

# TAREA HITO 4

## BASE DE DATOS II

**ESTUDIANTE:**

Andrés Vladimir Quiroga Huariste

**DOCENTE:**

William Roddy Barra Paredes

# MANEJO DE CONCEPTOS

- Defina que es el lenguaje procedural en MYSQL.

Conjunto de instrucciones SQL, mas una serie de estructuras de control que pueden almacenarse en el servidor.

- Defina que es una función en MYSQL.

Son definidas por el usuario, son rutinas que aceptan parámetros, realiza una acción y devuelven el resultado como un único valor.

- Cual es la diferencia entre funciones y procedimientos almacenados.

Cuando se llama al procedimiento almacenado, se debe especificar que es un parámetro externo, se pueden obtener varios parámetros, en funciones solo se puede devolver una variable o una tabla.

- Como se ejecuta una función y un procedimiento almacenado.

Expandir la base de datos que desee, expandir Programación, a continuación, expandir Procedimientos almacenados. Haga clic con el botón derecho en el procedimiento almacenado definido por el usuario que quiera y, luego, seleccione Ejecutar procedimiento almacenado.

- Defina que es una Trigger en MYSQL.

Los triggers son programas almacenados que se ejecutan automáticamente cuando ocurre un evento.

- En un Trigger que papel juega las variables OLD y NEW.

New: Es una sentencia Insert, tenemos la variable New en donde este tiene acceso a todas las columnas de la tabla.

Old: Es una sentencia Delete, tenemos la variable Old, en donde este tiene acceso a todas las columnas de la tabla.

- En un Trigger que papel juega los conceptos (clausuras) before or after.

Before: Antes de actualizar

After: Después de actualizar.

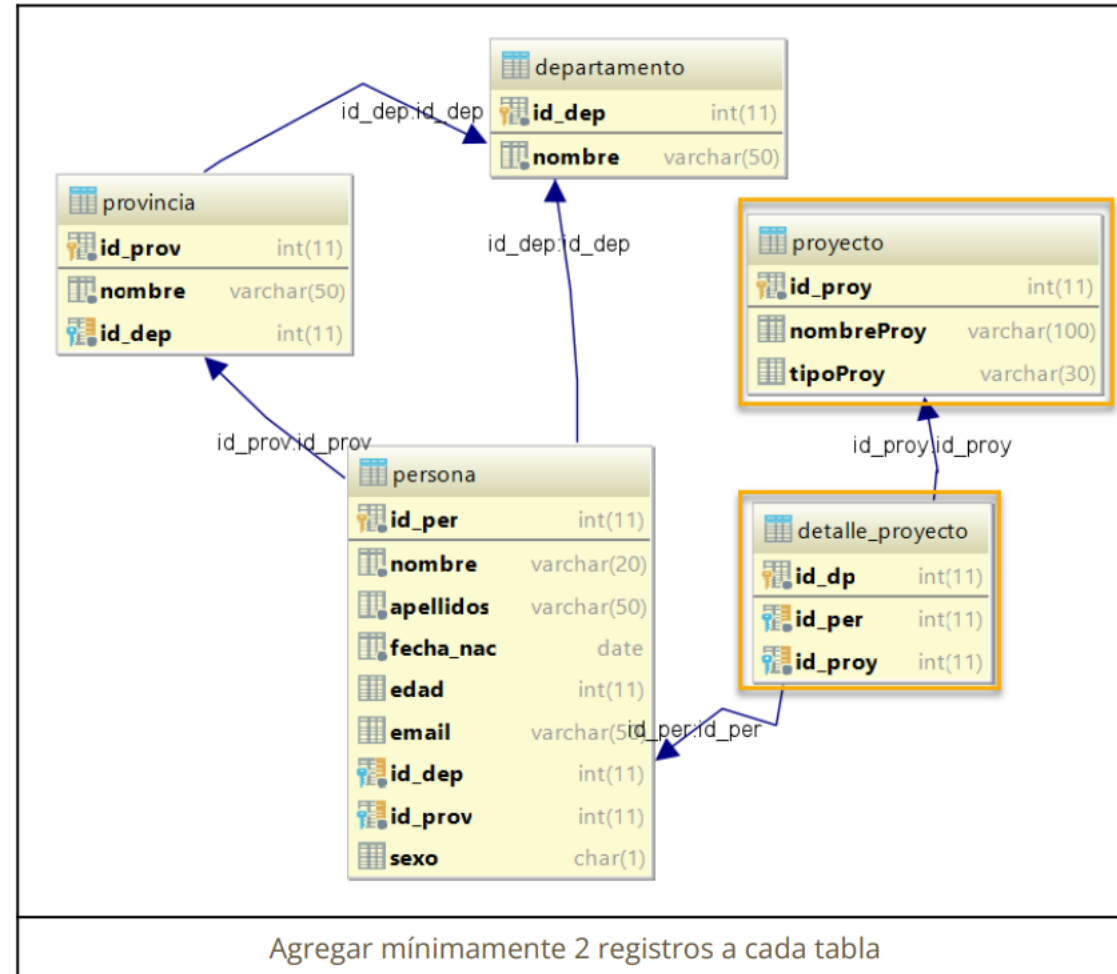
- A que se refiere cuando se habla eventos en Triggers.

Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

# PARTE PRACTICA

## Parte practica

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



## 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.
- Ejemplo: **suma\_serie\_fibonacci(mi\_metodo\_que\_retorna\_la\_serie(10))**
  - Note que previamente deberá crear una función que retorne una cadena con la serie fibonacci hasta un cierto valor.
    1. Ejemplo: 0,1,1,2,3,5,8,.....
  - Luego esta función se deberá pasar como parámetro a la función que suma todos los valores de esa serie generada.

	
FUNCTION QUE GENERA LA SERIE	FUNCTION QUE SUMA LA SERIE

- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
-- pregunta 10

create or replace function serie_fibonanci(numero integer) returns text
begin
    declare a integer default 0;
    declare b integer default 1;
    declare aux integer default 0;
    declare contador integer default 0;
    declare cadena text default '';
    set cadena = concat(a , ',' , b);
    if numero = 1 then set cadena = '0';
    elseif numero = 2 then set cadena = '0,1';
    elseif numero <= 0 then set cadena = 'El number debe ser mayor a cero';

    else
        repeat
            set aux = a + b;
            set cadena = concat(cadena, ',' , aux);
            set a = b;
            set b = aux;
            set contador = contador + 1;
            until contador = numero - 2 end repeat;
    end if;
    return cadena;
end;

create or replace function contar_fibonanci(serie text) returns integer
```

create or replace function contar\_fibonanci(serie text) returns integer

begin

declare suma integer default 0;

declare cont integer default 1;

declare final integer default char\_length(serie);

repeat

set suma = suma + substring(serie , cont , 1);

set cont = cont + 2;

until cont > final

end repeat;

return suma;

end;

select serie\_fibonanci( numero: 8);

select contar\_fibonanci( serie: serie\_fibonanci( numero: 8));|

-- pregunta 11

Output contar\_fibonanci(serie\_fibonanci(8)):int(11) X

|< < 1 row > >| ↺ ■ ↗

⌘ `contar\_fibonanci(serie\_fibonanci(8))` ⌵

1	21
---	----

```
128
129 -- pregunta 11
130
131 create or replace view refleja as
132     select concat(per.nombre, ' ', per.apellidos) as fullname , per.edad as edad , per.fecha_nac as fecha_nacimiento , proy.nombre_proy as nombre
133     from persona as per
134     inner join departamento dep on per.id_dep = dep.id_dep
135     inner join detalle_proyecto dp on per.id_per = dp.id_per
136     inner join proyecto proy on dp.id_proy = proy.id_proy
137     where per.genero = 'F' and dep.nombre = 'El Alto' and per.fecha_nac = '2000-10-10';
138
139 select * from refleja;
140
141 -- pregunta 12
142
143 alter table proyecto add (estado varchar(30));
144 insert into proyecto(nombre_proy, tipo_proy) VALUES
145 ('educacion a personas' , 'educacion'),
146 ('siembra' , 'forestacion'),
147 ('mayas' , 'cultura');
```

Output practica\_hito4.refleja

fullname	edad	fecha_nacimiento	nombre_del_proyecto
----------	------	------------------	---------------------

## 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:
  1. fecha\_nac = '2000-10-10'

**LA CONSULTA GENERADA PREVIAMENTE CONVERTIR EN UNA VISTA**

## 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.**

