

A graphic on the left side of the slide. It features a 3D effect with four stacked rectangular blocks in purple, orange, yellow, and blue. The text 'Agencia de Aprendizaje a lo largo de la vida' is written across these blocks in white. An orange arrow points to the right from the orange block.

Agencia de  
Aprendizaje  
a lo largo  
de la vida

# FULL STACK Node JS

## Clase 23

SQL 2

# Lenguaje SQL y Sublenguajes DDL y DML



# Les damos la bienvenida

Vamos a comenzar a grabar la clase

## Clase 22

**Introducción a Base de Datos**

- ¿Qué es una Base de datos?
- BBDD relacionales y no relacionales.
- Entorno MySQL. Instalación. Clientes MySQL.
- DER. Entidad, atributo y tipo de datos. Primary key.
- Creación de una BD.
- Backup y restauración de bases de datos.

## Clase 23

**Lenguaje y Sublenguajes SQL**

- Gestión y manipulación de datos con SQL.
- Gestión y manipulación de datos.
- Sublenguajes DDL y DML.
- Consultas: Estructura consulta SQL. Cláusulas SELECT, FROM, WHERE.
- Alias y literales. ORDER BY.

## Clase 24

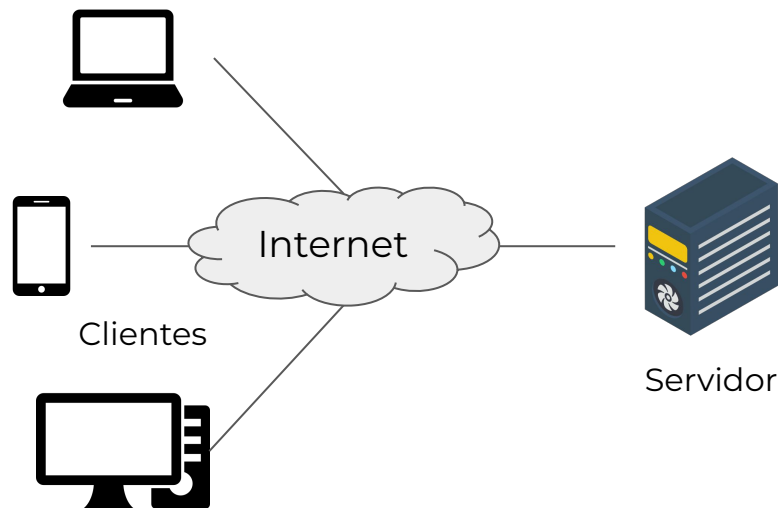
**JOIN y Subconsultas**

- JOIN: Inner, Left, Right.
- Funciones de agregación, GROUP BY, HAVING.
- Funciones escalares: Caracteres o cadena, Conversión, Fecha y tiempo, Matemáticas.
- Subconsultas.

# Arquitectura Cliente-Servidor

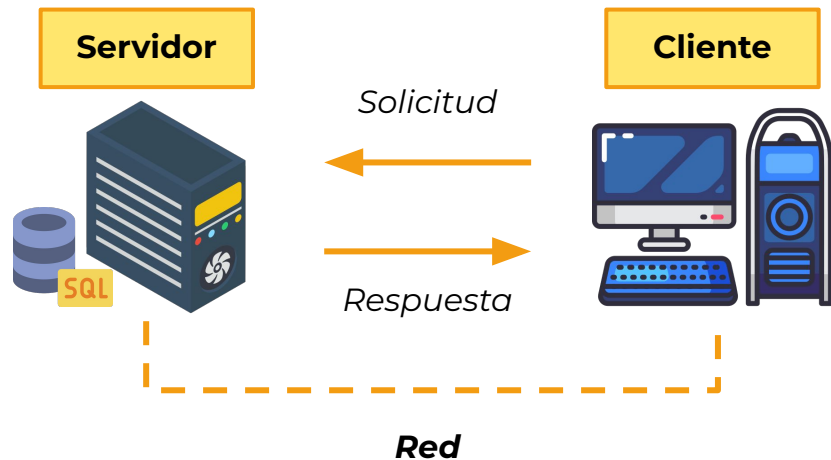
Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**.

- Un cliente realiza peticiones a otro programa.
- El servidor es quien le da respuesta.



