

# **BASE DE DATOS 2**

**ESTUDIANTE: ISRAEL ALEJANDRO ZAMBRANA**

**CARRERA: INGENIERÍA DE SISTEMAS**

**SEDE: EL ALTO**

**PARALELO: BDA (1)**

**DOCENTE: LIC. WILLIAM BARRA PAREDES**

# MANEJO DE CONCEPTOS:

1. Defina que es lenguaje procedural en MySQL.

R. Los procedimientos almacenados **MySQL** contienen una o más instrucciones SQL además de un procesamiento manipulador o lógico. La característica fundamental de los procedimientos almacenados **MySQL** es que estos comandos se quedan almacenados y se ejecutan en el servidor o en el motor de bases de datos.

2. Defina que es una FUNCION en MySQL.

R. Las **funciones EN MYSQL** son piezas de código que reciben datos de entrada, realizan operaciones con ellos y luego devuelven un resultado.

3.Cuál es la diferencia entre funciones y procedimientos almacenados.

R. Cuando llama al procedimiento almacenado, se debe especificar que es un parámetro externo. Una ventaja de los procedimientos almacenados es que puede obtener varios parámetros mientras que, en las **funciones**, solo se puede devolver una variable (función escalar) o una tabla (**funciones** con valores de tabla)

4. Cómo se ejecuta una función y un procedimiento almacenado.

R **Triggers, procedimientos y funciones en MySQL**

**Procedimiento almacenado:** Es un objeto que **se** crea con la sentencia CREATE PROCEDURE y **se** invoca con la sentencia CALL . ...

**Función almacenada:** Es un objeto que **se** crea con la sentencia CREATE **FUNCTION** y **se** invoca con la sentencia SELECT o dentro de una expresión



5. Defina que es una TRIGGER en MySQL.

R. El **trigger MySQL** es un objeto de la base de datos que está asociado con una tabla. Se activará cuando una acción definida se ejecute en la tabla. El **trigger** puede usarse para ejecutar una de las siguientes sentencias **MySQL** en la tabla: INSERT, UPDATE y DELETE. Se puede invocar antes o después del evento.

6. En un trigger que papel juega las variables OLD y NEW

R. El valor de ":new" NO puede modificarse en un trigger after, esto es porque el trigger se activa luego que los valores de "new" se almacenaron en la tabla.

El campo ":old" nunca se modifica, sólo puede leerse.

7. En un trigger que papel juega los conceptos(cláusulas) BEFORE o AFTER

R. El modificador **BEFORE o AFTER** indica que el **trigger** se ejecutará antes o después de ejecutarse la sentencia SQL definida por DELETE , INSERT

8. A que se refiere cuando se habla de eventos en TRIGGERS

R. Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador

# Parte practica

## 9. Crear la siguiente Base de datos y sus registros.

```
CREATE DATABASE Practica_procesual_hito_4;  
USE Practica_procesual_hito_4;
```

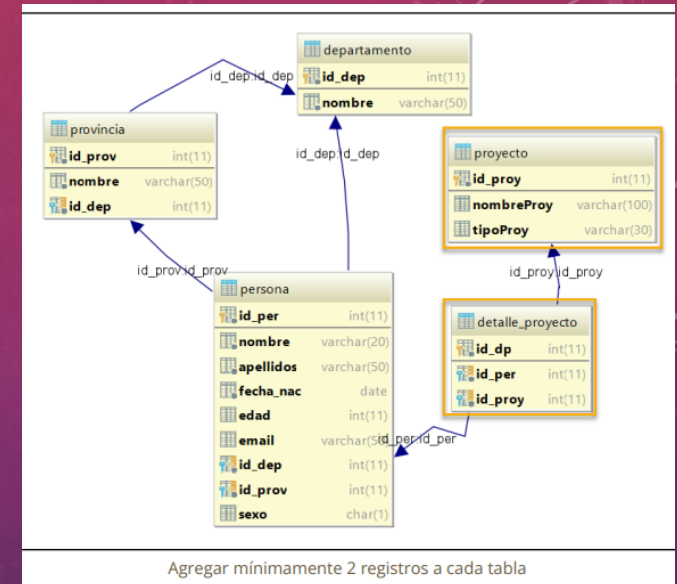
```
CREATE TABLE proyecto  
(  
  id_proy int primary key auto_increment,  
  nombreProy varchar(100),  
  tipoProy varchar(30)  
);
```

```
CREATE TABLE departamento  
(  
  id_dep int primary key auto_increment,  
  nombre varchar(50)  
);
```

