Relaciónes

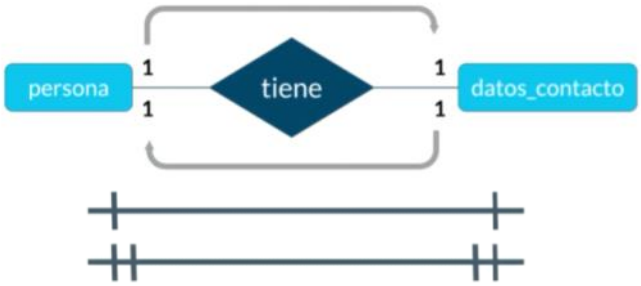
Friday, September 15, 2023 9:46 AM

Las relaciones nos permiten ligar o unir nuestras diferentes entidades y se representan con rombos. Por convención se definen a través de verbos.

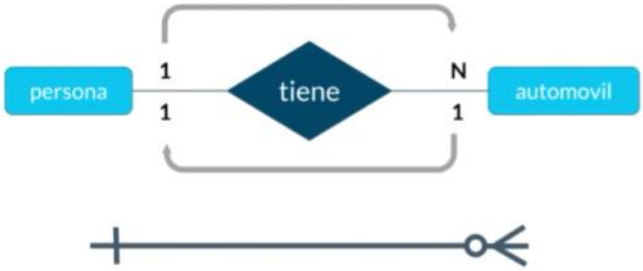
Las relaciones tienen una propiedad llamada cardinalidad y tiene que ver con números. Cuántos de un lado pertenecen a cuántos del otro lado:

Cardinalidad: 1 a 1
Cardinalidad: 0 a 1
Cardinalidad: 1 a N
Cardinalidad: 0 a N

Cardinalidad: 1 a 1



Cardinalidad: 1 a N



Cardinalidad: N a N

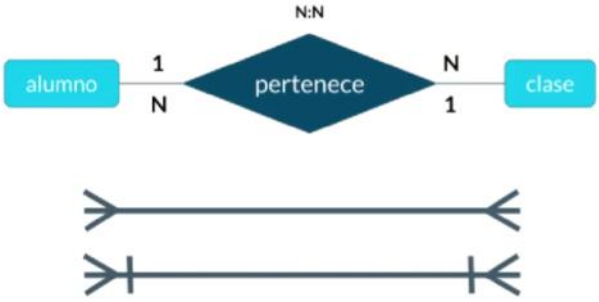
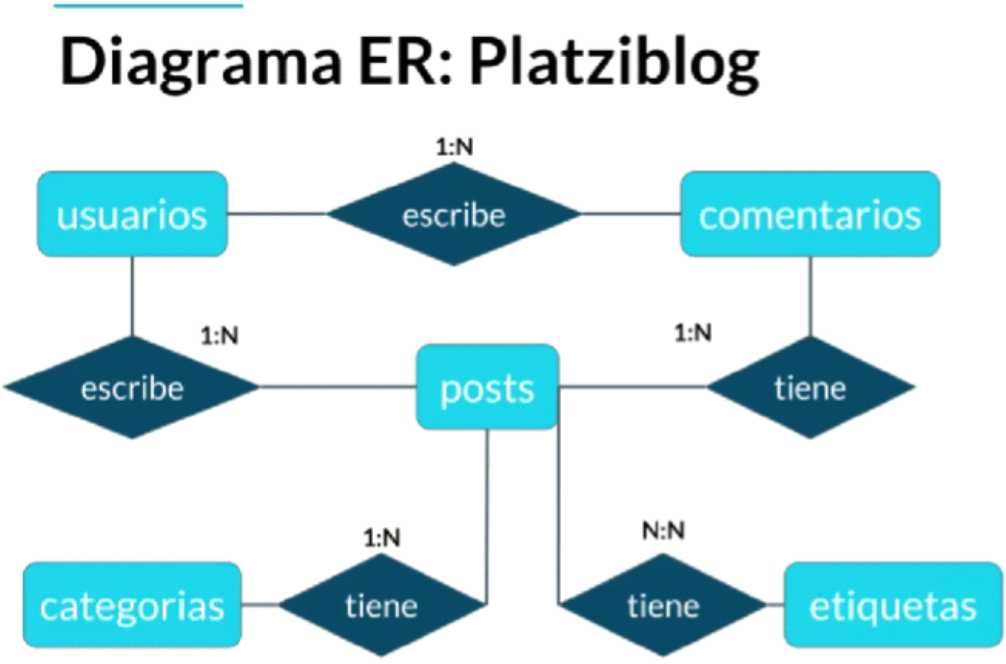


