BASE DE DATOS II

EVALUACION PROCESUAL HITO 4

CARLOS DANIEL FLORES PAUCARA
3ER SEMESTRE
2023

MANEJO DE CONCEPTOS

1.- Defina que es lenguaje procedural en MySQL.

El lenguaje procedural es la capacidad que tiene el servidor MySQL para crear y utilizar procedimientos almacenados y funciones definidas por el usuario, estos procedimientos, funciones pueden ser llamados desde cualquier lugar dentro de la base de datos MySQL estos pueden recibir parámetros de entrada y devolver valores de salida.

2.- Defina que es una función en MySQL.

La función en MySQL es un Objeto de programación que puede ser definido por el usuario y utilizarlo para realizar operaciones especificas donde estos pueden recibir uno o varios valores de entrada los cuales serán procesados y devolverán un resultado.

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

Una función es un bloque de código que acepta parámetros, realiza ciertas operaciones y devuelve un valor.

Un procedimiento almacenado es un conjunto de instrucciones SQL que se almacenan en la base de datos y se ejecutan de manera independiente.

MANEJO DE CONCEPTOS

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

Para ejecutar una función se utiliza la instrucción: select función();
Para ejecutar un procedimiento almacenado se utiliza la instrucción: Call procedimiento();

5.Defina que es una TRIGGER en MySQL.

En MySQL, una trigger (disparador en español) es un objeto de base de datos que permite ejecutar automáticamente un conjunto de instrucciones SQL cuando ocurre un evento específico en una tabla. Los disparadores se asocian con una tabla y se activan cuando se realiza una operación de inserción (INSERT), actualización (UPDATE) o eliminación (DELETE) en la tabla.

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

En un trigger en MySQL, las variables OLD y NEW se utilizan para acceder a los valores antiguos y nuevos de las columnas en una tabla, respectivamente. Estas variables son especialmente útiles en los triggers AFTER UPDATE y AFTER DELETE, donde puedes acceder a los valores antiguos antes de la operación de actualización o eliminación, así como a los nuevos valores después de la operación.

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

BEFORE:
En un trigger con la cláusula "BEFORE", el trigger se ejecuta antes de que se realice la operación en la tabla que lo activó. Esto significa que el trigger tiene acceso a los valores antiguos y puede modificarlos antes de que se realice la operación. Por ejemplo, puedes utilizar un trigger "BEFORE INSERT" para modificar o validar los valores que se van a insertar en la tabla.

AFTER:
En un trigger con la cláusula "AFTER", el trigger se ejecuta después de que se haya realizado la operación en la tabla que lo activó. Esto significa que el trigger tiene acceso a los valores nuevos y puede realizar acciones basadas en ellos después de que se haya completado la operación. Por ejemplo, puedes utilizar un trigger "AFTER UPDATE" para realizar un registro de auditoría de los cambios realizados en la tabla después de que se hayan actualizado los registros.

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

Cuando se habla de eventos en los triggers, se refiere a las operaciones específicas que ocurren en una tabla y que activan la ejecución del trigger. Estos eventos pueden ser de tres tipos principales:

INSERT: Este evento ocurre cuando se realiza una operación de inserción de datos en la tabla. Por ejemplo, cuando se agrega una nueva fila de datos a la tabla, se dispara un evento de inserción.

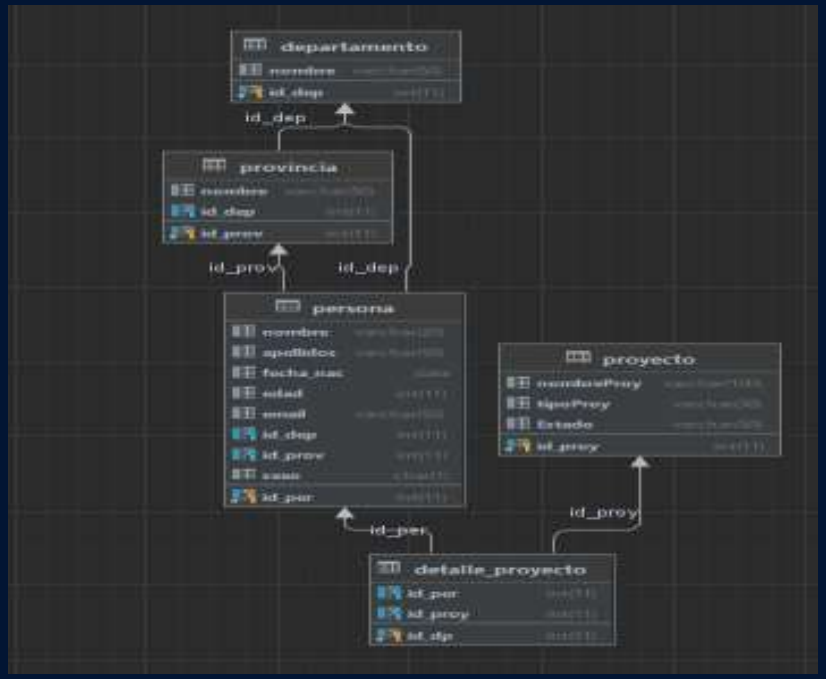
UPDATE: Este evento ocurre cuando se realiza una operación de actualización en la tabla. Por ejemplo, cuando se modifican los valores de una o varias columnas en una fila existente, se dispara un evento de actualización.

