**SQL y MYSQL**

**INTRODUCCIÓN Y FUNDAMENTOS DEL LENGUAJE DE BASES DE DATOS: SQL.**

Las **bases de datos** nos sirven para la conservación de los datos y a su vez organizarlos y estructurarlos.

Un **sistema gestor de bases de datos** (**SGBD**) un conjunto de programas que nos permiten la extracción, modificación, ingreso y eliminación de datos.

**¿Qué es un dato?**

Un **dato** es una representación cualitativa o cuantitativa de una variable que indica el valor que se le asigna a una cosa a través de símbolos, números o letras.

Los **programas** y las **aplicaciones** tienen como objetivo procesar datos de los lenguajes de programación.

**Tipos de datos**

**Numéricos**

* **Entero**: dato formado por variable numérica no decimal. 🡪28.
* **Real**: dato formado por variable numérica con parte decimal. 🡪29.78.

**Texto**

* **Caracter**: dato formado por unidad o símbolo (letra/número/mayúscula/signo puntuación). 🡪’L’.
* **Cadena**: dato formado por un conjunto de caracteres y se representa en “ ”. 🡪’Laura’.

**Lógico**

* **Boolean:** dato que representa valor true o false.

**¿Qué es un modelo de datos?**

Es la estructura lógica que determina cómo se almacenan, organizan o acceden a los datos de una base de datos. También determina cómo se manipulan los datos (operaciones con datos).

**Estilos de un modelo de datos**

* Modelo de datos conceptual:

Representa las entidades más importantes y sus relaciones. No se especifican los atributos ni las claves principales.

* Modelo de datos lógicos:

Representa datos organizados. Incluye entidades y sus relaciones, así como sus atributos y la clave principal para cada entidad. También las claves externas y la *normalización*.

* Modelo de datos físicos:

Representa objetos de datos y sus relaciones. Igual a la anterior y sumamos como característica la desnormalización.

En resumen, un **modelo de datos** debe ser ***claro, flexible, estable, independiente y reutilizable***.

**Conceptos generales de una base de datos**

* **Entidad**: todo aquello de lo que interesa guardar datos (**cine**).
* **Atributos**: características de una entidad.
* **Clave primaria:** cada entidad tiene asignada una clave primaria que especifica unívocamente al conjunto de datos (**PK**).
* **Clave foránea:** cuando en una entidad figura la clave primaria de otra entidad, se denomina clave foránea (**FK**):
* **Metadatos**: son datos acerca de los datos de la BD.
* **Tabla**: conjunto de filas y columnas bajo un mismo nombre que representa e conjunto de valores almacenados para una serie de datos.
* **Campos**: cada columna de una tabla.
* **Registros**: cada fila de una tabla.
* **Consulta**: instrucción para hacer una petición a una BD.
* **Índice**: estructura que almacena los campos clave de una tabla para organizarlos y encontrarlos más fácil.
* **Vista**: se obtiene al guardar una consulta de una o varias tablas. Es como una tabla virtual.
* **Informe**: listado ordenado de campos y registros seleccionados.
* **Guiones o scripts:** conjunto de instrucciones que se ejecutan ordenadamente y realizan operaciones y mantienen una BD.
* **Procedimientos**: tipo especial de script almacenada en BD y que forma parte de su esquema.

**Tipos de bases de datos**

* Bases de datos relacionales:

Los datos se almacenan en tablas formadas por filas y columnas.

* Bases de datos orientadas a objetos:

Agrupa la información en paquetes relacionados entre sí.

* Bases de datos distribuidas:

Consta de dos o más archivos ubicados en diferentes sitios.

* Almacenes de datos:

Tipo de BD para consultas.

* Bases de datos NoSQL:

También conocida como bases de datos no relacional almacena y manipula datos no estructurados y semiestructurados.

* Bases de datos orientadas a grafos.

Representan información en vértices y aristas (teoría de grafos).

* Bases de datos OLTP.

Orientadas a transacciones (genera un proceso atómico que puede involucrar operaciones de inserción, modificación y borrado.

**BASES DE DATOS RELACIONALES**

Constan de un conjunto de operaciones que toman como entrada una o más relaciones y dan como resultado una relación. Constan de una **tabla relacional** que es una representación extensional de una relación definida sobre un dominio.

**Operaciones fundamentales**

* *Selección (unitaria).*
* *Proyección (unitaria).*
* *Unión (binaria).*
* *Diferencia de conjuntos (binaria).*
* *Producto cartesiano (binaria).*
* *Renombramiento (unitaria).*

**Tupla**: función que asocia los nombres de los atributos con los valores de una relación. Es una fila de una tabla relacional.

**Operadores**: símbolos que permiten operar con los datos de una BD **(=, ¡= o <>, <, >, <=, >=**).

**SQL**

**SQL** se basa en un conjunto de reglas en inglés llamada sintaxis. Los comandos SQL se escriben a través de una interfaz llamada CLI.

**Elementos SQL**

* **Querys**: una query o consulta recupera datos en base a un criterio dado.
* **Cláusula**: son componentes de los estados y las querys.
* **Comandos**: los comandos o sentencias se usan en SQL para enviar consultas desde un programa cliente a un servidor donde se almacenan las BD.
* **Expresiones**: pueden producir valores escalares o tablas que se basan en columnas y filas de datos.
* **Predicados**: especifican las condiciones que se usan para limitar los efectos de los comandos o consultas o para cambiar el flujo del programa.

