



Lenguaje procedural, funciones, vistas y triggers

Romario Tola Quispe

Preguntas teóricas

1. Defina que es lenguaje procedural en MySQL.

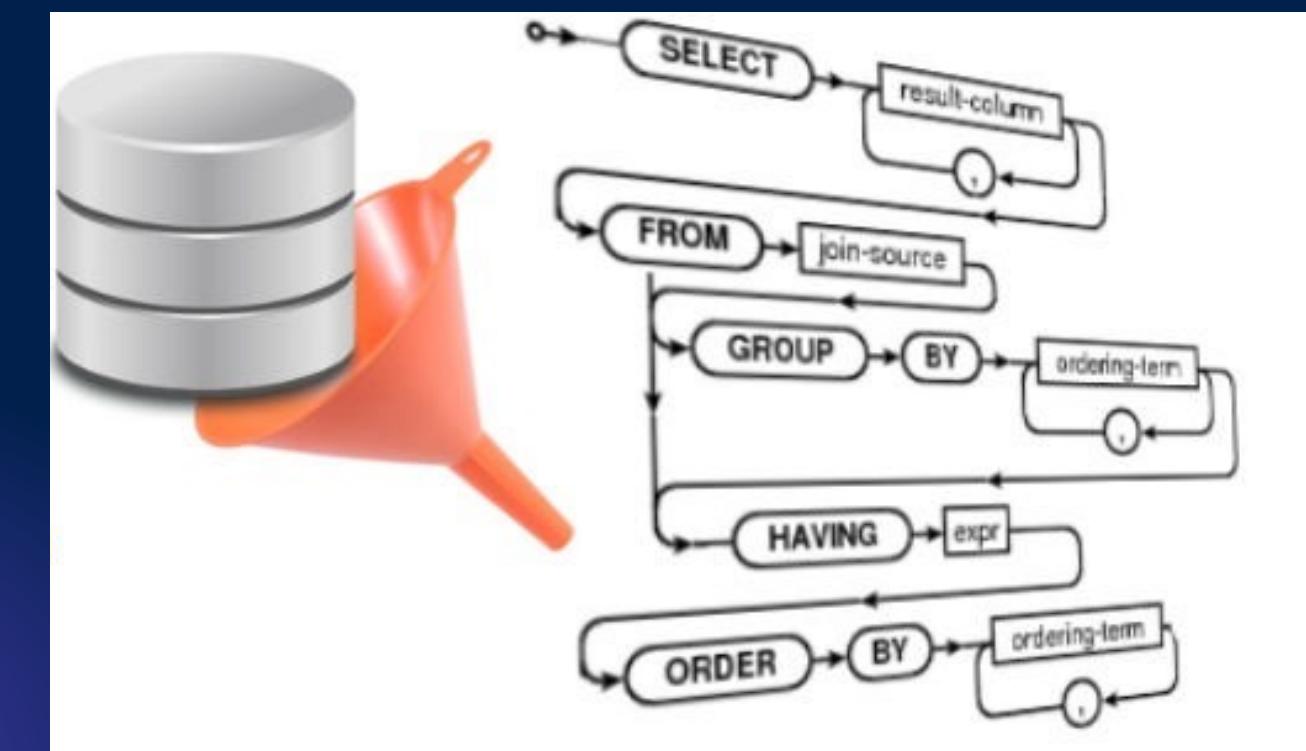


Es un tipo de lenguaje de programación que se basa en una serie de instrucciones que se ejecutan de manera secuencial para completar una tarea o resolver un problema.

El SQL procedimental de MySQL permite a los desarrolladores escribir bloques de código que se ejecutan de manera secuencial

2. Defina que es una función 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.

- Valor de retorno: Una diferencia fundamental es que las funciones siempre tienen un valor de retorno, mientras que los procedimientos almacenados pueden no tenerlo.
- Contexto de ejecución: Las funciones se evalúan en el contexto de una consulta y pueden interactuar directamente con los datos en la base de datos. Por otro lado, los procedimientos almacenados se ejecutan en un contexto separado y pueden contener operaciones más complejas, como transacciones y manipulación de variables, que no están directamente relacionadas con las consultas.



