EVALUACION PROCESUAL HITO4

BASE DE DATOS II

NILBER MAYTA CUNO

PARTE TEORICA

MANEJO DE CONCEPTOS

1. Defina que es lenguaje procedural en MySQL.

En MySQL, el lenguaje procedural se refiere a la capacidad de MySQL para admitir la creación de procedimientos almacenados y funciones, lo que permite escribir código procedural dentro de la base de datos.

2. Defina que es una FUCNTION en MySQL.

Una función nos ayuda a realizar cálculos mediante los datos que le proporcionemos y que nos retorne un valor resultado de todo el cálculo.

```
create or replace function N_NATURALESv2(limite
int)
    returns TEXT
BEGIN
    declare cont int default 2;
    declare resp text default ' ';
    while cont <= limite do
            set resp = concat(resp,cont,' , ');
            set cont = cont+2;
        end while;
    return resp;
end;
```

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

Aunque las dos estructuras son muy similares se puede decir que las funciones están más enfocadas en el manejo y cálculo de datos y los procedimientos almacenados en reservar lógica para llamarla de una manera más sencilla.

```
create or replace procedure insertar_datos(
    fecha text,
    usuario text,
    host_name text,
    accion text,
    antes text,
    despues text
)
begin
    insert into audit_usuarios_rrhh(fecha_mod,
usuario_log, hostname, acccion,
antes_del_cambio,dspues_del_cambio )
        values
(fecha,usuario,host_name,accion,antes,despues);
end;
```

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

Para usar una función se usa la palabra reservada selec y el nombre de la función:

```
select N_NATURALESv2(10);
```

Para llamar una función se usa la palabra reservada call y la lógica que usa:

```
call insertar_datos(
    now(),user(),@@hostname,'UPDATE','antes','despues'
    );
```

5. Defina que es una TRIGGER en MySQL

Un trigger es una estructura que nos permite realizar lógica al momento de que se realice algún evento

```
create or replace trigger GenerarContrasena
    before insert
    on Usuarios
    for each row
begin

    set NEW.password = lower(concat(substr(NEW.nombres,1,2),substr(NEW.apellidos,1,2),NEW.edad));

end;
```

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

Estos son para identificar cuando registro que está saliendo (OLD) o cunado está entrando (NEW).

```
create or replace trigger tr_audit_update_usuarios_rrhh
    before update
    on usuarios_rrhh
    for each row
begin

    insert into audit_usuarios_rrhh(fecha_mod, usuario_log, hostname, acccion, id_usr, nombre_completo,
password,antes_del_cambio,dspues_del_cambio) select
    now(),user(),@@hostname,'INSERT',new.id_usr,new.nombre_completo,new.pasword,concat(old.id_usr,old.nombre_completo,old.fecha_nac),
``concat(new.id_usr,new.nombre_completo,new.fecha_nac);
end;
```

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

Se usa Before para especificar que el trigger se ejecutara antes de que suceda el evento.

En cambio, cuando se una After especificamos que se ejecutara después de que suceda el evento.

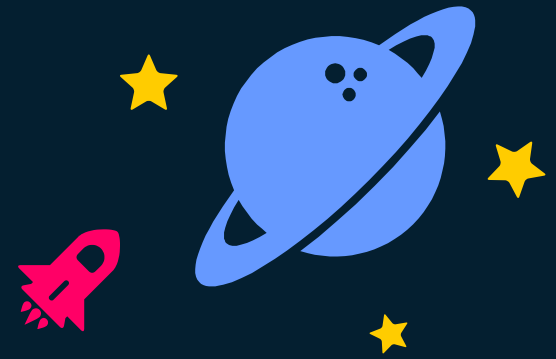
8. A que se refiere cuando se habla de eventos en TRIGGERS

