

PASO A PASO DE LA SOLUCIÓN DEL TALLER #4

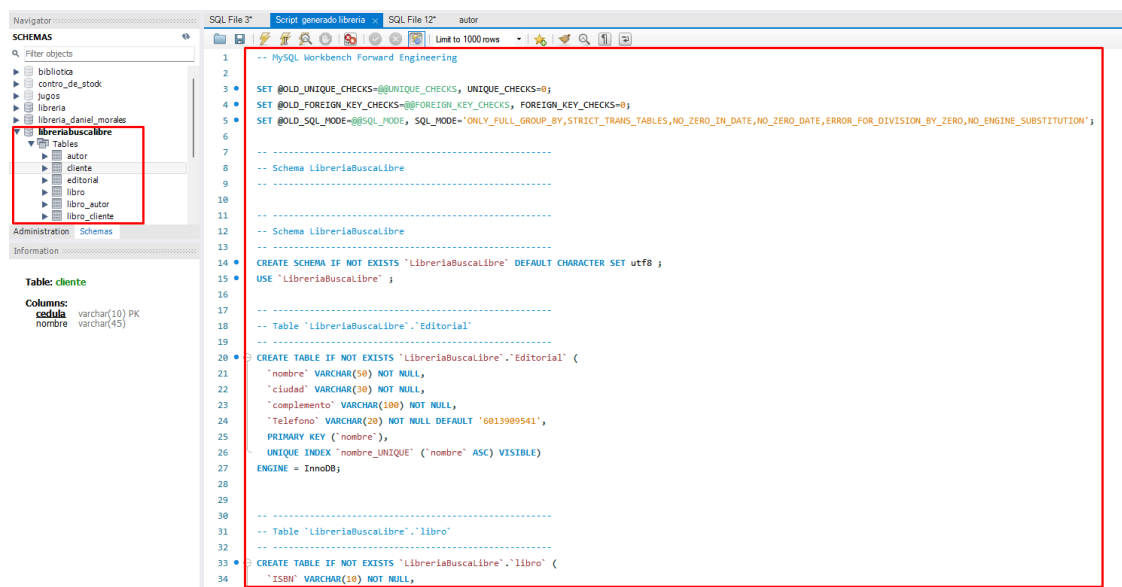
- **Punto 1 de la primera actividad:**

Complete la información para las tablas autor, libro, cliente, editorial, libro_cliente, libro_autor y teléfono_cliente con al menos (5,20,7,4,10,10, 12) registros respectivamente usando únicamente comandos SQL creados por usted.

1. Para abordar el punto 1 de la primera actividad, es crucial desglosar la información recibida con el fin de tener una idea clara de cuántos registros se deben crear para cada tabla. De esta manera:

- Autor = 5 registros
- Libro = 20 registros
- Cliente = 7 registros
- Editorial = 4 registros
- Libro_cliente = 10 registros
- Libro_autor = 10 registros
- Telefono_cliente = 12 registros

2. Se ejecuta en Workbench los scripts SQL recibidos



3. Se comienza a poblar las tablas.

- Comandos SQL para poblar la tabla de clientes.

```
1 • INSERT INTO libreriaescalibre.cliente (cedula, nombre)
2 VALUES
3 ('32567876', 'Leidy Laura Martinez Restrepo'),
4 ('1017543567', 'Paula Andrea Cordoba Sanchez'),
5 ('1077653789', 'Pablo Emilio Castillo'),
6 ('32456765', 'Sara Caicedo'),
7 ('1077654321', 'José Lopez'),
8 ('4567890', 'Pedro Gonzalez'),
9 ('1011654321', 'Gissela Moreno');
10
```

- Tabla cliente poblada con 7 registros

Table: cliente

cedula	nombre
1011654321	Gissela Moreno
1017543567	Paula Andrea Cordoba Sanchez
1077653789	Pablo Emilio Castillo
1077654321	José Lopez
32456765	Sara Caicedo
32567876	Leidy Laura Martinez Restrepo
4567890	Pedro Gonzalez

