

Bootcamp Java Developer

Fase 1 - Java Analyst
Módulo 2



Conceptos preliminares

Tablas

- Son el **objeto principal de una base de datos**, ya que en ellas se **almacenarán** los datos.
- Son objetos compuestos por una estructura que almacena datos relacionados acerca de algún **objeto/entidad** en general.
- Las tablas tienen un **nombre** y debe ser único en toda la base datos.
- Están compuestas por **registros (filas)** y **campos (columnas)**.
- Los registros y campos pueden estar en **diferentes órdenes**.
- Una base de datos puede contener **varias tablas**.

Código	Nombre	Precio	Stock
1	iPod	299	200
2	iPhone	399	300
3	iPad	499	250
4	MacBook Pro	1199	150

| Ejemplo de tabla de 4 campos y 4 registros.

Tipos de datos

Al diseñar tablas dentro de una base de datos, se debe **especificar el tipo de datos y la capacidad que podrá almacenar cada campo**.

Una correcta elección debe procurar que la tabla esté correctamente dimensionada en su capacidad, que destine un tamaño apropiado a la longitud de los datos y que apunte a lograr la máxima velocidad de ejecución de consultas posible.

MySQL admite dos tipos de datos: **números** y **cadenas de caracteres**. Además de estos dos grandes grupos, admite otros tipos de **datos especiales**, como formatos de fecha, lógicos, etc. *(No utilizaremos todos los tipos de datos en este curso).*



Caracteres o cadenas de texto

Las cadenas de texto se utilizan para almacenar una serie de **caracteres, palabras y/o frases de texto** en donde cada carácter es lo mismo que un *byte*.

CHAR	Este tipo se utiliza para almacenar cadenas de longitud fija . Su longitud abarca desde 1 a 255 caracteres . Un campo CHAR ocupará siempre el máximo de longitud que le hayamos asignado, aunque el tamaño del dato sea menor.
VARCHAR	Al igual que el anterior se utiliza para almacenar cadenas, en el mismo rango de 1-255 caracteres , pero en este caso, de longitud variable . VARCHAR solo almacena la longitud del dato, permitiendo que el tamaño de la base de datos sea menor.
TINYTEXT	Texto de longitud variable que puede tener hasta 255 caracteres .

Caracteres o cadenas de texto (continuación):

TEXT	Texto de longitud variable que puede tener hasta 65.535 caracteres .
MEDIUMTEXT	Texto de longitud variable que puede tener hasta 16.777.215 caracteres .
LONGTEXT	Texto de longitud variable que puede tener hasta 4.294.967.295 caracteres .
BLOB	Dato binario que puede almacenar archivos o texto . En este caso, los tipos TINYBLOB, BLOB, MEDIUMBLOB y LONGBLOB son idénticos a sus homólogos TEXT, con la diferencia de que las búsquedas en un tipo BLOB tienen en cuenta las mayúsculas y minúsculas .

Datos numéricos

En este tipo de campos **pueden almacenarse sólo números**, positivos o negativos, enteros o decimales, en notación hexadecimal, científica o decimal.



- Los tipos numéricos tipo **INTEGER** admiten los atributos **SIGNED** y **UNSIGNED**. En el primer caso, indica que pueden tener **valor negativo** y en el segundo, sólo **positivo**.
- Los tipos numéricos pueden además usar el atributo **ZEROFILL**, en cuyo caso los números se completarán **hasta la máxima capacidad disponible con ceros**.

Ejemplo de atributo ZEROFILL: *columna INT(5)*
zerofill → **valor 23**. Se almacenará como **00023**.

Números:

BIT o BOOL	Para un número entero que puede ser 0 ó 1 .
TINYINT	Es un número entero con rango de valores válidos desde -128 a 127 . Si se configura como <i>unsigned</i> (sin signo), el rango de valores es de 0 a 255 .
SMALLINT	Para números enteros , con rango desde -32.768 a 32.767 . Si se configura como <i>unsigned</i> , 0 a 65.535 .
MEDIUMINT	Para números enteros . El rango de valores va desde -8.388.608 a 8.388.607 . Si se configura como <i>unsigned</i> , 0 a 16.777.215 .
INT	Para almacenar números enteros , en un rango de -2.147.463.848 a 2.147.483.647 . Si configuramos este dato como <i>unsigned</i> , el rango es 0 a 4.294.967.295 .