**Cláusulas**

Las cláusulas son instrucciones que se le mandan a la BD. Algunas de ellas son:

-SELECT.

-FROM.

-WHERE.

-ORDER BY.

-GROUP BY.

-HAVING.

**Comandos**

Existen dos tipos de comandos:

* **DDL (*Data Definition Language*)** permite crear y definir nuevas BD, campos e índices.

-CREATE (crea nuevas tablas, campos o índices).

-DROP (elimina tablas e índices).

-ALTER (modifica tablas agregando campos o combinando los mismos).

* **DML (*Data Manipulation Language*)** manipula datos insertando, eliminando y actualizando. También puede generar consultas para ordenar, filtrar y extraer datos de un BD.

-INSERT (inserta datos).

-UPDATE (actualiza datos).

-DELETE (borra datos).

**CREAR BASE DE DATOS**

Para ello usamos la sintaxis:

**CREATE DATABASE nombredatabase;**

**CREATE DATABASE laura;**

**SELECCIONAR BASE DE DATOS**

Para seleccionar datos de una BD:

**SELECT \*FROM DATABASE;**

**SELECT\*FROM Laura;**

También podemos seleccionar solo una columna:

**SELECT** nombre **FROM** clientes;

**USAR BASE DE DATOS**

**USE DATABASE nombredatabase;**

**USE DATABASE Laura;**

**CREAR TABLA**

**CREATE TABLE nombretabla (datos a almacenar);**

**Créate table Laura (id nombre varchar(30), edad int, sueldo decimal);**

**INSERTAR DATOS**

**INSERT INTO nombretabla (registros) VALUES (campos);**

**INSERT INTO Laura (nombre, edad, sueldo) VALUES (‘lAURA’, 28, 1200.34);**

**WHERE**

Nos ayuda a situarnos en un campo específico de una columna.

**SELECT** nombre **FROM** clientes **WHERE** name=’Laura’;

**DISTINCT**

Se trata de un complemento de SELECT y sirve para ver los datos que no se repiten en una BD.

**SELECT DISTINCT** (columna a filtrar) **FROM** database;

**SELECT DISTINCT** (código) **FROM** clientes;

**SELECT DISTINCT** (id), **DISTINCT** (código) **FROM** clientes;

**ORDER BY**

Sirve para ordenar de forma ascendente (**ASC**) o descendente (**DESC**) un campo en una tabla.

**ORDER BY** campo **ASC/DESC;**

**SELECT \* FROM** clientes

**ORDER BY** edad **ASC;**

**OPERADORES BOOLEANOS**

Estos son operadores como **AND**, **OR** Y **NOT**.

**SELECT** \* **FROM** clientes **WHERE** nombre=’Laura’ **OR** apellidos=’García’;

**SELECT** \* **FROM** clientes **WHERE** nombre=’Laura’ **AND** apellidos=’García’;

**SELECT** \* **FROM** clientes **WHERE** **NOT** nombre=’Laura’;

**SELECT** \* **FROM** mundo **WHERE** continente=?Europa’ **NOT** región=’Suroeste’;

**SENTENCIA IN**

Esta sentencia es similar a OR, pero nos permite acortar el código.

**SELECT\* FROM** clientes **WHERE** apellidos **IN** (‘García’, ‘Lopez’);

**SELECT\* FROM** clientes **WHERE NOT** apellidos **IN** (‘García’, ‘Lopez’);

Esto es lo mismo que decir:

**SELECT\* FROM** cleientes **WHERE** apellido=’Garcia’ **AND/OR** apellido=’lopez’;

**SENTENCIA BETWEEN**

Se trata de una sentencia que nos permite filtrar con un segmento que no necesariamente es un número.

**USE** clientes;

**SELECT \* FROM** clientes

**WHERE** población **BETWEEN** 1 **AND** 10000000

**ORDER BY** población **DESC**;

**USE** clientes;

**SELECT \* FROM** clientes

**WHERE** población **NOT** **BETWEEN** 1 **AND** 10000000

**ORDER BY** población **DESC**;

**SENTENCIA LIKE**

Dicha sentencia nos permite filtrar a través de caracteres.

🡪’L%’ (empieza por L).

🡪’%L’ (acaba en L).

🡪’%L%’ (contenga L).

**🡪 ’---‘**(con 3 caracteres).

**🡪 ‘-----‘**(con 5 caracteres).

🡪‘Leo-\_’ (que empiecen con leo y siguiente carácter como por ejemplo león).

🡪 ‘ p\_ma’ (como puede ser puma).

**SELECT** \* **FROM** clientes

**WHERE** nombre **LIKE** ‘L%’;

**CONCATENAR Y SUM**

Esta sentencia sirve para concatenar.

**SELECT** nombre **+** apellidos **FROM** clientes;

Esta sentencia sirve para sumar datos de ese campo.

**SELECT SUM** (nombre) **FROM** clientes;

**MIN, MAX, AVG**

**MIN** nos calcular el valor mínimo.

**MAX** nos calcula el máximo valor.

**AVG** calcula la media.

**SELECT MAX** (Edad) **FROM** clientes;

**SELECT MIN** (Edad) **FROM** clientes;

**SELECT AVG** (Edad) **FROM** clientes;

**GROUP BY**

Esta sentencia sirve para agrupar campos.