- Comandos SQL para poblar la tabla editorial.

```
1 • INSERT INTO libreriaescalibre.editorial (nombre, ciudad, complemento, Telefono)
2 VALUES
3 ('Gato Malo', 'Bogotá', 'Calle 7 1 84', '672345678'),
4 ('Planeta', 'Cali', 'Carrera 3 7 1', '3234567898'),
5 ('Babel Libros', 'Bogotá', 'Calle 23 56 78', '3008765443'),
6 ('Siruela', 'Madrid', 'c/ Almagro 25. 28010', '+34 913555720');
7
```

- Tabla editorial poblada con 4 registros

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'editorial' table under the 'cliente' schema. The 'Table: editorial' information pane shows the following columns:

Column	DataType	PK
nombre	varchar(50)	PK
ciudad	varchar(30)	
complemento	varchar(100)	
Telefono	varchar(20)	

The main pane shows the SQL query: `SELECT * FROM libreriabuscabilib.re.editorial;` and the 'Result Grid' with 4 rows of data:

nombre	ciudad	complemento	Telefono
Babel Libros	Bogota	Calle 23 56 78	3008765443
Gato Molo	Bogota	Calle 7 1 84	672345678
Planeta	Cali	Carrera 3 7 1	3234567898
Sruela	Madrid	c/ Almagro 25. 28010	+34 913555720

- Comandos SQL para poblar la tabla autor.

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'autor' table under the 'cliente' schema. The 'Table: autor' information pane shows the following columns:

Column	DataType	PK
id	varchar(10)	PK
fecha_nacimiento	varchar(45)	
nacionalidad	varchar(20)	
nombre	varchar(45)	

The main pane shows the SQL command to populate the 'autor' table:

```
INSERT INTO libreriabuscabilib.autor (id, fecha_nacimiento, nacionalidad, nombre)
VALUES
('ID001', '1927-03-06', 'Mexicano', 'Gabriel Garcia Marquez'),
('ID002', '1899-07-21', 'Estadounidense', 'Ernest Hemingway'),
('ID003', '1947-09-21', 'Estadounidense', 'Stephen King'),
('ID004', '1947-08-24', 'Brasileño', 'Paulo Coelho'),
('ID005', '1809-01-19', 'Estadounidense', 'Edgar Allan Poe');
```

- Tabla autor poblada con 5 registros

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'autor' table under the 'cliente' schema. The 'Table: autor' information pane shows the following columns:

Column	DataType	PK
id	varchar(10)	PK
fecha_nacimiento	varchar(45)	
nacionalidad	varchar(20)	
nombre	varchar(45)	

The main pane shows the SQL query: `SELECT * FROM libreriabuscabilib.autor;` and the 'Result Grid' with 5 rows of data:

id	fecha_nacimiento	nacionalidad	nombre
ID001	1927-03-06	Mexicano	Gabriel Garcia Marquez
ID002	1899-07-21	Estadounidense	Ernest Hemingway
ID003	1947-09-21	Estadounidense	Stephen King
ID004	1947-08-24	Brasileño	Paulo Coelho
ID005	1809-01-19	Estadounidense	Edgar Allan Poe

- Comandos SQL para poblar la tabla libro.