Números (continuación):

BIGINT	Número entero con rango de valores desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 . Unsigned , desde 0 a 18.446.744.073.709.551.615 .
FLOAT (m,d)	Representa números decimales . Podemos especificar cuántos dígitos (m) pueden utilizarse (término también conocido como <i>ancho de pantalla</i>), y cuántos en la parte decimal (d) . MySQL redondeará el decimal para ajustarse a la capacidad.
DOUBLE	Número de coma flotante de precisión doble . Es un tipo de datos igual al anterior cuya única diferencia es el rango numérico que abarca.
DECIMAL	Almacena los números como cadenas de texto .

Fecha

DATE	Para almacenar fechas . El rango de valores va desde 0000-00-00 a 9999-12-31 . El formato por defecto es YYYY-MM-DD .
DATETIME	Combinación de fecha y hora . El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos . El formato de almacenamiento es de año-mes-día horas:minutos:segundos .
TIME	Almacena una hora . El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos . El formato de almacenamiento es de 'HH:MM:SS'
YEAR	Almacena un año . El rango de valores permitidos va desde el año 1901 al año 2155 . El campo puede tener tamaño dos o tamaño 4 , depende de si queremos almacenar el año con dos, o cuatro dígitos.

Creación de una base de datos

Para crear una base de datos (por ejemplo, con el nombre *ComercioIT*), la **instrucción SQL** a ejecutar será:

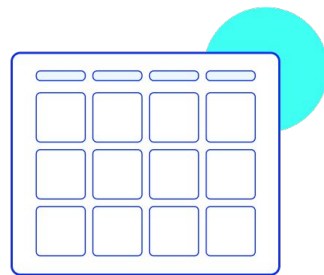
```
CREATE DATABASE ComercioIT;
```



Creación de una tabla

La sentencia **CREATE TABLE** creará una tabla con las **columnas (campos)** que indiquemos.

En el ejemplo de la [diapositiva siguiente](#), podemos ver una instrucción SQL para generar una tabla, con el nombre **Productos**. Dentro de la misma, se definen **7 campos** con sus **tipos de datos y modificadores** correspondientes.



Ejemplo:

```
CREATE TABLE Productos(  
  idProducto INT(11) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  Nombre VARCHAR(50) NOT NULL,  
  Precio DOUBLE,  
  Marca VARCHAR(30) NOT NULL,  
  Categoria VARCHAR(30) NOT NULL,  
  Stock INT(6) NOT NULL,  
  Disponible BOOLEAN DEFAULT false  
);
```



Eliminar una tabla

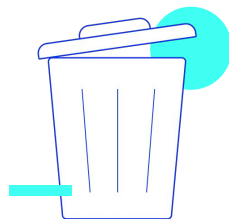
Para eliminar una tabla (por ejemplo, con el nombre *Productos*) de una base de datos, la **instrucción SQL** a ejecutar será:

```
DROP TABLE Productos;
```

o


```
DROP TABLE IF EXISTS Productos;
```

Nota: la cláusula ***IF EXISTS*** devuelve una advertencia en caso de que no exista la tabla a eliminar.




Restricciones de las tablas

Puntos claves

1. Los nombres de las tablas deben ser únicos en la base de datos.
 2. Los nombres de las columnas debe ser únicos en la tabla.
 3. No podrán existir dos registros con el mismo valor de la clave primaria.
- 

Columnas no descomponibles

1. Son aquellas columnas que contienen cierta información que no puede almacenarse en dos o más columnas.
 2. Son fáciles de actualizar.
 3. Son fáciles de consultar.
 4. Mejoran la implementación de la integridad de los datos.
- 

Modificadores aplicables a los campos de una tabla

- **NOT NULL:** no permite valores nulos; la carga del dato en ese campo será obligatoria. Es decir, el **campo no puede quedar vacío**.
- **DEFAULT:** permite especificar un **valor predeterminado**.
- **CLAVE PRIMARIA (PRIMARY KEY):** una tabla suele tener una columna, o una combinación de columnas, cuyos **valores identifican de forma única a cada registro de la tabla**.

Estas columnas se denominan **claves principales** de la tabla y exigen la integridad de entidad de la tabla (**un solo registro con ese valor de indicador único**). Cuando cree o modifique una tabla, puede crear una clave primaria mediante la definición de una restricción PRIMARY KEY.



Una tabla **sólo puede tener una restricción PRIMARY KEY** y **ninguna columna a la que se aplique una restricción PRIMARY KEY puede aceptar valores NULL**.

Ya que las restricciones PRIMARY KEY garantizan datos únicos, muchas veces se definen en una **columna de identidad**.

Cuando se especifica una restricción del tipo PRIMARY KEY en una tabla, el **motor de base de datos exige la unicidad de los datos** mediante la creación de un **índice único** para las columnas de la clave principal.

