

Actividad final hito 4

Martin Josue Mamani Pilco

Parte teórica

Manejo de conceptos

1. Defina que es lenguaje procedural en MySQL

El lenguaje procedural o PL/SQL es un lenguaje de programación por procedimientos, estructurado en bloques permitiendo programar funciones, triggers, procedimientos almacenados y scripts.

Es además una extensión de SQL que permite realizar las acciones anteriormente mencionadas permitiendo programar problemas más complejos.



2. Defina que es una Función en MySQL


Una función es un proceso que realiza una determinada acción, ya sea predeterminada o creada por el DBA, llegando a recibir un parámetro o varios según lo necesite.

La función se ejecutará bajo el comando 'SELECT' seguido del nombre, y posteriormente se pondrá entre paréntesis el o los parámetros.




3. Cual es la diferencia entre funciones y procedimientos almacenados

Las funciones realizan una acción predeterminada o una establecida por el DBA tomando un parámetro si lo requieren, mientras que los procedimientos almacenados pueden realizar la misma acción pero de una manera más rápida y con menos líneas de código. Sin embargo existen algunas acciones que son más convenientes realizarlas con una función y viceversa, por lo que es necesario analizar cual es más conveniente utilizar.



4. Como se ejecuta una función y un procedimiento almacenado

Una función se ejecuta llamando a la cláusula 'SELECT' seguido del nombre de la función y después entre paréntesis los parámetros que requiere, mientras que un procedimiento almacenado se ejecuta llamando a la cláusula 'EXEC' seguido del nombre del procedimiento y el parámetro, también podemos prescindir de la cláusula EXEC para llamar a un procedimiento almacenado.



5. Defina que es un TRIGGER en MySQL

Un TRIGGER es un procedimiento que puede ejecutarse antes o después de procesar los datos de las tablas, realizando su función al momento de introducir, borrar o modificar datos de las tablas.



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

Las variables NEW resultan de introducir nuevos datos a las tablas en las variables a las que se asigne la palabra NEW.

En el caso de las variables OLD estas resultan de modificar o sustituir datos existentes en las tablas, anteponiendo la palabra OLD en la variable que se quiera modificar.



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

El concepto BEFORE se utiliza para establecer que el proceso se realizará antes del procesamiento de los datos, contrario a AFTER que establece que el proceso se realizará después de procesar los datos.



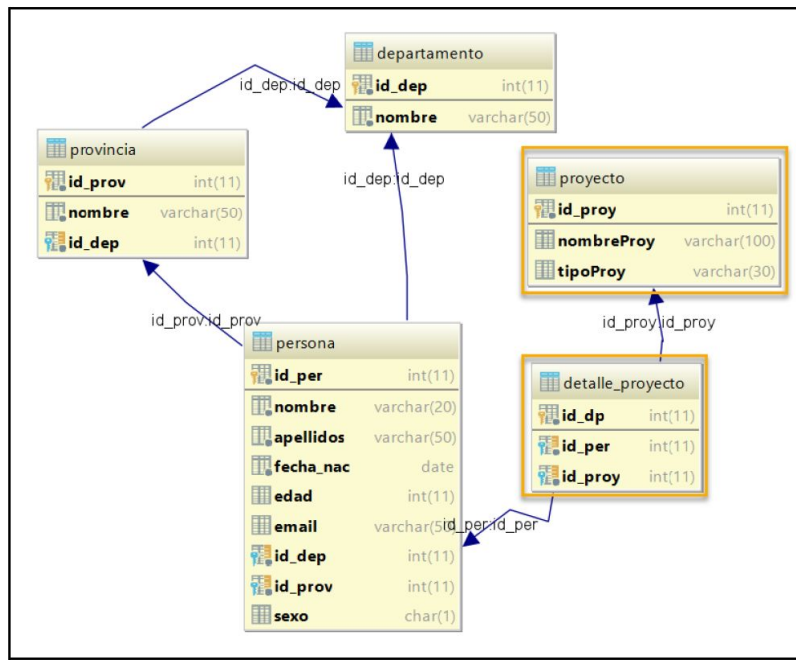
8. A que se refiere cuando se habla de eventos en TRIGGERS

