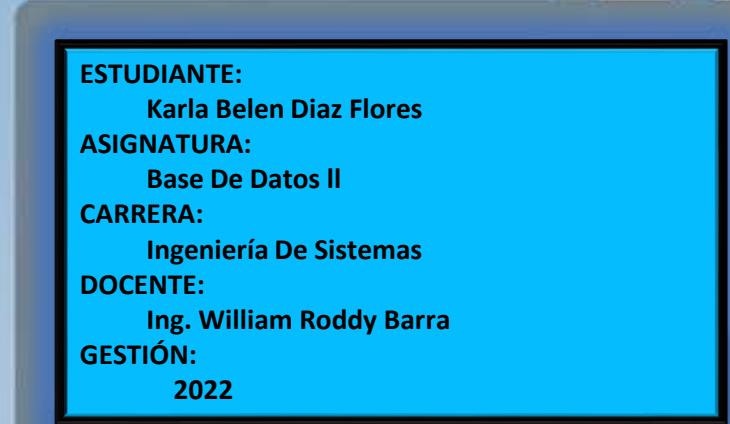


DEFENSA HITO 4



1. Defina que es lenguaje procedural en MySQL.

Lenguajes procedurales o procedimentales: 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

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.

Primeramente se debe crear una función para lo que usaremos create function y para el procedimiento almacenado seria create procedure con sus procedimientos ya realizados, para salir por pantalla necesitaremos un select.

Podrás usarlas en las sentencias SQL independientemente del lenguaje de programación del servidor sobre el que se ejecuten las consultas. 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

Cuando trabajamos con trigger a nivel de fila, provee de dos tablas temporales a las cuales se puede acceder, que contienen los antiguos y nuevos valores de los campos del registro afectado por la sentencia que disparó el trigger. El nuevo valor es ":new" y el viejo valor es ":old". Para referirnos a ellos debemos especificar su campo separado por un punto ":new.CAMPO" y ":old.CAMPO".



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

Los diferencia principal entre el disparador Antes y Desp  s en MySQL es que Before trigger realiza una acci  n antes de que una determinada operaci  n se ejecute en la tabla, mientras que After trigger realiza una acci  n despu  s de que cierta operaci  n se ejecute en la tabla.



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

Un "trigger" (disparador o desencadenador) es un bloque de c  digo que se ejecuta autom  ticamente cuando ocurre alg  n evento (como inserci  n, actualizaci  n 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 DefensaHito4;
USE DefensaHito4;

CREATE TABLE proyecto
(
    id_proy INTEGER(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombreProy VARCHAR(100),
    TipoProy VARCHAR(30),
    estado VARCHAR(30)
);

CREATE TABLE detalle_proyecto
(
    id_dp INTEGER(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_per INT(11),
    id_proy INT(11),
    FOREIGN KEY (id_proy) REFERENCES proyecto (id_proy),
    FOREIGN KEY (id_per) REFERENCES persona (id_per)
);

CREATE TABLE persona
(
    id_per INTEGER(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(20),
    apellidos VARCHAR(50),
    fecha_nac DATE,
    edad INTEGER(11),
    email VARCHAR(50),
    id_dep INT(11),
    id_prov INT(11),
    sexo CHAR(1),
    FOREIGN KEY (id_prov) REFERENCES provincia (id_prov),
    FOREIGN KEY (id_dep) REFERENCES departamento (id_dep)
);

CREATE TABLE provincia
(
    id_prov INTEGER(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(50),
    id_dep INT(11),
    FOREIGN KEY (id_dep) REFERENCES departamento (id_dep)
);

