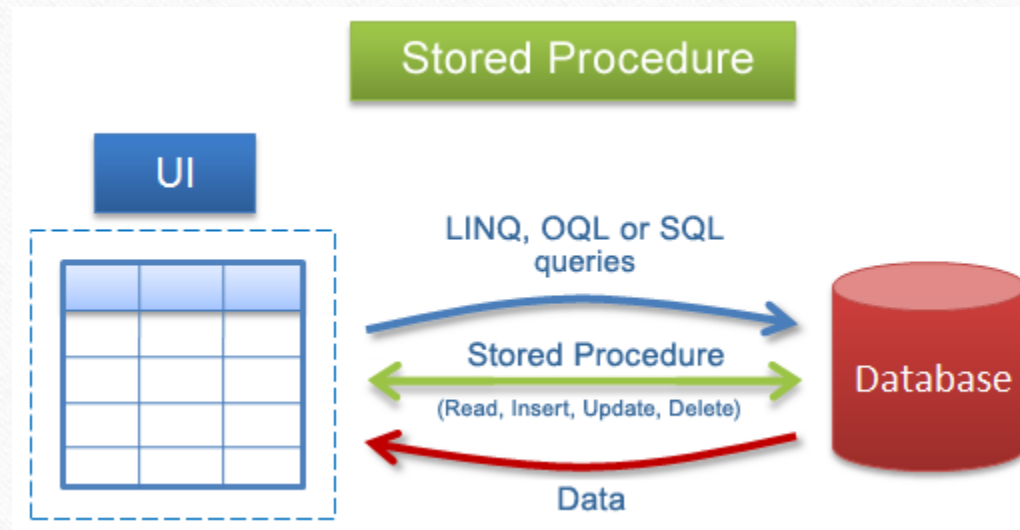


Tarea HITO4

Ludwing Antoni Vargas Ibarra

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.



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. Podemos utilizarlas en nuestras consultas para que el motor de base de datos resuelva eficazmente los cálculos de algunos datos, que de otra manera, resolviéndolos en nuestra capa de negocio nos podría traer problemas de performance en la aplicación.

```
create or replace function concatenaNumerosPares(numLimit integer)
returns varchar(200)
begin
  declare str varchar(200) default '';
  declare x integer default 1;
  WHILE x <= numLimit DO
    if x%2 = 0
    then
      set str = concat(str, x, ',');
    end if;
    set x = x + 1;
  END WHILE ;
  return str;
end;
select concatenaNumerosPares( numLimit: 7);
```

Cuál es la diferencia entre funciones y procedimientos almacenados.

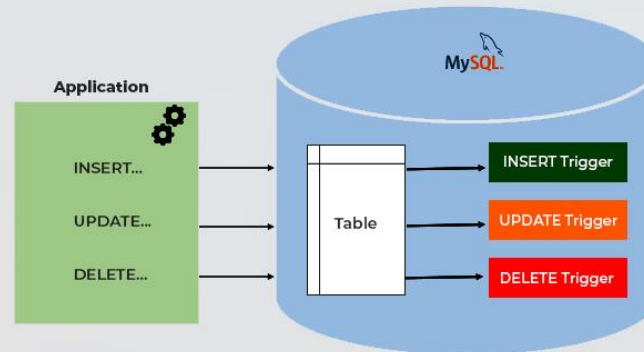
- Cuando llama al procedimiento almacenado, se debe especificar que es un parámetro externo. Una ventaja de los procedimientos almacenados es que puede obtener varios parámetros mientras que, en las funciones, solo se puede devolver una variable (función escalar) o una tabla (funciones con valores de tabla).

Cómo se ejecuta una función y un procedimiento almacenado

- Hay dos formas diferentes de ejecutar un procedimiento almacenado. El primer método y más común es que una aplicación o un usuario llame al procedimiento. El segundo método consiste en establecer el procedimiento para que se ejecute automáticamente cuando se inicie una instancia de SQL Server .
- Un parámetro no es parte de una transacción; por lo tanto, si se cambia un parámetro en una transacción que luego se revierte, el valor del parámetro no vuelve a su valor anterior. El valor devuelto a la persona que llama es siempre el valor en el momento de la devolución del módulo.

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, DELETE. Se puede configurar antes o después del evento.