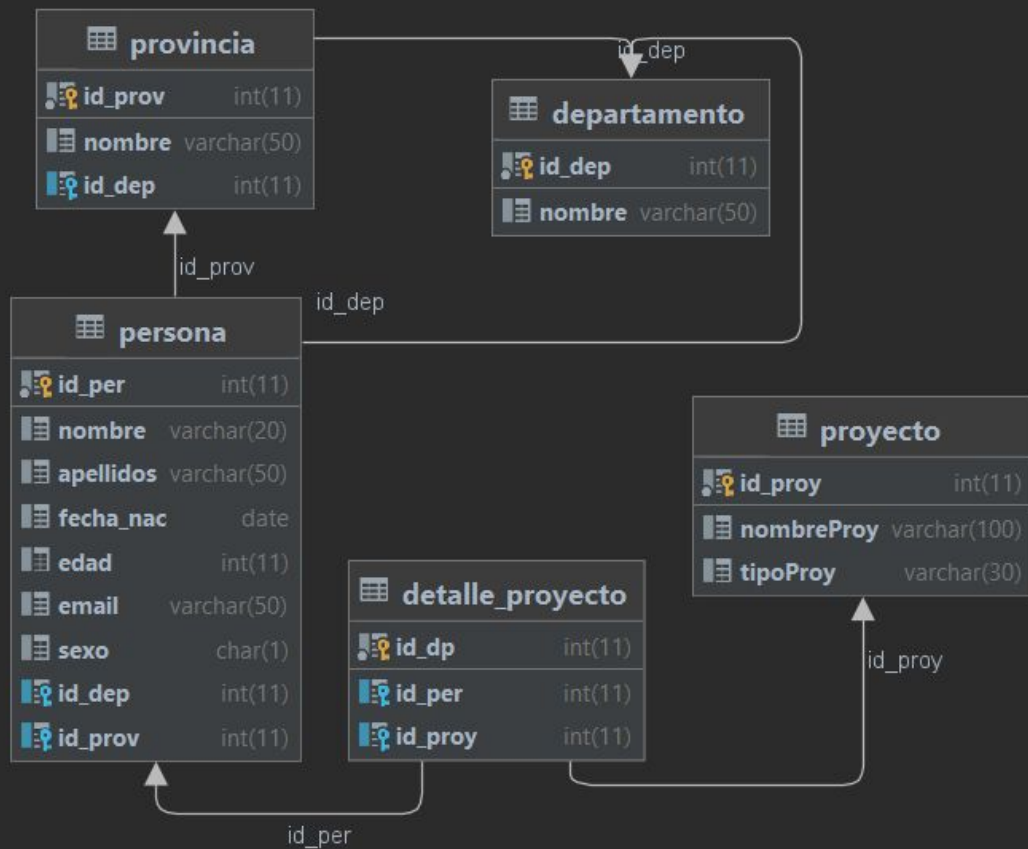
Se refiere a los tres eventos en los que pueden ejecutarse los TRIGGERS que son: Insert, Delete y Alter.



Parte práctica

9. Crear la siguiente Base de Datos y sus registros





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 function ejercicio_10(entrada int)
returns text
begin
    declare x int default 0;
    declare y int default 1;
    declare i int default 0;
    declare aux int default 0;
    declare serie text default '';
    while i < entrada
    do
        if i = 0
        then
            set serie = '0 ';
        else
            set serie = concat(serie, y, ' ');
            set aux = x;
            set x = y;
            set y = aux + x;
        end if;
        set i = i + 1;
    end while;
    return serie;
end;

```

```

select ejercicio_10(5) as serie fibonacci;

```

```

serie_fibonacci
1 0 1 1 2 3

```

```

create function suma_serie(entrada text)
returns int
begin
    declare espacio text default ' ';
    declare x int default 1;
    declare nVeces int default 0;
    declare letra char default '';
    declare limite int default char_length(entrada);
    declare a int default 0;
    declare b int default 1;
    declare cont int default 0;
    declare aux int default 0;
    declare suma int default 0;
    while x <= limite do
        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 suma = 0;
                else
                    set suma = suma + b;
                    set aux = a;
                    set a = b;
                    set b = aux + a;
                end if;
            set cont = cont + 1;
        end while;
        return suma;
    end;

```