CREATE TABLE departamento
(
    id_dep INTEGER(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(50)
);
```

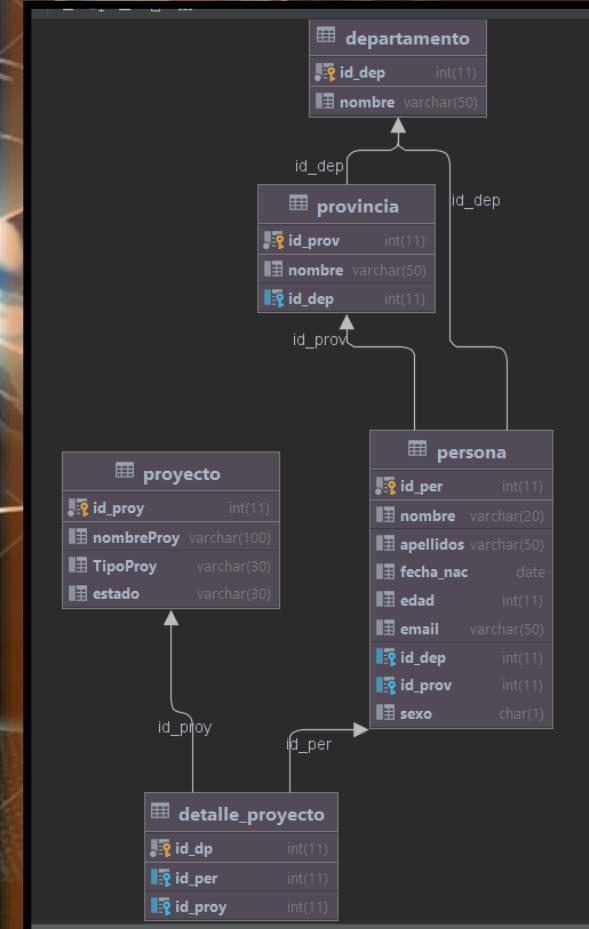
```
INSERT INTO proyecto (id_proy, nombreProy, TipoProy)
VALUES ( 4 , 'Sembrando esperanza' 'Forestacion');
INSERT INTO proyecto (id_proy, nombreProy, TipoProy)
VALUES ( 5 , 'Cambiando vidas','Adopcion');

INSERT INTO detalle_proyecto (id_dp, id_per, id_proy)
VALUES (1, 7, 4);
INSERT INTO detalle_proyecto (id_dp, id_per, id_proy)
VALUES (2,8, 5);

INSERT INTO persona (id_per, nombre, apellidos, fecha_nac, edad, email, id_dep, id_prov, sexo)
VALUES (7,'Luz','Peralta Chipana','2000-05-10',22,'luz@gmail.com',12,46,'f');
INSERT INTO persona (id_per, nombre, apellidos, fecha_nac, edad, email, id_dep, id_prov, sexo)
VALUES (8,'Jose','Fernandez Lopez','2001-04-04',21,'josefer@gmail.com',14,23,'m');

INSERT INTO provincia (id_prov, nombre, id_dep)
VALUES (46,'Bolivar',12);
INSERT INTO provincia (id_prov, nombre, id_dep)
VALUES (23,'Gran Chaco',14);

INSERT INTO departamento (id_dep, nombre)
VALUES (12, 'Cochabamba');
INSERT INTO departamento (id_dep, nombre)
VALUES (14,'Tarija');
```



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

```
CREATE FUNCTION fibo(limite integer) returns text
begin
    declare respuesta text default '';
    declare x integer default 0;
    declare y integer default 1;
    declare cont integer default 0;
    while cont != limite
        do
            set respuesta = concat(respuesta, x, ' ');
            set x = x + y;
            set y = x - y;
            set cont = cont + 1;
        end while;
    return respuesta;
end;

Select fibo(10);
```

```
CREATE FUNCTION sum_fibo(Limita integer)
returns text
begin
    declare entrada text default '';
    declare espacio text default '';
    declare x int default 1;
    declare nVeces int default 0;
    declare letra char default '';
    declare limite int default 0;
    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;
    set entrada = fibo(limita);
    set limite = char_length(entrada);
    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 sum_fibo(10);
```

```
92
93 ✓ Select fibo( limite: 10);
94
95 CREATE FUNCTION sum_fibo(Limita integer)
96     returns text
97 begin
98     declare entrada text default '';
99     declare espacio text default '';
100    declare x int default 1;
101    declare nVeces int default 0;
102    declare letra char default '';
103    declare limite int default 0;
104    declare a int default 0;
105    declare b int default 1;
106    declare cont int default 0;
107    declare aux int default 0;
108    declare sumar int default 0;
109    set entrada = fibo( limite: limita);
110    set limite = char_length(entrada);

Output × Id_pervarchar × fibo(10):text ×
ms
`fibo(10)`
1 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

```
125
126
127
128
129
130
131
132
133
134
135
136 ✓ Select sum_fibo( Limita: 10);
137
138
139
140 create trigger manejodetrig
141     before update
142     on proyecto
143     for each row

Output × sum_fibo(10):text × fibo(10):text
ms
`sum_fibo(10)`
1 88
```

