


---

Curso de  
**Introducción a  
MySQL y MariaDB**

@RetaxMaster

 ¿Cuál es la  
diferencia entre  
MySQL y MariaDB?



# La creación de MySQL ✨

Michael Widenius (Monty pa' los cuates), junto con un equipo de desarrolladores, lanza en 1995 un **DBMS** llamado MySQL, el cual en un futuro sería de los más populares.

Fue lanzado como un **sistema de código abierto.**

# Un motor muy popular

A lo largo de varios años, desde su lanzamiento, MySQL ha sido un motor de bases de datos muy popular. Incluso es el primer motor de bases de datos que se suele aprender cuando recién estás empezando.

# La salida de Monty

En 2008, MySQL fue vendida a **Sun Microsystems**. Monty estuvo trabajando con ellos durante un año hasta que en 2009 decidió salir junto con los demás desarrolladores iniciales de MySQL.

¿El motivo? Oracle...



# Conflicto de intereses

Oracle compró a Sun Microsystems. Monty junto con los demás desarrolladores temían que Oracle desarrollara su propio motor de base de datos (de paga) al mismo tiempo que desarrollaban MySQL (de código abierto).





# Sospechaban de Oracle

En esas épocas MySQL era una amenaza para Oracle. El DBMS que ellos tenían representaba aproximadamente el 80% de los ingresos de la compañía.

# ¿Y si era una compra hostil?

El equipo de desarrollo de MySQL temía que Oracle hubiese comprado MySQL...  
**solo para matarlo y quitarse la competencia de encima.**



# La creación de MariaDB ✨

Monty quería que siempre existiera una versión de MySQL gratuita, pero con Oracle como propietario, nada era seguro.

Entonces crearon una bifurcación de MySQL y la llamaron MariaDB. 🙄

# ¿El reemplazo de MySQL?

MariaDB fue concebido como un reemplazo directo de MySQL. De hecho, reemplazar MySQL por MariaDB en una aplicación es muy fácil, ya que son altamente compatibles.



# El objetivo de MariaDB

Los objetivos de MariaDB son:

- Mantener al equipo de desarrollo original de MySQL reunidos.
- Garantizar que siempre haya una versión de MySQL gratuita.

# Pero, ¿MySQL es gratis?

**Sí... por ahora...**

MySQL es “gratis”, pero nada le impide a Oracle volverla un DBMS de pago en un futuro. Actualmente tiene dos licencias:

- Licencia GPL
- Licencia comercial

# ♥ MariaDB

Por su parte,  
MariaDB cuenta con  
una única licencia  
GPL (de uso libre).



# Ventajas de MariaDB por sobre MySQL

- Es mucho más eficiente.
- Soporta muchos motores de almacenamiento diferentes.
- Ofrece un mejor rendimiento.
- Es muy similar a MySQL.

# Te doy la bienvenida 🍆



**RetaxMaster** 🖥️💚  
@retaxmaster

Course Director at @platzi 💚 Fullstack Web Developer, Software Engineer, Backend, Frontend, I love to code!! Making videos to teach code 🛠️

[Traducir la biografía](#)

[linktr.ee/RetaxMaster](https://linktr.ee/RetaxMaster) 📅 Se unió en diciembre de 2011

111 Siguiendo 2.302 Seguidores

Editar perfil



# Antes de tomar este curso...



Curso de Introducción a la Terminal y Línea de Comandos.



Curso de Fundamentos de Bases de Datos.