# Cliente-Servidor en Bases de Datos

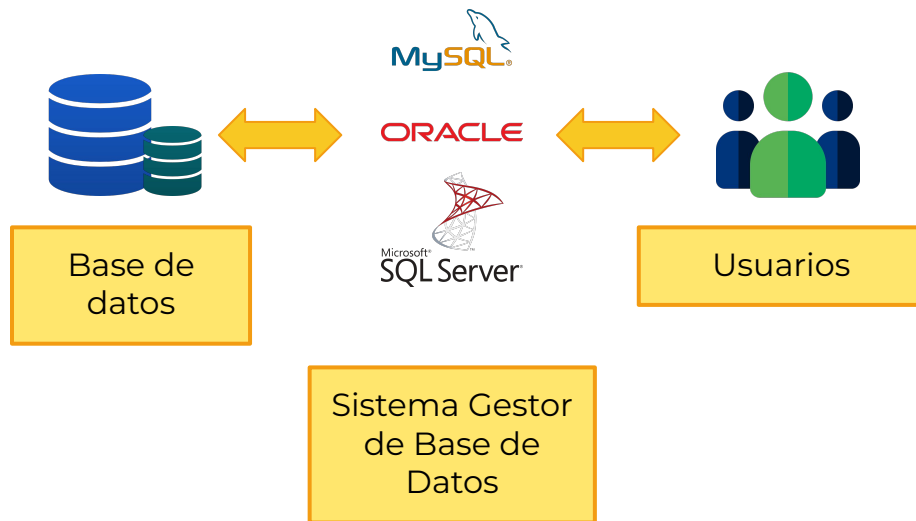
Las bases de datos en general utilizan la arquitectura Cliente-Servidor para proveer servicios de almacenamiento de información a determinados usuarios (Clientes).



# ¿Cómo se conecta un cliente a un servidor de BD?

El software intermediario entre un usuario y el servidor que provee el servicio de almacenamiento en bases de datos es conocido como **SGBD** (Sistema Gestor de Bases de Datos).

A través de los SGBD, los usuarios pueden hacer **CONSULTAS** en lenguaje **SQL** (*Structured Query Language* o Lenguaje de Consulta Estructurado) para realizar distintas operaciones.



# ¿Cómo armar un Servidor de BD?

Para armar un servidor de base de datos se pueden utilizar diferentes softwares, entre ellos, distribuciones de Linux, sistemas operativos especializados para bases de datos, servidores virtuales, servidores online, etc.

De forma experimental, el software que podremos utilizar para armar un servidor de BD es el **XAMPP Server**, visto en la presentación anterior. Para más detalles ver tutoriales recomendados.

El Sistema Gestor de Bases de Datos que utilizaremos será **MySQL**, uno de los más utilizados a nivel mundial.



# ¿Cómo empezamos a pensar en una BD?

La **manipulación** de los datos consiste en la realización de operaciones de *inserción*, *borrado*, *modificación* y *consulta* de la información almacenada en la base de datos. La **inserción** y el **borrado** son el resultado de añadir nueva información o eliminarla de nuestra base de datos, tomando en cuenta las restricciones marcadas por el DDL y las relaciones entre la nueva información y la antigua. La **modificación** nos permite alterar esta información, y la **consulta** nos permite el acceso a la información almacenada en la base de datos siguiendo criterios específicos.

Estas operaciones se podrán ejecutar a través de sentencias, que nos permitirán más adelante realizar los sistemas denominados **CRUD**.

**CRUD:** acrónimo de “Crear, Leer, Actualizar y Borrar” (*Create, Read, Update and Delete*), usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

# Sentencias DDL

**Lenguaje de definición de datos** (DDL: Data Definition Language): se encarga de la **modificación de la estructura** de los objetos de la base de datos.

Incluye órdenes o sentencias para crear, modificar o borrar las tablas en las que se almacenan los datos de la base de datos.

Utilizamos tres sentencias: CREATE, ALTER y DROP.

# Sentencias DDL: CREATE, ALTER y DROP

**CREATE:** Crear una base de datos

Con `CREATE DATABASE` creamos una base de datos con el nombre indicado en esa orden. Es necesario tener permisos del sistema de base de datos. Un equivalente es `CREATE SCHEMA`.

Si la base de datos ya existe se produce un error, para salvarlo podemos especificarle `IF NOT EXISTS`.

Ejemplo: `CREATE DATABASE IF NOT EXISTS databasename;`

# Sentencias DDL: CREATE, ALTER y DROP

**CREATE TABLE:** Crea una tabla:

```
CREATE TABLE alumnos (  
    dni int(11),  
    nombre varchar(30),  
    apellido varchar(30),  
    fecha_nac date  
);
```

Si intentamos crear una tabla con un nombre ya existente (existe otra tabla con ese nombre), mostrará un mensaje de error indicando que la acción no se realizó porque ya existe una tabla con el mismo nombre.

