MySQL: El Almacén y la Despensa de la Web

1. ¿Qué es MySQL? El Archivista Organizado 🧑‍💼

MySQL es un **Sistema Gestor de Bases de Datos Relacionales (RDBMS)**. No es un lenguaje de programación como PHP; es un software que se encarga de almacenar, gestionar y recuperar grandes cantidades de información de forma estructurada.

Piensa en MySQL como el **Archivista Experto** que trabaja en una enorme bodega de archivos (la base de datos).

- **HTML, CSS, PHP:** El restaurante (estructura, diseño y cocina).
- **MySQL:** La bodega donde se guardan las recetas, los inventarios, la lista de clientes y todos los pedidos históricos, todo perfectamente organizado en **tablas** (archiveros).

SQL: El Lenguaje para Hablar con el Archivista

Para interactuar con MySQL, usamos **SQL** (Structured Query Language - Lenguaje de Consulta Estructurada). SQL es el único idioma que entiende el Archivista. Todas las acciones (guardar, buscar, modificar, borrar) se hacen mediante consultas SQL.

Usos Comunes de MySQL en Desarrollo Web:

- **Gestión de Usuarios:** Almacenar nombres, contraseñas (encriptadas) y roles.
- **Inventarios y Catálogos:** Guardar listas de productos, precios y descripciones.
- **Contenido Dinámico:** Almacenar artículos de un blog, comentarios o mensajes.

---

2. Estructura Básica: Bases de Datos, Tablas y Campos

La información en MySQL no se guarda al azar; se organiza jerárquicamente.

2.1. Base de Datos (Database - El Archivador Principal)

Es el contenedor más grande. Una Base de Datos agrupa lógicamente toda la información de un proyecto (ej. `mi_enciclopedia_web`).

2.2. Tablas (Tables - Los Archiveros)

Dentro de la Base de Datos, la información se divide en Tablas. Cada Tabla almacena datos de un **único tipo** de entidad (ej. `usuarios`, `articulos`, `comentarios`).

2.3. Campos (Columns - Los Cajones del Archivero)

Cada Tabla tiene **Campos** (o Columnas), que definen el tipo de dato que se guardará. Por ejemplo, la tabla `usuarios` puede tener los campos `id`, `nombre`, `email` y `fecha_registro`.

2.4. Registros (Rows - Los Documentos)

Cada fila en una Tabla es un **Registro** o tupla, que representa una instancia única de esa entidad (ej. un usuario específico o un artículo).

---

3. Tipos de Datos Esenciales: Definiendo la Información

Al crear una Tabla, debemos ser muy estrictos con el tipo de dato que acepta cada Campo. Esto asegura la integridad y optimiza el almacenamiento.

Categoría	Tipo de Dato	Propósito	Ejemplo de Valor
Numéricos	`INT`	Números enteros (edades, IDs, cantidades).	10, 4500
	`FLOAT` / `DOUBLE`	Números con decimales (precios, coordenadas).	19.99, 3.1416
Texto	`VARCHAR(N)`	Cadenas de texto de longitud variable, donde `N` es el máximo de caracteres (nombres, títulos).	'Pedro', 'Título'
	`TEXT`	Cadenas de texto largas (descripciones, cuerpos de artículos).	*El texto completo de este artículo.*
Fecha/ Hora	`DATE`	Almacena solo la fecha (año, mes, día).	'2025-11-07'
	`DATETIME`	Almacena fecha y hora (para registros de actividad).	'2025-11-07 15:40:00'

4. Claves: La Identidad y la Relación 🔑

Las claves son los atributos que definen la estructura y las relaciones dentro del modelo de datos.

4.1. Clave Primaria (`PRIMARY KEY`) - La Cédula de Identidad

Es un campo (o conjunto de campos) que identifica de **forma única** cada Registro en la Tabla. No puede tener valores duplicados ni nulos.

> **Regla:** Cada tabla debe tener una única Clave Primaria. Es la forma más rápida de encontrar cualquier registro.

4.2. Clave Foránea (`FOREIGN KEY`) - El Conector de Relaciones

Es un campo en una Tabla que referencia (apunta) a la Clave Primaria de **otra Tabla**. Es el mecanismo que crea las **Relaciones** entre los datos.

> **Ejemplo:** En la tabla `comentarios`, el campo `id_articulo` sería una Clave Foránea que apunta al `id` de la tabla `articulos`. Esto significa que el comentario "pertenece a" ese artículo.

