Introducción a
Bases de Datos

Módulo 1

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 solo 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 (columnas) y 4 registros

Tipos de datos

Al diseñar nuestras tablas tenemos que 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 apunte a conseguir la máxima velocidad de ejecución de consultas posible.

Básicamente MySQL admite dos tipos de datos: números y cadenas de caracteres. Junto a estos dos grandes grupos, se admiten otros tipos de datos especiales: formatos de fecha, lógicos, etc.

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
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 sólo pueden almacenarse números, positivos o negativos, enteros o decimales, en notación hexadecimal, científica o decimal.

Los tipos numéricos tipo ***integer*** admiten los atributos **SIGNED** y **UNSIGNED** indicando en el primer caso que pueden tener valor negativo, y sólo positivos en el segundo.

Los tipos numéricos pueden además usar el atributo **ZEROFILL** en cuyo caso los números se completarán hasta la máxima capacidad disponible con ceros (columna INT(5) zerofill => valor 23 se almacenará como 00023).

Datos numéricos

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 unsigned (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 unsigned, 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 unsigned, 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 unsigned, el rango es 0 a 4.294.967.295
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 ancho de pantalla), 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

Tipos de datos

Datos Fecha

DATE	Para almacenar fechas. El formato por defecto es YYYY MM DD desde 0000 00 00 a 9999 12 31
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 dependiendo de si queremos almacenar el año con dos o cuatro dígitos.

Nota: *Existen otros tipos de datos que no estamos utilizando en este curso.*

Creación de una base de datos

Sintaxis

```
CREATE DATABASE ComercioIT;
```

Creación de una tabla

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

Sintaxis

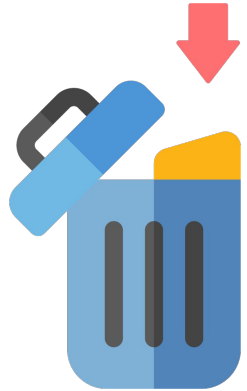
```
CREATE TABLE Productos(
idProducto INT(11) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(50) NOT NULL,
Precio DOUBLE,
Marca VARCHAR(30) NOT NULL,
Categoria VARCHAR(30) NOT NULL,
Stock INT(6) NOT NULL,
Disponible BOOLEAN DEFAULT false
);
```

Eliminar una tabla

Sintaxis

```
DROP TABLE Productos;
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.

Restricciones de las tablas

Restricciones en las columnas

1. NOT NULL (no permite valores nulos, el campo será de ingreso obligatorio).
2. 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). Puede crear una clave principal mediante la definición de una restricción **PRIMARY KEY** cuando cree o modifique una tabla.

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

Restricciones de las tablas

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.

Otros atributos de las columnas

- **Not Null:** indica que una columna no puede ser NULL. 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 ésta última, 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, ó sea que permitirá almacenar números positivos solamente.
- **Zero Fill:** indica que el contenido del campo se completará con ceros si es 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.

Anexo

Otros comandos MySQL

En las siguientes líneas se detallan comandos necesarios para la navegación dentro del motor MySQL.

- **Comando SHOW DATABASES**
  Muestra el catálogo de base de datos del servidor.

  Sintaxis

```
SHOW DATABASES;
```

- **Comando USE**
  Muestra el catálogo de base de datos del servidor.

  Sintaxis

```
USE NombredeBaseDeDatos;
```

Otros comandos MySQL

- **Comando SHOW TABLES**
  Muestra el catálogo o listado de tablas de la base de datos activa.

  Sintaxis

```
SHOW TABLES;
```

- **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:

  Sintaxis

```
## 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)
```

Otros comandos MySQL

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

  Sintaxis

```
DESCRIBE NombreDeTabla;
```

- **Comando SHOW CHARSET**
  Muestra los CHARSET (juegos de caracteres) instalados.

  Sintaxis

```
SHOW CHARSET;
```

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

  Sintaxis

```
SHOW COLLATION;
```

Conceptos avanzados

- Se puede crear una tabla a partir de una sentencia SELECT. De este modo podemos duplicar la estructura completa de una tabla o sólo parte de ella.

  Sintaxis

```
CREATE TABLE TablaNueva
SELECT (Columna1, ..., ColumnaX) FROM TablaExistente WHERE 1 = 2;
```

- También podría crearse una tabla y copiar datos en ella a partir de una sentencia SELECT. De este modo podemos duplicar la estructura completa de una tabla o sólo parte de ella, con registros incluidos.

  Sintaxis

```
CREATE TABLE TablaNueva
SELECT (Columna1, ..., ColumnaX) FROM TablaExistente;
```

Conceptos avanzados

En los casos mencionados previamente, se replica el tipo de datos, pero no la Primary Key.

- Existe un **tipo de dato ENUM**, y tiene estas 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, y si no se define un valor predeterminado con DEFAULT, será el primer valor de la lista.
  - Cada valor de la lista es enumerado con un índice (comenzando en 1).

Ejemplo

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

Conceptos avanzados

- Otro tipo de dato, **SET**, tiene estas 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, marcando los valores que contiene el campo, de manera que si todos los bits están a 1, es 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’).

Conceptos avanzados

- 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’));
```

¡Muchas gracias!

¡Sigamos trabajando!