

DG

DG



MariaDB

UNIVERSIDAD PRIVADA FRANZ TAMAYO

DEFENSA HITO 4 - TAREA FINAL

Estudiante: Rodrigo Cristhian Torrez De La Cruz

Asignatura: BASE DE DATOS II

Carrera: INGENIERÍA DE SISTEMAS

Sede: El Alto

Paralelo: BDA (1)

Docente: Lic. William Barra Paredes

fecha: 07/06/2023

GITHUB: <https://github.com/rodrigoT1/Base-de-Datos-II>

DG

DG

1. Defina que es lenguaje procedural en MySQL.

Resp. El lenguaje procedural en MySQL permite escribir y ejecutar programas o rutinas almacenadas dentro de la base de datos utilizando SQL procedimental. Esto incluye la creación de funciones, procedimientos almacenados, desencadenadores y eventos programados. Estas rutinas contienen instrucciones SQL, estructuras de control y manipulación de datos. El uso del lenguaje procedural en MySQL facilita la modularidad, mantenibilidad y rendimiento de las aplicaciones, al permitir encapsular y reutilizar lógica de programación compleja dentro de la base de datos. Algunos casos de uso comunes incluyen el cálculo y procesamiento de datos, la validación de reglas de negocio y la automatización de tareas recurrentes.

```
2
3 DROP PROCEDURE IF EXISTS procedimientol $$
4 CREATE PROCEDURE procedimientol ()
5 BEGIN
6     DECLARE i TINYINT UNSIGNED DEFAULT 1;
7     DROP TABLE IF EXISTS alumnos ;
8     CREATE TABLE alumnos
9         (id          INT PRIMARY KEY,
10          alumno VARCHAR(30))
11         ENGINE=innodb;
12 --
13 WHILE (i<=5) DO
14     INSERT INTO alumnos VALUES(i,CONCAT("alumno ",i));
15     SET i=i+1;
16 END WHILE;
17
18 END $$
19
```

2. Defina que es una FUCTION en MySQL.

Resp. Una función en SQL es un objeto de base de datos que realiza una tarea específica y devuelve un valor. Puede ser predefinida por el sistema de gestión de bases de datos o definida por el usuario. Las funciones predefinidas son proporcionadas por el DBMS y están listas para su uso, mientras que las funciones definidas por el usuario son creadas por el usuario para tareas específicas.

```
-- 10. Crear una función que genere los numeros serie Fibonacci.
CREATE FUNCTION generar_fibonacci(n INT)
RETURNS VARCHAR(1000)
BEGIN
    DECLARE respuesta VARCHAR(1000);
    DECLARE cont INT;
    DECLARE cont1 INT;
    DECLARE fib INT;
    DECLARE i INT;

    SET respuesta = '';
    SET cont = 1;
    SET cont1 = 0;
    SET i = 0;

    WHILE i < n DO
        SET respuesta = CONCAT(respuesta, cont, ',');

        SET fib = cont;
        SET cont = cont + cont1;
        SET cont1 = fib;

        SET i = i + 1;
    END WHILE;
    SET respuesta = LEFT(respuesta, LENGTH(respuesta) - 1); -- Eliminar la última coma
    RETURN respuesta;
END
```

```
SELECT generar_fibonacci( n: 9) AS fibonacci_sequence;
```

1 row	
	fibonacci_sequence
1	1,1,2,3,5,8,13,21,34



Manejo De Conceptos

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

Resp. Las funciones se utilizan para cálculos y devuelven un valor, mientras que los procedimientos almacenados se utilizan para realizar tareas más complejas y pueden no devolver valores o devolverlos mediante parámetros de salida. Las funciones se utilizan en consultas SQL, mientras que los procedimientos almacenados se llaman explícitamente desde una aplicación o una sentencia SQL.

