



# Evaluación Procesual

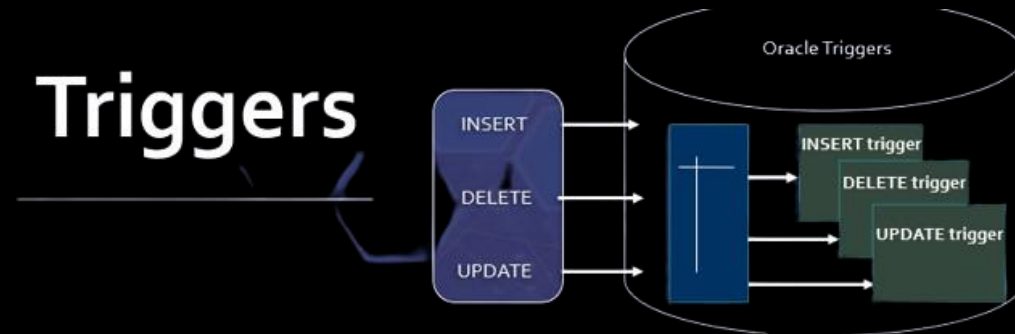
## Hito 4

**Estudiante:** Heber Mollericona Miranda

**CI:** 13409189

# Defina que es lenguaje procedural en MySQL

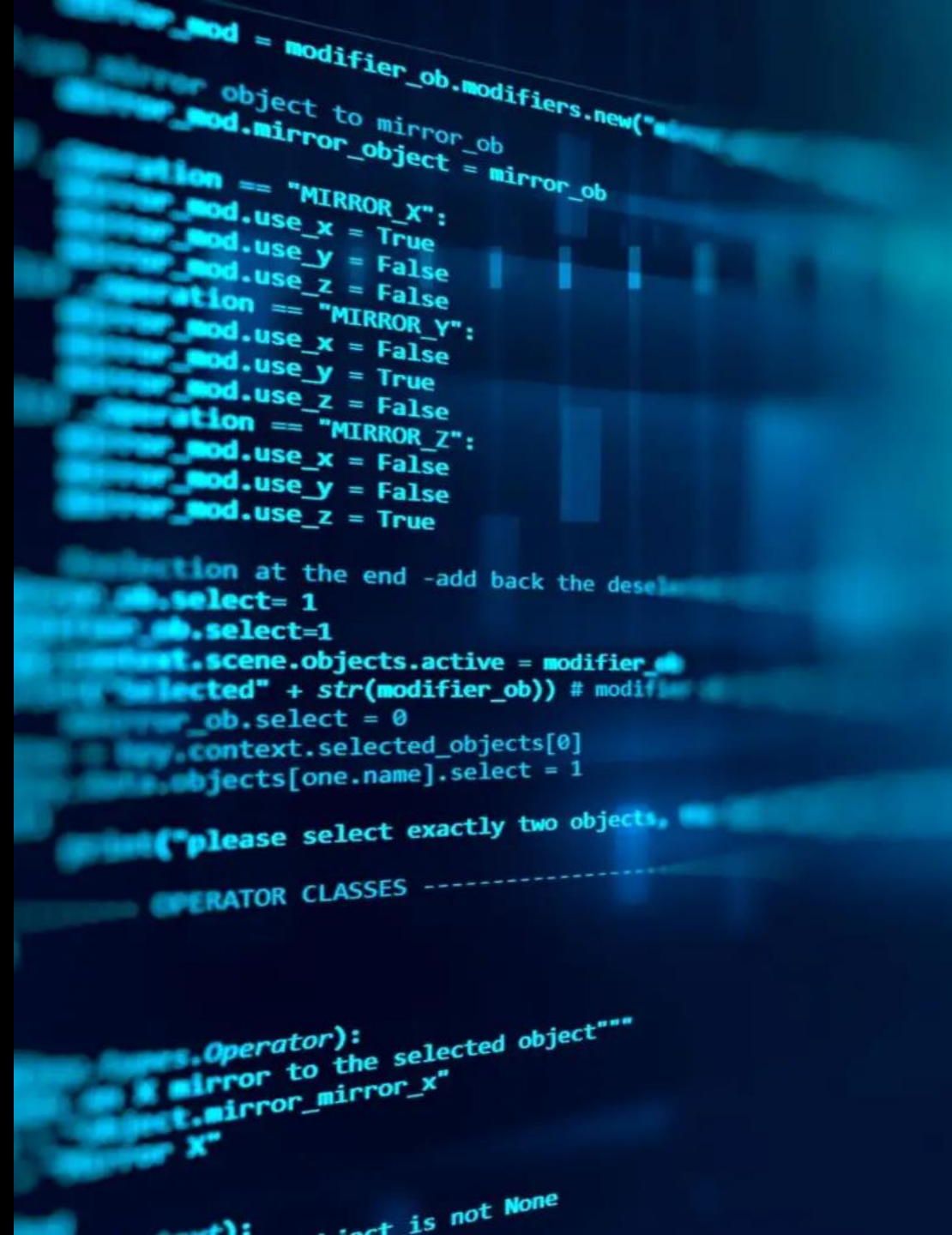
En MySQL, el lenguaje procedural se refiere a un conjunto de instrucciones y bloques de código que se utilizan para definir procedimientos almacenados y funciones. Estas son estructuras que permiten escribir código más complejo y lógico dentro del propio motor de la base de datos.