Diagrama ER

Friday, September 15, 2023 10:40 AM

Un diagrama es como un mapa y nos ayuda a entender cuáles son las entidades con las que vamos a trabajar, cuáles son sus relaciones y qué papel van a jugar en las aplicaciones de la base de datos.



Tipos de datos

Friday, September 15, 2023 12:14 PM

Tipos de dato:

Texto: CHAR(n), VARCHAR(n), TEXT

Números: INTEGER, BIGINT, SMALLINT, DECIMAL(n,s), NUMERIC(n,s)

Fecha/hora: DATE, TIME, DATETIME, TIMESTAMP

Lógicos: BOOLEAN

Constraints (Restricciones)

NOT NULL: Se asegura que la columna no tenga valores nulos

UNIQUE: Se asegura que cada valor en la columna no se repita

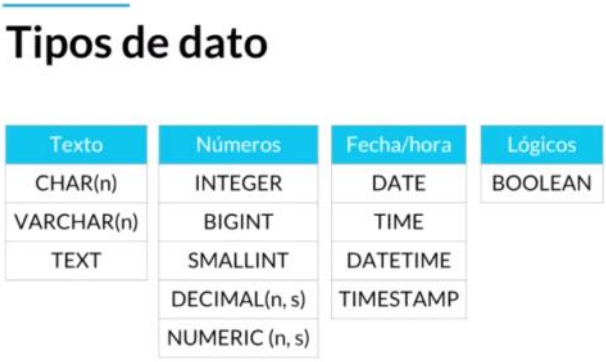
PRIMARY KEY: Es una combinación de NOT NULL y UNIQUE

FOREIGN KEY: Identifica de manera única una tupla en otra tabla

CHECK: Se asegura que el valor en la columna cumpla una condición dada

DEFAULT: Coloca un valor por defecto cuando no hay un valor especificado

INDEX: Se crea por columna para permitir búsquedas más rápidas



Normalización

Friday, September 15, 2023 12:56 PM

Primera forma normal (1FN)

Atributos atómicos (Sin campos repetidos)

No se deben repetir columnas con el mismo nombre, la siguiente tabla muestra la forma incorrecta:

Sin normalizar

alumno	nivel_curso	nombre_curso	materia_1	materia_2
Juanito	Maestría	Data engineering	MySQL	Python
Pepito	Licenciatura	Programación	MySQL	Python

La siguiente tabla se muestra como aplicamos la normalización para evitar repetir Columnas.

alumnos				
alumno_id	alumno	nivel_curso	nombre_curso	materia
1	Juanito	Maestría	Data engineering	MySQL
1	Juanito	Maestría	Data engineering	Python
2	Pepito	Licenciatura	Programación	MySQL
2	Pepito	Licenciatura	Programación	Python

Segunda forma normal (2FN)

Cumple 1FN y Cada campo de la tabla debe depender de una clave única

En la primera forma normal agregabamos una columna de 'materia' y ahi especificabamos la materia de cada alumno, pero esto provoca que un mismo alumno tenga muchas materias por lo tanto se empezaban a repetir los mismo alumnos y por ende los mismo ID. Para solucionar este problema aplicamos la segunda forma normal que seria separar las materias en otra tabla distinta como se muestra a continuación.

alumnos			
alumno_id	alumno	nivel_curso	nombre_curso
1	Juanito	Maestría	Data engineering
2	Pepito	Licenciatura	Programación

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

Tercera forma normal (3FN)

Cumple 1FN y 2FN y los campos que NO son clave NO deben tener dependencias.

alumnos		
alumno_id	alumno	curso_id
1	Juanito	1
2	Pepito	2

cursos		
curso_id	nivel_curso	nombre_curso
1	Maestría	Data engineering
2	Licenciatura	Programación

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

Cuarta forma normal (4FN)

Cumple 1FN, 2FN y 3FN los campos multivaluados se identifican por una clave única.