funciones

```
CREATE FUNCTION sumar_fibonacci(n INT)
RETURNS INT
BEGIN
    DECLARE fib_current INT;
    DECLARE fib_previous INT;
    DECLARE fib_temp INT;
    DECLARE suma INT;
    DECLARE i INT;

    SET fib_current = 1;
    SET fib_previous = 0;
    SET suma = 0;
    SET i = 0;

    WHILE i < n DO
        SET suma = suma + fib_current;

        SET fib_temp = fib_current;
        SET fib_current = fib_current + fib_previous;
        SET fib_previous = fib_temp;

        SET i = i + 1;
    END WHILE;

    RETURN suma;
END;
```

procedimientos almacenados

```
SELECT sumar_fibonacci( n: 9) AS suma_fibonacci;
```

#EJERCICIO 11

Output		suma_fibonacci:int(11)
1 row		
	suma_fibonacci	
1		88

```
CREATE OR REPLACE PROCEDURE inserta_datos(
    fecha TEXT,
    usuario TEXT,
    hostname TEXT,
    accion TEXT,
    antes TEXT,
    despues TEXT
)
BEGIN

    INSERT INTO audit_usuarios_rrhh1 (fecha_mod, usuario_log, histname, accion, antes_del_cambio, despues_del_cambio)
    VALUES (fecha, usuario, hostname, accion, antes, despues);

END;
```




Manejo De Conceptos

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

Resp. Para ejecutar una función en SQL, puedes llamarla dentro de una consulta SELECT u otra expresión donde se espere un valor. Aquí tienes un ejemplo de cómo ejecutar una función:

función

```
SELECT sumar_fibonacci(n: 9) AS suma_fibonacci;
```

Output suma_fibonacci: int(11) ×

	suma_fibonacci
1	88

procedimiento almacenado.

```
INSERT INTO audit_usuarios_rrhh1 (fecha_mod, usuario_log, histname, accion, antes_del_cambio, despues_del_cambio)  
VALUES (fecha, usuario, hostname, accion, antes, despues);
```



Manejo De Conceptos

5. Defina que es una TRIGGER en MySQL.

Resp. Un trigger en MySQL es un objeto de base de datos asociado a una tabla que se activa automáticamente en respuesta a eventos específicos, como inserciones, actualizaciones o eliminaciones de registros en la tabla. Su función principal es ejecutar un conjunto de instrucciones definidas por el usuario cuando ocurre el evento desencadenante. Estas instrucciones pueden incluir acciones como modificar registros en otras tablas, realizar cálculos o validar condiciones adicionales. Los triggers se definen mediante sentencias SQL y pueden utilizarse para mantener la integridad referencial de los datos, auditar cambios o realizar validaciones adicionales. Sin embargo, es importante tener en cuenta el impacto en el rendimiento y utilizarlos con precaución.

```
-- Manejo de Triggers II.
CREATE TRIGGER calculaEdad
BEFORE INSERT ON PERSONA
FOR EACH ROW
BEGIN
    DECLARE birth_date DATE;
    DECLARE age INT;

    SET birth_date = NEW.fecha_nac;
    SET age = TIMESTAMPDIFF(YEAR, birth_date, CURDATE());

    IF DATE_FORMAT(birth_date, '%m%d') > DATE_FORMAT(CURDATE(), '%m%d') THEN
        SET age = age - 1;
    END IF;

    SET NEW.edad = age;
END;
```

```
SELECT *,
    TIMESTAMPDIFF(YEAR, fecha_nac, CURDATE()) -
    (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(fecha_nac, '%m%d')) AS edad
FROM PERSONA;
```

-- EJERCICIO 14

_per	nombre	apellido	fecha_nac	persona.edad	email	id_dep	id_prov	sexo	edad
1	Ana	González	2000-10-10	23	Ana@Gonzalez.com	1	1	F	21
2	Pedro	López	1995-05-10	28	pedro@Lopez.com	1	2	M	28
3	Emeth	Brount	1885-01-20	36	Emeth@Brount.com	2	3	M	138



Manejo De Conceptos

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

Resp. En un trigger de MySQL, las variables OLD y NEW desempeñan un papel fundamental al permitir acceder a los valores antiguos y nuevos de los campos de una tabla durante un evento desencadenante.