# Defina que es una FUNCTION en MySQL

En MySQL, una función es un objeto de base de datos que realiza un cálculo o una operación y devuelve un valor.

Puede recibir parámetros de entrada, realizar una serie de operaciones y luego devolver un resultado. Las funciones se utilizan comúnmente para realizar cálculos y transformaciones de datos dentro de las consultas.





# Cuál es la diferencia entre funciones y procedimientos almacenados

La diferencia principal entre funciones y procedimientos almacenados en MySQL radica en su propósito y forma de uso. Las funciones están diseñadas para devolver un valor específico, mientras que los procedimientos almacenados son secuencias de comandos que pueden realizar operaciones complejas pero no necesariamente devuelven un valor.

# Cómo se ejecuta una función y un procedimiento almacenado

Para ejecutar una función en MySQL, se puede utilizar en una consulta SQL como cualquier otro elemento en la cláusula SELECT. Por ejemplo: `SELECT nombre_funcion(parametros).` En cambio, para ejecutar un procedimiento almacenado, se utiliza la instrucción `CALL` seguida del nombre del procedimiento y los parámetros si los requiere. Por ejemplo: `CALL nombre_procedimiento(parametros).`

```
create or replace function sumar_fibonacci(limite integer)
returns integer
begin
    declare cadena text default fibonacci(limite:limite);
    declare suma integer default 0;
    declare pos_coma integer;

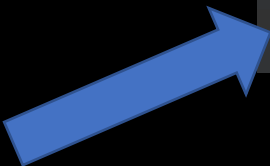
    while char_length(cadena) > 0 do

        set pos_coma = locate(',', cadena);

        if pos_coma = 0 then
            set suma = suma + cast(cadena as integer);
            set cadena = '';
        else
            set suma = suma + cast(substring(cadena, 1, pos_coma - 1) as integer);
            set cadena = substring(cadena, pos_coma + 2);
        end if;
    end while;

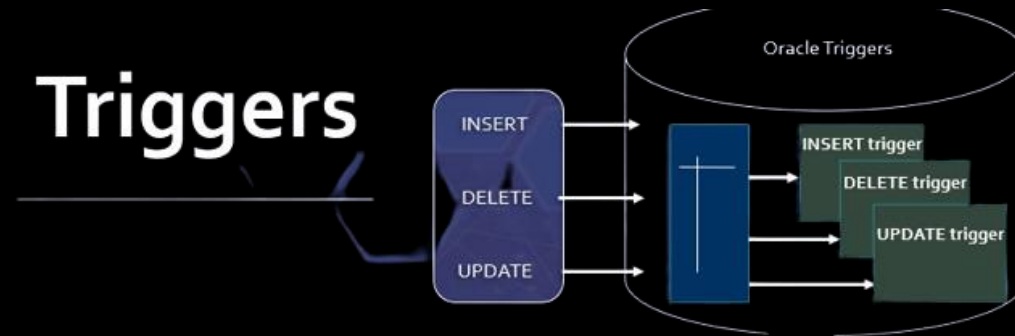
    return suma;
end;

select sumar_fibonacci(limite: 13);
```



# Defina que es una **TRIGGER** en MySQL

Un trigger en MySQL es un tipo especial de objeto de base de datos que se asocia con una tabla y se activa automáticamente cuando se realiza una operación (INSERT, UPDATE o DELETE) en esa tabla. Un trigger permite ejecutar un conjunto de instrucciones o acciones cuando se cumple una condición específica, como modificar otra tabla o generar un registro de auditoría.