---

5. El Lenguaje de Definición de Datos (DDL): Construyendo Tablas

El **DDL (Data Definition Language)** es la parte de SQL que se usa para crear, modificar y eliminar la estructura de la base de datos (las Tablas y Campos).

5.1. `CREATE DATABASE` y `USE`

Para empezar, creamos la Base de Datos y la seleccionamos.

```
-- 1. Crear la Base de Datos (si no existe)
CREATE DATABASE IF NOT EXISTS enciclopedia_dev;

-- 2. Seleccionar la Base de Datos para trabajar en ella
USE enciclopedia_dev;
```

5.2. `CREATE TABLE` (Construyendo el Archivero)

Aquí definimos el nombre de la Tabla, los Campos que tendrá y sus restricciones (como la Clave Primaria).

```
-- Ejemplo Vistoso: Creación de la tabla de Artículos
CREATE TABLE articulos (
    -- ID autoincremental, único y clave principal
    id INT(11) NOT NULL AUTO_INCREMENT,

    -- Título del artículo, máximo 200 caracteres, y no puede ser nulo
    titulo VARCHAR(200) NOT NULL,

    -- Cuerpo del texto (usamos TEXT para longitud)
    contenido TEXT NOT NULL,

    -- Clave Foránea que conecta con la tabla 'usuarios' (autor del artículo)
    autor_id INT(11) NOT NULL,

    -- Timestamp para guardar la fecha y hora de creación automáticamente
    fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP,

    -- Definición de las Claves
    PRIMARY KEY (id),

    -- Definición de la Clave Foránea que enlaza al ID de la tabla 'usuarios'
    FOREIGN KEY (autor_id) REFERENCES usuarios(id)
);
```

6. El Lenguaje de Manipulación de Datos (DML): El CRUD

El **DML (Data Manipulation Language)** es la parte de SQL que se usa para manejar la información real dentro de las tablas.

CRUD son las cuatro operaciones básicas: **C**rear, **R**ecuperar, **U**pdate (Actualizar) y **D**elete (Borrar).

6.1. `INSERT` (Crear - Guardar un Documento)

Añade nuevos registros (filas) a una tabla.

SQL

```
INSERT INTO articulos (titulo, contenido, autor_id)
VALUES ('Fundamentos de MySQL', 'Contenido completo del artículo...', 1);
```

6.2. `SELECT` (Recuperar - Buscar Documentos)

La consulta más usada. Permite obtener datos de una o más tablas.

SELECT titulo, fecha_creacion FROM articulos;

6.3. `UPDATE` (Actualizar - Modificar un Documento)

Modifica los datos de los registros que cumplen una condición. **¡Siempre usar `WHERE`!**

```
-- Actualiza el título del artículo que tenga ID = 5
UPDATE articulos
SET titulo = 'El nuevo título editado'
WHERE id = 5;
```

6.4. `DELETE` (Borrar - Destruir un Documento)

Elimina registros. **¡Siempre usar `WHERE`!** Sin `WHERE`, ¡borrarás toda la tabla!

```
-- Borra el registro del artículo que tenga ID = 10
DELETE FROM articulos
WHERE id = 10;
```

7. La Cláusula `WHERE`: Filtrando la Búsqueda 🔎

La cláusula `WHERE` es esencial en `SELECT`, `UPDATE` y `DELETE`. Actúa como un **filtro**, especificando la condición que los registros deben cumplir para ser seleccionados o modificados.

7.1. Operadores de Comparación

Usados para establecer las condiciones de filtrado (igualdad, mayor que, etc.).

Operador	Significado	Ejemplo
`=`	Igual a	`WHERE id = 5`
`>` o `<`	Mayor o menor que	`WHERE visitas > 1000`
`!=` o `<>`	Diferente de	`WHERE autor_id != 1`
`LIKE`	Búsqueda de patrones (uso con `%`)	`WHERE titulo LIKE 'CSS%'` (Empieza por CSS)

7.2. Operadores Lógicos (`AND`, `OR`)

Permiten combinar múltiples condiciones de filtrado.

```
-- Ejemplo Vistoso: Buscar artículos publicados después de 2024 y con menos de 50 comentarios.
SELECT titulo, fecha_creacion, comentarios
FROM articulos
WHERE
   fecha_creacion > '2024-01-01' -- Condición 1: Mayor a la fecha
   AND comentarios < 50;        -- Condición 2: Y (AND) menor a 50
```

8. Ordenamiento y Agrupación: Organizando el Catálogo

8.1. `ORDER BY` (Ordenando la Lista)

Esta cláusula se utiliza en la consulta `SELECT` para ordenar el conjunto de resultados por una o más columnas.