**SELECT** lenguaje, **COUNT** (\*) FROM clientes

**GROUP BY** lenguaje;

**FUNCIONALIDADES EXTRA**

Un ejemplo sería pasar datos de un campo o registro a mayúscula o minúscula con **UCASE** (UPPERCASE) o **LCASE** (LOWERCASE) y **CONCAT y CONCAT\_WS** para concatenar. Y **ROUND** para redondear. Y **CHAR\_LENGTH** para ver longitud.

**SELECT UCASE** (name) **FROM** clientes; ***mayúsculas***

**SELECT LCASE** (name) **FROM** clientes; ***minúsculas***

**SELECT** name, apellido, **CONCAT** (nombre, “ ”, apellido) **FROM** clientes;

**SELECT** name, apellido, **CONCAT\_WS** (“ “, apellido, nombre) **FROM** clientes; ***ver el caracter que se repite por vuelta***

**SELECT** esperanzavida, **ROUND** (esperanzavida, 0) **FROM** pacientes;

**SELECT** continente, **CHAR\_LENGTH** (continente) **FROM** clientes;

**SENTENCIA CASE**

Nos permite agregar una condición.

**SELECT** porcentaje,

**CASE**

**WHEN** porcentaje < 10 **THEN** “Es menor a 10”

**WHEN** porcentaje > 10 **THEN** “Es mayor a 10”

**END**

**FROM** clientes**;**

**UPDATE**

Dicha sentencia nos permite actualizar una tabla.

**UPDATE** clientes **SET** código=”12” **WHERE** código=”13”;

**ALTER TABLE**

Dicha sentencia nos permite modificar una tabla.

**ALTER TABLE** clientes

**ADD COLUMN** país **VARCHAR** (30) **NOT** **NULL**;

**DROP, DELETE Y TRUNCATE**

Estas dos sentencias son muy parecidas, la diferencia es que **DROP** borra un campo o registro, pero no la estructura de la tabla y **TRUNCATE** toda la tabla y la estructura tbn.

**DELETE** borra algo concreto.

**DELETE código=”12” FROM** clientes**;**

**DROP COLUMN** país;

**DROP TABLE** clientes;

**TRUNCATE** clientes;

**FUNDAMENTOS DE MYSQL**

**¿Qué es la normalización?**

La **normalización** es un proceso por el cual los atributos de una tabla son reorganizados para evitar su repetición, ya que su objeto principal es evitar la redundancia de datos.

**Formas de normalización**

* **Primera fase**: ver qué datos o campos se repiten innecesariamente.
* **Segunda fase**: separa los datos que se repiten y estudia la relación que existe entre ellos. Es decir, referenciamos los datos redundantes.
* **Tercera fase**: ver la relación que tienen entre sí las tablas.

Tras descargarnos **MYSQL workbench** creamos una base de datos y en esta cargaremos una tabla y datos. Podemos hacerlo mediante comandos en una interfaz o con el propio programa que ofrece MYSQL workbench.

**RESUMEN**

1. **CREATE DATABASE** tienda;
2. **USE** tienda;
3. **CREATE TABLE** clientes (

Nombre VARCHAR (30),

Apellido VARCHAR (30),

Edad int

);

1. **EXEC** sp\_tables/columns/etc tienda;
2. **INSERT INTO** clientes (nombre, apellido, edad)

**VALUES** (‘Laura’, ‘Rodríguez’, 27);

1. **SELECT \* FROM** clientes;
2. **SELECT** nombre, apellido **FROM** clientes**;**
3. **SELECT** nombre **FROM** clientes **WHERE** nombre=’Laura’;
4. **SELECT** edad **FROM** clientes **WHERE** edad <> 23;
5. **SELECT** nombre **AS** nombrecito **FROM** clientes **WHERE** nombre **LIKE** ‘La%’;
6. **ALTER TABLE** clientes **CHANGE** nombre name **VARCHAR** (20); Para cambiar datos o estructura de una tabla (REGISTROS)
7. **ALTER TABLE** clientes **RENAME** clientela; Para cambiar nombre de una tabla
8. **ALTER TABLE** clientes **ADD** domicilio **VARCHAR** (30); Para añadir columna
9. **ALTER TABLE** clientes **ADD** **COLUMN** teléfono **INT**;
10. **UPDATE** clientes **SET** nombre=’Laurita’ **WHERE** nombre=’Laura’;
11. **DROP TABLE** clientes;

**INNER JOIN**

Sirve para unir dos tablas mediante sus llaves foráneas y primarias.

**USE** combinaciones;

**SELECT** \* **FROM** tabla1

**INNER** **JOIN** tabla2

**ON** tabla1.idtabla1 = tabla2.tabla1.

**USE** combinaciones2;

**SELECT \* FROM** jugador

**INNER JOIN** afiliadosç

**ON** jugador.idjugador = afiliados.Njugador;

**EJERCICIO 1**

1. **Crear una base de datos (El estudiante pondrá el nombre)**

CREATE DATABASE Laura;

**2- Crear dos tablas, una para Productos y otra para Clientes**

**CREATE TABLE** clientes (

Id\_cliente **INT** **NOT NULL AUTO\_INCREMENT**,

Nombre\_cliente **VARCHAR** (30) **NOT NULL**,

Documento\_cliente **INT NOT** **NULL**

**PRIMARY KEY** (‘id\_cliente’)

);

