TAREA HITO 4

BASE DE DATOS II

PRESENTA:
RONALD URIEL CHOQUE PACO
SIS6972733

EL ALTO, JUNIO DE 2022

TABLA DE CONTENIDO

01

PARTE TEORICA:

Se hara la definicion de diferentes conceptos de DBA relacionales, principalmente TRIGGER y VISTAS.

02

PARTE PRACTICA:

Donde se hara la aplicacion de los conceprtos mencionados en MySQL.

01

PARTE TEORICA

Defina que es lenguaje procedural en MySQL.

Es en síntesis la programación a nivel de DBA, en donde generamos distintas estructuras de control dentro de funciones almacenadas.

Defina que es una FUCNTION en MySQL.

Es un proceso usado o creado para tomar parámetros y transformarlos en salidas; el proceso es rutinario.

Cuál es la diferencia entre funciones y procedimientos almacenados.

Una función retorna una salida y se ejecuta en tomando parámetros; en cambio un procedimiento almacenado puede devolver varios valores, normalmente usado como un método de Common Runtime Language (CLR) (Lenguaje de tiempo de ejecución común).

Cómo se ejecuta una función y un procedimiento almacenado.

Una función siempre se ejecuta bajo la clausula SELECT y retorna el valor.

Un procedimiento se ejecuta bajo la clausula CALL haciendo la llamada de los valores.

Defina que es una TRIGGER en MySQL.

Es un objeto del servidor (script) que esta asociado con TABLAS.

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

NEW y OLD son OBJETOS que acceden a las columnas de las tablas.

- NEW: INSERT, UPDATE.
- OLD: DELETE, UPDATE.

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

Indican el tiempo en el que se ejecutaran los eventos de TRIGGER; BEFORE (antes de la ejecución) y AFTER (después de la ejecución).

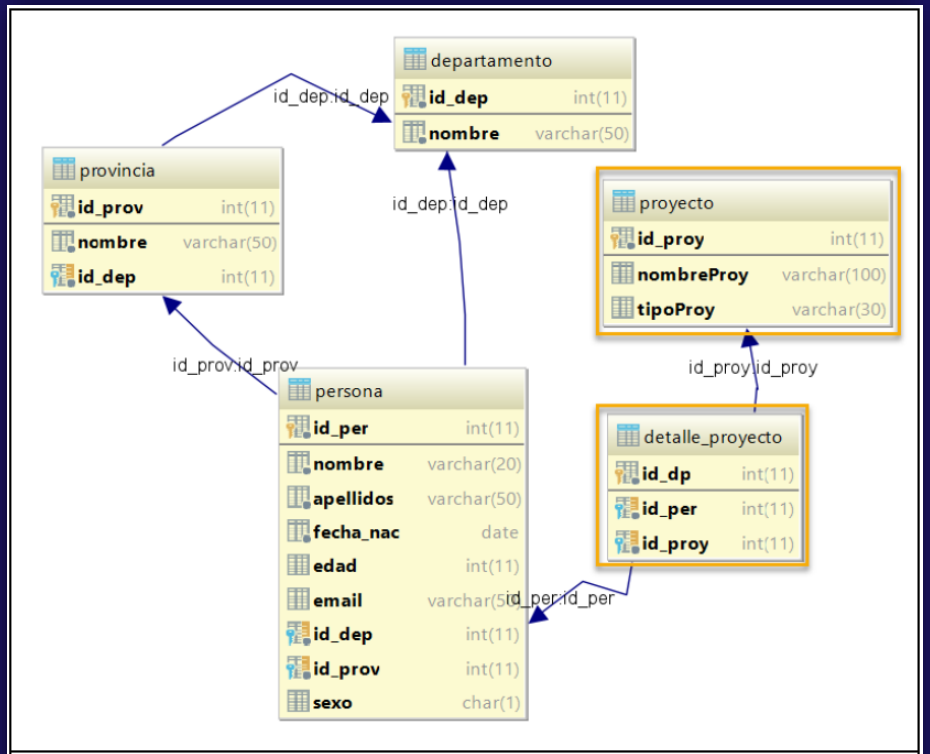
A que se refiere cuando se habla de eventos en TRIGGERS.

Se refiere a la ejecución del mismo; ya sea en los procesos de: INSERT, UPDATE y DELETE.