The screenshot shows a database management tool with a left sidebar displaying the schema structure. The 'libro' table is selected, showing its columns: ISBN (PK), titulo, numero_paginas, and nombre_editorial. The main window displays the following SQL command:

```

1 • INSERT INTO libreriausbcalibre.libro (ISBN, titulo, numero_paginas, nombre_editorial)
2 VALUES
3 ('978-0307474728', 'Cien años de soledad', '448', 'Gato Malo'),
4 ('978-0307389731', 'El amor en los tiempos del cólera', '368', 'Gato Malo'),
5 ('978-1400034956', 'Crónica de una muerte anunciada', '128', 'Gato Malo'),
6 ('978-0670813025', 'El otoño del patriarca', '272', 'Gato Malo'), -- Corregido el ISBN
7 ('978-1400033956', 'La hojarasca', '128', 'Gato Malo'),
8 ('978-8471668971', 'El viejo y el mar', '128', 'Planeta'),
9 ('978-8439915007', 'Adiós a las armas', '336', 'Planeta'),
10 ('978-8437619301', 'Por quién doblan las campanas', '576', 'Planeta'),
11 ('978-8499894141', 'Fiesta', '352', 'Planeta'),
12 ('978-8435018629', 'París era una fiesta', '272', 'Planeta'),
13 ('978-1501142970', 'It', '1138', 'Babel Libros'),
14 ('978-8432210295', 'El resplandor', '672', 'Babel Libros'),
15 ('978-8423346607', 'Cementerio de animales', '456', 'Babel Libros'),
16 ('978-0451169525', 'Misery', '368', 'Babel Libros'),
17 ('978-8401352327', 'Carrie', '304', 'Babel Libros'),
18 ('978-0062315007', 'El alquimista', '208', 'Siruela'),
19 ('978-0062502179', 'Brida', '240', 'Siruela'),
20 ('978-0060589288', 'Once minutos', '320', 'Siruela'),
21 ('978-0061124266', 'Veronika decide morir', '240', 'Siruela'),
22 ('978-0062512796', 'El peregrino de Compostela', '208', 'Siruela');

```

- Tabla libro poblada con 20 registros

The screenshot shows the same database management tool with the SQL command `SELECT * FROM libreriausbcalibre.libro;` executed. The result grid displays 20 records, which are highlighted with a red border. The records are as follows:

ISBN	titulo	numero_paginas	nombre_editorial
978-0060589288	Once minutos	320	Siruela
978-0061124266	Veronika decide morir	240	Siruela
978-0062315007	El alquimista	208	Siruela
978-0062502179	Brida	240	Siruela
978-0062512796	El peregrino de Compostela	208	Siruela
978-0307389731	El amor en los tiempos del cólera	368	Gato Malo
978-0307474728	Cien años de soledad	448	Gato Malo
978-0451169525	Misery	368	Babel Libros
978-0670813025	El otoño del patriarca	272	Gato Malo
978-1400033956	La hojarasca	128	Gato Malo
978-1400034956	Crónica de una muerte anunciada	128	Gato Malo
978-1501142970	It	1138	Babel Libros
978-8401352327	Carrie	304	Babel Libros
978-8423346607	Cementerio de animales	456	Babel Libros
978-8432210295	El resplandor	672	Babel Libros
978-8435018629	París era una fiesta	272	Planeta
978-8437619301	Por quién doblan las campanas	576	Planeta
978-8439915007	Adiós a las armas	336	Planeta
978-8471668971	El viejo y el mar	128	Planeta
978-8499894141	Fiesta	352	Planeta
NULL	NULL	NULL	NULL

- Comandos SQL para poblar la tabla libro_cliente.

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, with 'libro_cliente' selected under the 'libro' schema. The 'Table: libro_cliente' information pane shows the columns: 'ISBN_libro_cliente' (varchar(20), PK) and 'id_cliente' (varchar(10), PK). The main SQL editor window displays the following SQL command:

```
INSERT INTO libreriaescalibre.libro_cliente (ISBN_libro_cliente, id_cliente)
VALUES
('978-0307474728', '1011654321'),
('978-0307389731', '1017543567'),
('978-1400034956', '1077653789'),
('978-0670813025', '1077654321'),
('978-1400034956', '32456765'),
('978-8471668971', '32567876'),
('978-8439915007', '4567890'),
('978-8437619301', '1011654321'),
('978-8499894141', '1017543567'),
('978-8435018629', '1077653789');
```

- Tabla libro_cliente poblada, con 10 registros.