VS



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

1. Para ejecutar un procedimiento almacenado en MySQL, primero debes asegurarte de tener los permisos adecuados para ejecutarlo.
2. Utiliza la sentencia "CALL" seguida del nombre del procedimiento almacenado y los parámetros necesarios entre paréntesis.

```
CALL nombre_procedimiento(parametro1, parametro2);
```

1. Para ejecutar una función en MySQL, puedes utilizarla directamente dentro de una consulta SQL.
2. Utiliza la función en una expresión o como parte de una sentencia SELECT.

```
SELECT nombre_funcion(parametro1, parametro2);
```

5. Defina que es una TRIGGER en MySQL.

Un trigger o disparador es una regla que se asocia a una tabla. Mediante esta regla, se ejecutan una serie de instrucciones cuando se producen ciertos eventos sobre una tabla. Los eventos son: INSERT, UPDATE o DELETE.



A que se refiere cuando se habla de eventos en TRIGGERS

Son las operaciones o acciones específicas que ocurren en una tabla y que desencadenan la ejecución del trigger. Estos eventos pueden ser:

Inserción (INSERT): Ocurre cuando se agrega una nueva fila a la tabla.

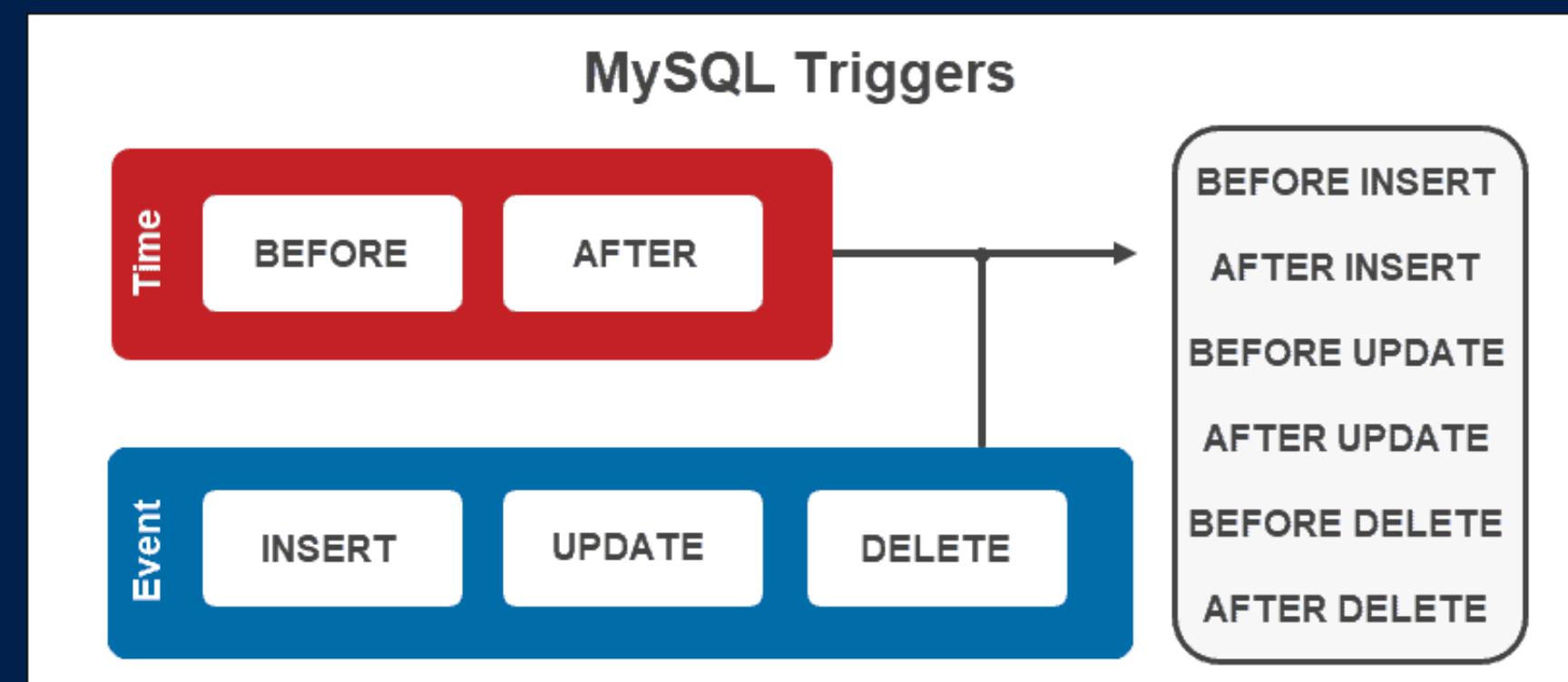
Actualización (UPDATE): Ocurre cuando se modifican los valores de una o varias filas existentes en la tabla.