**SHOW TABLES:** Nos permite ver las tablas existentes en una base de datos.

# Ejemplo CREATE

```
1 • ○ CREATE TABLE `alumnos` (  
2     `id` int(11) NOT NULL AUTO_INCREMENT,  
3     `id_escuela` int(11) DEFAULT NULL,  
4     `legajo` int(11) DEFAULT NULL,  
5     `nombre` varchar(45) DEFAULT NULL,  
6     `nota` decimal(10,0) DEFAULT NULL,  
7     `grado` int(11) DEFAULT NULL,  
8     `email` varchar(45) NOT NULL,  
9     PRIMARY KEY (`id`),  
10    KEY `fk_escuela_id_idx` (`id_escuela`)  
11 ) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;
```

# Sentencias DDL: CREATE, ALTER y DROP

**ALTER TABLE:** Nos permite agregar, eliminar o modificar una columna..

**ALTER TABLE ... ADD:** Agrega una columna:

```
ALTER TABLE nombre_de_tabla  
ADD nombre_de_columna tipo de dato;
```

**ALTER TABLE ... DROP:** Elimina una columna:

```
ALTER TABLE nombre_de_tabla  
DROP COLUMN nombre_de_columna;
```

# Sentencias DDL: CREATE, ALTER y DROP

**DROP TABLE:** Elimina una tabla.

```
DROP TABLE alumnos;
```

Si tipeamos nuevamente:

```
DROP TABLE alumnos;
```

Aparece un mensaje de error, indicando que no existe, ya que intentamos borrar una tabla inexistente. Para evitar este mensaje podemos tipear:

```
DROP TABLE IF EXISTS alumnos;
```

# Ejemplo ALTER y DROP

## ALTER TABLE

```
1 • ALTER TABLE `escuelas`.`escuelas`  
2   ADD COLUMN `partido` VARCHAR(45) NULL DEFAULT '' AFTER `localidad`;  
3
```

## DROP TABLE

```
1 • DROP TABLE escuelas;  
2
```



# Sentencias DDL: CREATE, ALTER y DROP

**DESCRIBE:** Nos permite ver la estructura de una tabla.

```
DESCRIBE alumnos;
```

Aparecerá lo siguiente:

	Field	Type	Null
	dni	int(11)	YES
	nombre	varchar(30)	YES
	apellido	varchar(30)	YES
	fecha nac	date	YES

# Sentencias DML

El **Lenguaje de Manipulación de Datos** (Data Manipulation Language, DML) permite a los usuarios de una base de datos llevar a cabo tareas de consulta o modificación. Para hacer estas actividades dentro del lenguaje SQL tenemos las siguientes sentencias:

**INSERT:** Para agregar un registro (fila o tupla).

**UPDATE:** Para modificar atributos de una o varios registros.

**DELETE:** Para borrar registros completos de una tabla.

**SELECT:** Para obtener datos de una base de datos.

# Sentencias de escritura

**INSERT INTO:** especifica en qué tabla se pretende **insertar** un dato.

**VALUES:** utilizada en conjunto con INSERT INTO especifica qué valores irán en la tabla.

La lista de atributos es opcional, pero si no se define entonces el DBMS espera una lista de valores coherente con todos los atributos de la tabla.

```
1 • INSERT INTO `escuelas`.`alumnos`  
2   (  
3     `id`,  
4     `id_escuela`,  
5     `legajo`,  
6     `nombre`,  
7     `nota`,  
8     `grado`,  
9     `email`)  
9   VALUES  
10  (  
11    <{id: }>,  
12    <{id_escuela: }>,  
13    <{legajo: }>,  
14    <{nombre: }>,  
15    <{nota: }>,  
16    <{grado: }>,  
17    <{email: }>);
```

Campos de datos que le voy a pasar a la tabla

Valores que le voy a pasar a la tabla **(en el mismo orden)**

Para hacer un **INSERT**: clic derecho en la tabla  
> Send to SQL Editor > Insert Statement



# Sentencias de modificación

**UPDATE:** especifica en qué tabla se pretende **modificar** un dato.

**SET:** utilizada en conjunto con UPDATE especifica cuál será el nuevo valor/dato para el campo de ese/os registro/s en particular.