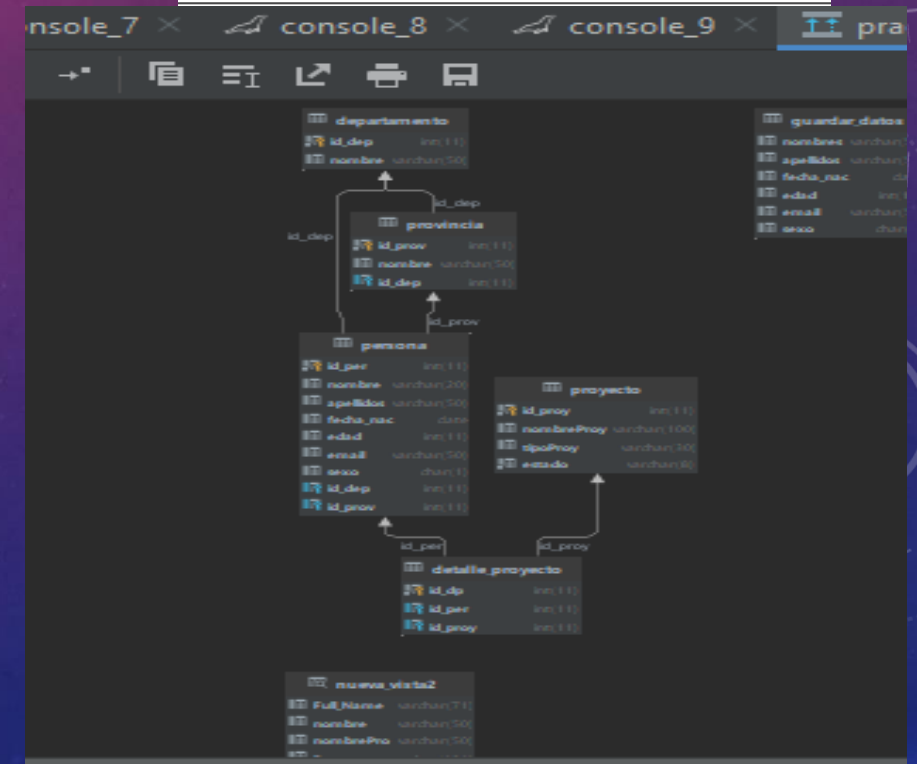
```
CREATE TABLE provincia  
(  
  id_prov int auto_increment primary key,  
  nombre varchar(50),  
  id_dep int,  
  foreign key (id_dep) references departamento(id_dep)  
);
```

```
CREATE TABLE persona  
(  
  id_per int auto_increment primary key,  
  nombre varchar(20),  
  apellidos varchar(50),  
  fecha_nac date,  
  edad int,  
  email varchar(50),  
  sexo char,  
  id_dep int,  
  id_prov int,  
  foreign key (id_dep) references departamento(id_dep),  
  foreign key (id_prov) references provincia(id_prov)  
);
```

```
CREATE TABLE detalle_proyecto  
(  
  id_dp int primary key auto_increment,  
  id_per int,  
  id_proy int,  
  foreign key (id_per) references persona(id_per),  
  foreign key (id_proy) references proyecto(id_proy)  
);
```