02

PARTE PRACTICA

02 Crear la siguiente Base de datos y sus registros.



Agregar mínimamente 2 registros a cada tabla

02 Crear la siguiente Base de datos y sus registros.

```sql
CREATE DATABASE evaluacionh4;
USE evaluacionh4;
```

```sql
CREATE TABLE departamento
(
    id_dep INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(50)
);
```

```sql
CREATE TABLE provincia
(
    id_prov INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(50),
    id_dep INTEGER,
    FOREIGN KEY (id_dep) REFERENCES departamento(id_dep)
);
```

```sql
CREATE TABLE proyecto
(
    id_proy INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombreProy VARCHAR(100),
    tipoProy VARCHAR(30)
);
```

```sql
CREATE TABLE persona
(
    id_per INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombres VARCHAR(20),
    apellidos VARCHAR(50),
    fecha_nac DATE,
    edad INTEGER,
    email VARCHAR(50),
    id_dep INTEGER,
    id_prov INTEGER,
    sexo CHAR,
    FOREIGN KEY (id_dep) REFERENCES departamento(id_dep),
    FOREIGN KEY (id_prov) REFERENCES provincia(id_prov)
);
```

```sql
CREATE TABLE detalle_proyecto
(
    id_dp INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_per INTEGER,
    id_proy INTEGER,
    FOREIGN KEY (id_per) REFERENCES persona(id_per),
    FOREIGN KEY (id_proy) REFERENCES proyecto(id_proy)
);
```

02 Crear la siguiente Base de datos y sus registros.

```
INSERT INTO departamento(nombre) VALUES ('La Paz'), ('Cochabamba'), ('Santa Cruz');
```

	id_dep	nombre
1	1	La Paz
2	2	Cochabamba
3	3	Santa Cruz

```
INSERT INTO provincia(nombre, id_dep) VALUES
('Murillo', 1), ('Cercado', 2), ('Andres Ibañez', 3);
```

	id_prov	nombre	id_dep
1	1	Murillo	1
2	2	Cercado	2
3	3	Andres Ibañez	3

```
INSERT INTO proyecto(nombreProy, tipoProy) VALUES
('Viviendas', 'Construccion'), ('Sisitema hospitalario', 'Salud'), ('Centro deportivo', 'Entretenimiento');
```

	id_proy	nombreProy	tipoProy
1	1	Viviendas	Construccion
2	2	Sisitema hospitalario	Salud
3	3	Centro deportivo	Entretenimiento

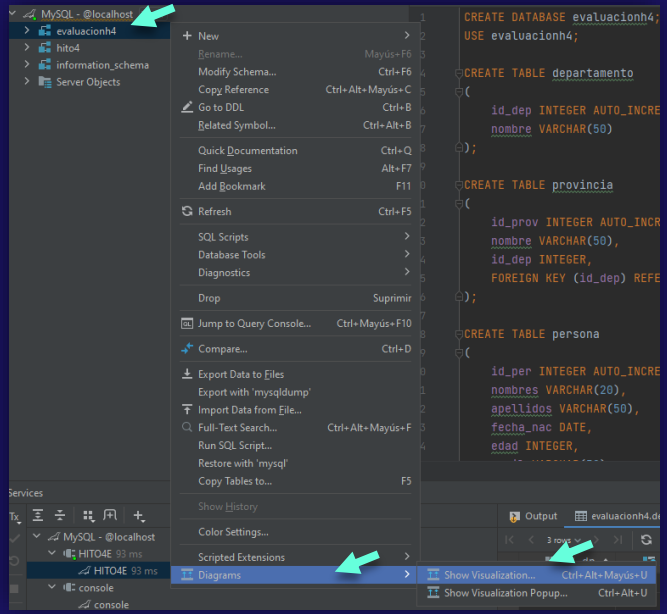
```
INSERT INTO detalle_proyecto(id_per, id_proy) VALUES
(2, 1), (1, 3), (3, 2);
```