**CREATE TABLE** productos (

Idproducto **INT NOT NULL AUTO\_INCREMENT**,

Nombre\_Productos **VARCHAR** (30) **NOT NULL**,

Precio **DECIMAL** (42) **NOT NULL**,

Fecha\_ingreso **DATE NOT NULL**,

Descripción **VARCHAR** (50) **NOT NULL,**

**PRIMARY KEY** (id\_productos’)

);

1. **La tabla productos tendrá los siguientes datos (idproducto, NombreProductos, Precio,Fecha\_Ingreso, Descripción)**

**CREATE TABLE** productos (

Idproducto **INT NOT NULL AUTO\_INCREMENT**,

Nombre\_Productos **VARCHAR** (30) **NOT NULL**,

Precio **DECIMAL** (42) **NOT NULL**,

Fecha\_ingreso **DATE NOT NULL**,

Descripción **VARCHAR** (50) **NOT NULL,**

**PRIMARY KEY** (id\_productos’)

);

**y la tabla Clientes tendrá los siguientes datos (Id\_Cliente, Nombre\_Cliente, Documento\_Cliente)**

**CREATE TABLE** clientes (

Id\_cliente **INT** **NOT NULL AUTO\_INCREMENT**,

Nombre\_cliente **VARCHAR** (30) **NOT NULL**,

Documento\_cliente **INT NOT** **NULL**

**PRIMARY KEY** (‘id\_cliente’)

);

1. **Hacer una selección de todos los datos de la tabla Clientes**

**USE** Laura**;**

**SELECT \* FROM** clientes**;**

1. **Hacer una proyección de Nombre\_Productos y Nombre\_Cliente siempre y cuando su ID sea mayor a 1**

**SELECT** clientes.Nombre\_cliente, producto. Nombre\_productos **FROM** clientes, productos **WHERE** clientes.id\_cliente >= 1 **AND** productos.id\_productos >=1;

**6- Hacer una selección de todos los datos de ambas tablas**

**SELECT \* FROM** clientes**,** productos**;**

**MODELO ENTIDAD-RELACIÓN**

**¿Qué es el modelo de bases de datos?**

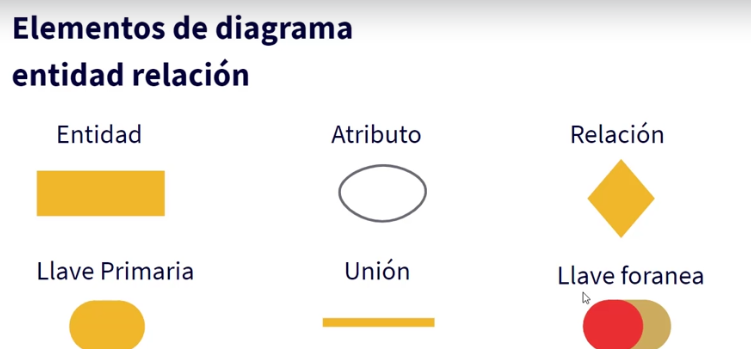
Son un conjunto de ideas lógicas que representan la estructura de una base de datos.

**¿Qué es el modelo relacional?**

Es un modelo que nos permite visualizar la estructura de esta base de datos.

**Diagrama entidad-relación**

Se trata de un diagrama que modela nuestra base de datos mediante un análisis y una representación gráfica.



**CARDINALIDAD**

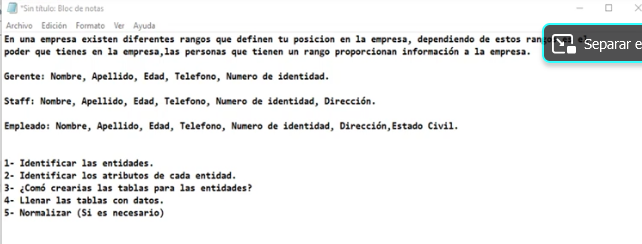
La **cardinalidad** es el número de entidades con la cual otra entidad se puede asociar mediante una relación binaria.

Puede ser:

* 1:1
* 1:M
* M:M

**Llave foránea**

Una **llave foránea** es una columna o un conjunto de columnas en una tabla cuyos valores se corresponden con las claves o llaves primarias de las tablas o entidades con las que está relacionada.



**CREATE DATABASE** tarea;

**USE** tarea;

**CREATE TABLE** gerente (

Id\_gerente **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_Gerente **VARCHAR** (30) **NOT NULL**,

Apellido\_Gerente **VARCHAR** (30) **NOT** **NULL**,

Edad\_Gerente **INT** (3) **NOT NULL**,

Telefono\_Gerente **INT** (8) **NOT NULL**,

N\_Identidad **VARCHAR** (8) **NOT NULL**,

**PRIMARY KEY** (‘Id\_Gerente)

);

**CREATE TABLE** Empleado (

Id\_Empleado **INT** (11) **NOT NULL AUTO\_INCREMENT,**

Nombre\_Empleado **VARCHAR** (40) **NOT NULL,**

Apellido\_Empleado **VARCHAR** (40) **NOT NULL,**

Edad **INT** (3) **NOT NULL,**

Telefono\_Empleado **INT** (8) **NOT NULL,**

N\_Identidad **INT** (8) **NOT NULL**,

Direccion\_Empleado **VARCHAR** (40) **NOT NULL,**

Estado\_Civil **VARCHAR** (10) **NOT NULL**

**PRIMARY KEY** (‘Id\_Empleado)

);

