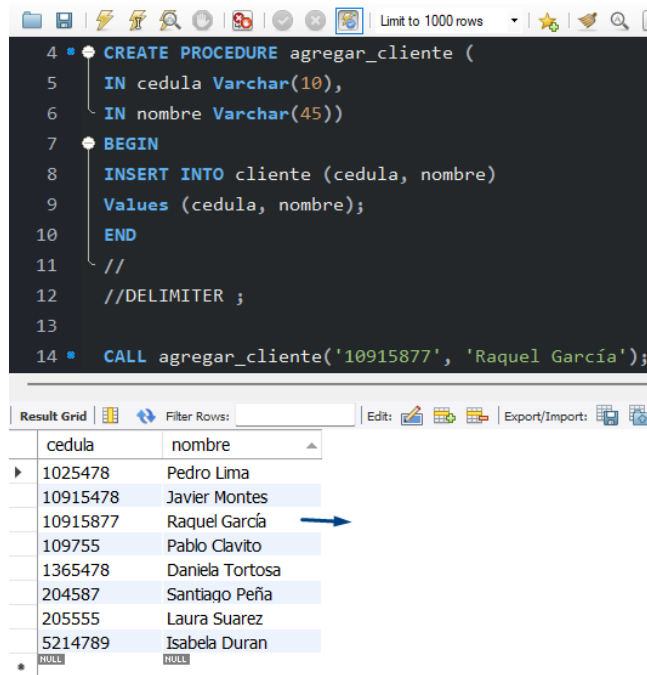


Documentación taller 6

Elabore 4 procedimientos almacenados que me permitan agregar, actualizar, consultar y borrar, en una de las tablas de la librería (primera actividad).

Tabla cliente:

- 1- Se agregar un cliente con su cedula y nombre a



The screenshot shows a SQL IDE window with a dark theme. The top toolbar includes icons for file operations, execution, and search. The main editor displays a SQL script for creating and calling a stored procedure. Below the editor, the 'Result Grid' shows the output of the query, which is a table with two columns: 'cedula' and 'nombre'. The table contains eight rows of data, with the third row highlighted in blue and a blue arrow pointing to it.

```
4 CREATE PROCEDURE agregar_cliente (  
5     IN cedula Varchar(10),  
6     IN nombre Varchar(45))  
7 BEGIN  
8     INSERT INTO cliente (cedula, nombre)  
9     Values (cedula, nombre);  
10 END  
11 //  
12 //DELIMITER ;  
13  
14 CALL agregar_cliente('10915877', 'Raquel García');
```

cedula	nombre
1025478	Pedro Lima
10915478	Javier Montes
10915877	Raquel García
109755	Pablo Clavito
1365478	Daniela Tortosa
204587	Santiago Peña
205555	Laura Suarez
5214789	Isabela Duran
NULL	NULL

Consultar:

```
#consultar
DELIMITER //
CREATE PROCEDURE consultar_libro(
IN ISBN VARCHAR(45)
)
BEGIN
SELECT * FROM libro
WHERE ISBN = ISBN;
END //
DELIMITER ;

CALL consultar_libro('1253');
```

Modificar:

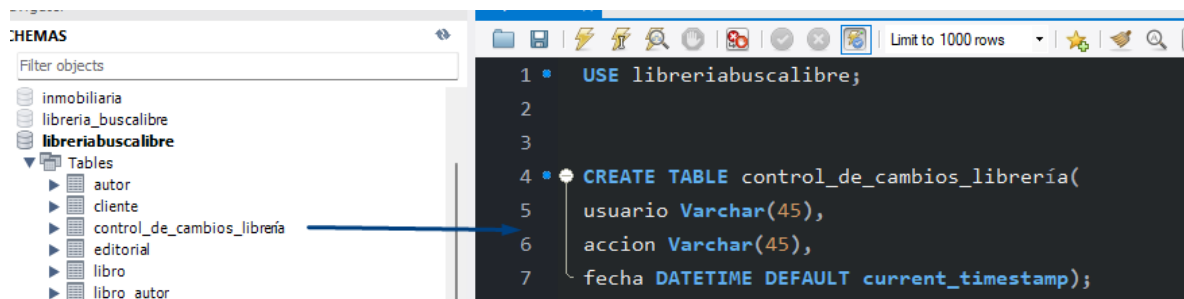
```
#Actualizar tabla

DELIMITER //
CREATE PROCEDURE modificar_nombre (
IN cedula Varchar(10),
IN nombre Varchar(45)
)
BEGIN
UPDATE cliente
SET nombre = nombre
WHERE cedula = cedula;
END
//
//DELIMITER ;
CALL modificar_nombre ('109755', 'Pablo Clavijo');
```

Borra:

```
34
35 #Borrar cliente
36 DELIMITER //
37 • CREATE PROCEDURE eliminar_cliente (
38     IN cedula Varchar(10),
39     IN nombre Varchar(45)
40 )
41 BEGIN
42     DELETE FROM cliente
43     WHERE cedula = cedula;
44 END
45 //
46 //DELIMITER ;
47
48 • CALL eliminar_cliente ('109755');
49
```

Elabore una nueva tabla llamada "control_de_cambios_librería" la cual debe contener 3 columnas (usuario, accion, fecha) y guarde utilizando 2 Triggers el nombre del usuario que agrego o elimino un registro en la tabla seleccionada en el punto anterior.