Agregar mínimamente 2 registros a cada tabla



# 10. Crear una función que sume los valores de la serie Fibonacci.

```
create function fibonacci(entrada int)
returns text
begin
    declare a int default 0;
    declare b int default 1;
    declare cont int default 0;
    declare aux int default 0;
    declare unir text default '';

    while cont < entrada
    do
        if cont = 0
        then
            set unir = '0';
        else
            set unir = concat(unir, b, ' ');
            set aux = a;
            set a = b;
            set b = aux + a;
        end if;
        set cont = cont + 1;
    end while;
    return unir;
end;
```

```
select fibonacci(10);
```

```
create function suma_fibonacci(entrada text)
returns int
begin
    declare espacio text default ' ';
    declare x int default 1;
    declare nVeces int default 0;
    declare letra char default '';
    declare limite int default char_length(entrada);
    declare a int default 0;
    declare b int default 1;
    declare cont int default 0;
    declare aux int default 0;
    declare sumar int default 0;

    while x <= limite do
        set letra = substring(entrada, x, 1);
        if letra = espacio
        then
            set nVeces = nVeces + 1;
        end if;
        set x = x + 1;
    end while;
    while cont < nVeces
    do
        if cont = 0
        then
            set sumar = 0;
        else
            set sumar = sumar + b;
            set aux = a;
            set a = b;
            set b = aux + a;
        end if;
        set cont = cont + 1;
    end while;
    return sumar;
end;
```

```
select suma_fibonacci(fibonacci(10)) as suma_fibonacci;
```

The screenshot shows a SQL IDE with multiple console windows. The main console window displays the execution of the `select fibonacci(10);` query. The output is shown in a table with one row: `1 0 1 1 2 3 5 8 13 21 34`. The code editor shows the `fibonacci` function definition and the query being executed.

The screenshot shows a SQL IDE with multiple console windows. The main console window displays the execution of the `select suma_fibonacci(fibonacci(10)) as suma_fibonacci;` query. The output is shown in a table with one row: `1 88`. The code editor shows the `suma_fibonacci` function definition and the query being executed.

# 11. MANEJO DE VISTAS

```
create view nueva_vista as
  select concat(per.nombre, ' ', per.apellidos) as Full_Name, per.edad,
per.fecha_nac, pro.nombreProy
  from persona as per
 inner join detalle_proyecto as dep on per.id_per = dep.id_per
 inner join proyecto as pro on dep.id_proy = pro.id_proy
 inner join departamento as depa on per.id_dep = depa.id_dep
 where per.sexo = 'F' and depa.nombre = 'El Alto' and per.fecha_nac = '2000-10-10';
```