**CREATE TABLE** Staff (

Id\_Staff **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_Staff **VARCHAR** (40) **NOT NULL,**

Apellido\_Staff **VARCHAR** (40) **NOT NULL,**

Edad\_Staff **INT** (3) **NOT NULL,**

Telefono\_Staff **INT** (8) **NOT NULL,**

Direccion\_Staff **VARCHAR** (40) **NOT NULL**

**PRIMARY KEY** (‘Id\_Staff)

);

**INSERT INTO** Staff (Nombre\_Staff, Apellido\_Staff, Edad\_Staff, Telefono\_Staff, Direccion\_Staff) **VALUES** (‘Isidoro’, ’García’, 34, ‘567 567 8’, ‘Ochoa’);

**INSERT INTO** Empleado (Nombre\_ Empleado, Apellido\_ Empleado, Edad\_ Empleado, Telefono\_ Empleado, Direccion\_ Empleado) **VALUES** (‘Isidoro’, ’García’, 34, ‘567 567 8’, ‘Ochoa’);

**INSERT INTO** Gerente (Nombre\_Gerente, Apellido\_ Gerente, Edad\_ Gerente, Telefono\_ Gerente, Direccion\_ Gerente) **VALUES** (‘Isidoro’, ’García’, 34, ‘567 567 8’, ‘Ochoa’);

**RUTINAS DE ALMACENAMIENTO**

Las **rutinas de almacenamiento** son un conjunto de instrucciones y procedimientos en MYSQL que simplifican la creación de código. El programador ha de especificar dichas instrucciones en el momento de llamar a la rutina (insertar datos, eliminar datos, etc).

**USE** ‘ferreteria’;

**DROP** procedure **IF** **EXISTS** ‘Dragon’;

**DELIMITER**

**$$USE** ‘ferreteria’

**$$CREATE PROCEDURE** ‘Dragon’ ()

**BEGINSELECT\*FROM** clientes;

**END$$DEMILITER;**

**EJERCICIO 2**

🡪El usuario deberá crear una base de datos

**CREATE DATABASE** ejemplo**;**

🡪Dentro de esta deberá tener 3 tablas, los datos que contendrán las tablas serán a elección del estudiante.

**USE** ejemplo**;**

**CREATE TABLE** tabla1 **(**

Id\_tabla1 **INT NOT NULL AUTO\_INCREMENT;**

Dato1 **VARCHAR (**30**) NOT NULL,**

Dato2 **VARCHAR (**30**) NOT NULL,**

Dato3 **VARCHAR (**30**) NOT NULL,**

**PRIMARY KEY** (‘Id\_tabla1’)

**) ;**

**CREATE TABLE** tabla2 **(**

Id\_tabla2 **INT NOT NULL AUTO\_INCREMENT;**

Dato1 **VARCHAR (**30**) NOT NULL,**

Dato2 **VARCHAR (**30**) NOT NULL,**

Dato3 **VARCHAR (**30**) NOT NULL,**

**PRIMARY KEY** (‘Id\_tabla2’)

**) ;**

**CREATE TABLE** tabla3 **(**

Id\_tabla3 **INT NOT NULL AUTO\_INCREMENT;**

Dato1 **VARCHAR (**30**) NOT NULL,**

Dato2 **VARCHAR (**30**) NOT NULL,**

Dato3 **VARCHAR (**30**) NOT NULL,**

**PRIMARY KEY** (‘Id\_tabla3’)

**) ;**

🡪Deberá crear tres consultas almacenadas llamando a los datos de la siguiente manera:

1. Datos de tabla 1 y 2.

**CREATE PROCEDURE ‘**Datos\_1y2’ ()

**BEGIN**

**SELECT\*FROM** tabla1, tabla2;

**END**

1. Datos de tabla 2 y 3.

**CREATE PROCEDURE ‘**Datos\_2y3’ ()

**BEGIN**

**SELECT\*FROM** tabla2, tabla3;

**END**

3- Datos de tabla 1, 2 y 3.

**CREATE PROCEDURE ‘**Datos\_1,2y3’ ()

**BEGIN**

**SELECT\*FROM** tabla1, tabla2, tabla3;

**END**

**CALL** datos\_2y3; etc etc

**TRIGGER (DISPARADORES)**

Los **triggers** o **disparadores** nos ayuda a ejecutar eventos en el gestor de base de datos ejecutando ciertas reglas, instrucciones o procedimientos. El programador asigna una serie de instrucciones que servirán como reglas que se iniciarán cada vez que ocurras una acción en el **GBD**.

La sintaxis:

**USE** tienda;

**DELIMITER//**

**CREATE TRIGGER** tabla:productos after inserto n productos for each row begin

**INSERT INTO** regis (IdRegis, Nombre\_Registro) VALUES (NEW.idProductos, NEW.Nombre\_producto);

**End//**

**DELIMILER;**

**PRÁCTICA1**

**CREATE SCHEMA** Registro;

o

**CREATE DATABASE** Registro;

**CREATE TABLE** Persona (

Id\_persona **INT NOT NULL AUTO\_INCREMENT,**

Nombre **VARCHAR** (30) **NOT NULL,**

Usuario\_Guardado **VARCHAR** (30) **NOT NULL,**

Contraseña\_Guardado **VARCHAR** (30) **NOT NULL,**

Fecha **DATE** **NOT NULL,**

**PRIMARY KEY** (‘Id\_persona’)

);

