TALLER # 6

ACTIVIDAD 1

Se inicia el diseño de los procedimientos donde se pueda agregar, actualizar, consultar y borrar en la base de datos de la librería:

Procedimiento para agregar:

```
1     USE libreriabuscalibre;
2
3     /* Se agrega tabla de procedimiento, para realizar
4     busquedas en la tabla de cliente por la cedula*/
5     DELIMITER //
6     CREATE PROCEDURE BUCAR_POR_ID(IN CODIGO_CLIENTE VARCHAR(10))
7     BEGIN
8     SELECT*FROM CLIENTE WHERE CEDULA = CODIGO_CLIENTE;
9     END
10    //
11    DELIMITER ;
```

Procedimiento para borrar:

```
14
15    /*Se realiza la contruccion de un procedimiento, el cual
16    me permite eliminar un libro cuando lo desee*/
17
18    DELIMITER //
19    CREATE PROCEDURE eliminar_libro (IN libro_id VARCHAR(10))
20    BEGIN
21      DELETE FROM libro WHERE ISBN = libro_id;
22    END
23    //
24    DELIMITER ;
```

Procedimiento para actualizar:

```
/*El siguiente procedimiento, es necesario para poder actualizar
los datos del cliente si es necesario*/

DELIMITER //
CREATE PROCEDURE actualizar_cliente(
    IN cedula VARCHAR(10),
    IN nombre VARCHAR(45)
)
BEGIN
    UPDATE cliente
    SET cedula = cedula,
        nombre = nombre
    WHERE nombre = nombre;
END //
DELIMITER ;
```

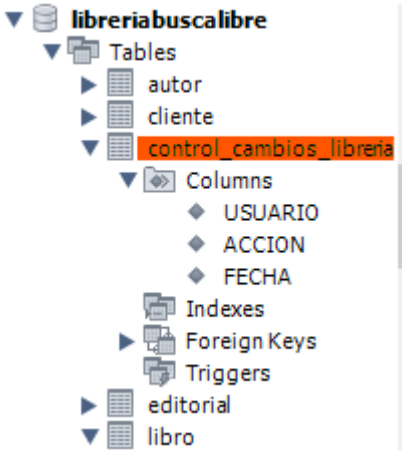
Procedimiento para consultar:

```
/*Se crea un procedimiento para consultar los libros
y a que editorial está relacionado*/

DELIMITER //
CREATE PROCEDURE consultar_libros()
BEGIN
    SELECT titulo, nombre_editorial FROM libro;
END //
DELIMITER ;
```

Luego de crear los procedimientos, se creará una nueva tabla, la cual tendrá como columnas un usuario, una fecha y una acción con las siguientes sentencias:

```
/*Se crea la tabla control de cambio libreria*/

CREATE TABLE control_cambios_libreria(
USUARIO VARCHAR(45),
ACCION VARCHAR (45),
FECHA DATE
);
```

Y como resultado, se creó la tabla en la base de datos:



A continuación, se crearán dos triggers, los cuales no ayudará a agregar y actualizar la información en la tabla nueva:

Para crear los triggers, se utilizaron las siguientes sentencias:

```
11    /*Se crean los triggers*/
12
13    DELIMITER //
14    CREATE TRIGGER guardar_nombre_cliente
15    AFTER INSERT ON cliente FOR EACH ROW
16    BEGIN
17      INSERT INTO control_cambios_libreria (usuario, accion, fecha)
18      VALUES (NEW.nombre, 'NUEVO CLIENTE AGREGADO', NOW());
19    END //
20    DELIMITER ;
21
22    DELIMITER //
23    CREATE TRIGGER guardar_nombre_cliente
24    AFTER UPDATE ON cliente FOR EACH ROW
25    BEGIN
26      INSERT INTO control_cambios_libreria (usuario, accion, fecha)
27      VALUES (NEW.nombre, 'SE ACTUALIZA EL CLIENTE', NOW());
28    END //
29    DELIMITER ;
```

ACTIVIDAD 2

Para esta nueva actividad, se tendrá en cuenta la base de datos del hospital, a la cual le agregaremos cuatro procedimientos, para facilitar las consultas:

El primer procedimiento es de buscar:

```
1     USE hospital;
2
3     /* Se agrega tabla de procedimiento, para realizar
4     busquedas en la tabla de medicamento por su nombre*/
5     DELIMITER //
6     CREATE PROCEDURE BUCAR_POR_NOMBRE(IN NOMBRE_MEDICAMENTO VARCHAR(45))
7     BEGIN
8     SELECT*FROM MEDICAMENTO WHERE NOMBRE = NOMBRE_MEDICAMENTO;
9     END
10    //
11    DELIMITER ;
```