Eliminación (DELETE): Ocurre cuando se elimina una o varias filas de la tabla.

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

La cláusula "BEFORE" activa el desencadenador antes de la operación de modificación, permitiendo realizar acciones previas o validaciones adicionales.

La cláusula "AFTER" activa el desencadenador después de la operación de modificación, lo que permite realizar acciones posteriores o trabajar con los valores actualizados.



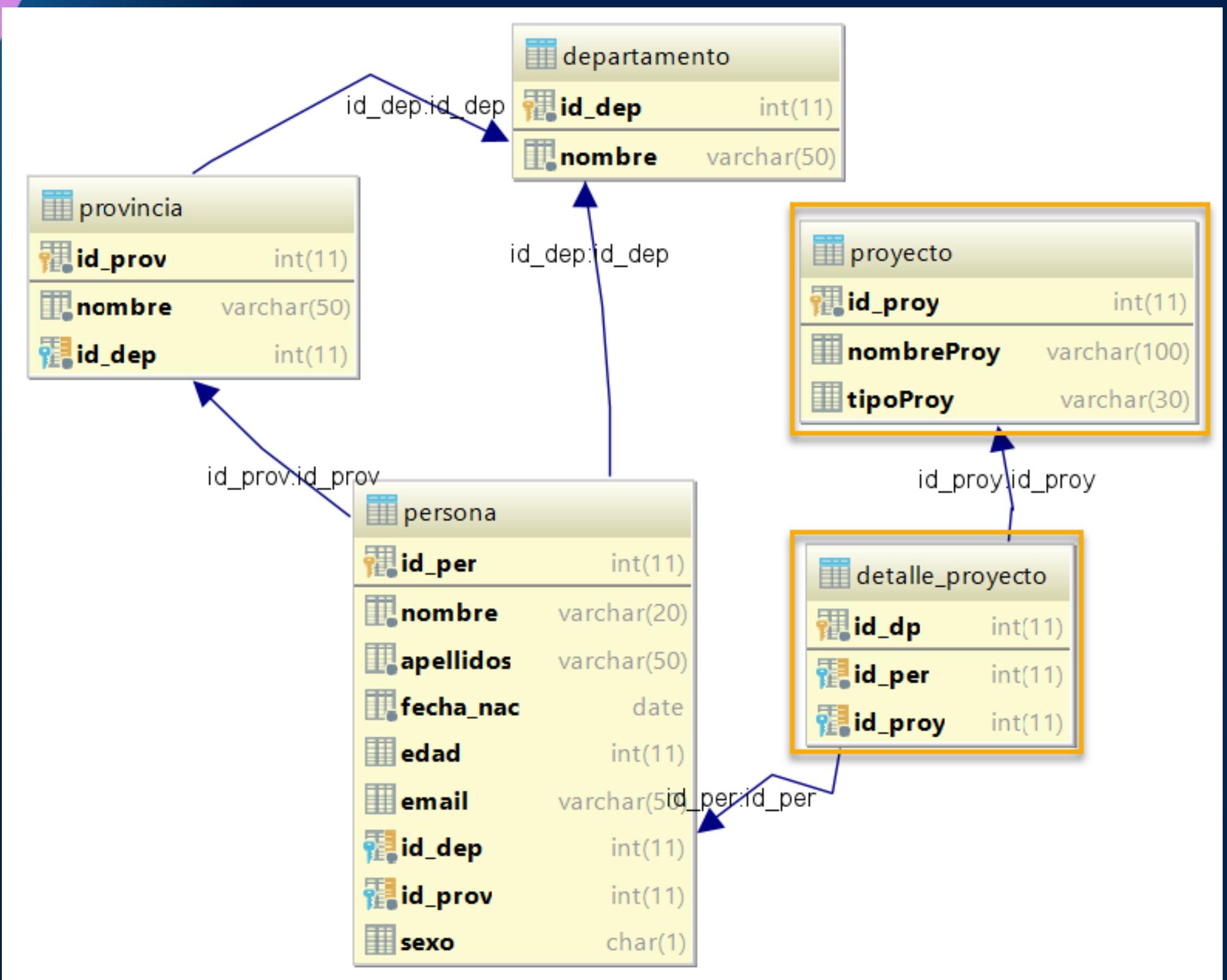
En un trigger que papel juega las variables OLD y NEW