# ¿Quieres aprender SQL profesional?



**End of class**

**- No eliminar please :3 -**



# **Presentación del proyecto**

**End of class**

**- No eliminar please :3 -**





# **Diagramamos nuestra base de datos**

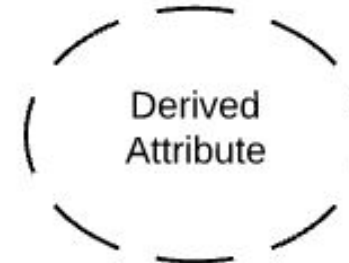
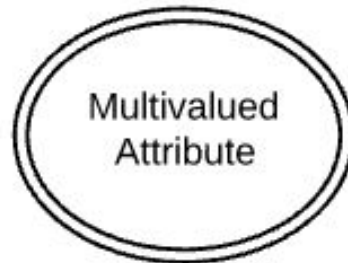
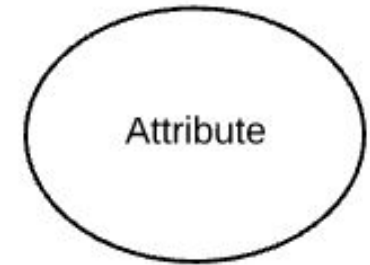
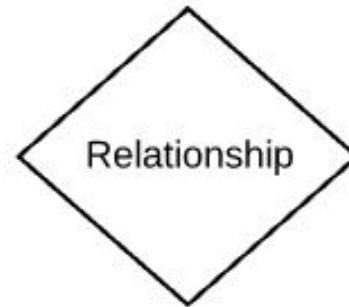
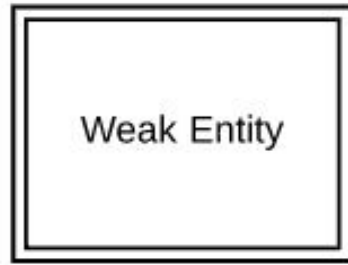
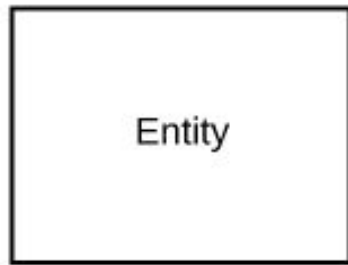
# Diagrama entidad relación

En todo buen proyecto, primero debemos tener una planeación de qué queremos hacer. En el caso de las bases de datos, los diagramas entidad relación son perfectos para tener esa planificación.