Un evento es lo que ocasiona que se dispare el trigger, estos pueden ser,

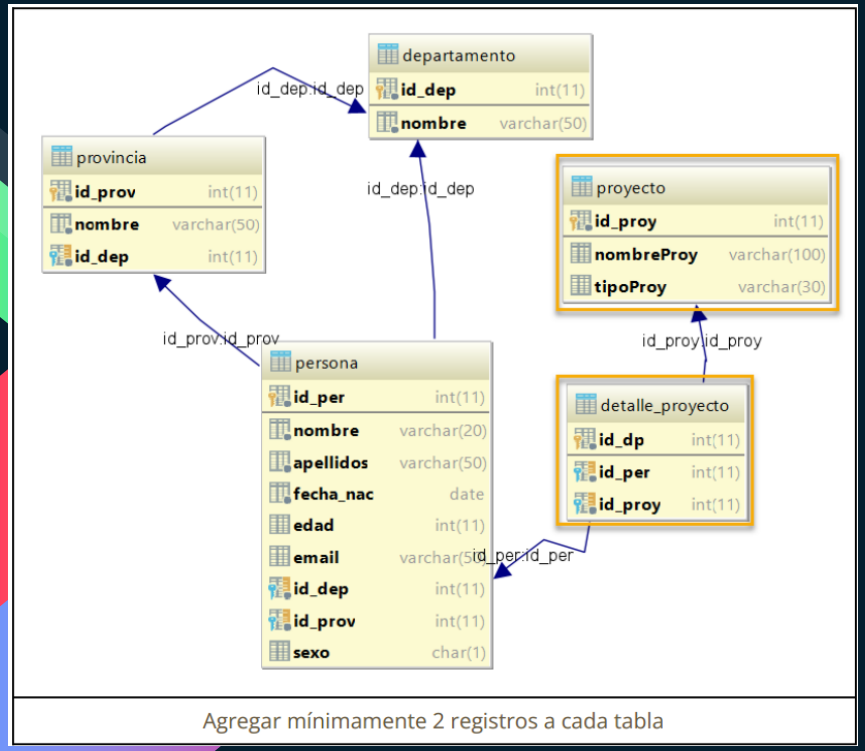
INSERT – UPDATE - DELETE

Cuando alguno de estos suceda se realizara el trigger

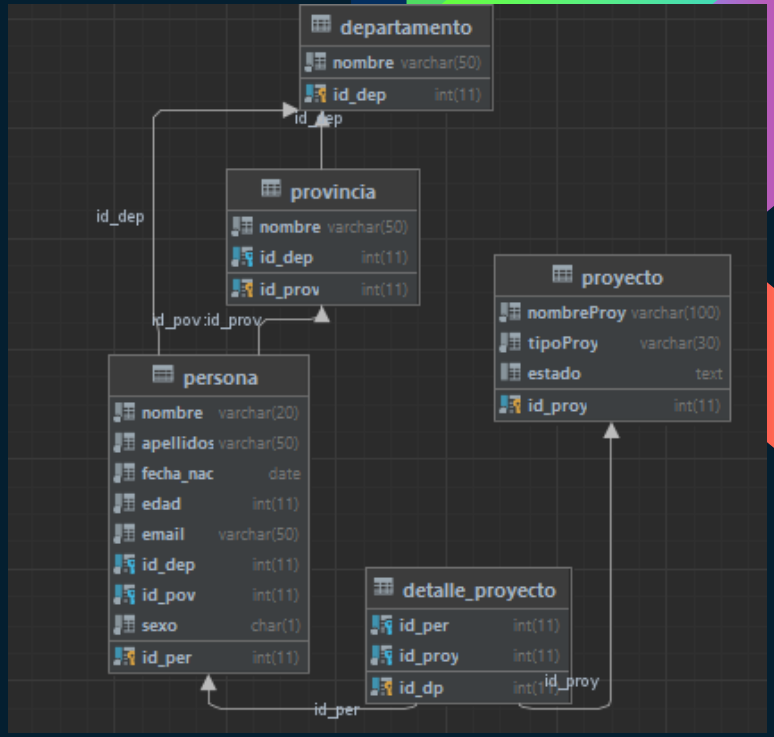
Parte practica



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



Agregar mínimamente 2 registros a cada tabla



```sql
create table departamento(
    id_dep int auto_increment primary key not null ,
    nombre varchar(50) not null
);

create table provincia(
    id_prov int auto_increment primary key not null ,
    nombre varchar(50) not null,
    id_dep int not null ,
    foreign key (id_dep) references departamento(id_dep)
);

create table persona(
    id_per int auto_increment primary key not null ,
    nombre varchar(20)not null ,
    apellidos varchar(50) not null ,
    fecha_nac date not null ,
    edad int not null ,
    email varchar(50) not null ,
    id_dep int not null ,
    id_pov int not null ,
    foreign key (id_dep) references departamento (id_dep),
    foreign key (id_pov) references provincia (id_prov),
    sexo char not null
);
```

```sql
create table proyecto(
    id_proy int auto_increment primary key not null ,
    nombreProy varchar(100) not null ,
    tipoProy varchar(30)not null
);

create table detalle_proyecto(
    id_dp int auto_increment primary key not null ,
    id_per int not null ,
    id_proy int not null ,
    foreign key (id_per) references persona (id_per),
    foreign key (id_proy) references proyecto (id_proy)
);
```

```sql
insert into departamento (nombre)
values ('La Paz'),
       ('Cochabamba');

insert into provincia (nombre, id_dep)
values ('Yungas',1),
       ('Quillacollo',2);

insert into persona (nombre, apellidos, fecha_nac, edad, email, id_dep, id_pov, sexo)
values ('Juan', 'García', '1990-05-10', 33, 'juan.garcia@example.com', 1, 1, 'M'),
       ('María', 'López', '1985-12-15', 38, 'maria.lopez@example.com', 2, 2, 'F');

insert into proyecto (nombreProy, tipoProy)
values ('Proyecto de Marketing', 'Publicidad'),
       ('Proyecto de Desarrollo', 'Tecnología');

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

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