En esta tabla estamos repitiendo las materias dentro de la tabla materias por lo tanto deberiamos de crear una tabla diferent, la tabla de abajo es la correcta.

alumnos		
alumno_id	alumno	curso_id
1	Juanito	1
2	Pepito	2

cursos		
curso_id	nivel_curso	nombre_curso
1	Maestría	Data engineering
2	Licenciatura	Programación

materias	
materia_id	materia
1	MySQL
2	Python

materias_por_alumno		
mpa_id	materia_id	alumno_id
1	1	1
2	2	1
3	1	2
4	2	2

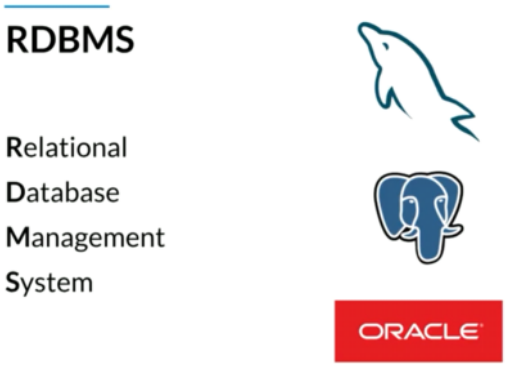
alumnos		
alumno_id	alumno	curso_id
1	Juanito	1
2	Pepito	2

cursos		
curso_id	nivel_curso	nombre_curso
1	Maestría	Data engineering
2	Licenciatura	Programación

materias		
materia_id	alumno_id	materia
1	1	MySQL
2	1	Python
3	2	MySQL
4	2	Python

RDBMS

Friday, September 15, 2023 4:46 PM



RDB (relational database)

RDBMS (Relational DataBase Management System) Sistema Manejador de Bases de datos relacionales.

La diferencia entre ambos es que las BBDD son un conjunto de datos pertenecientes (o al menos en teoría) a un mismo tipo de contexto, que guarda los datos de forma persistente para un posterior uso, y el Sistema de gestión de BBDD o sistema manejador, es el que nos permite acceder a ella, es un software, herramienta que sirve de conexión entre las BBDD y el usuario (nos presenta una interfaz para poder gestionarla, manejarla).

RDBMS

- MySQL
- PostgreSQL
- Etc

DDL Create

Saturday, September 16, 2023 1:39 PM

SQL tiene dos grandes sublenguajes:
DDL o **Data Definition Language** que nos ayuda a crear la estructura de una base de datos. Existen 3 grandes comandos:

Create: Nos ayuda a crear bases de datos, tablas, vistas, índices, etc.
Alter: Ayuda a alterar o modificar entidades.
Drop: Nos ayuda a borrar. Hay que tener cuidado al utilizarlo.
3 objetos que manipularemos con el lenguaje DDL:

Database o bases de datos
Table o tablas. Son la traducción a SQL de las entidades
View o vistas: Se ofrece la proyección de los datos de la base de datos de forma entendible.

DDL Command	Description
`CREATE DATABASE`	Creates a new database.
`DROP DATABASE`	Deletes a database.
`CREATE TABLE`	Creates a new table in a database.
`ALTER TABLE`	Alters the structure of an existing table.
`DROP TABLE`	Removes a table from a database.
`CREATE INDEX`	Creates an index on a table to improve a specific query performance.
`CREATE VIEW`	Creates a view, a virtual table based on one or more existing tables.
`CREATE PROCEDURE`	Creates a stored procedure, a precompiled SQL statement that can be run multiple times with different parameters.
`CREATE FUNCTION`	Creates a custom user-defined function that can be utilized in SQL statements.
`CREATE TRIGGER`	Creates a trigger, a type of stored procedure that is automatically executed when certain events occur, such as inserting, updating, or deleting data in a table.