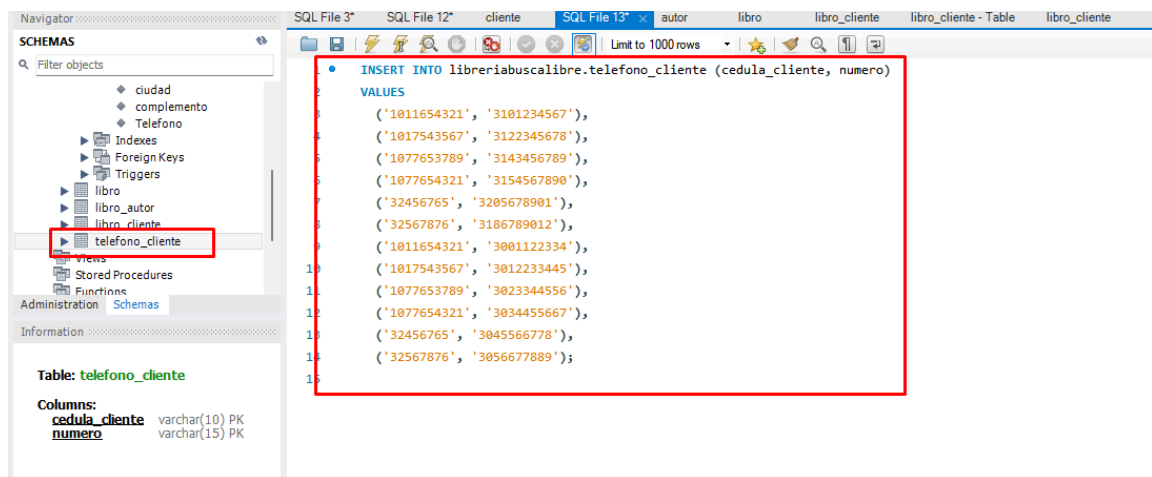
The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, with 'libro_cliente' selected under the 'libro' schema. The 'Table: libro_cliente' information pane shows the columns: 'ISBN_libro_cliente' (varchar(20), PK) and 'id_cliente' (varchar(10), PK). The main SQL editor window displays the following SQL command:

```
SELECT * FROM libreriaescalibre.libro_cliente;
```

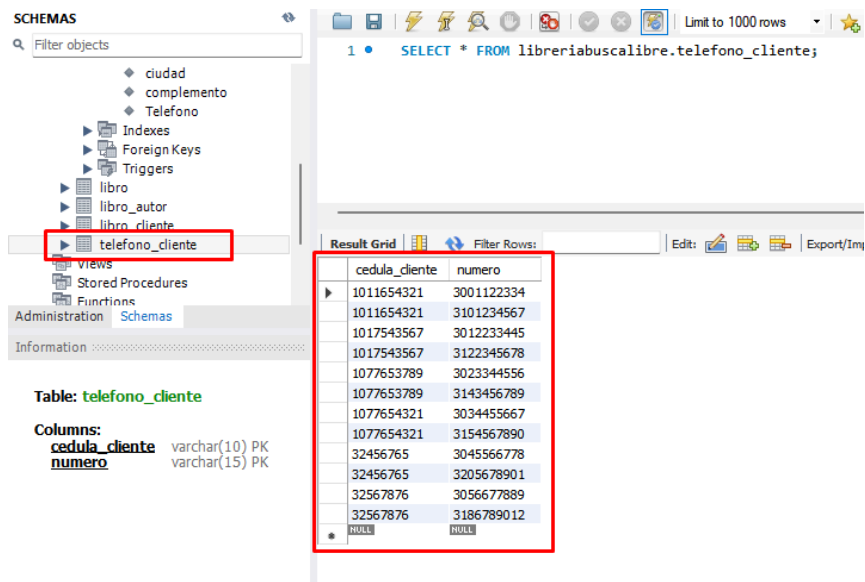
The 'Result Grid' pane shows the following data:

ISBN_libro_cliente	id_cliente
978-0307474728	1011654321
978-8437619301	1011654321
978-0307389731	1017543567
978-8499894141	1017543567
978-1400034956	1077653789
978-8435018629	1077653789
978-0670813025	1077654321
978-1400034956	32456765
978-8471668971	32567876
978-8439915007	4567890
NULL	NULL