**¡CUIDADO!** Si no hay cláusula WHERE lo que ocurrirá es que se actualizarán todos los registros de la tabla.

Es un error muy común que cometen algunos usuarios de base de datos.

```
1 • UPDATE `escuelas`.`alumnos`  
2 SET  
3 `nombre` = 'Roberto Luis Sánchez',  
4 `email` = 'robertoluissanchez@gmail.com'  
5 WHERE id` = 6; Condición (fundamental)
```

Original

6	1	190	Roberto Sanchez	8	3	
---	---	-----	-----------------	---	---	--

Actualizado


6	1	190	Roberto Luis Sánchez	8	3	robertoluissanchez@gmail.com
---	---	-----	----------------------	---	---	------------------------------

Para hacer un **UPDATE**: clic derecho en la tabla  
> Send to SQL Editor > Update Statement

# Sentencias de baja

**DELETE:** permite eliminar uno o varios registro/s de una tabla de forma permanente.

```
1 • DELETE FROM `escuelas`.`alumnos`  
2 WHERE `id` = 3; Condición (fundamental)
```



2	2	1002	Tomás Smith	8	1
4	1	101	Juan Perez	10	3

**¡CUIDADO!** Si no hay cláusula WHERE lo que ocurrirá es que se eliminarán todos los registros de la tabla y quedará vacía.

**El registro con id = 3 ha sido eliminado**

Para hacer un **DELETE**: clic derecho en la tabla  
> Send to SQL Editor > Delete Statement

# Sentencias SQL | Lectura

Las consultas SQL son los “diálogos” o “preguntas” que se generan entre el usuario y el SGBD dentro del cual se encuentran almacenados los datos. Las cláusulas más conocidas son:

## DE LECTURA:

- **SELECT:** especifica qué atributo (dato) se pretende obtener.
- **FROM:** utilizada en conjunto con SELECT, especifica desde qué tabla (entidad) se pretende traer el dato.
- **WHERE:** establece una condición específica que deberá cumplir el dato que se pretende traer (cláusula no obligatoria).

# Sentencias SQL | Lectura

**Ejemplo:** Supongamos que tenemos una tabla de empleados y queremos traer el nombre, apellido y fecha de nacimiento de todos aquellos que hayan nacido después del año 1970 inclusive.

EMPLEADO						
id_empleado	nombre	apellido	sexo	fecha_nacimiento	salario	puesto
1	Juan	Perez	M	22-09-1960	5000	administrador
2	Mario	Gimenez	M	10-02-1980	3000	secretario
3	Susana	Malcorra	F	11-03-1980	3000	secretaria
4	María	Casan	F	01-02-1965	6000	administrador

```
SELECT nombre, apellido, fecha_nacimiento  —————→ ¿Qué atributo/s quiero traer?  
FROM empleado                             —————→ ¿De dónde lo/straigo?  
WHERE empleado.fecha_nacimiento >= 01-01-1970; —————→ ¿Qué condición tiene/n que cumplir?
```

**El resultado de esta consulta será:**

Mario Gimenez 10-02-1980

Susana Malcorra 11-03-1980



# Sentencias SQL | Orden y agrupamiento

## DE ORDEN Y/O AGRUPAMIENTO:

- **ORDER BY:** utilizada para especificar por qué criterio se pretende ordenar los registros de una tabla.
- **GROUP BY:** utilizada para especificar por qué criterio se deben agrupar los registros de una tabla.

**Ejemplo:** obtener todos los empleados ordenados por apellido.

```
SELECT *  
FROM empleado  
ORDER BY apellido;
```

El resultado de esta consulta será traer TODOS los empleados, ordenados por apellido, en lugar de estar ordenados por id (identificación del empleado)

El \* significa que deberá traer TODOS los campos sin distinción.

# Sentencias SQL | Orden | Ejemplos

Orden por una columna  
(por defecto ascendente)

```
1 • SELECT *  
2 FROM escuelas.alumnos  
3 ORDER BY nombre;
```

	id	id_escuela	legajo	nombre	nota
▶	4	1	101	Juan Perez	10
	7	1	106	Martín Bossio	10
	9	4	1234	Pedro Gómez	6
	5	1	105	Pedro González	9

Orden por más  
de una columna

```
1 • SELECT *  
2 FROM escuelas.alumnos  
3 ORDER BY id_escuela, nombre;
```

	id	id_escuela	legajo	nombre	nota
	4	1	101	Juan Perez	10
	7	1	106	Martín Bo...	10
	5	1	105	Pedro Go...	9
	6	1	190	Roberto L...	8
	1	2	1000	Ramón M...	8

Orden descendente  
de una columna

```
1 • SELECT *  
2 FROM escuelas.alumnos  
3 ORDER BY id_escuela, nombre DESC;
```

	id	id_escuela	legajo	nombre	nota
	6	1	190	Roberto L...	8
	5	1	105	Pedro Go...	9
	7	1	106	Martín Bo...	10
	4	1	101	Juan Perez	10
	2	2	1002	Tomás Smith	8

# Sentencias SQL | LIMIT

- **SELECT LIMIT:** especifica el número de registros a devolver. Sintaxis:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

# Operadores de comparación o relacionales

Son utilizados en MySQL para comparar igualdades y desigualdades. Se utilizan en conjunto con la cláusula WHERE para determinar qué registros seleccionar.