DML

DML stands for **Data Manipulation Language**. It deals with data manipulation and includes the most common SQL statements such as `SELECT`, `INSERT`, `UPDATE`, `DELETE`, etc. DML statements are not auto-committed, meaning the changes can be rolled back if necessary. By mastering these DML commands, you can efficiently manipulate data in MySQL databases.

DML Command	Description
`SELECT`	Retrieves data from a table.
`INSERT`	Inserts new data into a table.
`UPDATE`	Updates existing data in a table.
`DELETE`	Deletes data from a table.
`REPLACE`	Updates or inserts a record into a table.
`MERGE`	Performs a UPSERT operation (insert or update) on a table.
`CALL`	Calls a stored procedure or Java subprogram.
`EXPLAIN`	Displays the execution plan for a given query.
`LOCK TABLE`	Locks a table to prevent other users from modifying it while a transaction progresses.

Views

Saturday, September 16, 2023 2:16 PM

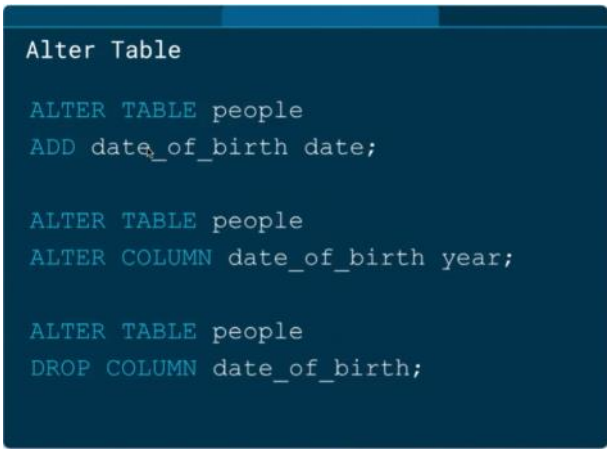
Vamos a asignarle un nombre a nuestra nueva vista, en este caso usaremos como convencion la letra 'v' indicando que es una vista y no otra nueva tabla. El nombre nos indica que solo listaremos los customers que se encuentran en brasil.
Y esos datos filtrados se guardaran en nuestra vista.

```sql
CREATE VIEW nombre_de_la_vista AS
SELECT columna1, columna2
FROM tabla
WHERE condicion;
```

Y en automatico se creara una nueva vista con los resultados. Podemos hacer una vista de diferentes forma, por ejemplo: donde unimos varias tablas, diferentes columnas y condiciones.

