

**UNIVERSIDAD PRIVADA FRANZ
TAMAYO**

Tarea HITO 4

Estudiante: Fabiva Veyra Ramos Verastegui

Asignatura: BASE DE DATOS II

Carrera: INGENIERÍA DE SISTEMAS

Paralelo: BDA (1)

Docente: Lic. William Barra Paredes



Manejo de conceptos

1. Defina que es lenguaje procedural en MySQL.

El usuario da órdenes para que se realicen las tareas pertinentes con el objetivo de recuperar los datos requeridos. Es la base del lenguaje de consulta SQL.

2. Defina que es una FUNCTION en MySQL.

Las funciones 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.



Que en los **Procedimiento almacenado**, se debe especificar que es un parámetro externo, se puede obtener varios parámetros mientras que, y en las **funciones**, solo se puede devolver una variable (función escalar) o una tabla (funciones con valores de tabla).

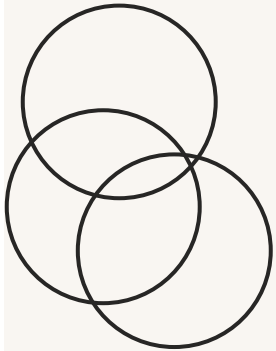
4. Cómo se ejecuta una función y un procedimiento almacenado.

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento y los valores para los parámetros separados por comas: `exec pa_libros_autor_editorial 'Richard Bach','Planeta'`. Para crear una función almacenada basta con que tengas permisos INSERT y DELETE sobre la base de datos.

5. Defina que es una TRIGGER en MySQL.

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.

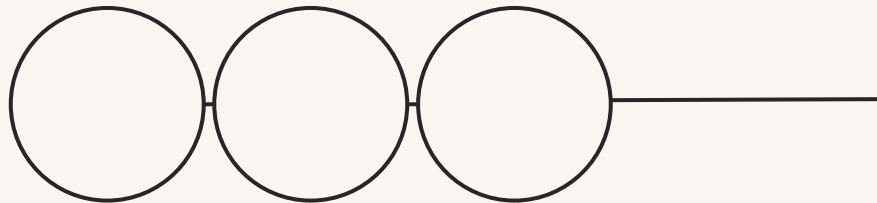
6. En un trigger que papel juega las variables OLD y NEW



La variable OLD hace referencia al valor de una columna antes de la incidencia se produzca; La variable NEW hace referencia a una columna afectada por la incidencia, una vez que haya pasado. Puede utilizar expresiones para realizar operaciones de lectura y asignación de valores a las variables de fila.

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

Si lo que falla es un disparador BEFORE, no se ejecuta la operación en el correspondiente registro.