Operador	Descripción
=	Igual
<>	Diferente
!=	Diferente
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Operador	Descripción
LIKE	Define un patrón de búsqueda y utiliza % y _
NOT LIKE	Negación de LIKE
IS NULL	Verifica si el Valor es NULL
IS NOT NULL	Verifica si el Valor es diferente de NULL
IN ()	Valores que coinciden en una lista
BETWEEN	Valores en un Rango (incluye los extremos)

# Sentencias SQL | LIKE

- Se utiliza para **comparaciones** con campos de tipo **cadenas de texto**.
- Esta sentencia se podría utilizar para consultar cuáles son los clientes que viven en una calle que contiene el texto “San Martín”.
- Al colocar el % al comienzo y al final estamos representando un texto que no nos preocupa cómo comienza ni cómo termina, siempre y cuando contenga la/s palabra/s que nos interesa. Sintaxis:

```
SELECT * FROM clientes c
WHERE calle LIKE '%San Martín%'
```

# Sentencias SQL | IS [NOT] NULL

Permiten seleccionar registros cuyo valor en un campo sea **null** o **no sea null (not null)**. No debemos confundir null con campo en blanco, es un campo que no tiene dato alguno asociado.

```
SELECT *  
FROM escuelas.alumnos  
WHERE nota IS NULL;
```

	id	id_escuela	legajo	nombre	nota	grado	email
	7	0	106	Martín Bo...	NULL	3	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
SELECT *  
FROM escuelas.alumnos  
WHERE nota IS NOT NULL;
```

	id	id_escuela	legajo	nombre	nota	grado	email
	1	2	1000	Ramón M...	8	1	rmesa...
	2	2	1002	Tomás Smith	8	1	
	4	1	101	Juan Perez	10	3	
	5	1	105	Pedro Go...	9	3	
	6	5	190	Roberto L...	8	3	roberto...
	8	4	100	Ramiro Es...	3	1	mail@m...
	9	4	1234	Pedro Gó...	6	2	

# Sentencias SQL | Uso de Alias

- Son muy utilizados en el uso de consultas o sentencias SQL extensas.
- Permite renombrar los nombres originales de tablas o campos de manera temporal. Esta propiedad es extensible a tablas y campos.
- Presenta algunas ventajas:
  - Permite acelerar la escritura de código SQL
  - Mejorar la legibilidad de las sentencias
  - Ocultar/Renombrar los nombres reales de las tablas o campos a usuarios
  - Permite asignar un nombre a una expresión, fórmula o campo calculado

**Ejemplo:** renombrar tablas y atributos calculados

```
SELECT V.precio , V.fecha, (V.precio * 1.21) AS precio_con_iva  
FROM ventas AS V
```

# Sentencias SQL | Operador IN | DISTINCT

**IN:** nos permite agregar una lista de posibilidades en lugar de encadenar cláusulas OR, ya que funciona de manera equivalente. Sintaxis:

```
SELECT codigo FROM productos  
WHERE descripción IN ('Harina' , 'Azúcar' , 'Leche')
```

**SELECT DISTINCT:** se usa para devolver solo valores distintos (diferentes) de una columna que puede tener registros duplicados. Sintaxis:

```
SELECT DISTINCT column1, column2, ...  
FROM table_name;
```



# Material extra

# Artículos de interés

- **Resumen SQL.pdf:** resumen con las sentencias SQL básicas más utilizadas.
- **Guía práctica de SQL:** guía de ejercicios con los que podrá poner en práctica los conocimientos de esta Unidad. Esta guía ***no es obligatoria*** pero les dará la práctica necesaria para poder trabajar sin problemas con las bases de datos.
- **world.sql:** script para generar la base de datos que deberá utilizar para resolver la guía práctica.
- **der-bd-world.jpg:** DER de la base de datos anteriormente mencionada.
- [Tutorial en w3schools](#)
- [Página Oficial MySQL](#)

# No te olvides de dar el presente

# Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar los Ejercicios de repaso.

**Todo en el Aula Virtual.**

**Muchas gracias por tu atención.**

**Nos vemos pronto**