# En un trigger que papel juega las variables OLD y NEW

En un trigger de MySQL, las variables OLD y NEW se utilizan para acceder a los valores antiguos y nuevos de las filas afectadas por la operación que disparó el trigger. La variable OLD contiene los valores antiguos antes de la operación y la variable NEW contiene los valores nuevos después de la operación. Estas variables se utilizan comúnmente en los triggers para realizar comparaciones y aplicar lógica basada en los cambios realizados en la tabla.



old



new

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

Las cláusulas BEFORE y AFTER en un trigger de MySQL definen cuándo se ejecutará el trigger en relación con la operación que lo activa.

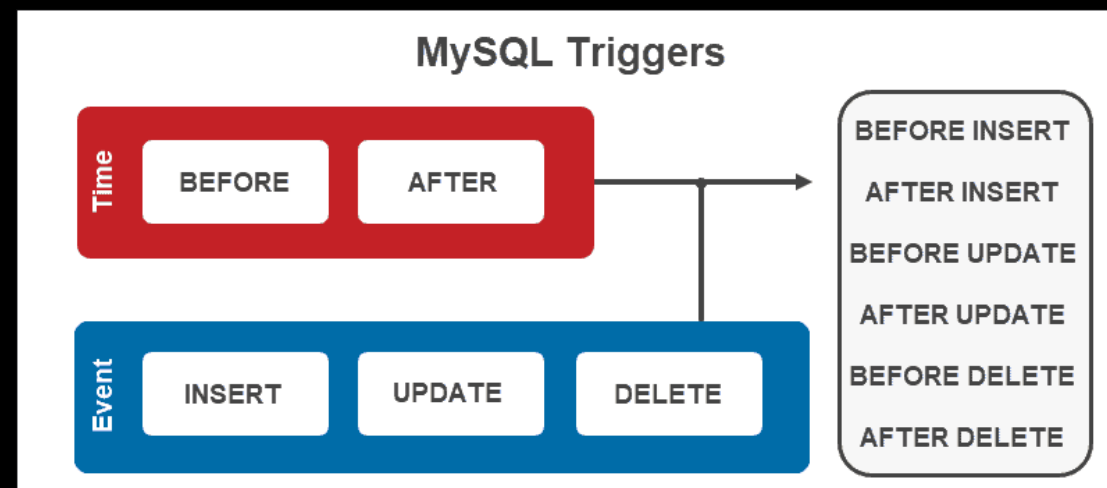
- BEFORE indica que el trigger se ejecutará antes de la operación que lo disparó.
- AFTER indica que el trigger se ejecutará después de la operación que lo disparó.