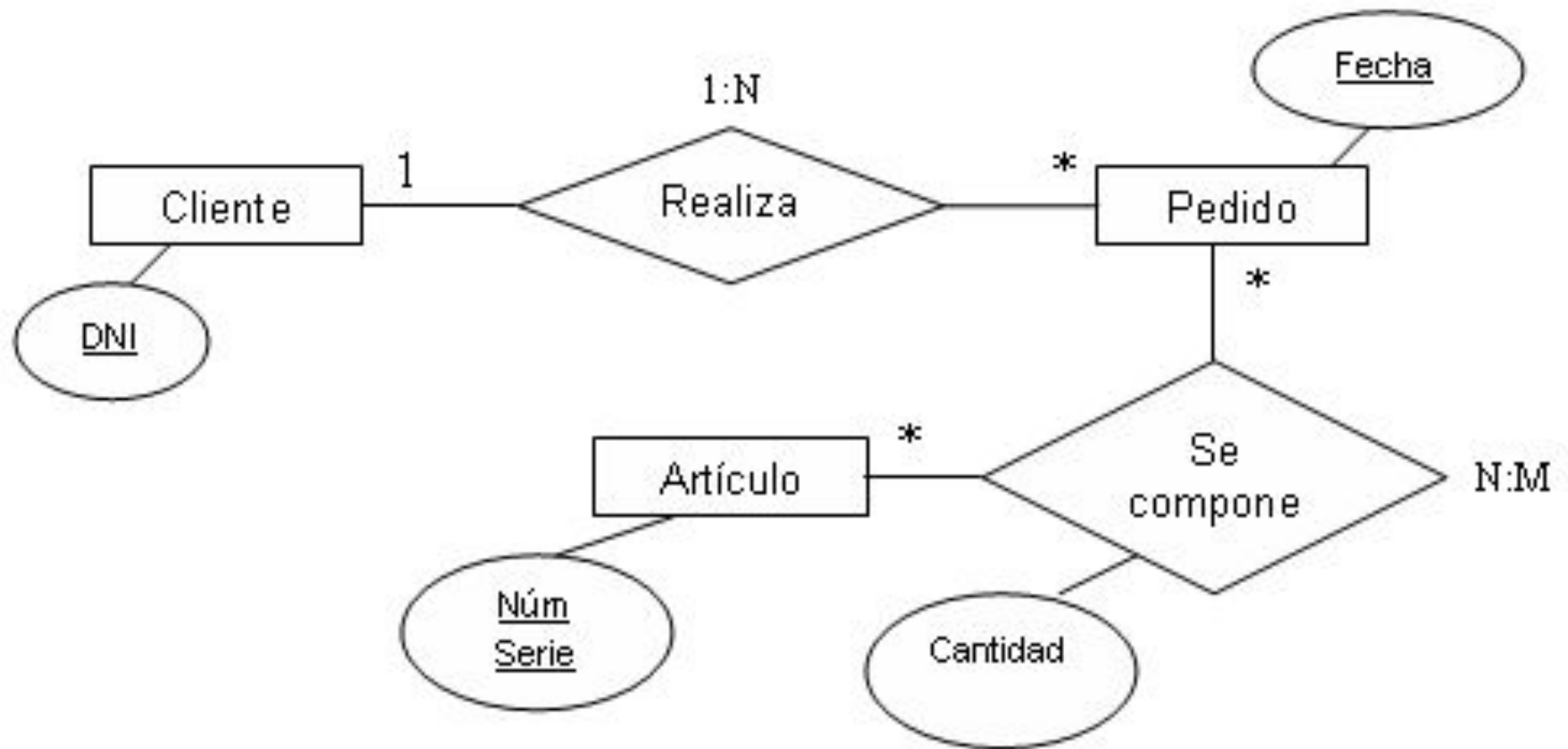
# Diagrama conceptual

Este diagrama expresa una visión amplia de lo que queremos representar en nuestra base de datos. Permiten describir entidades, tipos de entidades y atributos, y establecer relaciones entre entidades.

# Repasemos su simbología



# Diagrama conceptual





# Diagrama físico

Este es un diagrama más detallado. Incluye relaciones, tipos de relaciones, tablas, columnas e incluso tipos de datos.

Además, podemos definir los tipos de llaves que tendrá nuestra base de datos.

# Repasemos su simbología

Entity
Field
Field
Field

Entity	
Key	Field
Key	Field
Key	Field

Entity	
Field	Type
Field	Type
Field	Type

Entity		
Key	Field	Type
Key	Field	Type
Key	Field	Type

# Repasemos su simbología



One



Many



One (and only one)