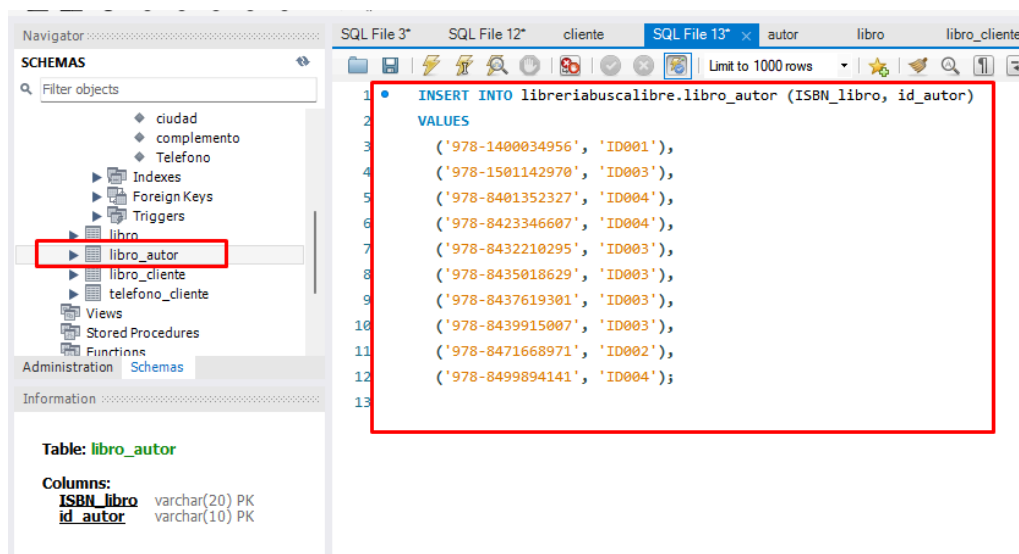
- Comandos SQL para poblar la tabla telefono_cliente.



- Tabla telefono_cliente poblada con 12 registros.



- Comandos SQL para poblar la tabla libro_autor.



- Tabla libro_autor poblada con 10 registros.

Table: libro_autor

Columns:

- ISBN_libro varchar(20) PK
- id_autor varchar(10) PK

ISBN_libro	id_autor
978-1400034956	ID001
978-8471668971	ID002
978-8432210295	ID003
978-8435018629	ID003
978-8437619301	ID003
978-8439915007	ID003
978-8423346607	ID004
978-8499894141	ID004
978-1501142970	ID005
978-8401352327	ID005

- **Punto 2 de la primera actividad:**

Realice 5 consultas que me permitan conocer el nombre y la fecha de nacimiento de cada escritor, la cantidad de libros diferentes vendidos, el nombre de su cliente acompañado de su número telefónico, el nombre del libro acompañado por su autor o sus autores, el nombre de las editoriales que han logrado vender libros.

4. Para realizar este punto se toman las consultas requeridas.

- ★ *nombre y la fecha de nacimiento de cada escritor*
- ★ *cantidad de libros diferentes vendidos*
- ★ *nombre de su cliente acompañado de su número telefónico*
- ★ *nombre del libro acompañado por su autor o sus autores*
- ★ *nombre de las editoriales que han logrado vender libros*

- **Consultas realizadas (Se adjunta archivo consultas_libreria.sql):**

- ★ *Nombre y la fecha de nacimiento de cada escritor:*

La consulta devuelve el nombre y la fecha de nacimiento de cada escritor almacenado en la tabla "autor".

```

1  -- nombre y fecha de nacimiento de cada escritor
2  • select nombre, fecha_nacimiento from autor;
3
4
5

```

nombre	fecha_nacimiento
Gabriel Garcia Marquez	1927-03-06
Ernest Hemingway	1899-07-21
Stephen King	1947-09-21
Paulo Coelho	1947-08-24
Edgar Allan Poe	1809-01-19

★ *Cantidad de libros diferentes vendidos:*

Esta consulta devuelve el número de libros diferentes que han sido vendidos, contando solo una vez cada libro, independientemente de la cantidad de veces que haya sido comprado.

```

3
4  -- Cantidad de libros diferentes vendidos
5  • select count(distinct(ISBN_libro_cliente)) from libro_cliente;
6