En un trigger que papel juega las variables OLD y NEW

- Las columnas de la tabla asociada con el disparador pueden referenciarse empleando los alias OLD y NEW.
- El uso de SET NEW.nombre_col = valor necesita que se tenga el privilegio UPDATE sobre la columna.
- El uso de SET nombre_var = NEW.nombre_col necesita el privilegio SELECT sobre la columna.

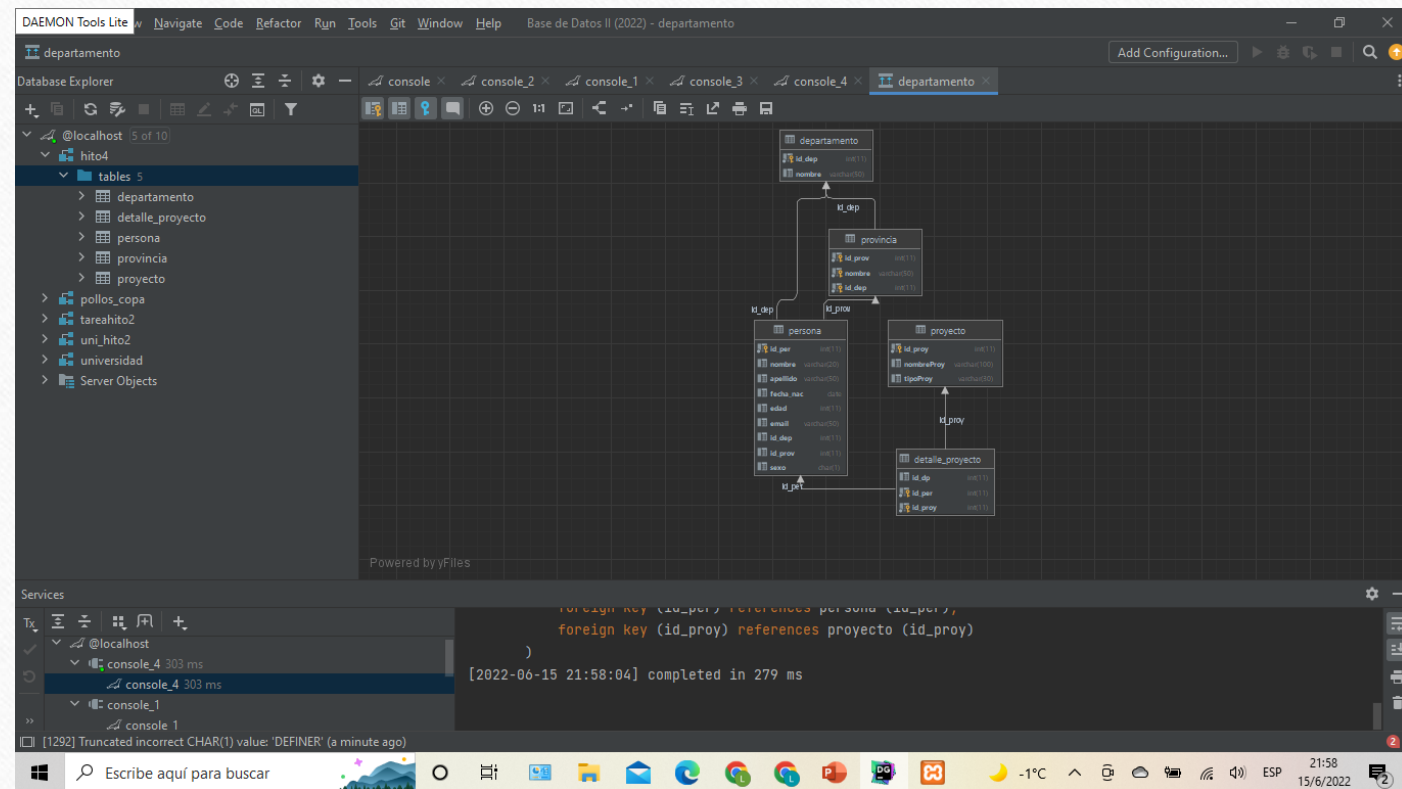
En un trigger que papel juega los conceptos(cláusulas) BEFORE o AFTER

- Un disparador es un objeto con nombre en una base de datos que se asocia con una tabla, y se activa cuando ocurre un evento en particular para esa tabla.
- momento_disp es el momento en que el disparador entra en acción.

A que se refiere cuando se habla de eventos en TRIGGERS

- Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador.

Crear la siguiente Base de datos y sus registros.