Alter

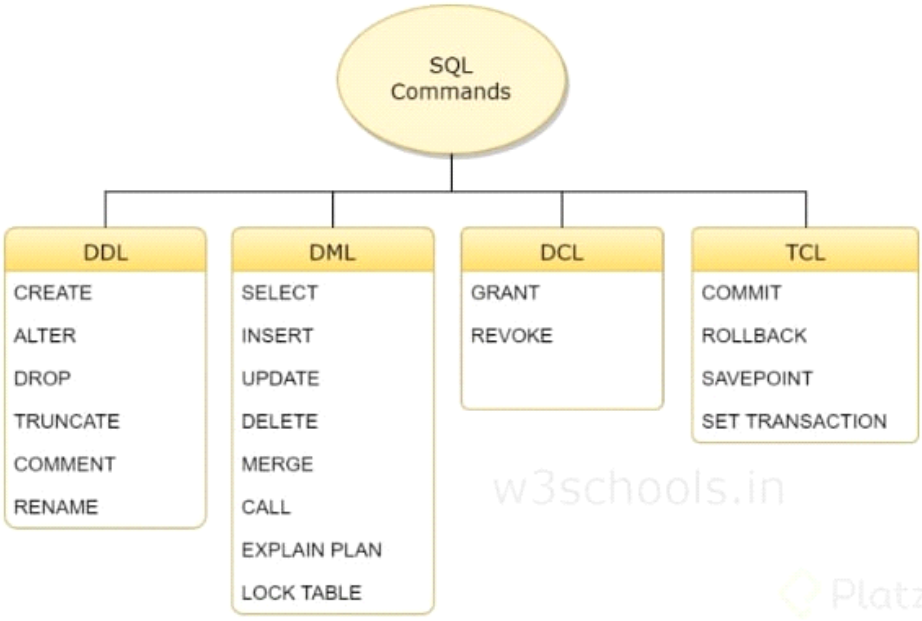
Saturday, September 16, 2023 2:29 PM



Nos sirve para modificar nuestra tabla, añadir, modificar columna, eliminar columna.

Comandos

Saturday, September 16, 2023 2:46 PM



DML

Saturday, September 16, 2023 2:59 PM

INSERT: Agrega un nuevo registro en la base de datos.

UPDATE: Actualiza un registro en la base de datos.

DELETE: Elimina un registro en la base de datos.

SELECT: Seleccionar algun elemento.

INSERT

```
Insert

INSERT INTO people (last_name,
first_name, address, city)
VALUES ('Hernández', 'Laura',
'Calle 21', 'Monterrey');
```

UPDATE

```
Update

UPDATE people
SET last_name = 'Chávez', city= 'Mérida'
WHERE person_id = 1;

UPDATE people
SET first_name = 'Juan'
WHERE city = 'Mérida';

UPDATE people
SET first_name = 'Juan';
```

DELETE

```
Delete

DELETE FROM people
WHERE person_id = 1;

DELETE FROM people;
```

SELECT

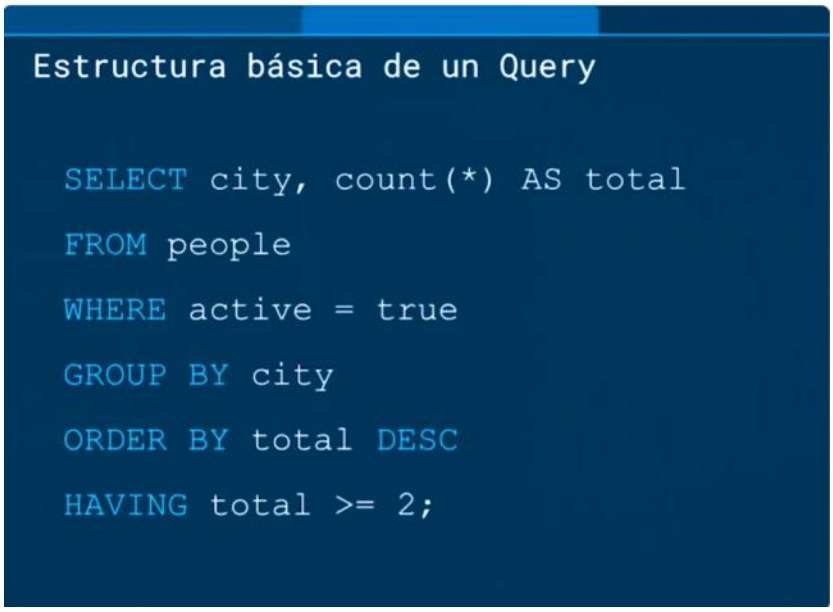
```
Select

SELECT fist_name, last_name
FROM people;
```

–

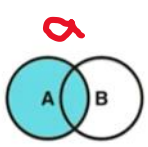
Querys

Saturday, September 16, 2023 6:58 PM

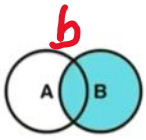


JOIN: Teoría de conjuntos

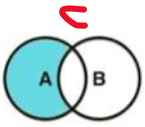
Wednesday, September 20, 2023 5:06 PM



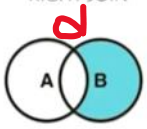
Diferencia



LEFT JOIN



RIGHT JOIN



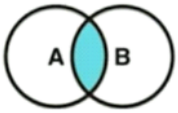
A LEFT JOIN: siginifica que traera todos los datos de la tabla A esten o no esten en la tabla B.