```
-- pregunta 12

alter table proyecto add (estado varchar(30));
insert into proyecto(nombre_proy, tipo_proy) VALUES
('educion a personas' , 'educacion'),
('siembra' , 'forestacion'),
('mayas' , 'cultura');

create or replace trigger tipo_proyecto
before update on proyecto
for each row
begin
    if NEW.tipo_proy = 'Educacion' or NEW.tipo_proy = 'Forestacion' or NEW.tipo_proy = 'Cultura'
    then set NEW.estado = 'Activo';
    else
        set NEW.estado = 'Inactivo';
    end if;
end;
```



```

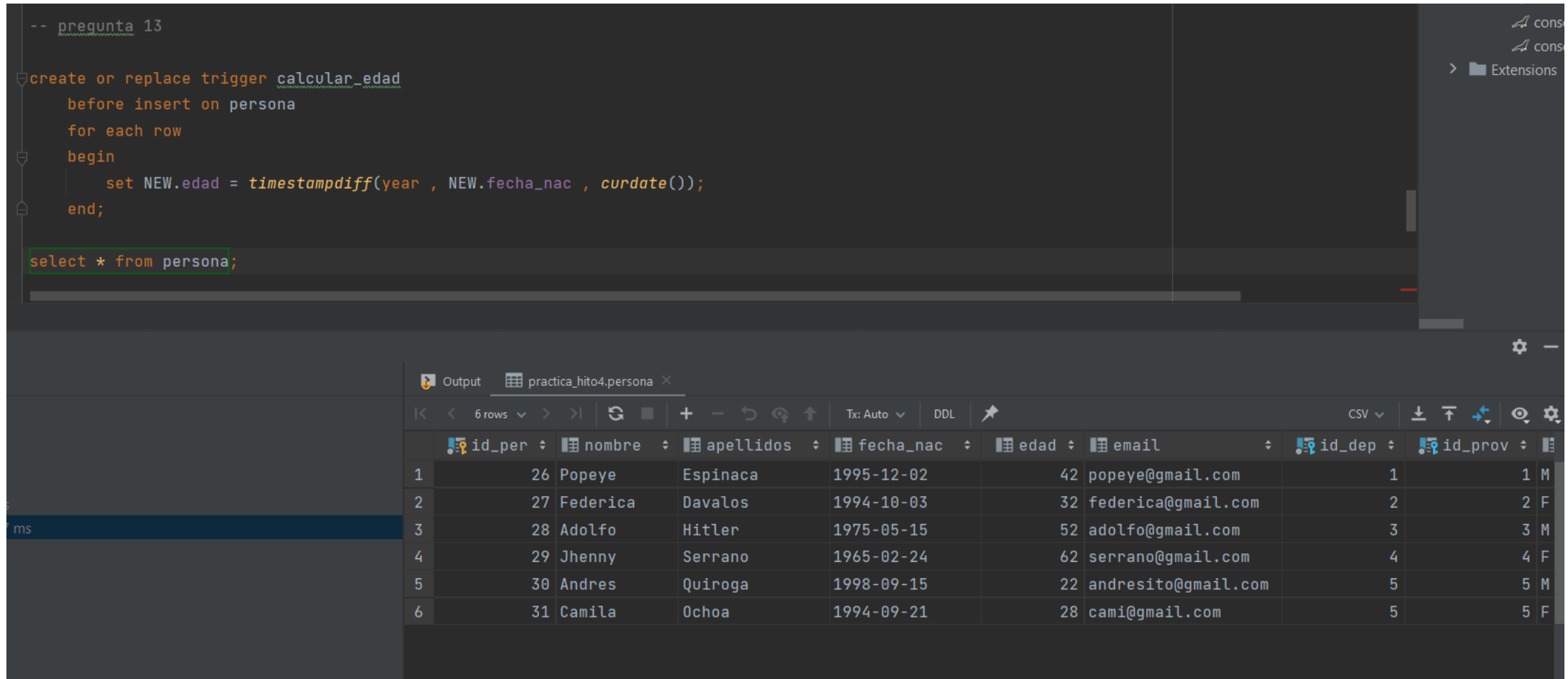
160 create or replace trigger agregacion
161     before insert on proyecto
162     for each row
163     begin
164         if NEW.tipo_proy = 'Educacion' or NEW.tipo_proy = 'Forestacion' or NEW.tipo_proy = 'Cultura'
165         then set NEW.estado = 'Activo';
166         else
167             set NEW.estado = 'Inactivo';
168         end if;
169     end;
170
171 ✓ select * from proyecto;
172
173

```

Output practica_hito4.proyecto				
Tx: Auto DDL				
	id_proy	nombre_proy	tipo_proy	estado
7	7	educion a personas	educacion	<null>
10	10	siembra	forestacion	<null>
11	11	mayas	cultura	<null>
12	12	educion a personas	educacion	<null>
13	13	siembra	forestacion	<null>
14	14	mayas	cultura	<null>
15	15	educion a personas	educacion	Activo
16	16	siembra	forestacion	Activo
17	17	mayas	cultura	Activo
18	18	Cachorritos	Mascotas	Inactivo
19	19	Robots	Tecnologia	Inactivo

## 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.
- Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento**.



The screenshot displays a SQL development environment. The top pane shows the SQL code for creating a trigger named `calcula_edad` that calculates age before inserting into the `persona` table. The bottom pane shows the output of a `select * from persona;` query, displaying a table with 6 rows of person data.