Zero or one

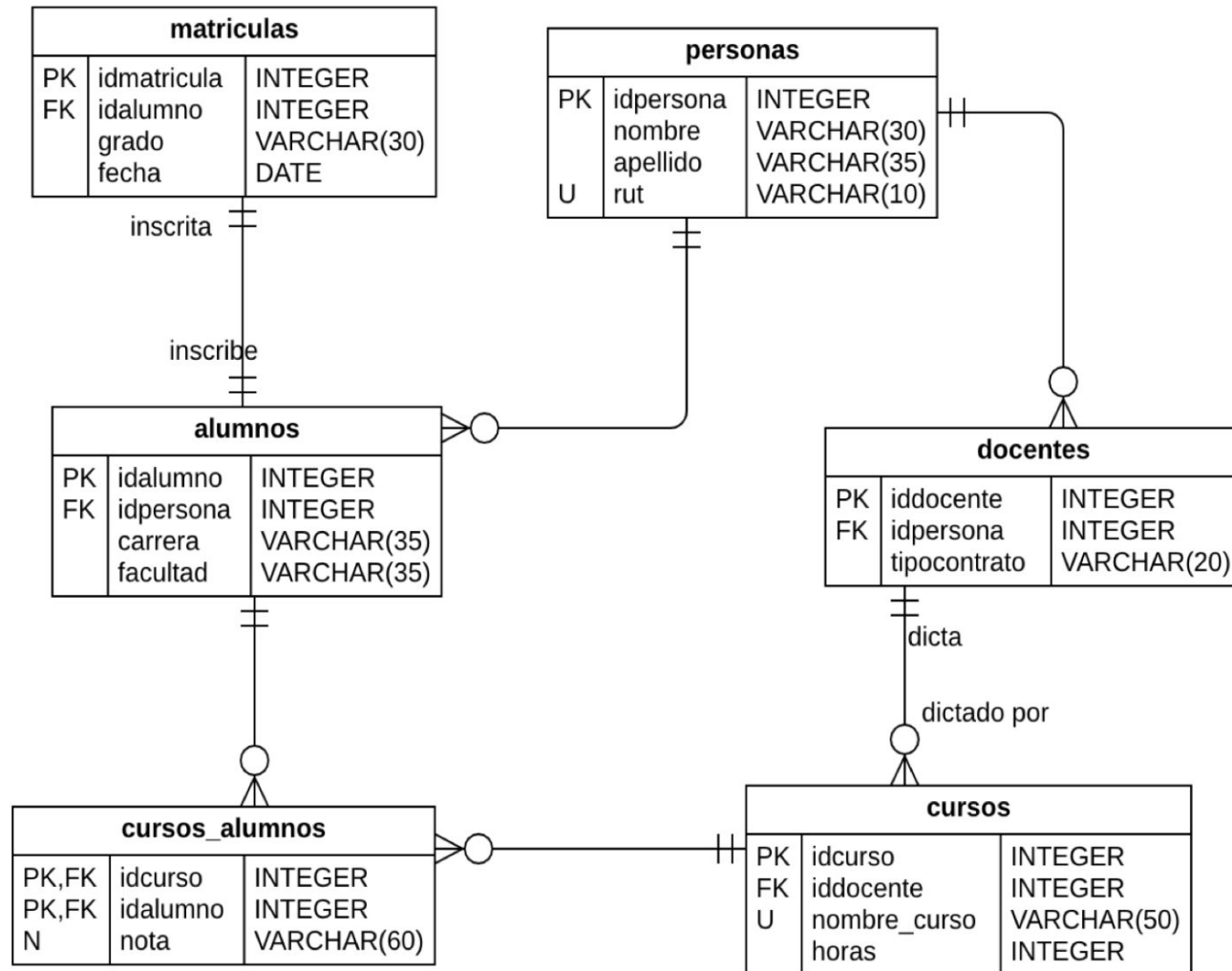


One or many



Zero or many

# Diagrama físico



**¡Vamos al navegador!**



**End of class**

**- No eliminar please :3 -**

---

# $\leftrightarrow$ Estableciendo relaciones

**End of class**

**- No eliminar please :3 -**





---

# **Creación del diagrama físico de nuestra base de datos**

**End of class**

**- No eliminar please :3 -**

---

**↔ Estableciendo  
relaciones en nuestro  
diagrama físico**

**End of class**

**- No eliminar please :3 -**



# **Instalación de MariaDB en Windows**

**End of class**

**- No eliminar please :3 -**



# **Instalación de MariaDB en Linux**

**End of class**

**- No eliminar please :3 -**



---

# **Instalación de MariaDB en macOS**

**End of class**

**- No eliminar please :3 -**




# Creando nuestra base de datos

# CREATE DATABASE



```
CREATE DATABASE database_name;
```

# CREATE TABLE



```
USE database_name;

CREATE TABLE `table_name` (
    `column_name` TYPE ATTRIBUTES,
    PRIMARY KEY(column_name)
)
CHARSET=charset_type
COLLATE=collate_type
```

**¡Vamos al código!**





---

# **Manejo de usuarios para la base de datos**



# GRANT



```
GRANT
    privileges
ON
    databases.tables
TO
    user;
```



**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**



# Creando nuestras tablas

**¡Vamos al código!**



**End of class**


**- No eliminar please :3 -**

---



# **Modificando tablas con ALTER TABLE**

# ALTER TABLE



```
USE database_name;
```

```
ALTER TABLE `table_name`  
MODIFY `column_name` MODIFICATIONS,  
ADD PRIMARY KEY(column_name)
```



**¡Vamos al código!**





**End of class**


**- No eliminar please :3 -**



---

# Llenando nuestra base de datos

# INSERT INTO



```
INSERT INTO `table_name`  
(col1, col2, col3, ...) VALUES  
( 'value1_col1', 'value1_col2', 'value1_col3', 'value1_col4' ),  
( 'value2_col1', 'value2_col2', 'value2_col3', 'value2_col4' ),  
( 'value3_col1', 'value3_col2', 'value3_col3', 'value3_col4' );
```

**¡Vamos al código!**



---



# **Actualizando información**

# UPDATE



```
UPDATE `table_name`  
SET column_name = "New value"  
WHERE column_condition = "value_condition";
```



**¡Vamos al código!**






---

# Borrando datos



# DELETE FROM WHERE



```
DELETE FROM `table_name`  
WHERE colum_condition = "value_condition";
```

# DELETE FROM



```
DELETE FROM `table_name`;
```

# TRUNCATE



```
TRUNCATE TABLE `table_name`;
```

**¡Vamos al código!**




**End of class**

**- No eliminar please :3 -**



# Listando datos

# SELECT FROM



```
SELECT field_name1, field_name  
FROM `table_name`  
WHERE column_condition = "value condition";
```



**¡Vamos al código!**





**End of class**

**- No eliminar please :3 -**

---

**□ ¿Qué son las  
consultas anidadas?**

# Consultas anidadas

En SQL podemos poner consultas dentro de consultas. A esto le llamamos “consultas anidadas”.