```

count(distinct(ISBN_libro_cliente))
9

★ *nombre de su cliente acompañado de su número telefónico:*

La consulta devuelve el nombre de los clientes junto con sus números de teléfono asociados, utilizando la relación entre las tablas cliente y telefono_cliente a través de la columna cedula.


```

8      -- nombre de cliente del libro acompañado de numero telefonico
9      • select cliente.nombre, telefono_cliente.numero
10     from cliente
11     inner join telefono_cliente on cliente.cedula = telefono_cliente.cedula_cliente;
12

```

nombre	numero
Gissela Moreno	3001122334
Gissela Moreno	3101234567
Paula Andrea Cordoba Sanchez	3012233445
Paula Andrea Cordoba Sanchez	3122345678
Pablo Emilio Castillo	3023344556
Pablo Emilio Castillo	3143456789
José Lopez	3034455667
José Lopez	3154567890
Sara Caicedo	3045566778
Sara Caicedo	3205678901
Leidy Laura Martinez Restrepo	3056677889
Leidy Laura Martinez Restrepo	3186789012

★ *nombre del libro acompañado por su autor o sus autores:*

La consulta devuelve el título del libro acompañado por el nombre de su autor o autores, utilizando la relación entre las tablas libro, libro_autor y autor.

```

13      -- nombre de libro acompañado por su autor o sus autores
14      • select libro.titulo, autor.nombre
15     from libro
16     join libro_autor on libro.ISBN = libro_autor.ISBN_libro
17     join autor on autor.id = libro_autor.id_autor;
18
19

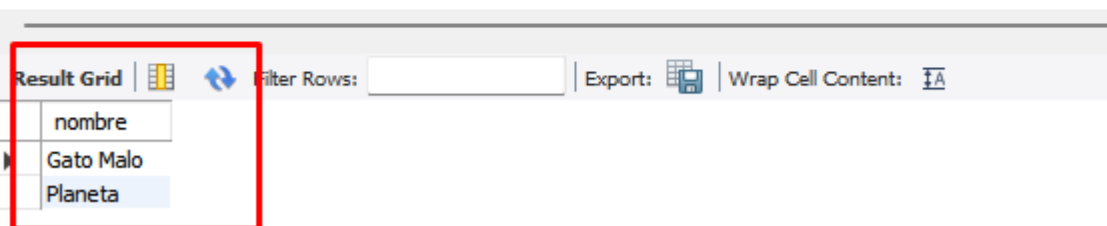
```

titulo	nombre
Crónica de una muerte anunciada	Gabriel Garcia Marquez
El viejo y el mar	Ernest Hemingway
El resplandor	Stephen King
París era una fiesta	Stephen King
Por quién doblan las campanas	Stephen King
Adiós a las armas	Stephen King
Cementerio de animales	Paulo Coelho
Fiesta	Paulo Coelho
It	Edgar Allan Poe
Carrie	Edgar Allan Poe

★ *nombre de las editoriales que han logrado vender libros:*

La consulta devuelve el nombre de las editoriales que han logrado vender libros, utilizando la relación entre las tablas editorial, libro y libro_cliente. La cláusula “distinct” asegura que solo se devuelvan los nombres de las editoriales sin duplicados.

```
21 -- nombre de las editoriales que hayan logrado vender libros
22 • select distinct editorial.nombre from editorial
23 join libro on libro.nombre_editorial = editorial.nombre
24 join libro_cliente on libro_cliente.ISBN_libro_cliente = libro.ISBN;
25
```