La variable OLD contiene los valores antiguos de los campos antes de que ocurra el evento desencadenante, mientras que la variable NEW contiene los valores nuevos o modificados después del evento. Estas variables se utilizan en los triggers de actualización, inserción y eliminación para acceder y utilizar los valores de los campos afectados.

El uso de las variables OLD y NEW en un trigger permite realizar acciones adicionales basadas en los valores antiguos y nuevos. Por ejemplo, se pueden realizar validaciones, realizar actualizaciones en otras tablas o realizar auditorías de cambios.

```
INSERT INTO copia_persona (nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo)
VALUES ('Maria', 'Flores', '1887-10-11', 21, 'Maria@Flores.com', 1, 1, 'F'),
       ('George', 'Mcfly', '1995-05-10', 36, 'George@Mcfly.com', 1, 2, 'M'),
       ('Aylin', 'Luna', '2000-04-05', 29, 'Aylin@Luna.com', 2, 3, 'F');

CREATE TRIGGER before_insert_persona
BEFORE INSERT ON Persona
FOR EACH ROW
BEGIN
    INSERT INTO copia_persona (nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo)
    VALUES (NEW.nombre, NEW.apellido, NEW.fecha_nac, NEW.edad, NEW.email, NEW.id_dep, NEW.id_prov, NEW.sexo);
END;

SELECT * FROM copia_persona;
```


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

Resp. Con la cláusula BEFORE, el trigger se ejecuta previamente al evento desencadenante, mientras que con la cláusula AFTER, el trigger se ejecuta posteriormente al evento desencadenante. Estas cláusulas permiten realizar acciones y lógica adicional tanto antes como después de una operación en la tabla. La elección de usar una u otra depende de las necesidades específicas y el momento adecuado para realizar ciertas acciones en relación con la operación en la tabla.

BEFORE

```
BEFORE INSERT ON PERSONA
FOR EACH ROW
BEGIN
    DECLARE birth_date DATE;
    DECLARE age INT;

    SET birth_date = NEW.fecha_nac;
    SET age = TIMESTAMPDIF(YEAR, birth_date, CURDATE());

    IF DATE_FORMAT(birth_date, '%m%d') > DATE_FORMAT(CURDATE(), '%m%d') THEN
        SET age = age - 1;
    END IF;

    SET NEW.edad = age;
END;

SELECT *,
    TIMESTAMPDIF(YEAR, fecha_nac, CURDATE()) -
    (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(fecha_nac, '%m%d')) AS edad
FROM PERSONA;
```

AFTER

```
CREATE OR REPLACE TRIGGER tr_audit_usuarios_rrhh1
AFTER UPDATE
ON usuarios_rrhh
FOR EACH ROW
BEGIN
    declare antes text default concat(OLD.id_usr, ' ', OLD.nombre_completo, OLD.correo);
    declare despues text default concat(OLD.id_usr, ' ', NEW.nombre_completo, NEW.correo);

    call inserta_datos(
        now(),
        user(),
        @@hostname,
        'UPDATE',
        antes,
        despues
    );
end;

select * from usuarios_rrhh;
select * from audit_usuarios_rrhh;
```




Manejo De Conceptos

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

Resp. Cuando se habla de "eventos" en el contexto de los triggers, se refiere a las acciones que ocurren en una tabla y que pueden desencadenar la ejecución de un trigger. Estos eventos pueden ser inserciones, actualizaciones o eliminaciones de registros en la tabla.