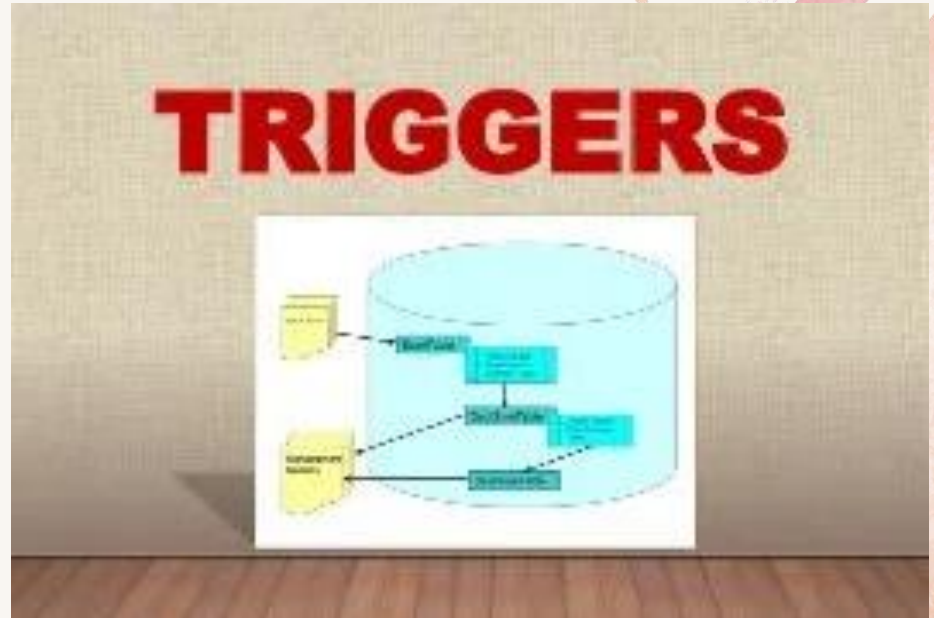


Un disparador AFTER se ejecuta solamente si el disparador BEFORE (de existir) y la operación se ejecutaron exitosamente.

Un error durante la ejecución de un disparador BEFORE o AFTER deriva en la falla de toda la sentencia que provocó la invocación del disparador.

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

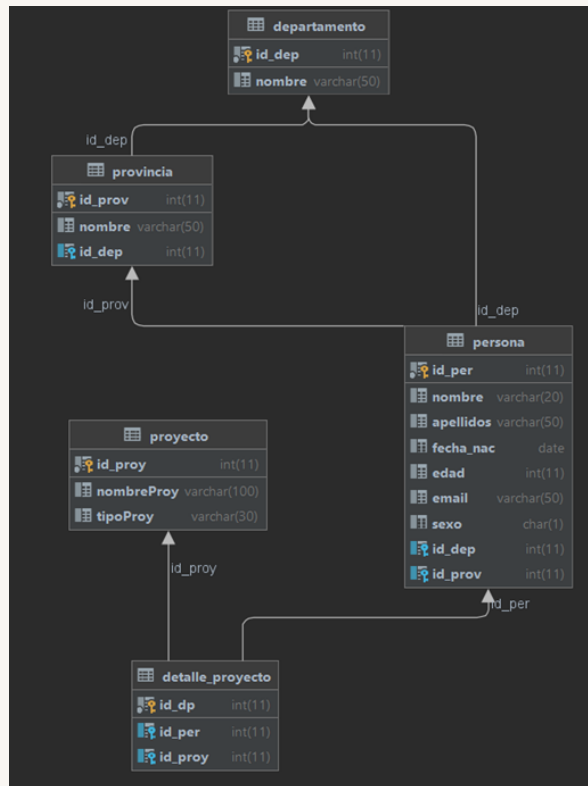
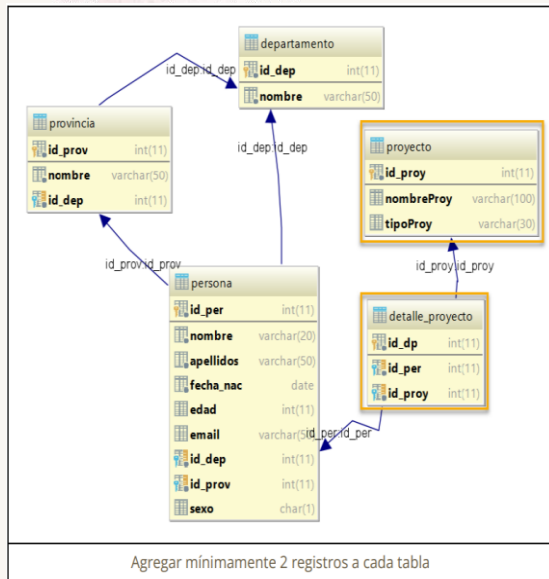
Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador.



The background features soft watercolor washes in shades of pink, peach, and light orange. A thin, hand-drawn pink line forms a looping, scribbled shape on the right side. In the bottom right corner, there are faint, concentric white circles resembling a fingerprint or a stylized spiral.