C LEFT JOIN: significa que traera todos los datos de la tabla A pero solo los que no esten asociados a la tabla B.

B RIGHT JOIN: siginifica que traera todos los datos de la tabla B esten o no esten en la tabla A.

D LEFT JOIN: significa que traera todos los datos de la tabla B pero solo los que no esten asociados a la tabla A.

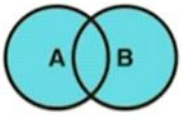
Intersección



Intersección

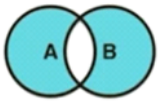
INNER JOIN

INNER JOIN: se refiere a traer las filas de la tabla A que esten asociadas a una fila de la tabla B.



Unión

OUTER JOIN: Traerá todos los registros de ambas tablas sin excepción.



Diferencia simétrica

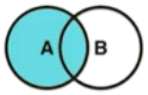
OUTER JOIN: Se traerán todos los registros que no esten asociados entre ambas tablas.

JOIN: práctico

Wednesday, September 20, 2023 5:20 PM

```
SELECT *
FROM usuarios
	LEFT JOIN posts ON usuarios.id = posts.usuario_id;
```

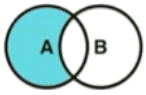
Primero va a traer a todos los usuarios y despues traera todos los post donde haya usuarios ligados a ese post.



LEFT JOIN

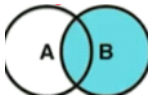
```
SELECT *
FROM usuarios
	LEFT JOIN posts ON usuarios.id = posts.usuario_id
WHERE posts.usuario_id IS NULL;
```

Treremos todos los posts que no estan ligados a un usuario.



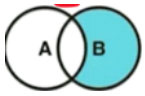

```
SELECT *
FROM usuarios
	RIGHT JOIN posts ON usuarios.id = posts.usuario_id;
```

Primero va a traer a todos los posts y despues traera todos los post donde haya usuarios ligados a ese post o aunque no esten ligados.




```
SELECT *
FROM usuarios
	RIGHT JOIN posts ON usuarios.id = posts.usuario_id
WHERE posts.usuario_id IS NULL;
```

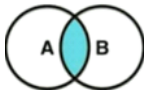
Traer solo los de un lado que no tengan nada del otro.



INNER JOIN

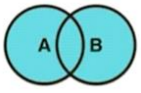
```
SELECT *
FROM usuarios
	INNER JOIN posts ON usuarios.id = posts.usuario_id;
```

Traera solo los datos ligados de ambas tablas.




```
SELECT *
FROM usuarios
	LEFT JOIN posts ON usuarios.id = posts.usuario_id
UNION
SELECT *
FROM usuarios
	RIGHT JOIN posts ON usuarios.id = posts.usuario_id;
```

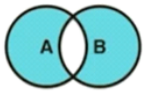
Nos traera todo el universo de ambas tablas.



Unión

```
SELECT *
FROM usuarios
	LEFT JOIN posts ON usuarios.id = posts.usuario_id
WHERE posts.usuario_id IS NULL
UNION
SELECT *
FROM usuarios
	RIGHT JOIN posts ON usuarios.id = posts.usuario_id
WHERE posts.usuario_id IS NULL;
```

Traeremos lo que existe en la Tabla A pero no en la tabla B.



Diferencia simétrica

WHERE

Wednesday, September 20, 2023 11:37 PM

Es la setencia que nos ayuda filtrar que datos deseamos mostrar dependiendo a ciertas condiciones o criterios.

```
SELECT * FROM posts
WHERE id < 50;
```

Traera todos los registros menores a 50.

```
SELECT * FROM posts WHERE titulo LIKE '%hola%'
```

Traera los registros que contengan en el titulo la palabra "hola". Si quitamos el simbolo de porcentaje del final, significa: trae el registro que contenga al final la cadena. Y si se quita al principio seria lo contrario que inicie con esa palabra el parrafo.

```
SELECT * FROM posts
WHERE fecha > '2025-01-01'
```

Traer los registros que sean posteriores a la fecha que se le asigna.

```
SELECT * FROM posts
WHERE fecha BETWEEN '2023-01-01' AND '2025-12-31'
```

```
SELECT * FROM posts
WHERE YEAR(fecha) BETWEEN '2023' AND '2025'
```

Todos los registros entre los años 2023 y 2025.

```
SELECT * FROM posts
WHERE MONTH(fecha) = '04'
```

Traera todod los registros del mes de abril.