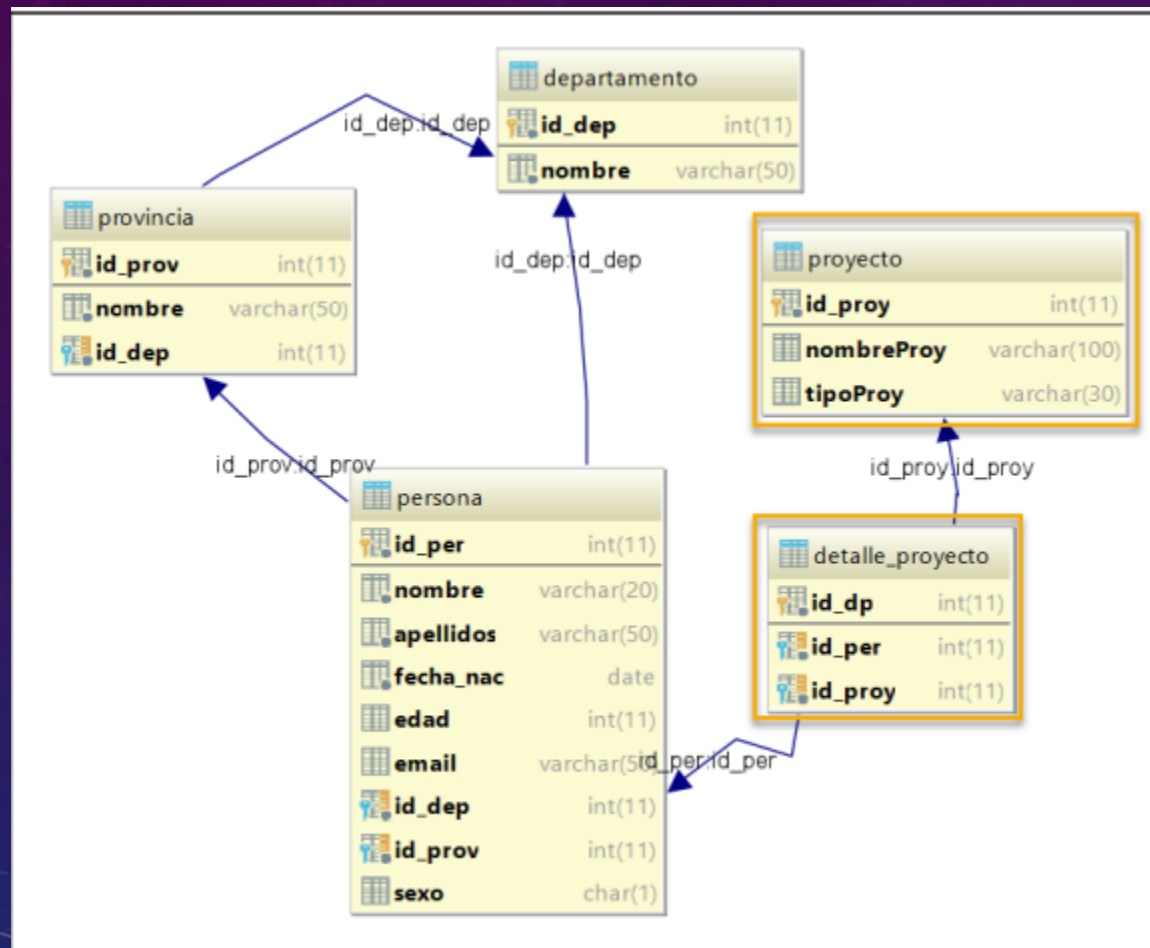
En otras palabras, un evento es una operación específica que se realiza en una tabla y que activa un trigger asociado a esa tabla. Por ejemplo, si se inserta un nuevo registro en una tabla, se realiza una acción de inserción y puede activar un trigger BEFORE INSERT o AFTER INSERT según cómo esté configurado.