# ¿Para qué me sirven?

Con este tipo de consultas podemos traernos información de una tabla para usarla en una consulta hacia otra tabla.

# ¿Para qué me sirven?

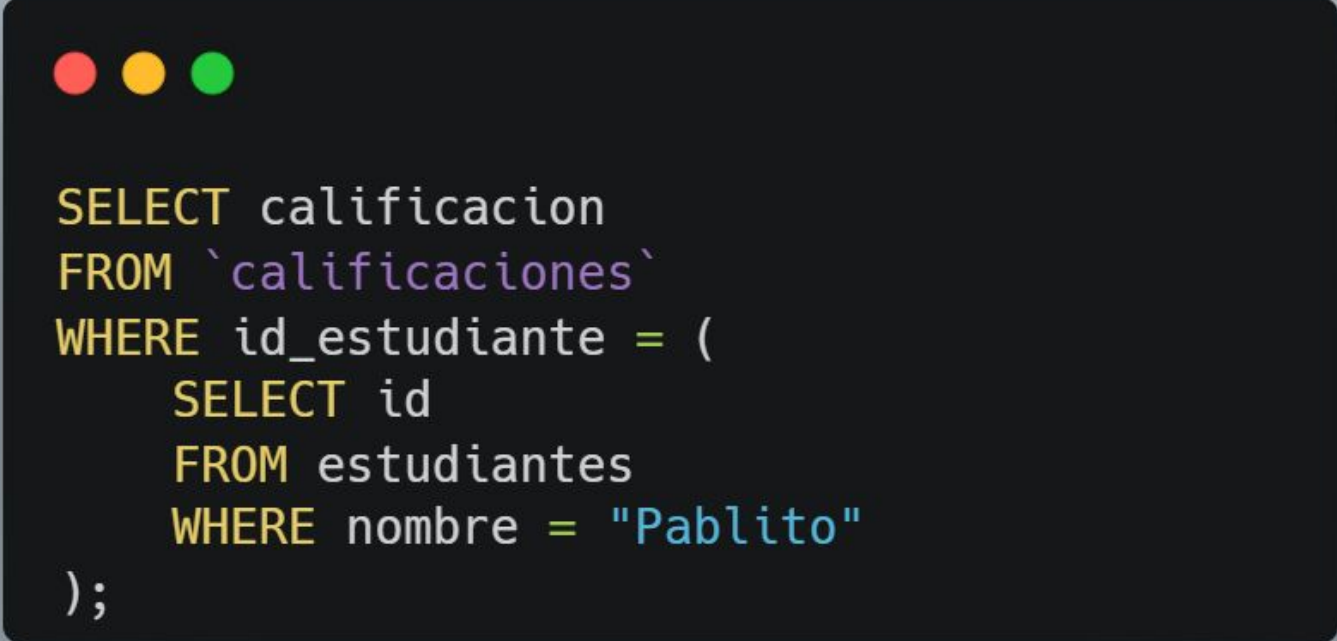
Imagina que necesitas el ID de algún registro que tienes en una tabla para buscar algo dentro de otra tabla, entonces puedes usar consultas anidadas.

# Nested queries

estudiantes	
id	nombre
1	José
2	Pablito
3	María

calificaciones		
id	id_estudiante	calificacion
1	1	80
2	2	95
3	3	100

# Nested queries



```
SELECT calificacion
FROM `calificaciones`
WHERE id_estudiante = (
    SELECT id
    FROM estudiantes
    WHERE nombre = "Pablito"
);
```

# Hay ciertas reglas

Por ejemplo, en este tipo de consultas, tu consulta anidada debe devolver únicamente un solo registro y una única columna.