Parte practica

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



```

Create database Defensa_H4;
USE Defensa_H4;
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)
);
    
```

```
INSERT INTO proyecto(nombreProy,
tipoProy)
values ('Nombre1', 'Educacion'),
('Nombre2', 'Gastronomia'),
('Nombre3', 'Cultura'),
('Nombre4', 'Deporte'),
('Nombre5', 'Deporte');
```

```
INSERT INTO departamento(nombre)
values ('El Alto'),
('Cochabamba'),
('Santa Cruz'),
('Beni'),
('pando');
```

```
INSERT INTO provincia(nombre,
id_dep)
values ('NombrE1', 1),
('NombrE2', 2),
('NombrE3', 3),
('NombrE4', 4),
('NombrE5', 5);
```

```
INSERT INTO persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
values ('Nombre1', 'Apellido1', '2000-10-10', 22, 'email1@gmail.com', 'F', 1, 1),
('Nombre2', 'Apellido2', '2001-10-10', 21, 'email2@gmail.com', 'M', 2, 2),
('Nombre3', 'Apellido3', '2002-10-10', 20, 'email3@gmail.com', 'F', 3, 3),
('Nombre4', 'Apellido4', '2003-10-10', 19, 'email4@gmail.com', 'F', 4, 4),
('Nombre5', 'Apellido5', '2004-10-10', 18, 'email5@gmail.com', 'F', 5, 5);

INSERT INTO detalle_proyecto(id_per, id_proy)
values (1,1),
(2,2),
(3,3),
(4,4),
(5,5);
```

```
SELECT dep.*
FROM departamento AS dep;
SELECT dp.*
FROM detalle_proyecto AS dp;
SELECT per.*
FROM persona AS per;
SELECT pro.*
FROM provincia AS pro;
SELECT proy.*
FROM proyecto AS proy;
```

	id_dep	nombre	id_dep
1	1	NombreE1	1
2	2	NombreE2	2
3	3	NombreE3	3
4	4	NombreE4	4
5	5	NombreE5	5

	id_proy	nombreProy	tipoProy
1	1	Nombre1	Educacion
2	2	Nombre2	Gastronomia
3	3	Nombre3	Cultura
4	4	Nombre4	Deporte
5	5	Nombre5	Deporte

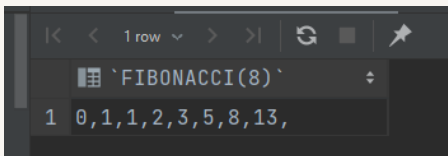
	id_dep	nombre
1	1	El Alto
2	2	Cochabamba
3	3	Santa Cruz
4	4	Beni
5	5	pando

	id_per	nombre	apellidos	fecha_nac	edad	email	sexo	id_dep	id_prov
1	1	Nombre1	Apellido1	2000-10-10	22	email1@gmail.com	F	1	1
2	2	Nombre2	Apellido2	2001-10-10	21	email2@gmail.com	M	2	2
3	3	Nombre3	Apellido3	2002-10-10	20	email3@gmail.com	F	3	3
4	4	Nombre4	Apellido4	2003-10-10	19	email4@gmail.com	F	4	4
5	5	Nombre5	Apellido5	2004-10-10	18	email5@gmail.com	F	5	5

	id_dep	id_per	id_proy
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

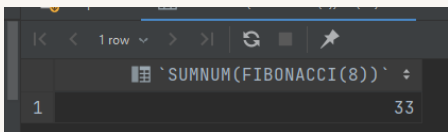
10. Crear una función que sume los valores de la serie Fibonacci.

- El objetivo es sumar todos los números de la serie fibonacci desde una cadena.
- Es decir usted tendrá solo la cadena generada con los primeros N números de la serie fibonacci y a partir de ellos deberá sumar los números de esa serie.



A screenshot of a database query result. The query is ``FIBONACCI(8)``. The result is a single row with the value `0,1,1,2,3,5,8,13,`.

1 0,1,1,2,3,5,8,13,



A screenshot of a database query result. The query is ``SUMNUM(FIBONACCI(8))``. The result is a single row with the value `33`.