En MySQL, los términos "OLD" y "NEW" se utilizan en los desencadenadores (triggers) para referirse a los valores antiguos y nuevos de una fila afectada durante una operación de modificación de datos (como una inserción, actualización o eliminación).

Cuando se ejecuta un desencadenador, se activa en respuesta a un evento específico, como una inserción, actualización o eliminación de una fila en una tabla. Durante la ejecución del desencadenador, se pueden acceder a los valores antiguos y nuevos de la fila afectada utilizando las palabras clave "OLD" y "NEW".

Preguntas prácticas

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



Agregar
minima
mente 2
registro
s a cada
tabla

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.

	<code>^ fibonacci(10)</code>
1	0, 1, 1, 2, 3, 5, 8, 13, 21, 34,



<	<	1 row	>	>	G	↻	✖	✖

	<code>^ sumFibonacci(10)</code>
1	88

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

```
-- 10.Crear una función que sume los valores de la serie Fibonacci.  
--  
CREATE OR REPLACE FUNCTION fibonacci_serie(limite INT)  
RETURNS TEXT  
BEGIN  
    DECLARE i INT DEFAULT 0;  
  
    DECLARE cadena TEXT DEFAULT '';  
    DECLARE num INT DEFAULT 0;  
    DECLARE nuevoNumero INT DEFAULT 1;  
    DECLARE acum INT;  
  
    WHILE (i < limite) DO  
        SET cadena = concat(cadena, num, ',');  
        SET acum = num;  
        SET num = nuevoNumero;  
        SET nuevoNumero = acum + nuevoNumero;  
  
        SET i = i+1;  
    END WHILE;  
  
    RETURN cadena;  
END;
```

Output fibonacci_serie(8):text

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

```
CREATE OR REPLACE FUNCTION suma_serie_fibonacci (cadenaFibonacci text)
RETURNS INT
BEGIN
    DECLARE ubicacion_Coma INT;
    DECLARE nuevo_numero int;
    DECLARE resultado int default 0;
    WHILE (cadenaFibonacci != '') DO
        SET ubicacion_Coma = locate(',',cadenaFibonacci);
        set nuevo_numero = substr(cadenaFibonacci,1,ubicacion_Coma-1);
        set resultado = resultado + nuevo_numero;
        set cadenaFibonacci = substr(cadenaFibonacci,ubicacion_Coma+1);
    END while;
    return resultado;
END;
select suma_serie_fibonacci( fibonacci_serie( limite: 40));
```

The screenshot shows the MySQL Workbench interface with the following details:

- Output Tab:** The tab is titled "suma_serie_fibonacci(fibonacci_serie(12)):int(11)".
- Result Grid:** The grid displays the output of the function call. The first row has the header "suma_serie_fibonacci(fibonacci_serie(12))". The second row contains the value "232".
- Status Bar:** The status bar at the bottom right shows the number "232".