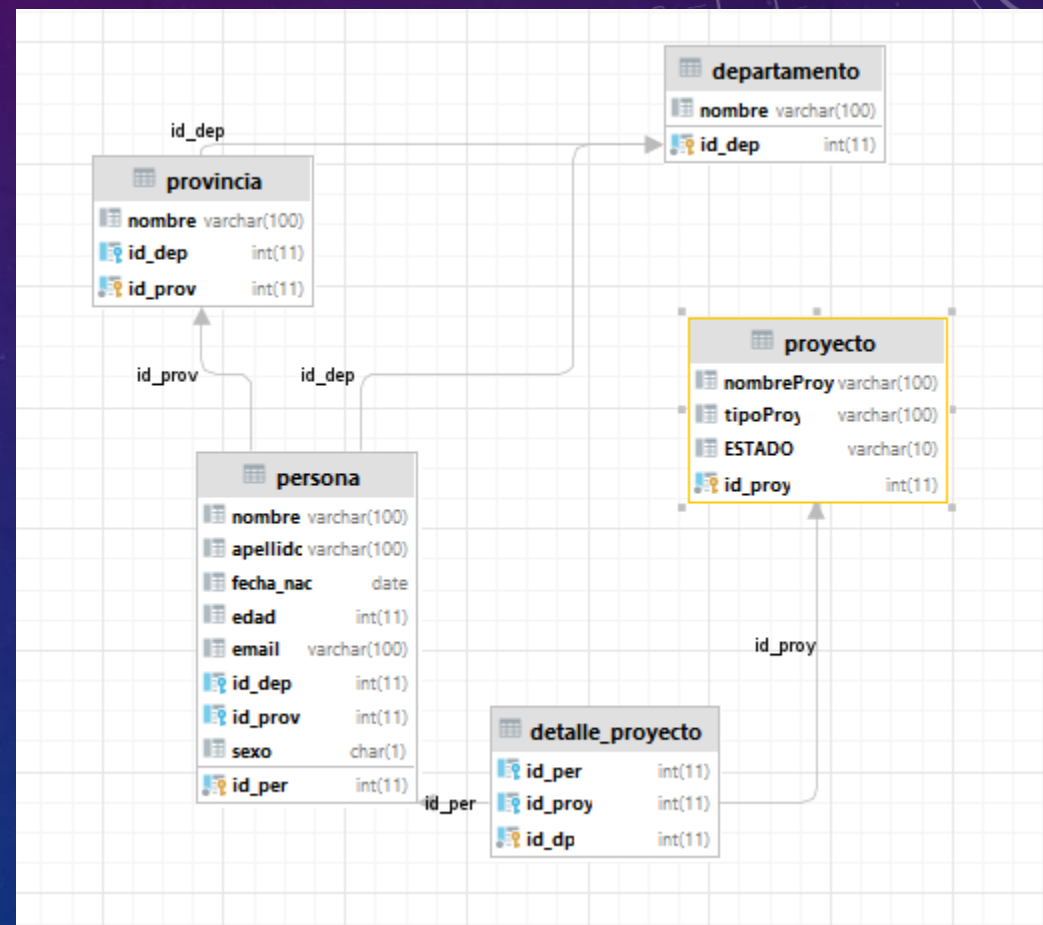
PARTE PRACTICA



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



Agregar mínimamente 2 registros a cada tabla



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

```
-- 10. Crear una función que genere los numeros serie Fibonacci.
CREATE FUNCTION generar_fibonacci(n INT)
RETURNS VARCHAR(1000)
BEGIN
    DECLARE respuesta VARCHAR(1000);
    DECLARE cont INT;
    DECLARE cont1 INT;
    DECLARE fib INT;
    DECLARE i INT;

    SET respuesta = '';
    SET cont = 1;
    SET cont1 = 0;
    SET i = 0;

    WHILE i < n DO
        SET respuesta = CONCAT(respuesta, cont, ',');

        SET fib = cont;
        SET cont = cont + cont1;
        SET cont1 = fib;

        SET i = i + 1;
    END WHILE;
    SET respuesta = LEFT(respuesta, LENGTH(respuesta) - 1);
    RETURN respuesta;
END;
```

```
SELECT generar_fibonacci(n: 9) AS fibonacci_sequence;
```