```

select suma_serie(ejercicio_10(5)) as Ejercicio 10;

```

Ejercicio_10 ▾

1		7
---	--	---

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 view ejercicio 11 as
  select concat(per.nombre, ' ', per.apellidos) as Full_Name, per.edad,
per.fecha_nac,pro.nombreProy
  from persona as per
 inner join detalle proyecto as dep on per.id_per = dep.id_per
 inner join proyecto as pro on dep.id_proy = pro.id_proy
 inner join departamento as depa on per.id_dep = depa.id_dep
 where per.sexo = 'F' and depa.nombre = 'El Alto' and per.fecha_nac =
'2000-10-10';
```

	Full_Name	edad	fecha_nac	nombreProy
1	Nombre1 Apellido1	18	2000-10-10	Proyecto1

12. Manejo de TRIGGERS I

- Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO
 - Deberá de crear 2 triggers mínimamente.
- Agregar un nuevo campo a la tabla PROYECTO.
 - El campo debe llamarse ESTADO
- Actualmente solo se tiene habilitados ciertos tipos de proyectos.
 - EDUCACIÓN, FORESTACIÓN 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
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
alter table proyecto add estado varchar(8) not null;
```

```
create trigger insertando datos
before insert on proyecto
for each row
begin
    case
        when new.tipoProy = 'Educacion' or new.tipoProy = 'Forestacion' or new.tipoProy =
'Cultura' then
            set new.estado = 'Activo';
        else
            set new.estado = 'Inactivo';
    end case;
end;
```

```
create trigger actualizando datos
before update on proyecto
for each row
begin
    case
        when new.tipoProy = 'Educacion' or new.tipoProy = 'Forestacion' or new.tipoProy =
'Cultura' then
            set new.estado = 'Activo';
        else
            set new.estado = 'Inactivo';
    end case;
end;
```

	id_proy	nombreProy	tipoProy	estado
1	1	Proyecto1	Educacion	Activo
2	2	Proyecto2	Forestacion	Activo
3	3	Proyecto3	Cultura	Activo
4	4	Proyecto4	Gastronomia	Inactivo
5	5	Proyecto5	Deportes	Inactivo
6	6	Proyecto6	Educacion	Activo

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
    declare year int default 0;
    declare year actual int default 0;
    set year = (select max(substr(new.fecha_nac, 1, 4))
    from persona as per);
    set year actual = (SELECT substr(curdate(),1, 4));
    set new.edad = year actual - year;
  end;

```

```

insert into persona (nombre, apellidos, fecha_nac, email, sexo, id_dep, id_prov)
values ('Nombre6', 'Apellidos6', '1998-10-10', 'nombre6@gmail.com', 'F', 1, 1);

```

	id_per	nombre	apellidos	fecha_nac	edad
1	1	Nombre1	Apellido1	2000-10-10	18
2	2	Nombre2	Apellido2	2004-08-06	20
3	3	Nombre3	Apellido3	2002-02-10	22
4	4	Nombre4	Apellido4	2000-10-10	25
5	5	Nombre5	Apellido5	2000-10-10	19
6	6	Nombre6	Apellidos6	1998-10-10	24

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 trigger guardar datos
  before insert on persona
  for each row
  begin
    insert into copia persona (nombres, apellidos, fecha_nac, edad, email, sexo)
    values (new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo);
  end;
```

```
insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id dep, id prov)
values ('Nombre7', 'Apellido7', '1999-10-10', 23, 'email7@gmail.com', 'M', 1, 1);
```

	nombres	apellidos	fecha_nac	edad	email	sexo
1	Nombre7	Apellido7	1999-10-10	23	email7@gmail.com	M

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

La consulta generada convertirlo a vista

```
create view ejercicio 15 as
select concat(per.nombre, ' ', per.apellidos) as Full Name,
       depa.nombre, pro.nombre as nombrePro, concat(proy.nombreProy, ' ', proy.tipoProy) as
Proyecto
from departamento as depa
inner join provincia as pro on depa.id_dep = pro.id_dep
inner join persona as per on depa.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;
```

	Full_Name	nombre	nombrePro	Proyecto
1	Nombre1 Apellido1	El Alto	Murillo	Proyecto1 Educacion
2	Nombre2 Apellido2	Cochabamba	Cercado	Proyecto2 Forestacion
3	Nombre3 Apellido3	Santa Cruz	Chiquitos	Proyecto3 Cultura
4	Nombre4 Apellido4	Tarija	Mendez	Proyecto4 Gastronomia
5	Nombre5 Apellido5	Sucre	Tomina	Proyecto5 Deportes