WHERE NULO Y NO NULO

```
SELECT * FROM posts WHERE usuario_id IS NULL;
```

Me traera los registros donde los posts tengan el valor nulo en una columna.

```
SELECT * FROM posts WHERE usuario_id IS NOT NULL;
```

Me traera todos los posts donde tengan usuario asignado y los que no tengan los excluira.

```
SELECT *
FROM posts
WHERE usuario_id IS NOT NULL
    AND status = 'activo'
    AND id<50;
```

Me traera todos los posts donde el valor de usuario no sea nulo y ademas el status sea activo y que sean menores que el id 50 y asi puedo seguir concatenando condiciones a mi script.

GROUP BY

Friday, September 22, 2023 1:05 PM

```
SELECT estatus, COUNT(*) post_quantity
FROM posts
GROUP BY estatus;
```

Nos mostrara cuantos registros estan activos y cuantos registros estan inactivos.

```
SELECT YEAR(fecha) AS post_year, COUNT(*) AS post_quantity
FROM posts
GROUP BY post_year;
```

Traera cuantos registros se hicieron por año.

```
SELECT MONTHNAME(fecha) AS post_month, COUNT(*) AS post_quantity
FROM posts
GROUP BY post_month;
```

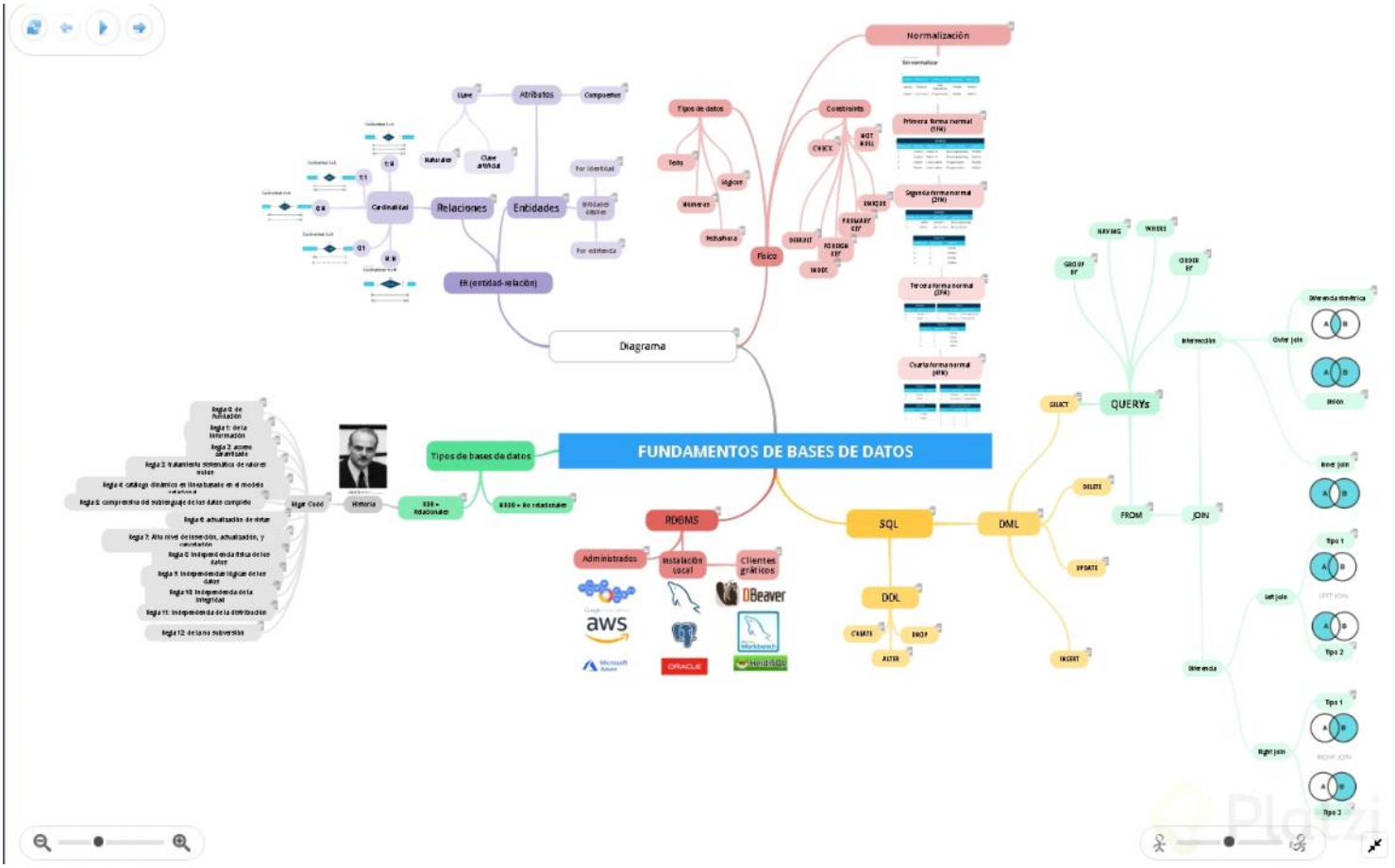
Agrupar por mes.

```
SELECT status, MONTHNAME(fecha) AS post_month, COUNT(*) AS post_quantity
FROM posts
GROUP BY status, post_month;
```