**CREATE TABLE** Users (

Id\_users **INT NOT NULL AUTO\_INCREMENT,**

Usuario **VARCHAR** (30) **NOT NULL,**

Contraseña **VARCHAR** (30) **NOT NULL,**

**PRIMARY KEY** (‘Id\_users)

);

**USE** Registro;

**DELIMITER//**

**CREATE TRIGGER** Tabla\_persona **AFTER INSERT ON** persona

**For each row begin**

**INSERT INTO** users (idUsuario, Usuarios, Contraseña) VALUES (**NEW**.IdPerdona, **NEW**.Usuario\_Guardado, **NEW**.Contraseña\_Guardado);

**End//**

**DELIMITER;**

**DROP TRIGGER** Tabla\_Persona;

**VISTAS**

Las **vistas** o **tablas virtuales** almacenan instrucciones para mostrar datos al usuario. Se usan para tener una opción más de hacer selecciones de datos y mostrarlas en pantalla, vendrían siendo igual que las rutinas almacenadas solos que estas tienen algunas limitaciones.

Sintaxis:

**CREATE**

**ALGORITHM** = **UNDEFINED**

**DEFINER** = ‘root’ @ ‘localhost’

**SQL SECURITY DEFINER**

**VIEW** ‘escuela’ . ‘new\_view’ AS

**SELECT**

‘escuela’ . ’publico’ . ’NIE’ AS ‘NIE’,

‘escuela’ . ’publico’ . ‘Nombre\_Estudiante’ AS ‘Nombre\_Estudiante’,

‘escuela’ . ’publico’ . ‘Grado’ as ‘Grado’,

‘escuela’ . ’publico’ . ‘Edad’ AS ‘Edad’

**FROM**

‘escuela’ . ‘publico’

**TRANSACCIONES**

Las **transacciones** son un conjunto de instrucciones las cuales se ejecutan para cumplir un funcionamiento, estas instrucciones deben cumplirse en su totalidad de lo contrario la transacción no se ejecutará.

**¿Cómo funcionan?**

Funcionan siguiendo el término ‘**ACID’** que está formado por 4 propiedades:

* **Atomicidad** (Todo o nada).
* **Consistencia** (Como el principio el final).
* **Aislamiento** (No interferir).
* **Durabilidad** (Permanente).

**CREATE TABLE** productos ( …….);

**SELECT\*FROM** **super**.productos;

**USE** super;

**EJERCICIO**

Creamos bd y tabla clientes;

**USE** transaction;

**START TRANSACTION;**

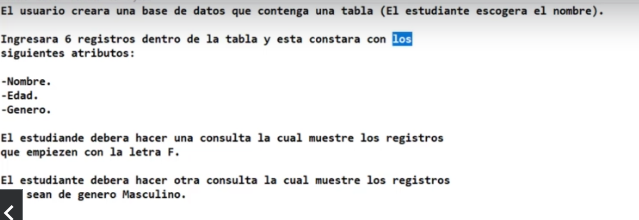
**INSERT INTO** clientes (Nombre\_cliente, Description) VALUES (‘Jorge’ ‘Estudiante’);

**INSERT INTO** clientes (Nombre\_cliente, Description) VALUES (‘Jeniffer’ ‘Estudiante’);

**ROLLBACK;**

**COMMIT;**

**TRIPLE EJERCICIO**



**CREATE SCHEMA** practica1;

**USE** practica1;

**CREATE TABLA** personal (

Idpersonal **INT NOT NULL AUTOINCREMENT,**

Nombre **VARCHAR** (30) **NOT NULL,**

Edad **INT** (3) **NOT NULL,**

Genero **VARCHAR** (45) **NOT NULL,**

**PRIMARY KEY** (‘idpersonal’));

**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Laura’, 23, ‘Femenino’);

**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Carlos, 34, ‘Masculino);

**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Isidora, 45, ‘Femenino’);

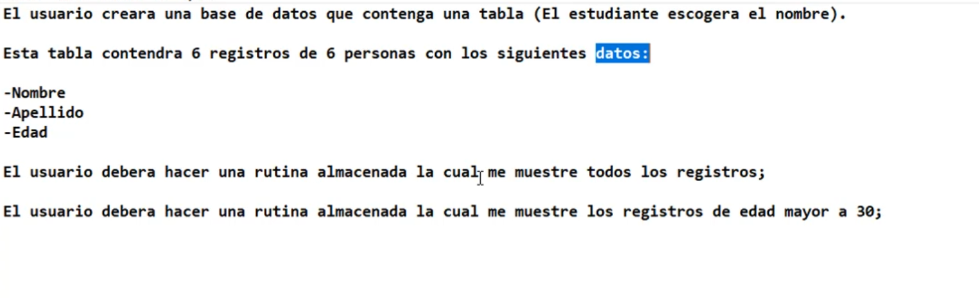
**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Lucia, 45, ‘Femenino’);

**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Veli, 67, ‘Femenino’);

**INSERT INTO** personal (Nombre, Edad, Genero) **VALUES** (‘Gloria, 83, ‘Femenino’);

**SELECT \* FROM** personal **WHERE** Nombre **LIKE** ‘F%’;

**SELECT \* FROM** personal **WHERE** Genero = ‘Masculino;



**CREATE SCHEMA** practica2;

**USE** practica2;

**CREATE TABLA** personal2 (

Idpersonal2 **INT NOT NULL AUTOINCREMENT,**

Nombre **VARCHAR** (30) **NOT NULL,**

Edad **INT** (3) **NOT NULL,**

Genero **VARCHAR** (45) **NOT NULL,**

**PRIMARY KEY** (‘idpersonal2’));

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Laura’, 23, ‘Femenino’);