Dirección	Significado
`ASC`	Ascendente (A-Z, 1-10) - **Es el valor por defecto.**
`DESC`	Descendente (Z-A, 10-1)

```
-- Listar todos los artículos, ordenados primero por la fecha más reciente (DESC)
-- y si dos tienen la misma fecha, ordenarlos por título alfabéticamente (ASC).
SELECT titulo, fecha_creacion
FROM articulos
ORDER BY fecha_creacion DESC, titulo ASC;
```

8.2. `GROUP BY` (Agrupando por Categorías)

Agrupa filas que tienen los mismos valores en una o más columnas en un solo resumen. Se utiliza casi siempre con **Funciones de Agregación** (como `COUNT`, `SUM`, `AVG`).

```
-- Contar cuántos artículos ha escrito cada autor
SELECT autor_id, COUNT(id) AS total_articulos
FROM articulos
GROUP BY autor_id;
-- Resultado: Muestra el ID del autor y la suma de sus artículos.
```

9. Funciones de Agregación: Haciendo Resúmenes y Cálculos

Las funciones de agregación operan sobre un conjunto de filas y devuelven un único valor de resumen.

Función	Propósito
`COUNT()`	Cuenta el número de filas o valores no nulos.
`SUM()`	Calcula la suma total de valores en una columna.
`AVG()`	Calcula el promedio (media) de valores.
`MAX()`	Encuentra el valor máximo.
`MIN()`	Encuentra el valor mínimo.

```
-- Ejemplo Vistoso: Encontrar el artículo con más visitas y el total de artículos.
SELECT
   COUNT(id) AS total_articulos,   -- Cuenta el número total de artículos
   MAX(visitas) AS maxima_visita   -- Encuentra la visita máxima
FROM articulos;
```

10. `JOIN`s: Conectando Archiveros (Tablas Relacionadas) 🤝

Los **JOIN** son esenciales porque permiten combinar filas de dos o más tablas basándose en un campo relacionado (normalmente una Clave Foránea). Es la forma de obtener datos que están separados.

10.1. `INNER JOIN` (La Intersección Exacta)

Devuelve solo los registros que tienen coincidencias **en ambas tablas**. Es el más común.

10.2. `LEFT JOIN` (Mantener el Lado Izquierdo)

Devuelve todos los registros de la Tabla de la izquierda, y los registros coincidentes de la Tabla de la derecha. Si no hay coincidencia, los campos de la derecha serán `NULL`.

Ejemplo Vistoso: Saber el Nombre del Autor (Conectando Tablas)

```
-- Quiero el título del artículo y el nombre del autor, que están en tablas separadas.
SELECT
   A.titulo AS Titulo_Articulo,
   U.nombre AS Nombre_Autor       -- Alias (A y U) para simplificar la escritura.
FROM
   articulos AS A                 -- Tabla Izquierda
INNER JOIN
   usuarios AS U ON A.autor_id = U.id; -- Conexión basada en la Clave Foránea

-- Resultado: Una lista con el título del artículo y el nombre completo del autor.
```

11. Optimización y Seguridad Inicial

11.1. Índices (`INDEX`) - El Cajón de Búsqueda Rápida

Un **Índice** es una estructura de datos que mejora la velocidad de las operaciones de recuperación de datos. Piensa en el índice de un libro: te permite saltar directamente a la página en lugar de leer todo el libro.

- Las Claves Primarias y Foráneas ya tienen índices implícitos.
- **Recomendación:** Crea índices en cualquier columna que uses frecuentemente en tu cláusula `WHERE` o `ORDER BY`.

```
-- Crear un índice en la columna 'titulo' para acelerar las búsquedas por título
CREATE INDEX idx_titulo ON articulos (titulo);
```

11.2. Inyección SQL (La Peligrosa Brecha de Seguridad) 🛡

La **Inyección SQL** es el ataque más común a las bases de datos. Ocurre cuando un atacante introduce código SQL malicioso a través de un campo de entrada (como un formulario). Si PHP no valida y limpia la entrada, el atacante puede ejecutar consultas no autorizadas (como borrar una tabla).

Solución (PHP): Para prevenir esto, usa sentencias preparadas (Prepared Statements) con la librería **MySQLi** o **PDO**. Estas sentencias separan el comando SQL de los datos que ingresa el usuario, asegurando que los datos siempre sean tratados como datos y nunca como código ejecutable. (Visto en PHP punto 10).