1 33

```
CREATE FUNCTION FIBONACCI(LIM INT)
RETURNS TEXT
BEGIN
  DECLARE Y INT DEFAULT 0;
  DECLARE Z INT DEFAULT 1;
  DECLARE U INT DEFAULT 0;
  DECLARE X INT DEFAULT 1;
  DECLARE RESPONSE TEXT;
  IF LIM >= 1
  THEN
    SET RESPONSE = CONCAT(Y, ',');
  END IF;
  IF LIM >= 2
  THEN
    SET RESPONSE = CONCAT(RESPONSE, Z, ',');
  END IF;
  IF LIM >= 3
  THEN
    WHILE X <= (LIM - 2) DO
      SET U=Y+Z;
      SET RESPONSE = CONCAT(RESPONSE, U, ',');
      SET Y = Z;
      SET Z= U;
      SET X = X+1;
    END WHILE;
  END IF;
  RETURN RESPONSE;
END;
```

```
CREATE FUNCTION SUMNUM(CADENA VARCHAR(100))
RETURNS INT
BEGIN
  DECLARE NUM VARCHAR(50) DEFAULT "";
  DECLARE RES INTEGER DEFAULT 0;
  DECLARE FIBO VARCHAR(1);
  DECLARE C INTEGER DEFAULT 1;

  IF LENGTH(CADENA) > 0 THEN
    WHILE (C <= LENGTH(CADENA)) DO
      SET FIBO = SUBSTRING(CADENA, C, 1);
      IF FIBO = ',' THEN
        SET RES = RES + NUM;
        SET NUM = "";
      ELSE
        SET NUM = CONCAT(NUM, FIBO);
      END IF;
      SET C = C + 1;
    END WHILE;
    RETURN RES;
  ELSE
    RETURN 0;
  END IF;
END;

# Ejemplo: 0,1,1,2,3,5,8,.....
SELECT FIBONACCI(8);
SELECT SUMNUM(FIBONACCI(8));
```

11. Manejo de vistas.

Crear una consulta SQL para lo siguiente.

■ La consulta de la vista debe reflejar como campos:

1. nombres y apellidos concatenados

2. la edad

3. fecha de nacimiento.

4. Nombre del proyecto

○ Obtener todas las personas del sexo femenino que hayan nacido en el

departamento de El Alto en donde la fecha de nacimiento sea:

fecha_nac = '2000-10-10'

```
create view manejo_de_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 dp on per.id_per = dp.id_per
  inner join proyecto as pro on dp.id_proy = pro.id_proy
  inner join departamento as dep on per.id_dep = dep.id_dep
  where per.sexo = 'F' and dep.nombre = 'El Alto' and per.fecha_nac = '2000-10-10';
```



12

Manejo de TRIGGERS I.

- Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO.
- Debera de crear 2 triggers minimamente.
- Agregar un nuevo campo a la tabla PROYECTO.
- El campo debe llamarse ESTADO
- Actualmente solo se tiene habilitados ciertos tipos de proyectos.
- EDUCACION, FORESTACION y CULTURA
- Si al hacer insert o update en el campo tipoProy llega los valores EDUCACION, FORESTACIÓN o CULTURA, en el campo ESTADO colocar el valor ACTIVO. Sin embargo se llegat un tipo de proyecto distinto colocar INACTIVO
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

	id_proy	nombreProy	tipoProy	estado
1	1	Nombre1	Educacion	Activo
2	2	Nombre1123	Gastronomia	Inactivo
3	3	Nombre113	Cultura	Activo
4	4	Nombre4	Deporte	
5	5	Nombre5	Deporte	

```
alter table proyecto add estado varchar(15) not null;
```

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

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

```
update proyecto as proy
set proy.estado = 'Activo'
where proy.tipoProy = 'Cultura' or proy.tipoProy = 'Educacion' or proy.tipoProy = 'Forestacion';
```

```
update proyecto as proy
set proy.estado = 'Inactivo'
where proy.tipoProy = 'Gastronomia';
```

```
select *
from proyecto;
```


13. Manejo de Triggers II.

- El trigger debe de llamarse calculaEdad.
- El evento debe de ejecutarse en un BEFORE INSERT.
- Cada vez que se inserta un registro en la tabla PERSONA, el trigger debe de calcular la edad en función a la fecha de nacimiento.