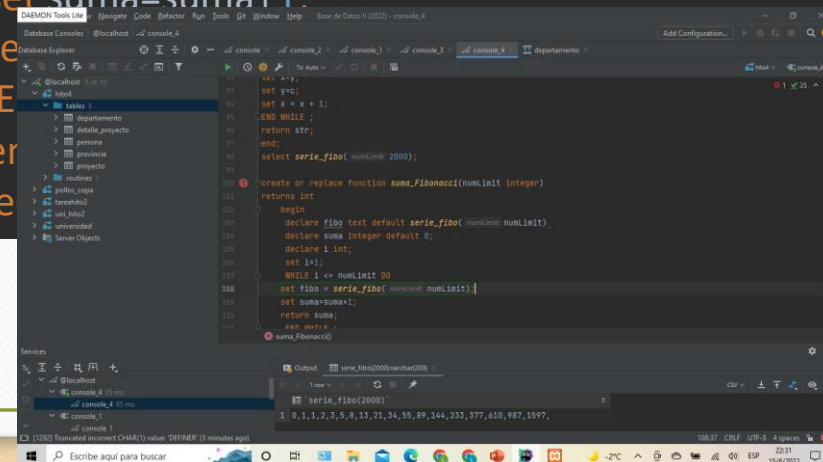


Crear una función que sume los valores de la serie Fibonacci.

```
create or replace function serie_fibo(numLimit
integer)
returns varchar(200)
begin
declare str varchar(200) default '';
declare x integer default 0;
declare y integer default 0;
declare c integer;
WHILE x <= numLimit DO
set str = concat(str, x, ',');
set c=x+y;
set x=y;
set y=c;
set x = x + 1;
END WHILE ;
return str;
end;
select serie_fibo(2000);
```

```
create or replace function suma_Fibonacci(numLimit integer)
returns int
begin
declare fibo text default serie_fibo(numLimit)
declare suma integer default 0;
declare i int;
set i=1;
WHILE i <= numLimit DO
set fibo = serie_fibo(numLimit);
set suma=suma+1;
```

re
E
er
sele



The screenshot shows a SQL IDE with a database explorer on the left and a query editor on the right. The query editor contains the following SQL code:

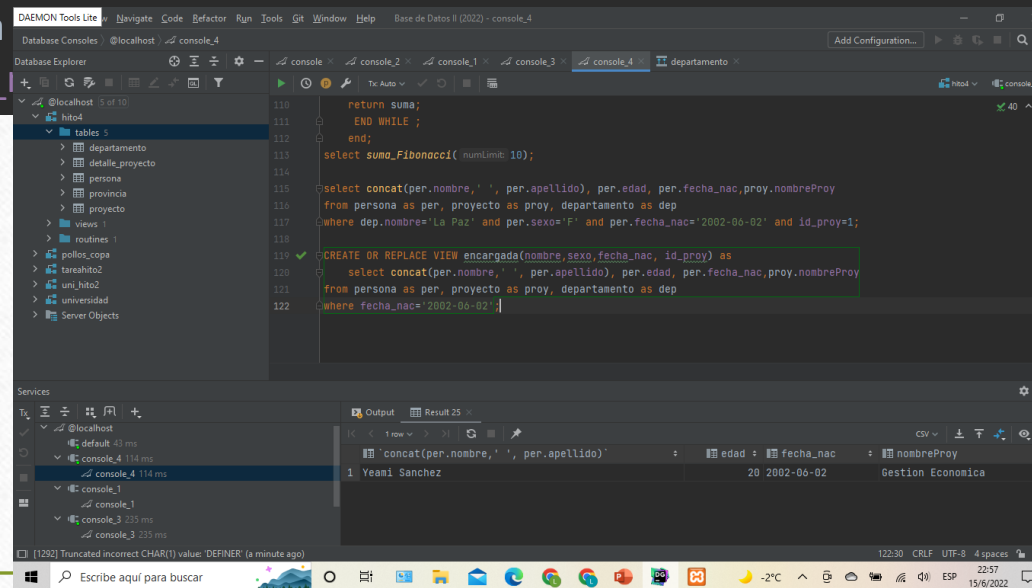
```
create or replace function suma_Fibonacci(numLimit integer)
returns int
begin
declare fibo text default serie_fibo(numLimit)
declare suma integer default 0;
declare i int;
set i=1;
WHILE i <= numLimit DO
set fibo = serie_fibo(numLimit);
set suma=suma+1;
return suma;
end;
select suma_Fibonacci(2000);
```

The output window at the bottom shows the result of the query: 1, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597.