DELETE: Este evento ocurre cuando se realiza una operación de eliminación de datos en la tabla. Por ejemplo, cuando se elimina una fila de datos de la tabla, se dispara un evento de eliminación.

PARTE PRACTICA

Agregar mínimamente 2 registros a cada tabla

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




```
create table departamento
(
   id_dep int auto_increment primary key,
   nombre varchar(50)
);

create table provincia
(
   id_prov int auto_increment primary key,
   nombre varchar(50),
   id_dep int,
   foreign key (id_dep) references
departamento(id_dep)
);

create table proyecto
(
   id_proy int auto_increment primary key,
   nombreProy varchar(100),
   tipoProy varchar(30)
);
```

```
create table detalle_proyecto
(
   id_dp int auto_increment primary key,
   id_per int,
   id_proy int,
   foreign key (id_per) references persona(id_per),
   foreign key (id_proy) references proyecto(id_proy)

);

create table persona
(
   id_per int auto_increment primary key,
   nombre varchar(20),
   apellidos varchar(50),
   fecha_nac date,
   edad int,
   email varchar(50),
   id_dep int,
   id_prov int,
   sexo char(1),
   foreign key (id_dep) references
departamento(id_dep),
   foreign key (id_prov) references provincia(id_prov)
);
```

```
insert into departamento(nombre)
values ('Oruro'),('Potosi');

insert into provincia(nombre, id_dep)
values ('colquiri',1),('libertad',2);

insert into persona(nombre, apellidos, fecha_nac, edad, email, id_dep, id_prov,
sexo)
values ('Carlos','Flores Paucara','2001-01-01',21,'carlosflo@gmail.com',2,2,'M'),
      ('Melanny','Boyan  Quispe','2002-02-07',21,'Melabya@gmail.com',1,1,'F'),
      ('Yeisi','Barra Alannoca','2003-12-15',23,'yeisibarr@gmail.com',1,1,'F');
insert into proyecto(nombreProy, tipoProy)
values ('robot','tecnologico'),
      ('medio ambiente','reciclar');

insert into detalle_proyecto(id_per, id_proy)
values (1,1),(2,2),(3,1);
```

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))

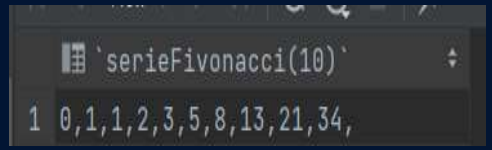
■ Note que previamente deberá crear una función que retorne una cadena con la serie fibonacci hasta un cierto valor.

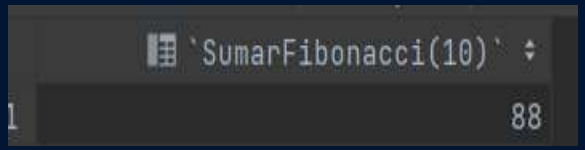
1. Ejemplo: 0,1,1,2,3,5,8,.....

■ Luego esta función se deberá pasar como parámetro a la función que suma todos los valores de esa serie generada.

```
create or replace function serieFivonacci(numero
int)
returns TEXT
begin
    declare respuesta TEXT default '';
    declare cont int default 1;
    declare numR int default 0;
    declare num1 int default 0;
    declare num2 int default 1;

    while(numero >= cont) do
    set respuesta = concat(respuesta,num1,',');
    set numR = num1+num2;
    set num1 = num2 ;
    set num2 = numR ;
    set cont = cont+1;

    end while;
    return respuesta;
end;

select serieFivonacci(10);
```

```
create or replace function SumarFibonacci(limite int)
returns int
begin
    declare cont int default limite;
    declare ubi int default 0;
    declare cant int default 0;
    declare numb int default 0;
    declare ubi2 text default 0;
    declare sum int default 0;
    while (cont > 0) do
        set ubi2 = ubi;
        set ubi = locate(',',serieFivonacci(limite),ubi2+1);
        set cant = ubi-ubi2;
        set numb= substr(serieFivonacci(limite),ubi2+1,cant-1);
        set sum = numb + sum;
        set cont = cont-1;

        end while;
    return sum;
end;

select SumarFibonacci(10);
```





