TAREA HITO 4

MELANIE INGRID VILLCA COPA

MANEJO DE CONCEPTOS

1. **Defina que es un lenguaje procedural en MySQL**

Lenguajes procuderales o procedimentales: El usuario da órdenes para que se realicen las tareas pertinentes con el objetico de recuperar los datos requeridos. Es la base del lenguaje de consulta SQL

2. **Defina que es una función en MySQL.**

Una función en MySQL es una rutina creada para tomar unos parámetros, procesarlos y retornar en un salida.

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.

MANEJO DE CONCEPTOS

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

Variable NEW

NEW almacena el valor que aporta la consulta a la base de datos. Con esta variable podemos acceder a los datos introducidos. Con NEW.nombre_columna se almacenará la información con el nuevo valor que tendrá ese registro modificado (desde un UPDATE o INSERT) en la tabla. Los trigger relacionados con DELETE no tendrán disponible la variable NEW.

Variable OLD

OLD a diferencia de NEW, almacena el valor de las columnas que van a ser borradas o eliminadas. Al igual que pasa con NEW, OLD no está disponible en todas las instrucciones, más concretamente el valor no se puede recuperar cuando la instrucción es un INSERT.

MANEJO DE CONCEPTOS

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

Before trigger es un disparador que se ejecuta antes de una operación como insertar, actualizar, eliminar.

After trigger es un disparador que se ejecuta después de una operación como insertar, actualizar, eliminar.

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

Un trigger o disparador es un objeto que se asocia con tablas y se almacena en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

PARTE PRÁCTICA

9. Crear la siguiente Base de datos ysus 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)
);
```

PARTE PRÁCTICA

```
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)
);
```

PARTE PRÁCTICA

```
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');
```

PARTE PRÁCTICA

```
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');
```

PARTE PRÁCTICA

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

```
CREATE FUNCTION fibo(limiti 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 != limiti
     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);
```

PARTE PRÁCTICA

```
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
```

PARTE PRÁCTICA

```
      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);
```

PARTE PRÁCTICA

11. Manejo de vistas.

PARTE PRÁCTICA

12. Manejo de TRIGGERS I.

```
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 ;
```

PARTE PRÁCTICA

13. Manejo de Triggers II.

```
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;
```

PARTE PRÁCTICA

14. Manejo de TRIGGERS III.

```
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;
```

PARTE PRÁCTICA

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

```
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;
```

GRACIAS