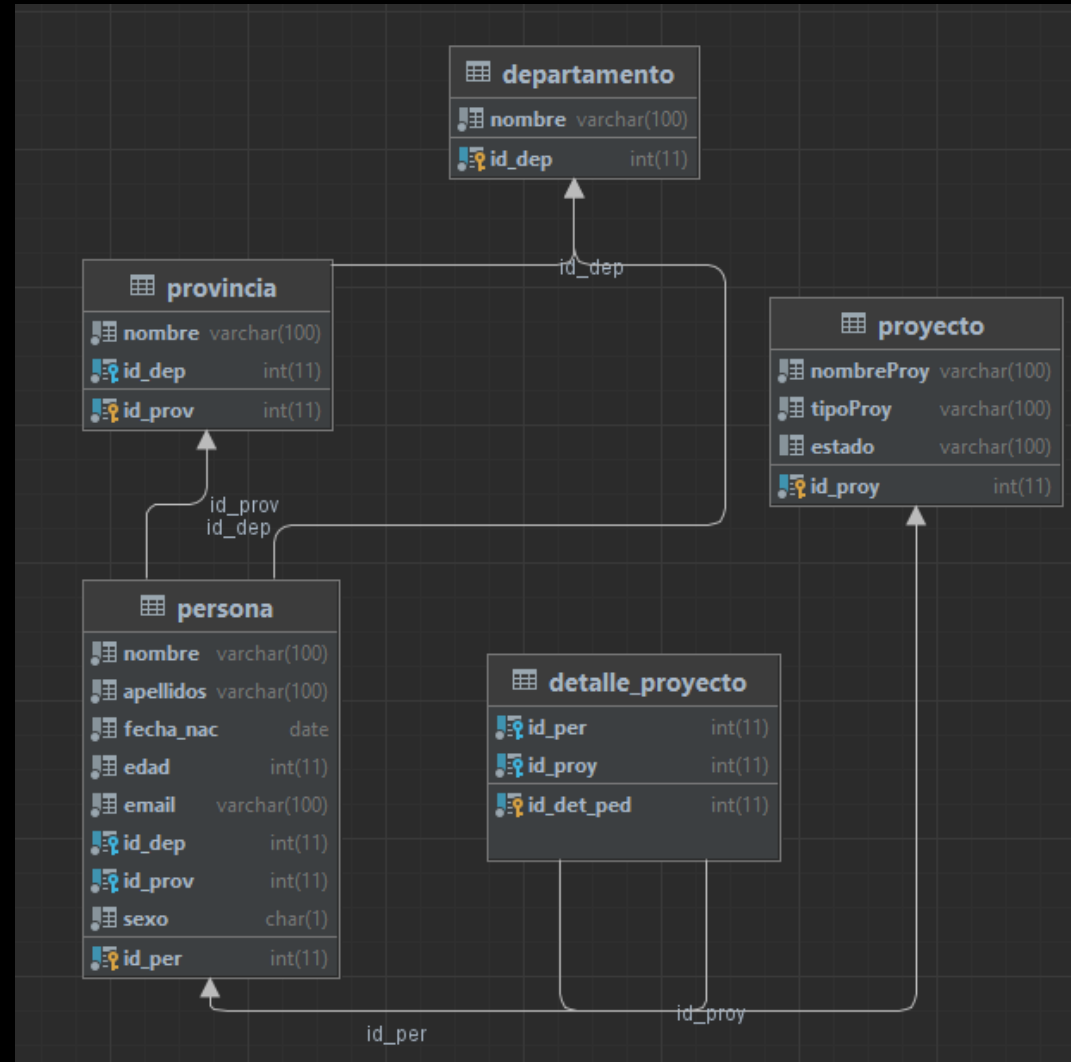
# A que se refiere cuando se habla de eventos en TRIGGERS

Cuando se habla de eventos en triggers de MySQL, se refiere a las operaciones que activan el trigger. Estos eventos pueden ser INSERT (cuando se agrega un nuevo registro), UPDATE (cuando se actualiza un registro existente) o DELETE (cuando se elimina un registro). El trigger se dispara automáticamente cuando se realiza uno de estos eventos en la tabla asociada al trigger.



# PARTE PRACTICA

Crear la siguiente Base de datos y sus registros.



# Crear una función que sume los valores de la serie Fibonacci

```
create or replace function sumar_fibonacci(limite integer)
returns integer
begin
  declare cadena text default fibonacci(limite: limite);
  declare suma integer default 0;
  declare pos_coma integer;

  while char_length(cadena) > 0 do

    set pos_coma = locate(',', cadena);

    if pos_coma = 0 then
      set suma = suma + cast(cadena as integer);
      set cadena = '';
    else
      set suma = suma + cast(substring(cadena, 1, pos_coma - 1) as integer);
      set cadena = substring(cadena, pos_coma + 2);
    end if;
  end while;

  return suma;
end;
select sumar_fibonacci(limite: 13);
```

```
sumar_fibonacci(10) ÷
88
```