12. Manejo de TRIGGERS I.

- Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO

```
create trigger manejodetrig
  before update
  on proyecto
  for each row
begin
  if new.tipoProy =
    'Forestacion' or new.tipoProy =
    'Adopcion' or new.tipoProy = 'CULTURA' then
    set new.estado = 'activo';
  else
    set new.estado = 'inactivo';
  end if;
end;
update proyecto
set tipoProy =
  'Educacion'
where id_proy = 4;

SELECT * from proyecto ;
```

The screenshot shows a MySQL Workbench session window. The SQL editor contains the following code:

```
140  create trigger manejodetrig
141    before update
142    on proyecto
143    for each row
144  begin
145    if new.tipoProy =
146      'Forestacion' or new.tipoProy =
147        'Adopcion' or new.tipoProy = 'CULTURA' then
148        set new.estado = 'activo';
149    else
150        set new.estado = 'inactivo';
151    end if;
152  end;
153  update proyecto
154  set tipoProy =
155    'Educacion'
156  where id_proy = 4;
157
158  SELECT * from proyecto ;
159
160  create trigger calculaEdad
161    before insert
```

The results pane shows the following table data:

	id_proy	nombreProy	TipoProy	estado
1	1	Sembrando esperanza	Educacion	inactivo
2	2	Cambiando vidas	Adopcion	activo

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.

```
create trigger calculaEdad
before insert
on persona
for each row
begin
    declare edad_calc integer;
    set edad_calc = timestampdiff(year, new.fecha_nac,
curdate());
    set new.edad = edad_calc;
end;
INSERT INTO persona (nombre, fecha_nac)
VALUES ('Benjamin', '1955-07-19');

select * from persona;
```

The screenshot shows the MySQL Workbench interface. On the left, the SQL editor contains the trigger definition and some test data. On the right, the Results tab displays the contents of the 'persona' table.

```
for each row
begin
    declare edad_calc integer;
    set edad_calc = timestampdiff(year, new.fecha_nac, curdate());
    set new.edad = edad_calc;
end;
INSERT INTO persona (nombre, fecha_nac)
VALUES ('Sol', '2002-03-19');

select * from persona;

CREATE TABLE PERSON
```

	id_per	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	7	Luz	Peralta Chipana	2000-05-10	22	luz@gmail.com	12	46	f
2	8	Jose	Fernandez Lopez	2001-04-04	21	josefer@gmail.com	14	23	m
3	9	Benjamin	<null>	1955-07-19	66	<null>	<null>	<null>	<null>
4	10	Joel	Pocoaca Castillo	2002-12-31	19	Joe@gmail.com	<null>	<null>	m
5	11	Sol	<null>	2002-03-19	20	<null>	<null>	<null>	<null>