Este índice también permite un acceso rápido a los datos cuando se utiliza la clave principal en las consultas. De esta forma, las claves principales que se eligen deben seguir las reglas para crear índices únicos.

Si se define una restricción PRIMARY KEY para más de una columna, puede haber valores duplicados dentro de la misma columna, pero cada combinación de valores de todas las columnas de la definición de la restricción PRIMARY KEY debe ser única.



- **NOT NULL:** indica que una **columna debe ser de carga obligatoria**. NULL constituye un valor en sí (valor desconocido). No es vacío en un campo de tipo texto, ni 0 en un campo numérico.
- **UNIQUE:** indica que **la columna no tendrá ningún valor repetido**. Similar a PRIMARY KEY, pero a diferencia de esta última, UNIQUE **permite un valor nulo, y puede haber varias columnas de este tipo** en una tabla.
- **BINARY:** indica que los valores en esta columna se almacenarán en **modo binario**, respetando **mayúsculas y minúsculas**.
- **UNSIGNED:** indica que esta columna no usará un byte para el signo; es decir, permitirá almacenar **números positivos únicamente**.
- **ZERO FILL:** indica que el contenido del campo se completará con **ceros** (siempre que sea numérico).
- **AUTO_INCREMENT:** el **motor de base de datos incrementará automáticamente su valor**. Una tabla sólo puede tener un campo autoincremental y éste tiene que formar parte de la PK (PRIMARY KEY).



Anexo

Otros comandos *MySQL*

A continuación, se detallan comandos necesarios para la navegación dentro del motor *MySQL*. La sintaxis de cada uno se muestra a la derecha de su definición.

- **Comando SHOW DATABASES:** muestra el catálogo de base de datos del servidor.
- **Comando USE:** pone en uso una base de datos del servidor. Todos los comandos SQL que se ejecuten, se llevarán a cabo sobre la base de datos en uso.

```
SHOW DATABASES;
```

```
USE NombredeBaseDeDatos;
```

- **Comando SHOW TABLES:** muestra el **catálogo o listado de tablas** de la base de datos activa.
- **Comentarios:** permiten escribir **texto que no será interpretado como parte de sentencias SQL**, útiles para documentar y comentar acciones realizadas por las sentencias. Se pueden utilizar las siguientes formas de escribir comentarios:

```
SHOW TABLES;
```

```
# Esto es un comentario de 1 línea  
(Propia de MySQL)
```

```
/* Esto es un comentario de 1 o más  
líneas */  
(Soportada en MySQL y otros motores)
```

```
-- Esto es un comentario de 1 línea  
(Soportada en MySQL y otros motores)
```

- **Comando DESCRIBE:** devuelve la **descripción de campos y detalles de una tabla** (estructura física).

```
DESCRIBE NombreDeTabla;
```

- **Comando SHOW CHARSET:** muestra los CHARSET (juegos de caracteres).

```
SHOW CHARSET;
```

- **Comando SHOW COLLATION:** Muestra los COLLATIONS instalados.

```
SHOW COLLATION;
```

Conceptos avanzados

ENUM

Existe un tipo de dato denominado ENUM, que presenta las siguientes características:

- Sólo puede contener **un valor**.
- Se puede definir una lista de **hasta 65.535 valores** distintos.
- Si se permiten valores **null**, éste será el valor **predeterminado**; si no se define un valor predeterminado con DEFAULT, será el primer valor de la lista.
- Cada valor de la lista es **numerado con un índice** (comenzando en 1).

Ejemplo:

```
CREATE TABLE Medida (medida ENUM('pequeño', 'mediano', 'grande') NOT NULL DEFAULT 'mediano');
```

SET

Otro tipo de dato es SET, el cual posee las siguientes características:

- **0, 1 ó varios valores.**
- Se puede definir una lista de **hasta 64 valores** distintos.
- Los valores **no pueden contener comas**, ya que los valores asignados en la lista están separados por ese carácter.
- **Cada valor** de la lista **representa un *bit*** de la cadena de bits del campo.
- El **valor decimal del campo determina los *bits***, al marcar los valores que contiene el campo. De manera que, si todos los *bits* están a 1, significa que ese campo contiene todos los valores.

Por ej.: Si el valor decimal es 7, en binario sería *0111* y eso quiere decir que el campo contiene los valores '*a*', '*b*' y '*c*'.

Tipo de dato SET - Ejemplo:

SET	Decimal	Bytes
'a'	1	0001
'b'	2	0010
'c'	4	0100
'd'	8	1000

```
CREATE TABLE Letra (letra SET('a', 'b', 'c', 'd'));
```



**¡Sigamos
trabajando!**