmente, el borrado de una tabla es irreversible y no hay ninguna petición de confirmación, por lo que conviene ser muy cuidadoso con esta operación.

## Modificación de tablas

- Cambiar el nombre a una tabla.

```
ALTER TABLE nombre_viejo RENAME nombre_nuevo;
```

- Añadir columnas.

```
ALTER TABLE Alumno ADD (Direccion VARCHAR(100));
```

- Eliminar columnas.

```
ALTER TABLE Alumno DROP (Direccion);
```

- Modificar una columna. Basta con volver a definirla mediante la siguiente sintaxis:

```
ALTER TABLE Alumno Modify (Email VARCHAR(100));
```

Se ha cambiado el tipo de dato de la columna email y además ahora se permite dejar en blanco.

Se pueden también añadir y eliminar restricciones, pero esto y más se puede ver en la documentación oficial de SQL.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU



Financiado por  
la Unión Europea  
NextGenerationEU



Plan de Recuperación,  
Transformación  
y Resiliencia



GOBIERNO DE ESPAÑA  
MINISTERIO DE EDUCACIÓN  
Y FORMACIÓN PROFESIONAL



GOBIERNO  
DE ARAGON

Revision #6

Created 25 August 2022 20:47:17 by Berta

Updated 17 January 2023 15:46:55 by Equipo CATEDU