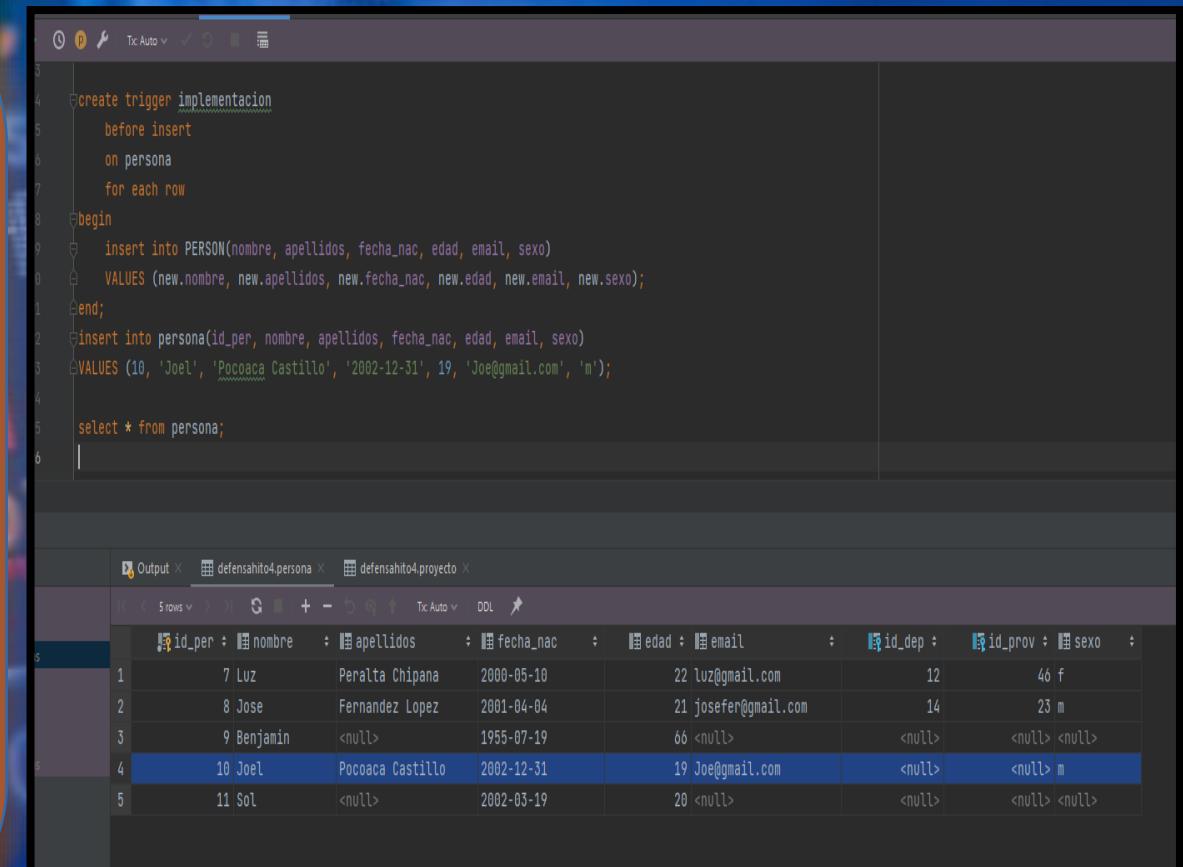
14. Manejo de TRIGGERS III.

- Crear otra tabla con los mismos campos de la tabla persona(Excepto el primary key id_per).

```
CREATE TABLE PERSON
(
    nombre VARCHAR(50),
    apellidos VARCHAR(50),
    fecha_nac date,
    edad INTEGER(11),
    email VARCHAR(50),
    sexo CHAR(1)
);
```

```
create trigger implementacion
before insert
on persona
for each row
begin
    insert into PERSON(nombre, apellidos, fecha_nac, edad, email, sexo)
    VALUES (new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo);
end;
insert into persona(id_per, nombre, apellidos, fecha_nac, edad, email, sexo)
VALUES (10, 'Joel', 'Pocoaca Castillo', '2002-12-31', 19, 'Joe@gmail.com', 'm');

select * from persona;
```



```
create trigger implementacion
before insert
on persona
for each row
begin
    insert into PERSON(nombre, apellidos, fecha_nac, edad, email, sexo)
    VALUES (new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo);
end;
insert into persona(id_per, nombre, apellidos, fecha_nac, edad, email, sexo)
VALUES (10, 'Joel', 'Pocoaca Castillo', '2002-12-31', 19, 'Joe@gmail.com', 'm');

select * from persona;
```

	id_per	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	7	Luz	Peralta Chipana	2000-05-10	22	luz@gmail.com	12	46	f
2	8	Jose	Fernandez Lopez	2001-04-04	21	josefer@gmail.com	14	23	m
3	9	Benjamin	<null>	1955-07-19	66	<null>	<null>	<null>	<null>
4	10	Joel	Pocoaca Castillo	2002-12-31	19	Joe@gmail.com	<null>	<null>	m
5	11	Sol	<null>	2002-03-19	20	<null>	<null>	<null>	<null>