Veremos cuantos posts estan activos e inactivos por mes.

Mapa mental

Friday, September 22, 2023 1:20 PM



Order by, having

Friday, September 22, 2023 1:21 PM

```
SELECT *
FROM posts
ORDER BY fecha DESC;
```

Asi lo va a ordenar de forma descendente.

```
SELECT *
FROM posts
ORDER BY fecha ASC;
```

Asi lo va a ordenar de forma ascendente.

```
SELECT *
FROM posts
ORDER BY titulo;
```

Lo va ordenar de manera alfabetica.

HAVING

```
SELECT MONTHNAME(fecha) AS post_month, status, COUNT(*) AS posts_quantity
FROM posts
GROUP BY estatus, post_month
HAVING posts_quantity > 1
ORDER BY post_month;
```

Convertir a una pregunta a un query

Friday, September 22, 2023 2:21 PM

De pregunta a Query

Lo que quieres mostrar = SELECT

De donde voy a tomar los datos = FROM

Los filtros de los datos que quieres mostrar = WHERE

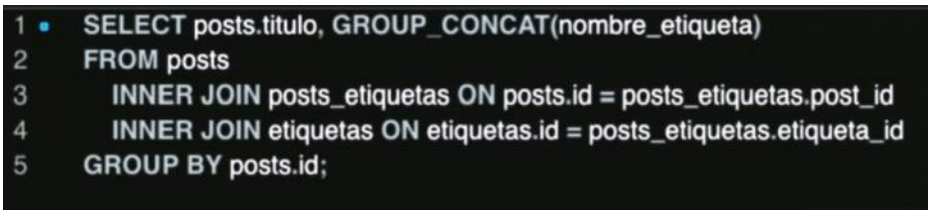
Los rubros por los que me interesa agrupar la información = GROUP BY

El orden en que quiero presentar mi información ORDER BY

Los filtros que quiero que mis datos agrupados tengan HAVING

Scripts

Friday, September 22, 2023 2:38 PM



```
SELECT posts.titulo, GROUP_CONCAT(nombre_etiqueta)
FROM posts
    INNER JOIN posts_etiquetas ON posts.id = posts_etiquetas.post_id
    INNER JOIN etiquetas ON etiquetas.id = posts_etiquetas.etiqueta_id
GROUP BY posts.id;
```

Sirve para saber que etiquetas estan ligados a que post.