El segundo procedimiento es de eliminar:

```
15    /*Se realiza la contruccion de un procedimiento, el cual
16    me permite eliminar algun paciente cuando lo desee*/
17
18      DELIMITER //
19    CREATE PROCEDURE ELIMINAR_PACIENTE (IN ID_PACIENTE VARCHAR(10))
20    BEGIN
21        DELETE FROM PACIENTE WHERE ID = ID_PACIENTE;
22    END
23    //
24    DELIMITER ;
```

El tercer procedimiento es de actualizar:

```
26    call ELIMINAR_PACIENTE ("CARLOS")
27
28    /*El siguiente procedimiento, es necesario para poder actualizar
29    los datos del paciente si es necesario*/
30
31    DELIMITER //
32    CREATE PROCEDURE ACTUALIZAR_PACIENTE(
33        IN ID VARCHAR(10),
34        IN NOMBRE VARCHAR(45),
35        IN APELLIDO VARCHAR(45),
36        IN DIRECCION VARCHAR(45)
37    )
37    )
38    BEGIN
39        UPDATE PACIENTE
40        SET ID = ID,
41            NOMBRE = NOMBRE,
42            APELLIDO = APELLIDO,
43            DIRECCION = DIRECCION
44        WHERE ID = ID;
45    END //
46    DELIMITER ;
```

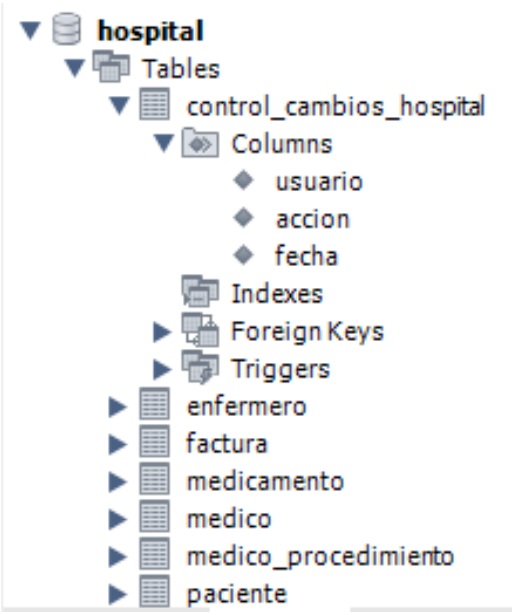
El procedimiento de consultar:

```
50    /*Se crea un procedimiento para consultar los pacientes
51    y que procedimiento se le ha realizado*/
52
53    DELIMITER //
54    CREATE PROCEDURE CONSULTAR_PACIENTE()
55    BEGIN
56        SELECT NOMBRE, ID_PROCEDIMIENTO FROM PACIENTE;
57    END //
58    DELIMITER ;
59
60    CALL CONSULTAR_PACIENTE()
```

Para finalizar, se crea una tabla, la cual nos va a servir para controlar los cambios en el hospital con las siguientes sentencias:

```
1   USE HOSPITAL;
2
3   /*Se crea la tabla control de cambio hospital*/
4
5   CREATE TABLE control_cambios_hospital (
6     usuario VARCHAR(45),
7     accion VARCHAR(45),
8     fecha DATE
9   );
```

Y como resulta, tenemos agregada la tabla como se visualiza en la siguiente imagen:



Seguido de lo anterior, se agregarán dos triggers, los cuales nos servirán para la actualización en la tabla de un cliente o cuando se actualiza algún dato del mismo.

El primer trigger que vemos en la siguiente imagen, es para agregar o insertar:

```
11      /* Se crea el primer trigger para guardar los cambios en la tabla paciente*/
12
13      DELIMITER //
14      CREATE TRIGGER insertar_paciente_control_cambios
15      AFTER INSERT ON paciente FOR EACH ROW
16      BEGIN
17          INSERT INTO control_cambios_hospital (usuario, accion, fecha)
18          VALUES (USER(), 'Se agregó un paciente', NOW());
19      END//
20      DELIMITER ;
```

El segundo trigger es para cuando se actualiza algún dato:

```
22      /*Se crea el segundo trigger para actualizar los datos en la tabla paciente*/
23      DELIMITER //
24      CREATE TRIGGER actualizar_paciente_control_cambios
25      AFTER UPDATE ON paciente FOR EACH ROW
26      BEGIN
27          INSERT INTO control_cambios_hospital (usuario, accion, fecha)
28          VALUES (USER(), 'Se ha actualzado un paciente', NOW());
29      END //
30      DELIMITER ;
```

Conclusión:

Para finalizar se concluye que, es importante cada uno de las tablas, procedimientos y triggers agregados anteriormente, para tener un mayor orden en la gestión de una base de datos. Además, para facilitar incluso las consultas que podrían ser más fundamentales o tediosas.