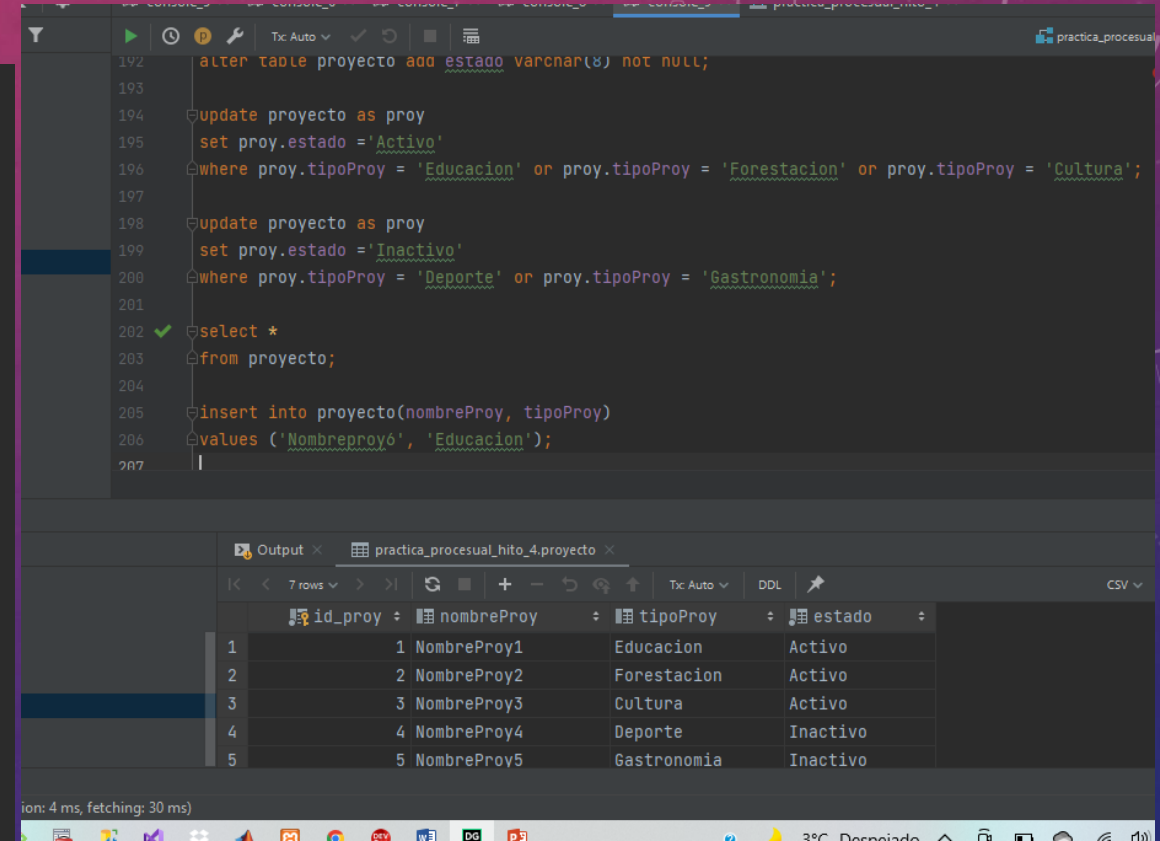
WHERE		ORDER BY	
Full_Name	edad	fecha_nac	nombreProy
1 Nombre1 Apellido1	22	2000-10-10	NombreProy1



# 12. Manejo de TRIGGERS I.

```
create trigger al_insertar_datos
before insert on proyecto
for each row
begin
    case
        when new.tipoProy = 'Educacion' or new.tipoProy = 'Forestacion' or new.tipoProy = 'Cultura' then
            set new.estado = 'Activo';
        else
            set new.estado = 'Inactivo';
        end case;
    end;
```

```
create trigger al_actualizar_datos
before update on proyecto
for each row
begin
    case
        when new.tipoProy = 'Educacion' or new.tipoProy = 'Forestacion' or new.tipoProy = 'Cultura' then
            set new.estado = 'Activo';
        else
            set new.estado = 'Inactivo';
        end case;
    end;
```



The screenshot shows a SQL IDE with a script editor and an output window. The script editor contains the following SQL code:

```
192 alter table proyecto add estado varchar(8) not null;
193
194 update proyecto as proy
195     set proy.estado = 'Activo'
196     where proy.tipoProy = 'Educacion' or proy.tipoProy = 'Forestacion' or proy.tipoProy = 'Cultura';
197
198 update proyecto as proy
199     set proy.estado = 'Inactivo'
200     where proy.tipoProy = 'Deporte' or proy.tipoProy = 'Gastronomia';
201
202 select *
203 from proyecto;
204
205 insert into proyecto(nombreProy, tipoProy)
206 values ('Nombreproy6', 'Educacion');
207
```

The output window shows the result of the 'select \* from proyecto;' query, displaying 7 rows of data:

id_proy	nombreProy	tipoProy	estado
1	NombreProy1	Educacion	Activo
2	NombreProy2	Forestacion	Activo
3	NombreProy3	Cultura	Activo
4	NombreProy4	Deporte	Inactivo
5	NombreProy5	Gastronomia	Inactivo
6	Nombreproy6	Educacion	Activo
7			

The status bar at the bottom indicates 'ion: 4 ms, fetching: 30 ms'.