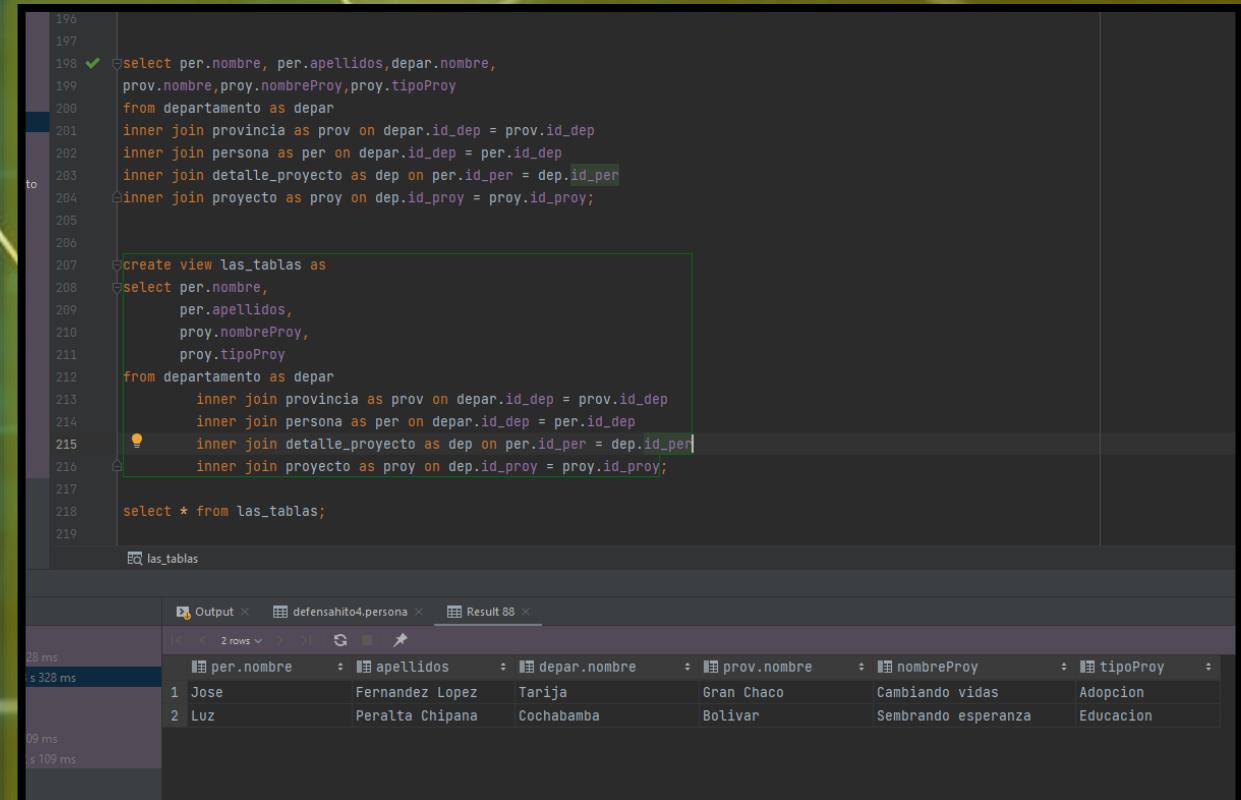
15. Crear una consulta SQL que haga uso de todas las tablas.

- La consulta generada convertirlo a VISTA

```
select per.nombre, per.apellidos, depar.nombre,  
prov.nombre, proy.nombreProy, proy.tipoProy  
from departamento as depar  
inner join provincia as prov on depar.id_dep = prov.id_dep  
inner join persona as per on depar.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;
```

```
create view las_tablas as  
select per.nombre,  
       per.apellidos,  
       proy.nombreProy,  
       proy.tipoProy  
  from departamento as depar  
         inner join provincia as prov on depar.id_dep = prov.id_dep  
         inner join persona as per on depar.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;
```

```
select * from las_tablas;
```



The screenshot shows a code editor with a dark theme. A green box highlights the SQL code for creating the 'las_tablas' view. The code includes joins between five tables: 'departamento', 'provincia', 'persona', 'detalle_proyecto', and 'proyecto'. The 'Output' tab at the bottom shows the results of the query, displaying two rows of data:

per.nombre	per.apellidos	depar.nombre	prov.nombre	nombreProy	tipoProy
1 Jose	Fernandez Lopez	Tarija	Gran Chaco	Cambiando vidas	Adopcion
2 Luz	Peralta Chipana	Cochabamba	Bolivar	Sembrando esperanza	Educacion



**¡GRACIAS POR SU
ATENCIÓN!**

ENLACE AL VIDEO

<https://drive.google.com/file/d/1yqgs4Ejyl9KT0aWIzM5dCGDgOho0xyUm/view?usp=sharing>