	id_dp	id_per	id_proy
1	1	2	1
2	2	1	3
3	3	3	2

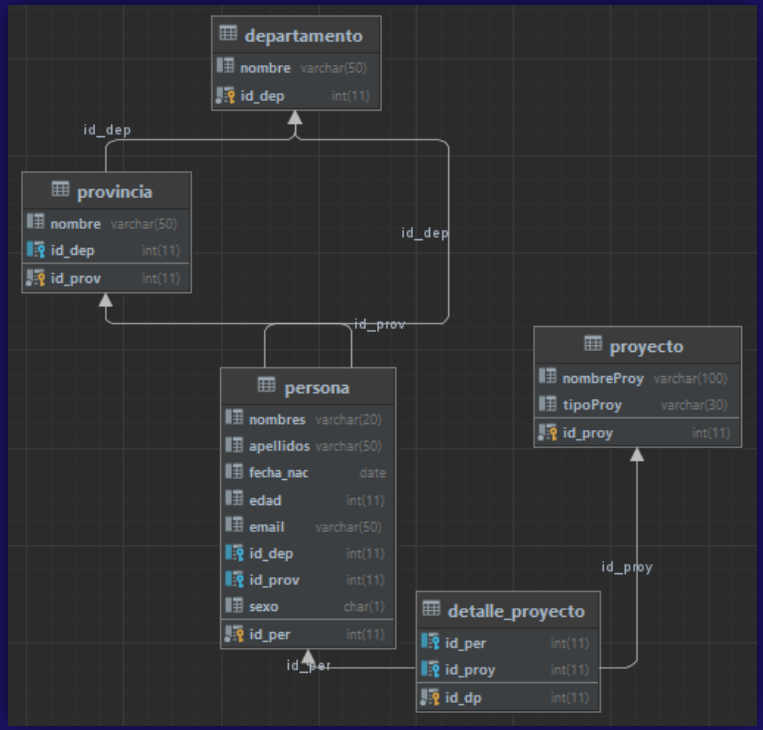
```
INSERT INTO persona(nombres, apellidos, fecha_nac, edad, email, id_dep, id_prov, sexo) VALUES
('Ronald', 'Choque', '2002-11-21', 19, 'ronald@gmail.com', 1, 1, 'm'),
('persona2', 'apellido2', '2004-01-01', 18, 'presona2@gmail.com', 3, 3, 'f'),
('persona3', 'apellido3', '2001-06-09', 20, 'persona3@gmail.com', 2, 2, 'm');
```

	id_per	nombres	apellidos	fecha_nac	edad	email	id_dep	id_prov	sexo
1	1	Ronald	Choque	2002-11-21	19	ronald@gmail.com	1	1	m
2	2	persona2	apellido2	2004-01-01	18	presona2@gmail.com	3	3	f
3	3	persona3	apellido3	2001-06-09	20	persona3@gmail.com	2	2	m

02 Crear la siguiente Base de datos y sus registros.



Haciendo un clic derecho sobre la carpeta de tablas, vaya y seleccione DIAGRAMS y la opción SHOW VISUALIZATION para poder ver el diagrama.



02

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

CODIGO.

```sql
CREATE OR REPLACE FUNCTION seriefibonacci(limite INT)
RETURNS TEXT
BEGIN
    DECLARE resp TEXT DEFAULT '';
    DECLARE i INT DEFAULT 0;
    DECLARE siguiente INT DEFAULT 1;
    DECLARE uno INT DEFAULT 0;
    DECLARE dos INT DEFAULT 1;

    WHILE i < limite DO
        IF i <= 1
        THEN
            SET siguiente = i;
        ELSE
            SET siguiente = uno + dos;
            SET uno = dos;
            SET dos = siguiente;
        end if;
        SET i = i + 1;
        SET resp = CONCAT(resp, siguiente, ' , ');
        end while;
    RETURN resp;
end;

SELECT seriefibonacci( limite: 10);
```

```sql
CREATE OR REPLACE FUNCTION sumafibonacci(limite INT)
RETURNS INT
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE siguiente INT DEFAULT 1;
    DECLARE uno INT DEFAULT 0;
    DECLARE dos INT DEFAULT 1;
    DECLARE suma INT DEFAULT 0;

    WHILE i < limite DO
        IF i <= 1
        THEN
            SET siguiente = i;
        ELSE
            SET siguiente = uno + dos;
            SET uno = dos;
            SET dos = siguiente;
        end if;
        SET i = i + 1;
        SET suma = suma + siguiente;
        end while;
    RETURN suma;
end;

SELECT sumafibonacci( limite: 10);
```

EJECUCION.

	`seriefibonacci(10)`
1	0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 ,

	`sumafibonacci(10)`
1	88

02 Manejo de vistas.

CODIGO.

```sql
CREATE OR REPLACE VIEW femnacElAlto AS
    SELECT CONCAT(per.nombres, ' ', per.apellidos) AS NOMBRE_COMPLETO, per.edad, per.fecha_nac, p.nombreProy
    FROM persona AS per
    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 departamento d on per.id_dep = d.id_dep
    WHERE per.fecha_nac = '2000-10-10' AND  per.sexo = 'f' AND d.id_dep = 2;

SELECT fec.* FROM femnacElAlto AS fec;
```

EJECUCION.