The screenshot shows a database management tool interface. On the left, a tree view displays the database structure. The 'libreria_buscabilibre' database is selected, and its tables are listed: autor, cliente, control_de_cambios_librería, editorial, libro, and libro_autor. A blue arrow points from the 'control_de_cambios_librería' table to the SQL editor on the right. The SQL editor contains the following code:

```
1 • USE libreria_buscabilibre;
2
3
4 • CREATE TABLE control_de_cambios_librería(
5     usuario Varchar(45),
6     accion Varchar(45),
7     fecha DATETIME DEFAULT current_timestamp);
```

```

9      #creación del TIGGER
10     DELIMITER //
11 •   CREATE TRIGGER insertar_registro
12     AFTER INSERT ON cliente
13     FOR EACH ROW
14     BEGIN
15         INSERT INTO control_de_cambios_librería (usuario, accion, fecha)
16         VALUES (usuario(), 'agregar', NOW());
17     END;
18     // DELIMITER ;
19
20
21
22     DELIMITER //
23 •   CREATE TRIGGER eliminar_registro
24     AFTER DELETE ON cliente
25     FOR EACH ROW
26     BEGIN
27         INSERT INTO control_de_cambios_librería (usuario, accion, fecha)
28         VALUES (USER(), 'eliminar', NOW());
29     END;
30     // DELIMITER ;
31
32

```

```

#Agregar un medico

DELIMITER //
•   CREATE PROCEDURE insertar_medico(
    IN id_medico VARCHAR(10),
    IN nombre VARCHAR(45),
    IN especialidad VARCHAR(45)
)
•   BEGIN
    INSERT INTO medico (id_medico, nombre, especialidad)
    VALUES (id_medico, nombre, especialidad);
    END//
    // DELIMITER ;

CALL insertar_medico('120M', 'pedro Carrillo', 'Cardiología');

```

Elabore 4 procedimientos almacenados que me permitan agregar, actualizar, consultar y borrar, en una de las tablas del Hospital (segunda actividad).

Agregar medico:

```
#Agregar un medico

DELIMITER //

CREATE PROCEDURE insertar_medico(
  IN id_medico VARCHAR(10),
  IN nombre VARCHAR(45),
  IN especialidad VARCHAR(45)
)

BEGIN
  INSERT INTO medico (id_medico, nombre, especialidad)
  VALUES (id_medico, nombre, especialidad);
END//

// DELIMITER ;

CALL insertar_medico('120M', 'pedro Carrillo', 'Cardiología');
```

consultar médico:

```
9 #consultar un medico
10
11 DELIMITER //
12 • CREATE PROCEDURE consultar_medico(
13   IN id_medico VARCHAR(10),
14   IN nombre VARCHAR(45),
15   IN especialidad VARCHAR(45)
16 )
17
18 • BEGIN
19   SELECT * FROM medico
20   WHERE id_medico = id_medico;
21   END//
22   // DELIMITER ;
23
24 • CALL insertar_medico('120M');
```

Actualizar medico:

```
#actualizar medico

DELIMITER //

• CREATE PROCEDURE actualizar_medico_nombre(
  IN id_medico VARCHAR(10),
  IN nombre VARCHAR(45),
  IN especialidad VARCHAR(45)
)

• BEGIN
  UPDATE medico
  SET nombre = nombre
  WHERE id_medico = id_medico AND especialidad = especialidad;
END//
// DELIMITER ;

• CALL insertar_medico('100M', 'Juan Carrillo', 'Cardiología');
```

Elabore una nueva tabla llamada "control_de_cambios_hospital" la cual debe contener 3 columnas (usuario, accion, fecha) y guarde utilizando 2 Triggers el nombre del usuario que agrego o elimino un registro en la tabla seleccionada en el punto anterior.

```
USE hospital;

CREATE TABLE control_de_cambios_hospital(
    usuario Varchar(45),
    accion Varchar(45),
    fecha DATETIME DEFAULT current_timestamp);
```

```
7
8  #creación del TIGGER
9  DELIMITER //
10 • CREATE TRIGGER agregar_medico
11  AFTER INSERT ON medico
12  FOR EACH ROW
13  BEGIN
14      INSERT INTO control_de_cambios_hospital (usuario, accion, fecha)
15      VALUES (usuario(), 'agregar', NOW());
16  END;
17  // DELIMITER ;
18
19
20  DELIMITER //
21 • CREATE TRIGGER eliminar_medico
22  AFTER DELETE ON medico
23  FOR EACH ROW
24  BEGIN
25      INSERT INTO control_de_cambios_hospital (usuario, accion, fecha)
26      VALUES (USER(), 'eliminar', NOW());
27  END;
28  // DELIMITER ;
29
30
```