La consulta de la vista debe reflejar como campos:  
Nombres y apellidos concatenados, la edad, fecha de nacimiento,  
Nombre del proyecto

```
create or replace view vista_proyecto_femenino as
select concat(per.nombre, ' ', per.apellidos) as Nombre_Apellidos, per.edad as Edad,
       per.fecha_nac as Fecha_de_nacimiento, proy.nombreProy as Nombre_de_Proyecto
       from persona as per
       inner join detalle_proyecto as detPed on per.id_per = detPed.id_per
       inner join proyecto as proy on proy.id_proy = detPed.id_proy
       inner join departamento as dep on dep.id_dep = per.id_dep
       where per.sexo = 'F' and per.fecha_nac = '2000-10-10' and dep.nombre = 'La Paz';

select* from vista_proyecto_femenino;
```

	Nombre_Apellidos	Edad	Fecha_de_nacimiento	Nombre_de_Proyecto
1	María López	26	2000-10-10	Proyecto B



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

```
create trigger tr_agregar_estado_pry_insert
before insert
on proyecto
for each row
begin
    if new.tipoProy = 'EDUCACION' or new.tipoProy = 'FORESTACION' or new.tipoProy = 'CULTURA' then
        set new.estado = 'ACTIVO';
    else
        set new.estado = 'INACTIVO';
    end if;
end;
```

```

create or replace trigger tr_agregar_estado_update
before update
on proyecto
for each row
begin
    if NEW.tipoProy = 'EDUCACION' or NEW.tipoProy = 'FORESTACION' or NEW.tipoProy = 'CULTURA' then
        set new.estado = 'ACTIVO';
    else
        set new.estado = 'INACTIVO';
    end if;
end;

```

	id_proy	nombreProy	tipoProy	estado
4	4	Proyecto 4	Educacion	ACTIVO
5	5	Proyecto 5	Investigacion	INACTIVO
6	6	Proyecto 5	FORESTACION	ACTIVO
7	7	Proyecto 6	Conferencia	INACTIVO

# Crear un trigger, debe de llamarse calculaEdad

```
create trigger tr_cacular_edad_Persona
before insert
on persona
for each row
begin
    set new.edad = timestampdiff(year, new.fecha_nac, curdate());
end;
```

	id_per	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	1	Juan	Pérez	2000-01-01	30	juan@gmail.com	2	1	M
2	2	María	López	2000-10-10	26	maria@gmail.com	1	2	F
3	3	Pedro	Gómez	2004-03-03	36	pedro@gmail.com	3	3	M
4	4	Heber	Miranda	2003-11-02	19	hebergmail.com	1	1	M
5	5	Daniel	Chura	2000-10-15	22	daniel@gmail.com	1	1	M

Crear otra tabla con los mismos campos de la tabla persona. Crear un trigger before insert para la tabla PERSONA.

```
create trigger tr_copia_persona
before insert
on persona
for each row
begin
    insert into copia_persona(nombre, apellido, fecha_nacimiento, edad, email, genero, id_dep, id_prov)
    select new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo, new.id_dep, new.id_prov;
end;
```

	nombre	apellido	fecha_nacimiento	edad	email	genero	id_dep	id_prov
1	Daniel	Chura	2000-10-15	22	daniel@gmail.com	M	1	1

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

```
create view manejo_de_proyectos as
select concat(per.nombre, ' ', per.apellido) as NombresApellido, per.edad as Edad,
       dep.nombre as Departamento, provi.nombre as Provincia, proy.nombreProy as Proyecto, proy.tipoProy AS TipoProyecto
       from persona as per
       inner join provincia as provi on per.id_prov = provi.id_prov
       inner join departamento as dep on per.id_dep = dep.id_dep
       inner join detalle_proyecto as deta on deta.id_per = per.id_per
       inner join proyecto as proy on proy.id_proy = deta.id_proy
       where year(per.fecha_nac)= 2003 and dep.nombre = 'La Paz';
```

	NombresApellido	Edad	Departamento	Provincia	Proyecto	TipoProyecto
1	Heber Mollericona	19	La Paz	Los Andes	Proyecto 1	Investigacion
2	Ana Lefonzo	20	La Paz	Los Andes	Proyecto 2	Educacion