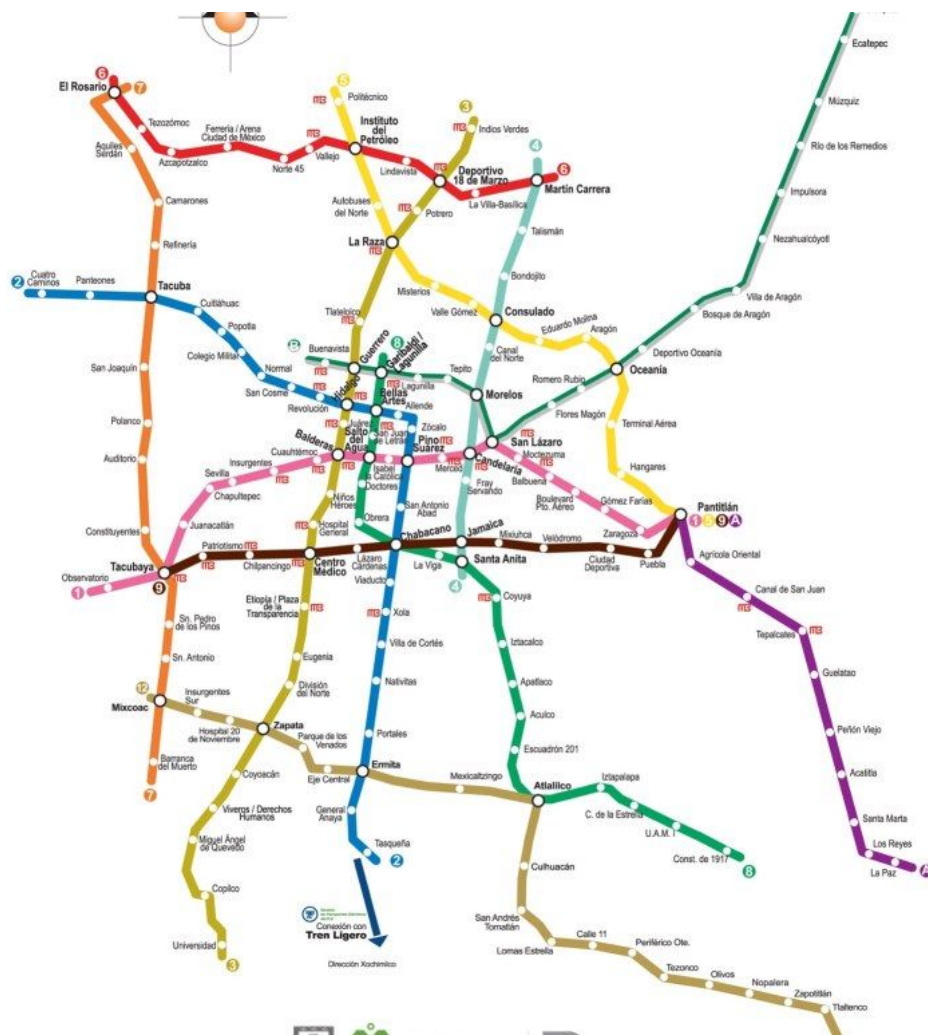
**End of class**

**- No eliminar please :3 -**

---

**□ Creando nuestra  
primera consulta  
anidada**

# Mapa metro





**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**



---



# ¿Cómo funciona la geolocalización?

# Spatial Functions

Las últimas versiones de MySQL y MariaDB proveen de una serie de funciones que permiten hacer cálculos de geolocalización.



# Spatial Functions

Esta es una serie de funciones que permiten hacer diversos cálculos como:

- Medir distancias.
- Encontrar si un punto está dentro de un área.
- Guardar polígonos.

# Tipo POINT

Para usar estas funciones necesitaremos de un tipo de dato especial llamado “**POINT**”. De esta forma podremos usar las funciones para calcular distancias.

**¡Vamos al navegador!**



**End of class**

**- No eliminar please :3 -**



---

# **Creando nuestras consultas de geolocalización**

**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**



---

# **Reto:** **Geolocalización con** **consultas anidadas**



**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**

---



# ¿Qué son las Stored Procedures?

# Procedimientos almacenados

Los Stored Procedures (o procedimientos almacenados), en resumidas palabras, son funciones del lado de la base de datos.