Manejo de vistas.

```
select concat(per.nombre, ' ', per.apellido), per.edad, per.fecha_nac, proy.nombreProy
from persona as per, proyecto as proy, departamento as dep
where dep.nombre='La Paz' and per.sexo='F' and per.fecha_nac='2002-06-02' and id_proy=1;
```

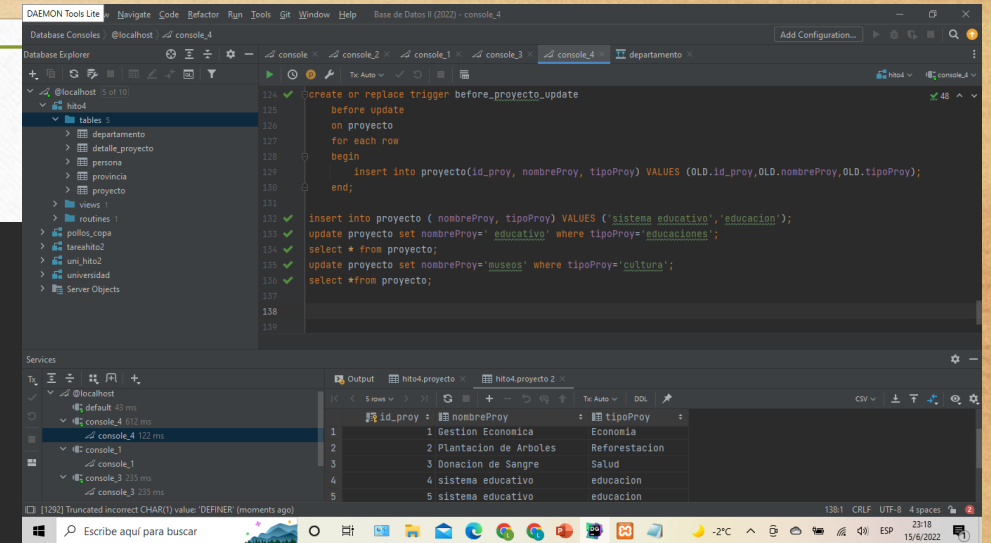
```
CREATE OR REPLACE VIEW encargada(nombre, sexo, fecha_nac, id_proy) as
select concat(per.nombre, ' ', per.apellido), per.edad, per.fecha_nac, proy.nombreProy
from persona as per, proyecto as proy, departamento as dep
where dep.nombre='La Paz' and per.sexo='F' and per.fecha_nac='2002-06-02' and id_proy=1;
```




```
create or replace trigger before_proyecto_update
before update
on proyecto
for each row
begin
```

```
    insert into proyecto(id_proy, nombreProy, tipoProy) VALUES
(OLD.id_proy,OLD.nombreProy,OLD.tipoProy);
end;
```

```
insert into proyecto ( nombreProy, tipoProy) VALUES ('sistema educativo','educacion');
update proyecto set nombreProy=' educativo' where tipoProy='educaciones';
select * from proyecto;
update proyecto set nombreProy='museos' where tipoProy='cultura';
select *from proyecto;
```



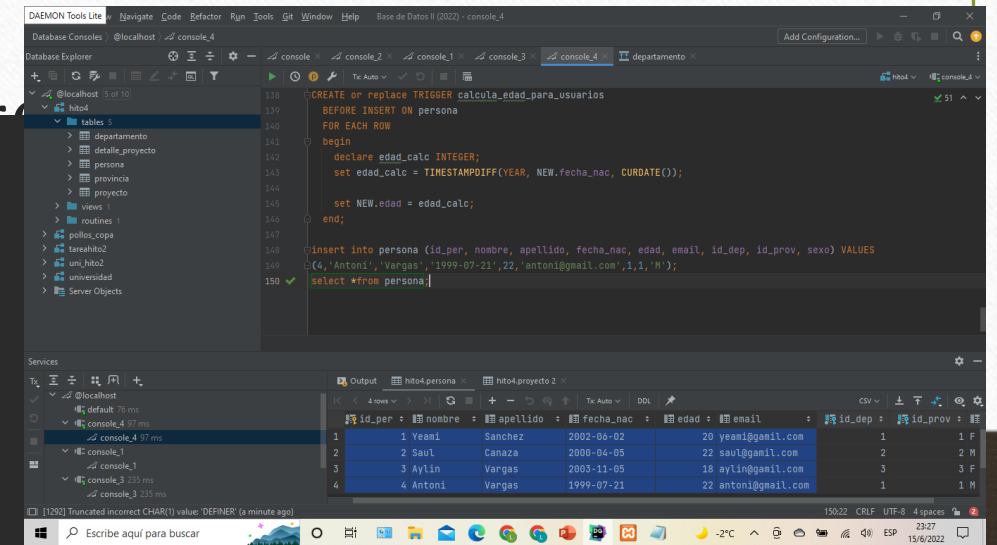
Monje de Trigo

```
CREATE or replace TRIGGER calcula_edad_para_usuarios
BEFORE INSERT ON persona
FOR EACH ROW
begin
    declare edad_calc INTEGER;
    set edad_calc = TIMESTAMPDIF(YEAR, NEW.fecha_nac, CURDATE());

    set NEW.edad = edad_calc;
end;
```

```
insert into persona (id_per, nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo) VALUES
(4,'Antoni','Vargas','1999-07-21',22,'antoni@gmail.com',1,1,'M');
```

```
select *from persona;
```