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'

```
create or replace view Identifica_Persona as
select concat(per.nombre,' ', per.apellidos), per.edad,
per.fecha_nac,p.nombreProy
from persona per
inner join departamento d on per.id_dep = d.id_dep
inner join detalle_proyecto dp on per.id_per = dp.id_per
inner join proyecto p on dp.id_proy = p.id_proy
where d.nombre = 'Oruro' and per.fecha_nac = '2000-10-10' and
per.sexo = 'F';

select * from Identifica_Persona;
```

	`concat(per.nombre,' ', per.apellidos)`	edad	fecha_nac	nombreProy
1	Melanny,Boyan Quispe	21	2000-10-10	medio ambiente

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**
- 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**

```
alter table proyecto add column  Estado
varchar(50);

create or replace  trigger Estado_Activo_Inactivo
   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;
```

```
create or replace  trigger
Activo_Inactivo_update
   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;
```

```
insert into proyecto (nombreProy, tipoProy)
VALUES ('Cortar Arboles','Forestacion');
insert into proyecto (nombreProy, tipoProy)
VALUES ('Cortar Arboles','Venta');

select * from proyecto;

update proyecto
set tipoProy = 'estudio'
where id_proy =2;

update proyecto
set tipoProy = 'tecno'
where id_proy =1;
select * from proyecto;
```

	id_proy	nombreProy	tipoProy	Estado
1	1	robot	tecno	Inactivo
2	2	medio ambiente	estudio	Inactivo
3	3	Cortar Arboles	Forestacion	Activo
4	4	Cortar Arboles	Venta	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.
- Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento.**

```
create or replace trigger 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, id_dep, id_prov,
sexo)
values ('julian','Alvarez','1990-03-
13','julianAlvarez@gmail.com',2,2,'M');

select * from persona;
```

4	4	julian	Alvarez	1990-03-13	33	julianAlvarez@gmail.com	2	2	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.**
- Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento**.

```
create table persona_dos(
    nombre varchar(20),
    apellidos varchar(50),
    fecha_nac date,
    edad int,
    email varchar(50),
    id_dep int,
    id_prov int,
    sexo char(1)
);
```

```
create or replace trigger manejodetriggerIII
    before insert
    on persona
    for each row
    begin
        insert into persona_dos
(nombre, apellidos, fecha_nac, edad, email, id_dep,
id_prov, sexo)
 SELECT

new.nombre,new.apellidos,new.fecha_nac,new.eda
d,new.email,new.id_dep,new.id_prov,new.sexo;

end;
```

	id_per	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	1	Carlos	Flores Paucara	2001-01-01	21	carlosflo@gmail.com	2	2	M
2	2	Melanny	Boyan Quispe	2000-10-10	21	Melaby@gmail.com	1	1	F
3	3	Yeisi	Barra Alannoca	2003-12-15	23	yeisibarr@gmail.com	1	1	F
4	4	julian	Alvarez	1998-03-13	33	julianAlvarez@gmail.com	2	2	M
5	5	Marcos	Quispe	1998-03-23	25	MarcosQuispe@gmail.com	1	1	M

```
insert into persona(nombre, apellidos,
fecha_nac, email, id_dep, id_prov, sexo)
values ('Marcos','Quispe','1998-03-
23','MarcosQuispe@gmail.com',1,1,'M');

select *from persona;

select * from persona_dos;
```

	nombre	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	Marcos	Quispe	1998-03-23	25	MarcosQuispe@gmail.com	1	1	M

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

```sql
create or replace view Uso_de_Todas_las_tablas as
select concat(per.nombre,' ', per.apellidos), per.edad, per.fecha_nac,
p.nombreProy as Proyecto,p.tipoProy,p.Estado,
prov.nombre as provincia ,d.nombre as departamento,dp.id_proy
from persona per

inner join departamento d on per.id_dep = d.id_dep
inner join detalle_proyecto dp on per.id_per = dp.id_per
inner join proyecto p on dp.id_proy = p.id_proy
inner join provincia prov on per.id_prov = prov.id_prov;

select * from Uso_de_Todas_las_tablas;
```

	`concat(per.nombre,' ', per.apellidos)`	edad	fecha_nac	Proyecto	tipoProy	Estado	provincia
1	Melanny Boyan Quispe	21	2000-10-10	medio ambiente	estudio	Inactivo	colquiri
2	Yeisi Barra Alannoca	23	2003-12-15	robot	tecno	Inactivo	colquiri
3	Carlos Flores Paucara	21	2001-01-01	robot	tecno	Inactivo	libertad

Gracias