sequence

Output fibonacci_sequence:varchar(1000) X

1 row

fibonacci_sequence

1 1,1,2,3,5,8,13,21,34



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

```
-- crear una funcion que sume los numeros fibonacci
CREATE FUNCTION sumar_fibonacci(n INT)
RETURNS INT
BEGIN
    DECLARE fib_current INT;
    DECLARE fib_previous INT;
    DECLARE fib_temp INT;
    DECLARE suma INT;
    DECLARE i INT;

    SET fib_current = 1;
    SET fib_previous = 0;
    SET suma = 0;
    SET i = 0;












    WHILE i < n DO
        SET suma = suma + fib_current;

        SET fib_temp = fib_current;
        SET fib_current = fib_current + fib_previous;
        SET fib_previous = fib_temp;

        SET i = i + 1;
    END WHILE;

    RETURN suma;
END;
```

```
SELECT sumar_fibonacci(n: 9) AS suma_fibonacci;
```

<div><div> Output</div><div> suma_fibonacci:int(11) ×</div></div>	
<div><div>  1 row  </div><div>   </div></div>	
	<div><div> suma_fibonacci ⌵</div></div>
1	88



11. Manejo de vistas.

- o 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
- o 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'

```
CREATE VIEW Vista_Personas AS
SELECT CONCAT(Persona.nombre, ' ', Persona.apellido) AS nombres_apellidos,
       Persona.edad,
       Persona.fecha_nac,
       Proyecto.nombreProy AS nombre_proyecto
FROM Persona
JOIN Detalle_proyecto ON Persona.id_per = Detalle_proyecto.id_per
JOIN Proyecto ON Detalle_proyecto.id_proy = Proyecto.id_proy;

SELECT * FROM Vista_Personas;
```

```
SELECT CONCAT(Persona.nombre, ' ', Persona.apellido) AS nombres_apellidos,
       Persona.edad,
       Persona.fecha_nac,
       Proyecto.nombreProy AS nombre_proyecto
FROM Persona
JOIN Departamento ON Persona.id_dep = Departamento.id_dep
JOIN Detalle_proyecto ON Persona.id_per = Detalle_proyecto.id_per
JOIN Proyecto ON Detalle_proyecto.id_proy = Proyecto.id_proy
WHERE Persona.sexo = 'F' AND
       Departamento.nombre = 'El Alto' AND
       Persona.fecha_nac = '2000-10-10';
```

Output Result 5				
1 row				
	nombres_apellidos	edad	fecha_nac	nombre_proyecto
1	Ana González	23	2000-10-10	carretera



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 6 ○ 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

```
-- Agregar el nuevo campo "ESTADO" a la tabla "Proyecto"
ALTER TABLE Proyecto
ADD COLUMN ESTADO VARCHAR(10);
```

```
-- Trigger 1: Se ejecuta antes de insertar un nuevo registro en la tabla
CREATE TRIGGER before_insert_proyecto
BEFORE INSERT ON PROYECTO
FOR EACH ROW
BEGIN
    IF NEW.tipoProy IN ('EDUCACION', 'FORESTACION', 'CULTURA') THEN
        SET NEW.ESTADO = 'ACTIVO';
    ELSE
        SET NEW.ESTADO = 'INACTIVO';
    END IF;
END;

SELECT *,
CASE
    WHEN tipoProy IN ('EDUCACION', 'FORESTACION', 'CULTURA') THEN 'ACTIVO'
    ELSE 'ACTIVOS'
END AS ESTADO
FROM PROYECTO;
```

```
-- Trigger 2: Se ejecuta antes de actualizar un registro existente en la tabla
CREATE TRIGGER before_update_proyecto
BEFORE UPDATE ON PROYECTO
FOR EACH ROW
BEGIN
    IF NEW.tipoProy IN ('EDUCACION', 'FORESTACION', 'CULTURA') THEN
        SET NEW.ESTADO = 'ACTIVO';
    ELSE
        SET NEW.ESTADO = 'INACTIVO';
    END IF;
END;

SELECT *,
CASE
    WHEN tipoProy IN ('EDUCACION', 'FORESTACION', 'CULTURA') THEN 'ACTIVO'
    ELSE 'INACTIVO'
END AS ESTADO
FROM PROYECTO;
```



13. Manejo de Triggers II.

- o El trigger debe de llamarse calculaEdad.