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

- Deberá 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


```
INSERT INTO proyecto
( nombre_proy, tipo_proy)
VALUES('nombreProy','cultura'),
      ('nombreProy2','diseño');

select * from proyecto;

UPDATE proyecto AS proy
SET proy.tipo_proy = 'invsetigacion' where proy.tipo_proy = 'cultura';

select * from proyecto;
```

#	SISTEMA DE FACTURACION	PROYECTO	ESTADO
3	nombreProy	cultura	ACTIVO
4	nombreProy2	diseño	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.

```
CREATE OR REPLACE TRIGGER tr_calculaEdad
  BEFORE INSERT
  ON persona
  FOR EACH ROW
BEGIN
  SET NEW.edad = TIMESTAMPDIFF(YEAR, NEW.fecha_nac , CURDATE());
END;

insert into persona ( nombre, apellidos, fecha_nac, email, direccion, id_dep, id_prov, sexo)
values ('Mijail','Torn','2004-09-16','alb@gmail.com','Av. Paseo Tablado',1,1,'M');

select * from persona;
```

7	Mijail	Torn	2004-09-16	18	alb@gmail.com	Av. Paseo Tablado	1	1	M

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.


```
insert into persona ( nombre, apellidos, fecha_nac, email, direccion, id_dep, id_prov, sexo)
values ('albert','Torn','2000-06-20','alb@gmail.com','Av. Paseo Tablado',1,1,'M');

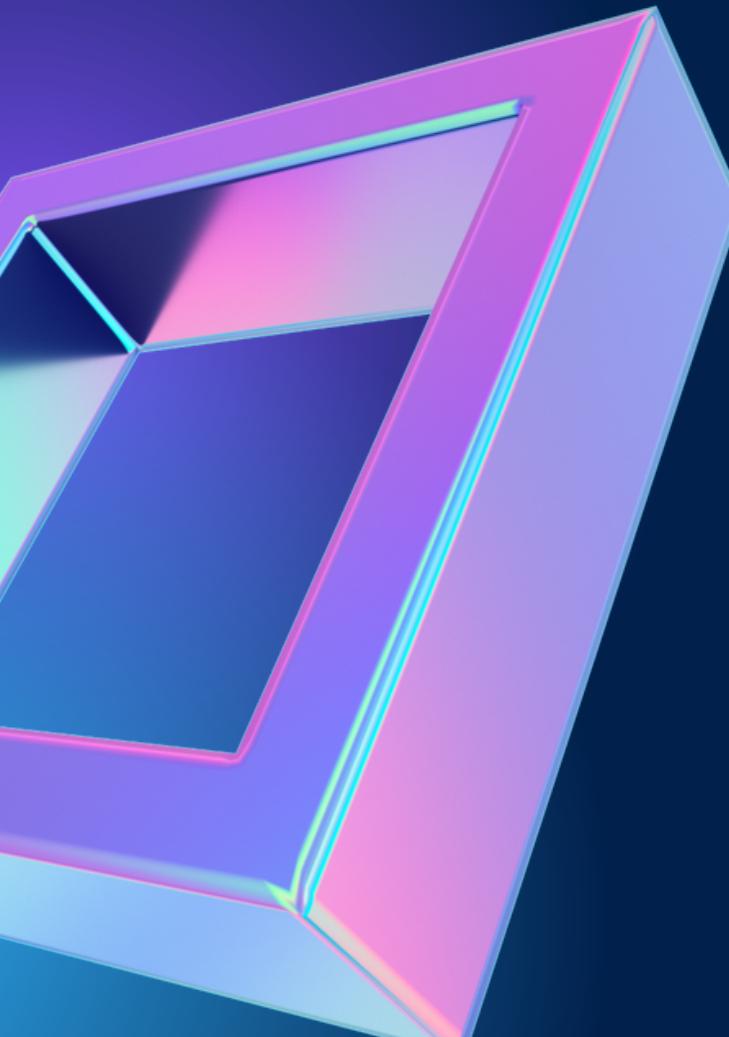
select * from persona;
select * from audit_persona;
```

Output procesual_hito4.audit_persona

	nombre	apellidos	fecha_nac	edad	email	direccion	id_dep	id_prov	sexo
1	albert	Torn	2000-06-20	22	alb@gmail.com	Av. Paseo Tablado	1	1	M

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

- La consulta generada convertirlo a VISTA**
- Resultado esperado.**



GRACIAS!!