# 14. Manejo de TRIGGERS III.

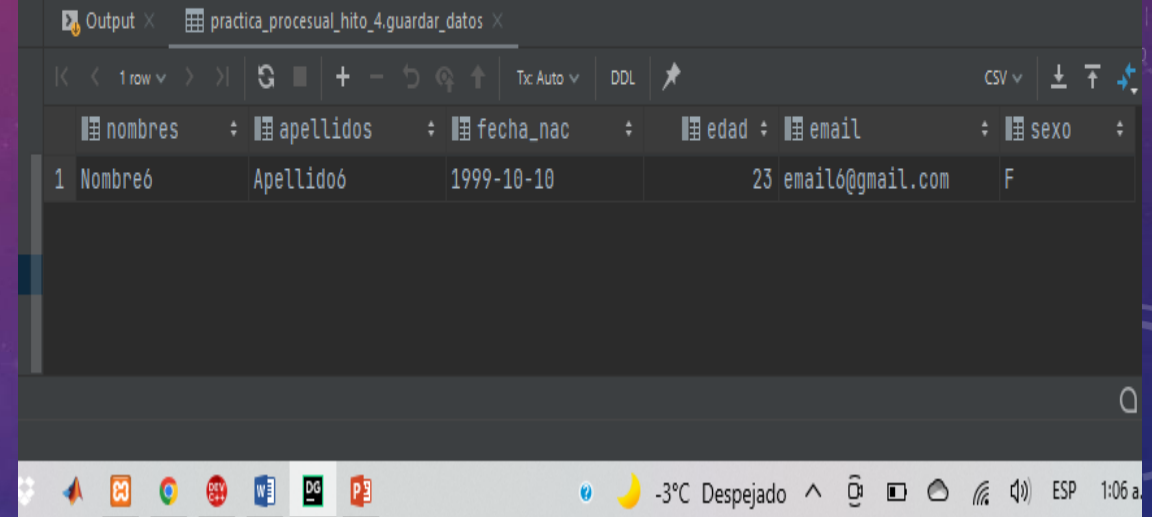
```
create table guardar_datos
(
  nombres varchar(50),
  apellidos varchar(50),
  fecha_nac date,
  edad int,
  email varchar(50),
  sexo char
);
```

```
create trigger guardar_datos
before insert on persona
for each row
begin
  insert into guardar_datos (nombres, apellidos, fecha_nac, edad, email, sexo)
  values (new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo);
end;
```

```
insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
values ('Nombre6', 'Apellido6', '1999-10-10', 23, 'email6@gmail.com', 'F', 1, 1);
```

```
select gu.*
from guardar_datos as gu;
```

```
select gu.*
from guardar_datos as gu;
```



	nombres	apellidos	fecha_nac	edad	email	sexo
1	Nombre6	Apellido6	1999-10-10	23	email6@gmail.com	F