- o El evento debe de ejecutarse en un BEFORE INSERT.

- o 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.

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

```
-- Manejo de Triggers II.
CREATE TRIGGER calculaEdad
BEFORE INSERT ON PERSONA
FOR EACH ROW
BEGIN
    DECLARE birth_date DATE;
    DECLARE age INT;

    SET birth_date = NEW.fecha_nac;
    SET age = TIMESTAMPDIFF(YEAR, birth_date, CURDATE());

    IF DATE_FORMAT(birth_date, '%m%d') > DATE_FORMAT(CURDATE(), '%m%d') THEN
        SET age = age - 1;
    END IF;

    SET NEW.edad = age;
END;
```

```
SELECT *,
    TIMESTAMPDIFF(YEAR, fecha_nac, CURDATE()) -
    (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(fecha_nac, '%m%d')) AS edad
FROM PERSONA;
```

_per	nombre	apellido	fecha_nac	persona.edad	email	id_dep	id_prov	sexo	edad
1	Ana	González	2000-10-10	23	Ana@Gonzalez.com	1	1	F	21
2	Pedro	López	1995-05-10	28	pedro@Lopez.com	1	2	M	28
3	Emeth	Brount	1885-01-20	36	Emeth@Brount.com	2	3	M	138



14. Manejo de TRIGGERS III.

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

■ No es necesario que tenga PRIMARY KEY.

o Cada vez que se haga un INSERT a la tabla persona estos mismos valores deben insertarse a la tabla copia.

o Para resolver esto deberá de crear un trigger before insert para la tabla PERSONA.

```
-- Manejo de TRIGGERS III.
CREATE TABLE copia_persona (
  nombre VARCHAR(100),
  apellido VARCHAR(100),
  fecha_nac DATE,
  edad INT,
  email VARCHAR(100),
  id_dep INT,
  id_prov INT,
  sexo CHAR(1)
);
```

```
INSERT INTO copia_persona (nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo)
VALUES ('Maria', 'Flores', '1887-10-11', 21, 'Maria@Flores.com', 1, 1, 'F'),
      ('George', 'Mcfly', '1995-05-10', 36, 'George@Mcfly.com', 1, 2, 'M'),
      ('Aylin', 'Luna', '2000-04-05', 29, 'Aylin@Luna.com', 2, 3, 'F');
```

```
CREATE TRIGGER before_insert_persona
BEFORE INSERT ON Persona
FOR EACH ROW
BEGIN
  INSERT INTO copia_persona (nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo)
  VALUES (NEW.nombre, NEW.apellido, NEW.fecha_nac, NEW.edad, NEW.email, NEW.id_dep, NEW.id_prov, NEW.sexo);
END;

SELECT * FROM copia_persona;
```




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

```
-- Crear una consulta SQL que haga uso de todas las tablas.
-- Ejecutar consulta
SELECT CONCAT(Persona.nombre, ' ', Persona.apellido) AS nombres_apellidos,
       Persona.edad,
       Persona.fecha_nac,
       Proyecto.nombreProy AS nombre_proyecto
FROM Persona
JOIN Detalle_proyecto ON Persona.id_per = Detalle_proyecto.id_per
JOIN Proyecto ON Detalle_proyecto.id_proy = Proyecto.id_proy;

-- Crear vista
CREATE VIEW Vista_Personas AS
SELECT CONCAT(Persona.nombre, ' ', Persona.apellido) AS nombres_apellidos,
       Persona.edad,
       Persona.fecha_nac,
       Proyecto.nombreProy AS nombre_proyecto
FROM Persona
JOIN Detalle_proyecto ON Persona.id_per = Detalle_proyecto.id_per
JOIN Proyecto ON Detalle_proyecto.id_proy = Proyecto.id_proy;

CREATE PROCEDURE obtenerSuma(IN numero1 INT, IN numero2 INT, OUT suma INT)
BEGIN
    SET suma = numero1 + numero2;
END
```