0

```
create trigger calculaedad
before insert on persona
for each row
begin
    declare an int default 0;
    declare an_act int default 0;

    set an = (select max(substr(new.fecha_nac, 1, 4))
    from persona as per);
    set an_act = (SELECT substr(curdate(),1,4));

    set new.edad = an_act - an;
end;

insert into persona (nombre, apellidos, fecha_nac, email, sexo, id_dep, id_prov)
values ('Fabi', 'Ramos', '2002-09-06', 'nom@gmail.com', 'F', 1, 1);

select per.*
from persona as per;
```

	id_per	nombre	apellidos	fecha_nac	edad	email	sexo	id_dep	id_prov
1	1	Nombre1	Apellido1	2000-10-10	22	email1@gmail.com	F	1	1
2	2	Nombre2	Apellido2	2001-10-10	21	email2@gmail.com	M	2	2
3	3	Nombre3	Apellido3	2002-10-10	20	email3@gmail.com	F	3	3
4	4	Nombre4	Apellido4	2003-10-10	19	email4@gmail.com	F	4	4
5	5	Nombre5	Apellido5	2004-10-10	18	email5@gmail.com	F	5	5
6	6	Fabi	Ramos	2002-09-06	20	nom@gmail.com	F	1	1

—14. Manejo de TRIGGERS III—

- Crear otra tabla con los mismos campos de la tabla persona (Excepto el primary key id_per).
- No es necesario que tenga PRIMARY KEY.
- Cada vez que se haga un INSERT a la tabla persona estos mismos valores deben insertarse a la tabla copia.
- Para resolver esto deberá de crear un trigger before insert para la tabla PERSONA.

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

create trigger secu_perso
before insert on persona
for each row
begin
  insert into secu_perso (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 ('Diana', 'Colque', '2000-05-07', 20, 'diaco@gmail.com', 'F', 1, 1);

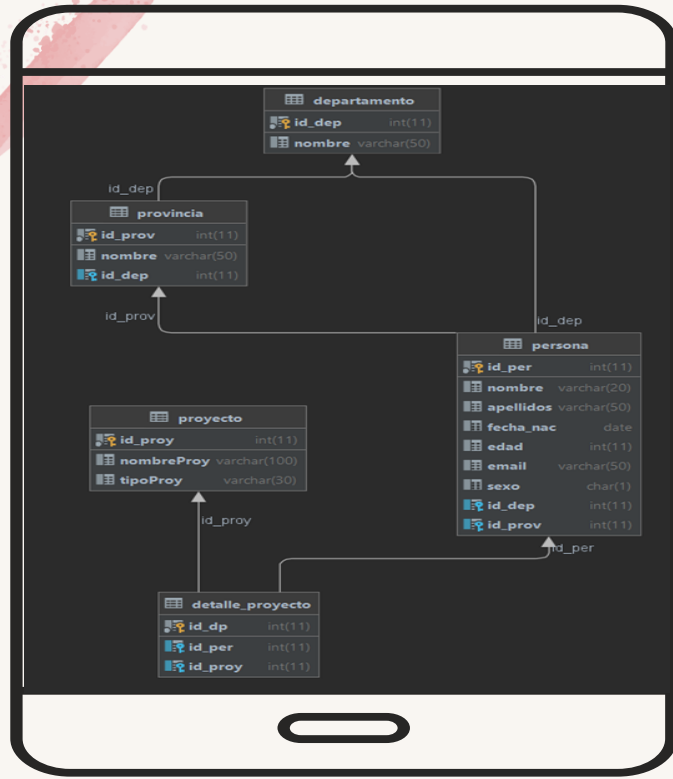
select sp.*
from secu_perso as sp;
```

1 row						
	nombres	apellidos	fecha_nac	edad	email	sexo
1	Diana	Colque	2000-05-07	22	diaco@gmail.com	F

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

La consulta generada convertirlo a VISTA

```
create view uso_de_todas_las_tablas 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;
```



Gracias. ..:)