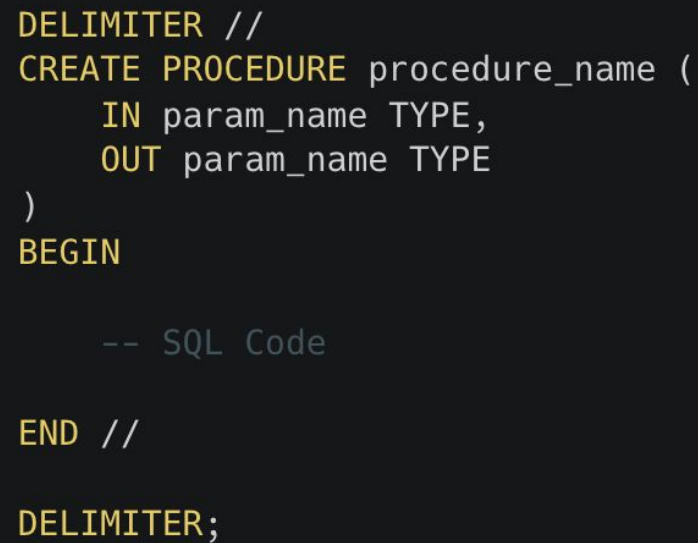
# Procedimientos almacenados

Literalmente, podemos programar funciones que podemos invocar desde nuestra terminal, o incluso desde algún lenguaje de programación.

# ¿Para qué sirven?

Nos permiten crear consultas frecuentes. En lugar de que desde un lenguaje de programación escribamos una consulta tal cual, podemos hacer una ***rutina*** que haga dicha consulta 😊.

# Stored Procedures



```
DELIMITER //  
CREATE PROCEDURE procedure_name (  
    IN param_name TYPE,  
    OUT param_name TYPE  
)  
BEGIN  
  
    -- SQL Code  
  
END //  
DELIMITER;
```

# Prepared Statements

En un procedimiento almacenado también podemos tener **consultas preparadas**. Esto nos permite tener una cadena de texto con código SQL que podremos ejecutar 🙄.



# ¿Debería aplicar esta lógica desde un lenguaje mejor?

Depende de cómo se organice tu equipo de trabajo y de quién quiera tener el control de las consultas.

**End of class**

**- No eliminar please :3 -**



---

# **Creando nuestras primeras Stored Procedures**

**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**

---



# **Procedimientos almacenados con Prepared Statements**

**¡Vamos al código!**

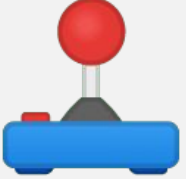


**End of class**

**- No eliminar please :3 -**



---



# ¿Qué son los triggers?

# Triggers

Los triggers son, en pocas palabras, eventos de nuestra base de datos. Podemos escuchar ciertos eventos y hacer acciones cada vez que sucedan.

# Triggers

Son similares a las Stored Procedures, con la diferencia de que estas se ejecutan cada vez que algún evento sucede.

# NEW y OLD

Al ser eventos, podemos acceder a los datos nuevos que estamos insertados e incluso a los datos viejos antes de que los nuevos los sustituyan 🙄.

**End of class**

**- No eliminar please :3 -**



# Creando nuestros primeros triggers

**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**





# Triggers compuestos

**¡Vamos al código!**



**End of class**

**- No eliminar please :3 -**



---

**Confesión  
personal:  
phpMyAdmin es  
superior**

**¡Vamos al navegador!**



**End of class**

**- No eliminar please :3 -**



---



# **Cómo seguir aprendiendo SQL**

# ¿Qué aprendiste?

- Crear bases de datos.
- Crear y modificar tablas.
- Sentencias básicas.
- Consultas anidadas.
- Stored procedures.
- Triggers.
- Geolocalización.





# ¿Qué sigue?

¡Ahora te toca poner todo en práctica!  
También, ya que sabes cómo usar un DBMS, puedes empezar a integrarlo con tu lenguaje de programación de preferencia para empezar a desarrollar tu backend. 🙄🙄

# ¡Nos vemos! 🖐️



**@RetaxMaster**



**/RetaxMaster**



**/RetaxMaster**



**@RetaxMaster**