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Carlos, 34, ‘Masculino);

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Isidora, 45, ‘Femenino’);

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Lucia, 45, ‘Femenino’);

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Veli, 67, ‘Femenino’);

**INSERT INTO** personal2 (Nombre, Edad, Genero) **VALUES** (‘Gloria, 83, ‘Femenino’);

**CREATE PROCEDURE** ‘general’ ()

**BEGIN**

**SELECT\*FROM** personal2;

**END**

**USE** practica2**;**

**CALL** general**;**

**CREATE PROCEDURE** ‘mayor30’ ()

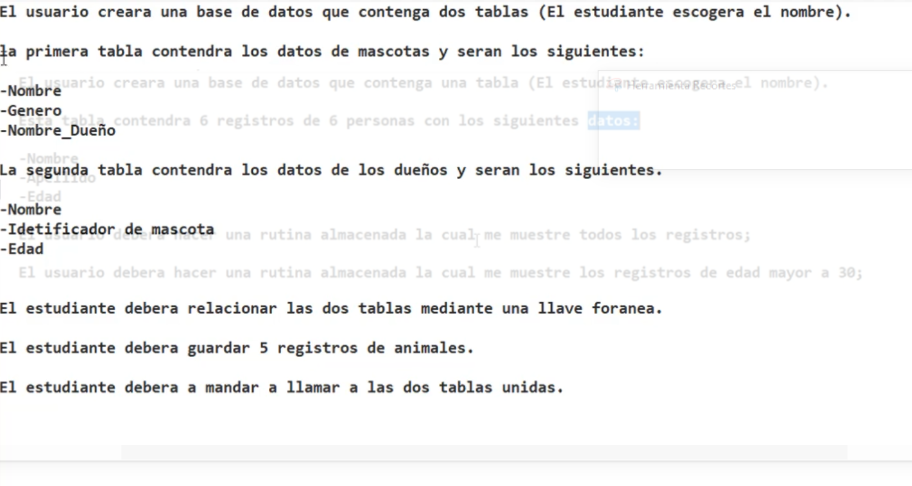
**BEGIN**

**SELECT\*FROM** personal2 **WHERE** edad >30;

**END**

**USE** practica2**;**

**CALL** mayor30**;**



**CREATE SCHEMA** veterinaria;

**USE** veterinaria;

**CREATE TABLE** mascotas (

Id\_mascotas **INT NOT NULL AUTO\_INCREMENT,**

Nombre **VARCHAR** (30) **NOT NULL**,

Genero **VARCHAR** (30) **NOT NULL**,

Nombre\_Dueño **VARCHAR** (30) **NOT NULL**,

**PRIMARY KEY** (‘Id\_mascotas’) ) ;

**CREATE TABLE** dueños (

Id\_dueños **INT NOT NULL AUTO\_INCREMENT,**

Nombre **VARCHAR** (30) **NOT NULL,**

Id\_mascota **INT NOT NULL,**

Edad **INT** (2) **NOT NULL,**

**PRIMARY KEY** (‘Id\_dueños’),

**FOREING KEY** (‘Id\_mascota’) );

**INSERT INTO** mascotas (Nombre, Genero, Nombre\_dueño) **VALUES** (‘Lolo’, ‘Femenino’, ‘Paco’);

**INSERT INTO** mascotas (Nombre, Genero, Nombre\_dueño) **VALUES** (‘Lolo’, ‘Femenino’, ‘Paco’);

**INSERT INTO** mascotas (Nombre, Genero, Nombre\_dueño) **VALUES** (‘Lolo’, ‘Femenino’, ‘Paco’);

**INSERT INTO** mascotas (Nombre, Genero, Nombre\_dueño) **VALUES** (‘Lolo’, ‘Femenino’, ‘Paco’);

**INSERT INTO** mascotas (Nombre, Genero, Nombre\_dueño) **VALUES** (‘Lolo’, ‘Femenino’, ‘Paco’);

**INSERT INTO** dueños (Nombre, Edad) **VALUES** (‘Laura’, 23);

**INSERT INTO** dueños (Nombre, Edad) **VALUES** (‘Laura’, 23);

**INSERT INTO** dueños (Nombre, Edad) **VALUES** (‘Laura’, 23);

**INSERT INTO** dueños (Nombre, Edad) **VALUES** (‘Laura’, 23);

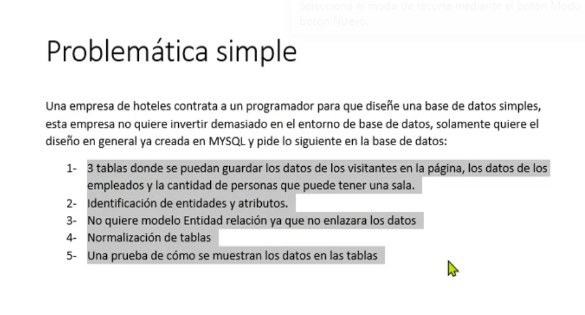
**INSERT INTO** dueños (Nombre, Edad) **VALUES** (‘Laura’, 23);