	NOMBRE_COMPLETO	edad	fecha_nac	nombreProy
1	nombre4 apellido4	21	2000-10-10	proyectoE
2	nombre6 apellido6	21	2000-10-10	proyectoE

NOTA: No se adjunta la ejecución debido a que esto conlleva a la muestra de las tablas, la explicación y muestra completa se encuentra en el video.

02 Manejo de TRIGGERS I.

CODIGO INSERTAR.

```
ALTER TABLE proyecto ADD estado VARCHAR(20);
```

```
ALTER TABLE proyecto ADD estado VARCHAR(20);

#para insetar
CREATE OR REPLACE TRIGGER activo_o_no
  BEFORE INSERT ON proyecto
  FOR EACH ROW
  begin
      IF NEW.tipoProy = 'EDUCACION' OR NEW.tipoProy = 'FORESTACION' OR NEW.tipoProy = 'CULTURA'
      THEN
          SET NEW.estado = 'ACTIVO';
      ELSE
          SET NEW.estado = 'INACTIVO';
      end if;
  end;
```

NOTA: No se adjunta la ejecución debido a que esto conlleva a la muestra de las tablas, la explicación y muestra completa se encuentra en el video.

CODIGO MODIFICAR.

```
CREATE OR REPLACE TRIGGER activo_o_no2
  BEFORE UPDATE ON proyecto
  FOR EACH ROW
  begin
      IF NEW.tipoProy = 'EDUCACION' OR NEW.tipoProy = 'FORESTACION' OR NEW.tipoProy = 'CULTURA'
      THEN
          SET NEW.estado = 'ACTIVO';
      ELSE
          SET NEW.estado = 'INACTIVO';
      end if;
  end;
```

02 Manejo de TRIGGERS II.

CODIGO.

```
CREATE OR REPLACE TRIGGER calcular_edad
  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;
```

NOTA: No se adjunta la ejecución debido a que esto conlleva a la muestra de las tablas, la explicación y muestra completa se encuentra en el video.

02 Manejo de TRIGGERS III.

TABLA AÑADIDA.

```sql
CREATE TABLE persona_copia
(
    nombres VARCHAR(20),
    apellidos VARCHAR(50),
    fecha_nac DATE,
    edad INTEGER,
    email VARCHAR(50),
    id_dep INTEGER,
    id_prov INTEGER,
    sexo CHAR,
    FOREIGN KEY (id_dep) REFERENCES departamento(id_dep),
    FOREIGN KEY (id_prov) REFERENCES provincia(id_prov)
);
```

CODIGO.

```sql
CREATE OR REPLACE TRIGGER copiar
  BEFORE INSERT ON persona
  FOR EACH ROW
  begin
      INSERT INTO persona_copia(nombres, apellidos, fecha_nac, edad, email, id_dep, id_prov, sexo)
      SELECT nombres, apellidos, fecha_nac, edad, email, id_dep, id_prov, sexo FROM persona;
  end;
```

NOTA: No se adjunta la ejecución debido a que esto conlleva a la muestra de las tablas, la explicación y muestra completa se encuentra en el video.

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

CODIGO.

```sql
CREATE OR REPLACE VIEW usar_todo AS
    SELECT CONCAT(pe.nombres, ' ' ,pe.apellidos) AS NOMBRES, d.nombre AS DEP, p.nombre AS PROV, p2.nombreProy AS TRABAJO
    FROM persona AS pe
    INNER JOIN departamento d on pe.id_dep = d.id_dep
    INNER JOIN provincia p on d.id_dep = p.id_dep
    INNER JOIN detalle_proyecto dp on pe.id_per = dp.id_per
    INNER JOIN proyecto p2 on dp.id_proy = p2.id_proy
    WHERE d.id_dep = 1 AND p.id_prov = 3;

SELECT usu.* FROM usar_todo AS usu;
```

EJECUCION.

	NOMBRES	DEP	PROV	TRABAJO
1	nombre2 apellido2	La Paz	Ingavi	proyectoE
2	nombre3 apellido3	La Paz	Ingavi	proyectoF
3	nombre1 apellido1	La Paz	Ingavi	proyectoC
4	nombre7 apellido7	La Paz	Ingavi	proyectoC

GRACIAS POR SU ATENCION!

¿Tiene alguna pregunta?

Ronald.choque2111@gmail.com

+591 65648933

EL ALTO, JUNIO DE 2022