```
-- pregunta 13

create or replace trigger calcula_edad
before insert on persona
for each row
begin
    set NEW.edad = timestampdiff(year , NEW.fecha_nac , curdate());
end;

select * from persona;
```

	id_per	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	
1	26	Popeye	Espinaca	1995-12-02	42	popeye@gmail.com	1	1	M
2	27	Federica	Davalos	1994-10-03	32	federica@gmail.com	2	2	F
3	28	Adolfo	Hitler	1975-05-15	52	adolfo@gmail.com	3	3	M
4	29	Jhenny	Serrano	1965-02-24	62	serrano@gmail.com	4	4	F
5	30	Andres	Quiroga	1998-09-15	22	andresito@gmail.com	5	5	M
6	31	Camila	Ochoa	1994-09-21	28	cami@gmail.com	5	5	F

## 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**.
- Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento**.

```
-- pregunta 14

create table copia_persona(
    nombre varchar(20),
    apellidos varchar(50),
    fecha_nac date,
    edad int,
    email varchar(50),
    id_dep int not null,
    id_prov int not null,
    genero varchar(1),
    foreign key (id_prov) references provincia(id_prov),
    foreign key (id_dep) references departamento(id_dep)
);

create or replace trigger copias_persona
before insert on persona
for each row
begin
    insert into copia_persona(nombre, apellidos, fecha_nac, edad, email, id_dep, id_prov, genero)
    values (NEW.nombre , NEW.apellidos , NEW.fecha_nac , NEW.edad , NEW.email , NEW.id_dep , NEW.id_prov , NEW.genero);
end;
```

```
insert into persona(nombre, apellidos, fecha_nac, edad, email, id_dep, id_prov, genero) values  
('Camila' , 'Ochoa' , '1994-09-21' , 23 , 'cami@gmail.com' , 5 , 5 , 'F');
```

```
select * from copia_persona;
```

```
-- pregunta 15
```

```
create or replace view usando_todas_las_tablas as
```

Output practica\_hito4.copia\_persona X

	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	genero
1	Camila	Ochoa	1994-09-21	28	cami@gmail.com	5	5	F

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

- La consulta generada convertirlo a VISTA

```
-- pregunta 15
create or replace view usando_todas_las_tablas as
select
    concat(per.nombre , per.apellidos) as fullname , per.edad as edad , dep.nombre as departamento , prov.nombre as provincia,
    concat(proy.nombre_proy , ' ' , tipo_proy) as proyecto
from persona as per
inner join departamento dep on per.id_dep = dep.id_dep
inner join provincia prov on dep.id_dep = prov.id_dep
inner join detalle_proyecto dp on per.id_per = dp.id_per
inner join proyecto proy on dp.id_proy = proy.id_proy...

select * from usando_todas_las_tablas;
```

Output practica\_hito4.usando\_todas\_las\_tablas

	fullname	edad	departamento	provincia	proyecto
1	PopeyeEspinaca	42	La Paz	Coroico	CachorritosMascotas
2	FedericaDavalos	32	El Alto	Caranavi	RobotsTecnologia
3	AdolfoHitler	52	Cochabamba	Palos Blancos	ProgramasSistemas
4	JhennySerrano	62	Santa Cruz	Laja	VacunasSalud