```
create or replace function  serie_fibonacci(num int)
    returns text
begin
    declare a int default 0;
    declare b int default 1;
    declare resp text default '';

    while num>0 do
            set resp = concat(resp,a,',');
            set b = a + b;
            set a = b - a;
            set num = num - 1;
        end while;
    return resp;
end;
```

```
select serie_fibonacci(7);
```

	`serie_fibonacci(7)`
1	0,1,1,2,3,5,8,

```
create or replace function  suma_serie_fibonacci(Limite int)
    returns int
begin
    declare resp int default 0;
    declare cont int default 1;
    declare aux char default '';
    declare num int default 0;
    declare fibonacci text;
    set fibonacci = serie_fibinacci(Limite);

    while (cont <= length(fibonacci)) do
            set aux = substr(fibonacci, cont, 1);
            if (aux not in (','))then
                set num = cast(aux as unsigned);
                set resp = resp + num;
            end if;
            set cont = cont + 1;
        end while;

    return resp;
end;
```

```
select suma_serie_fibonacci(7);
```

	`suma_serie_fibonacci(7)`
1	20

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'

```sql
select concat(pe.nombre,' ',pe.apellidos) as fullname,pe.edad,pe.fecha_nac,pro.nombreProy
from persona as pe
    inner join detalle_proyecto det on pe.id_per = det.id_per
    inner join proyecto pro on det.id_proy = pro.id_proy
    inner join departamento dep on pe.id_dep = dep.id_dep
where pe.sexo = 'F' and dep.nombre = 'El Alto' and pe.fecha_nac = '2000-10-10';
```

```sql
create or replace view DatosPersonales as
select concat(pe.nombre,' ',pe.apellidos) as fullname,pe.edad,pe.fecha_nac,pro.nombreProy
from persona as pe
        inner join detalle_proyecto det on pe.id_per = det.id_per
        inner join proyecto pro on det.id_proy = pro.id_proy
        inner join departamento dep on pe.id_dep = dep.id_dep
where pe.sexo = 'F' and dep.nombre = 'El Alto' and pe.fecha_nac = '2000-10-10';
```

```sql
select *
from datospersonales as date;
```

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, si llega un tipo de proyecto distinto colocar INACTIVO
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
alter table proyecto
add column (estado
text);
```

```
create or replace trigger tr_insert
   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;
```

```
insert into proyecto (nombreProy, tipoProy)
values ('Talacion de arboles','FORESTACION');
```

```
create or replace trigger tr_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 ('Construccion de puentes','CNSTRUCCION');
```

```
select *
from proyecto;

update proyecto set tipoProy = 'FORESTACION' where id_proy = 1;

select *
from proyecto;
```

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.

```sql
create or replace trigger tr_calculaEdad
    before insert
    on persona
    for each row
begin
    declare edad int default timestampdiff(YEAR, New.fecha_nac,curdate());

    set NEW.edad = edad;

end;

insert into persona (nombre, apellidos, fecha_nac, edad, email, id_dep, id_pov, sexo)
values ('Nilber','Mayta','2003-08-18',1,'nilber@gmail.com',1,1,'M');

select *
from persona;
```

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.

```sql
create table copia_persona(
        nombre varchar(20)not null ,
        apellidos varchar(50) not null ,
        fecha_nac date not null ,
        edad int not null ,
        email varchar(50) not null ,
        id_dep int not null ,
        id_pov int not null ,
        sexo char not null
);
```

```sql
create or replace trigger tr_CopiaPersona
    before insert
    on persona
    for each row
begin
    insert into copia_persona(nombre, apellidos, fecha_nac, edad, email, id_dep, id_pov, sexo)
        select NEW.nombre,NEW.apellidos,NEW.fecha_nac,NEW.edad,NEW.email,new.id_dep,NEW.id_pov,NEW.sexo;
end;
```

```sql
insert into persona(nombre, apellidos, fecha_nac, edad, email, id_dep, id_pov, sexo)
values ('Gabriel','Quispe','2004-05-12',18,'juan@gmail.com',2,2,'M');
```

```sql
select *
from copia_persona;
```

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

- La consulta generada convertirlo a VISTA

```
select concat(pe.nombre,' ',pe.apellidos)as fullname,dep.nombre,prov.nombre,pro.tipoProy
from persona as pe
        inner join detalle_proyecto det on pe.id_per = det.id_per
        inner join proyecto pro on det.id_proy = pro.id_proy
        inner join departamento dep on pe.id_dep = dep.id_dep
        inner join provincia prov on dep.id_dep = prov.id_dep;
```

```
create or replace view ConsultaGeneral as
select concat(pe.nombre,' ',pe.apellidos)as fullname,dep.nombre as departamento,prov.nombre as provincia,pro.tipoProy as tipo
from persona as pe
        inner join detalle_proyecto det on pe.id_per = det.id_per
        inner join proyecto pro on det.id_proy = pro.id_proy
        inner join departamento dep on pe.id_dep = dep.id_dep
        inner join provincia prov on dep.id_dep = prov.id_dep;
```

```
select * from ConsultaGeneral;
```

GRACIAS POR SU ATENCION