Manejo de TRIG

```
CREATE or replace TRIGGER calcula_edad_para_usuarios  
BEFORE INSERT ON persona  
FOR EACH ROW
```

```
begin
```

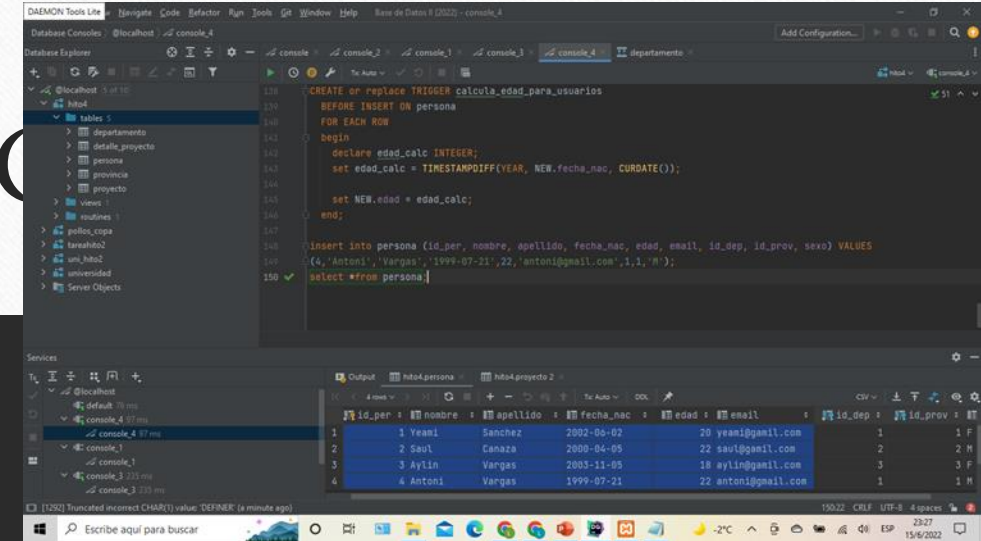
```
declare edad_calc INTEGER;
```

```
set edad_calc = TIMESTAMPDIFF(YEAR, NEW.fecha_nac, CURDATE());
```

```
set NEW.edad = edad_calc;
```

```
end;
```

```
insert into persona (id_per, nombre, apellido, fecha_nac, edad, email, id_dep, id_prov, sexo) VALUES  
(4,'Antoni','Vargas','1999-07-21',22,'antoni@gmail.com',1,1,'M');  
select *from persona;
```



Crear una consulta SQL que haga uso de todas

```
select concat(per.nombre, ' ', per.apellido), per.edad, per.fecha_nac, proy.nombreProy, prov.nombre, dp.id_dp  
from persona as per, proyecto as proy, departamento as dep, provincia as prov, detalle_proyecto as dp  
where dep.nombre='La Paz' and per.sexo='F' and per.fecha_nac='2002-06-02' and prov.nombre='El Alto';
```

```
CREATE OR REPLACE VIEW mejor(nombre, sexo, fecha_nac, id_dp) as
```

```
select concat(per.nombre, ' ', per.apellido), per.edad, per.fecha_nac, proy.nombreProy, prov.nombre, dp.id_dp  
from persona as per, proyecto as proy, departamento as dep, provincia as prov, detalle_proyecto as dp  
where fecha_nac='2002-06-02';
```