**USE** veterinaria;

**SELECT\*FROM** mascotas **INNER** **JOIN** dueños

**ON** mascotas.id\_mascotas = dueños.Imascotas;

**CASO PRÁCTICO DE CREACIÓN DE BD**



**CREATE SCHEMA** anonimo;

**CREATE TABLE** visitantes (

Id\_visitantes **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_v **VARCHAR** (45) **NOT NULL,**

Fecha **DATE NOT NULL**,

**PRIMARY KEY** (‘Id\_visitantes’) );

**CREATE TABLE** empleados (

Id\_empleados **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_e **VARCHAR** (45) **NOT NULL,**

Apellido\_e **VARCHAR** (45) **NOT NULL,**

Edad\_e **INT** (3) **NOT NULL**

Fecha **DATE NOT NULL**,

**PRIMARY KEY** (‘Id\_empleados’) );

**CREATE TABLE** sala (

Id\_sala **INT NOT NULL AUTO\_INCREMENT,**

Sala1 **INT (3) NOT NULL**,

Sala2 **INT (3) NOT NULL**,

Sala3 **INT (3) NOT NULL**,

**PRIMARY KEY** (‘Id\_sala’) );

**USE** anonimo;

**SELECT\* FROM** empleados, salas, visitantes;

**INSERT INTO** visitantes (Nombre\_v, Fecha) **VALUES** (‘Lura’, ‘2020/12/13’);

**INSERT INTO** visitantes (Nombre\_v, Fecha) **VALUES** (‘Isi, ‘2020/12/14’);

**INSERT INTO** visitantes (Nombre\_v, Fecha) **VALUES** (‘Gloria, ‘2020/12/15’);

**INSERT INTO** visitantes (Nombre\_v, Fecha) **VALUES** (‘Maria, ‘2020/12/16’);

Así con el resto de las tablas.

**FUNCIONES AVANZADAS EN MYSQL**

**Incrementar fechas y utilizarlas**

**CREATE SCHEMA** avanzada;

**CREATE TABLE** productos (

idProductos **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_Producto **VARCHAR** (45) **NOT NULL**,

Precio\_Producto **DECIMAL** (40) **NOT NULL**,

Fecha\_Garantia **DATE** (45) **NOT NULL,**

**PRIMARY KEY (‘**idProductos’**));**

**USE** avanzada**;**

**INSERT INTO** productos (Nombre\_Producto, Precio\_Producto, Fecha\_Garantia) **VALUES** (‘Lampara’, 12.34, ‘02/12/2020’);

**INSERT INTO** productos (Nombre\_Producto, Precio\_Producto, Fecha\_Garantia) **VALUES** (‘Lampara’, 12.34, ‘02/12/2020’);

**INSERT INTO** productos (Nombre\_Producto, Precio\_Producto, Fecha\_Garantia) **VALUES** (‘Lampara’, 12.34, ‘02/12/2020’);

**SELECT** Fecha\_Garantia **AS** ‘Fecha Ingreso’, **ADDDATE** (Fecha\_Garantia, 90) **AS** ‘Fceha de expiración’ **FROM** productos; 🡪 **le sumamos 90 a la fecha**

**Encriptación de datos**

**CREATE SCHEMA** avanzada1;

**CREATE TABLE** usuarios (

idUsuarios **INT NOT NULL AUTO\_INCREMENT,**

Nombre\_Usuario **VARCHAR** (45) **NOT NULL,**

Contraseña\_Usuario **VARCHAR** (35) **NOT NULL,**

**PRIMARY KEY** (‘idUsuarios’));

**USE** avanzada1;

**INSERT INTO** usuarios (Nombre\_Usuario, Contraseña\_Usuario) **VALUES** (‘Laura’, ‘deknfjhejfjh’);

**INSERT INTO** usuarios (Nombre\_Usuario, Contraseña\_Usuario) **VALUES** (‘Laura’, ‘deknfjhejfjh’);

**INSERT INTO** usuarios (Nombre\_Usuario, Contraseña\_Usuario) **VALUES** (‘Laura’, ‘deknfjhejfjh’);

**SELECT\*FROM** usuarios;

**SELECT** Contraseña\_Usuario **AS** ‘Contraseña’**, AES\_ENCRYPT** (Contraseña\_Usuario, ‘Base de datos’) **AS** ‘Encriptado’ **FROM** usuarios;🡪**PARA ENCRIPTAR DATOS**

**Conteo de caracteres**

**SELECT** Contraseña\_Usuario **AS** ‘Contraseña’**, COMPRESS** (Contraseña\_Usuario) **AS** ‘Encriptado’ **FROM** usuarios;🡪**PARA comprimir CARACTERES**

**SELECT** Contraseña\_Usuario **AS** ‘Contraseña’**, UNCOMPRESS** (Contraseña\_Usuario) **AS** ‘Encriptado’ **FROM** usuarios;🡪**PARA descomprimir CARACTERES**

**SELECT** Nombre\_Usuario **AS** ‘Usuarios’ **LENGTH** (Nombre\_Usuario) **AS** ‘Carcateres’ **FROM** usuarios;🡪**PARA CONTAR CARACTERES**

**Contando datos**

**SELECT CONCAT (**Nombre\_Empleado, Apellido\_Empleado**) AS** ‘Nombre\_Empleado’ **FROM empleado;🡪**Para contar datos