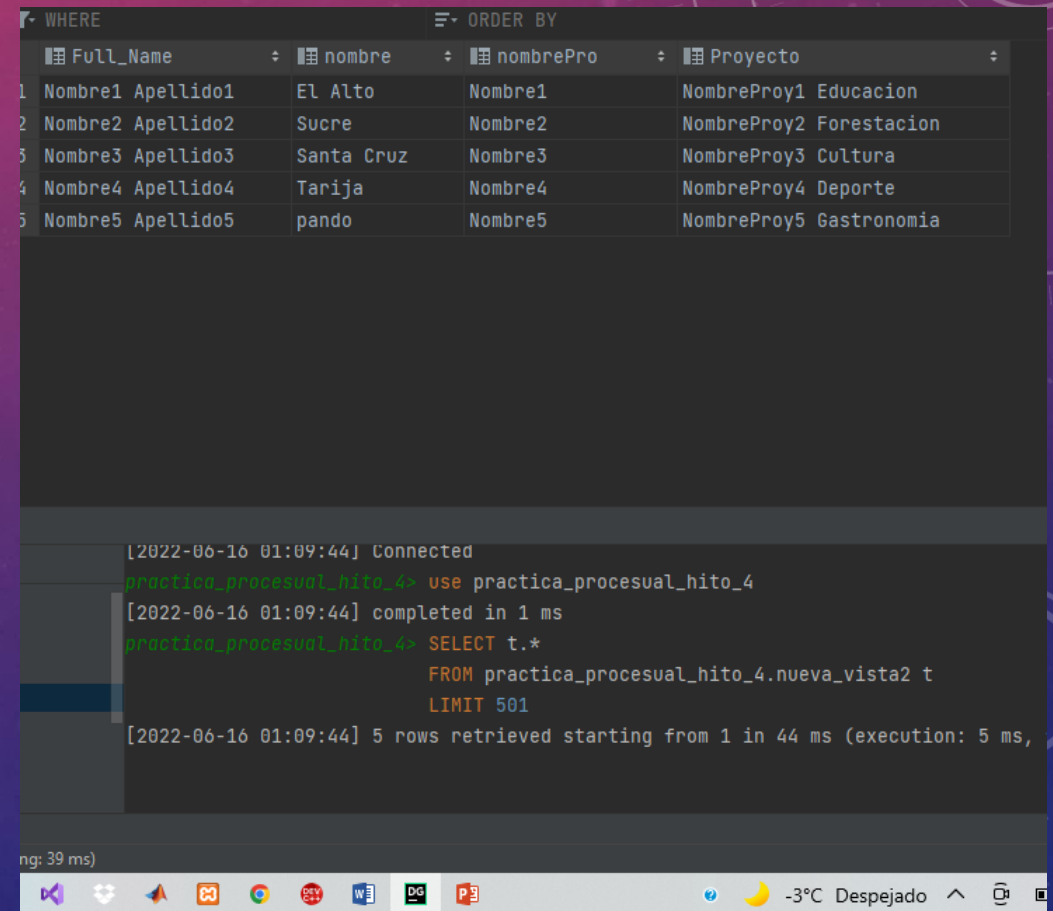
## 15. Crear una consulta SQL que haga uso de todas las tablas.

- La consulta generada convertirlo a **VISTA**

```
create view nueva_vista2 as
select concat(per.nombre, ' ', per.apellidos) as Full_Name,
       depa.nombre, pro.nombre as nombrePro, concat(proy.nombreProy, ' ', proy.tipoProy) as Proyecto
from departamento as depa
inner join provincia as pro on depa.id_dep = pro.id_dep
inner join persona as per on depa.id_dep = per.id_dep
inner join detalle_proyecto as dep on per.id_per = dep.id_per
inner join proyecto as proy on dep.id_proy = proy.id_proy
```

Link del video de mi defensa de la practica en drive.

[https://drive.google.com/file/d/1Vuddhcht\\_6rJhNLOmU\\_-my0LJdVK8ebV/view?usp=sharing](https://drive.google.com/file/d/1Vuddhcht_6rJhNLOmU_-my0LJdVK8ebV/view?usp=sharing)



The screenshot shows a SQL IDE interface. At the top, there's a table view with columns: Full\_Name, nombre, nombrePro, and Proyecto. The table contains 5 rows of data. Below the table view, there's a terminal window showing the execution of a SQL query. The query is: `SELECT t.* FROM practica_procesual_hito_4.nueva_vista2 t LIMIT 501`. The terminal output shows the query was executed successfully, retrieving 5 rows in 44 ms.

	Full_Name	nombre	nombrePro	Proyecto
1	Nombre1 Apellido1	El Alto	Nombre1	NombreProy1 Educacion
2	Nombre2 Apellido2	Sucre	Nombre2	NombreProy2 Forestacion
3	Nombre3 Apellido3	Santa Cruz	Nombre3	NombreProy3 Cultura
4	Nombre4 Apellido4	Tarija	Nombre4	NombreProy4 Deporte
5	Nombre5 Apellido5	pando	Nombre5	NombreProy5 Gastronomia

```
[2022-06-16 01:09:44] Connected
practica_procesual_hito_4> use practica_procesual_hito_4
[2022-06-16 01:09:44] completed in 1 ms
practica_procesual_hito_4> SELECT t.*
                        FROM practica_procesual_hito_4.nueva_vista2 t
                        LIMIT 501
[2022-06-16 01:09:44] 5 rows retrieved starting from 1 in 44 ms (execution: 5 ms,
```



GRACIAS