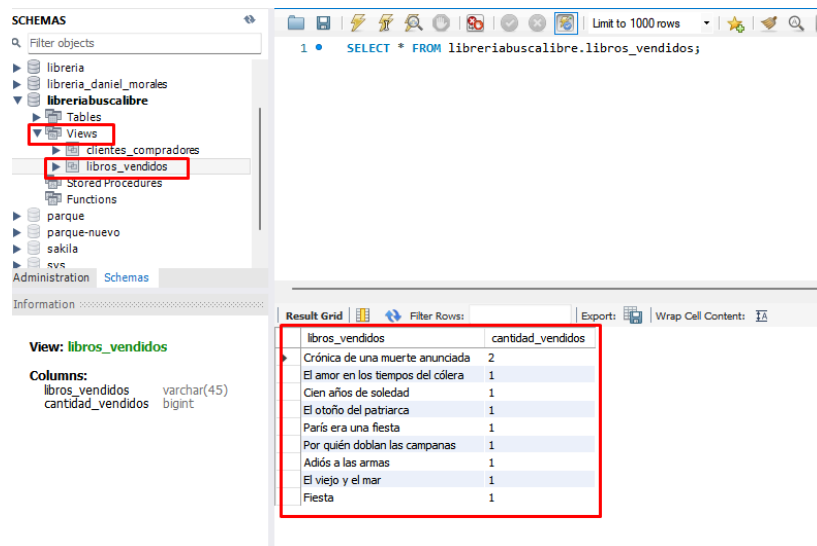
- **Punto 3 de la primera actividad (Se adjunta archivo vistas_libreria.sql):**

Realice las dos vistas que considere sean las más importantes y explique el motivo de su selección.

5. La primera vista que considero más importante es una vista llamada "libros_vendidos" porque proporciona a la librería información valiosa y detallada sobre sus ventas de libros, lo que le permite tomar decisiones más informadas, optimizar su inventario y mejorar la experiencia del cliente. Esto puede contribuir significativamente al éxito y la rentabilidad de la librería.

- Esta vista "libros_vendidos" mostrará el título de los libros vendidos junto con la cantidad de veces que cada libro ha sido vendido, ordenados de mayor a menor según la cantidad de libros vendidos

```
CREATE VIEW libros_vendidos AS
select libro.titulo as 'libros_vendidos', count(*) as 'cantidad_vendidos' from libro
join libro_cliente on libro.ISBN = libro_cliente.ISBN_libro_cliente
group by(libro_cliente.ISBN_libro_cliente)
order by(cantidad_vendidos) desc;
```



- La segunda vista que proporciona información importante se llamará “clientes_compradores”, se crea porque tener información de sus clientes y la cantidad de libros que han comprado, puede ayudar a la librería a mejorar sus estrategias de marketing, aumentar las ventas y fidelizar a los clientes existentes. Esto puede contribuir significativamente al éxito y la rentabilidad del negocio.

Esta vista "clientes_compradores" proporciona una visión agrupada de los clientes que han comprado libros, incluyendo sus nombres, números de teléfono y la cantidad de libros que han comprado. Esto puede ser útil para realizar análisis sobre los clientes y para tomar decisiones comerciales informadas, por ejemplo enviar promociones.

```
CREATE VIEW clientes_compradores as
select cliente.nombre, group_concat(telefono_cliente.numero separator ', ') as 'telefonos', count(libro_cliente.ISBN_libro_cliente) as 'cantidad_libros_comprados'
from cliente
join libro_cliente on cliente.cedula = libro_cliente.id_cliente
join telefono_cliente on cliente.cedula = telefono_cliente.cedula_cliente
group by cliente.nombre
order by cantidad_libros_comprados desc;
```

View: **clientes_compradores**

Columns:

- nombre: varchar(45)
- telefonos: text
- cantidad_libros_comprados: bigint

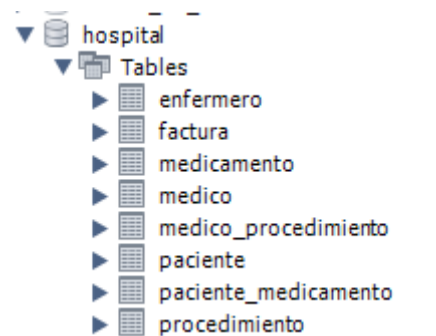
nombre	phone_numbers	cantidad_libros_comprados
Gissela Moreno	3001122334, 3101234567, 3001122334, 3101...	4
Pablo Emilio Castillo	3023344556, 3143456789, 3023344556, 3143...	4
Paula Andrea Cordoba Sanchez	3012233445, 3122345678, 3012233445, 3122...	4
José Lopez	3034455667, 3154567890	2
Leidy Laura Martinez Restrepo	3056677889, 3186789012	2
Sara Caicedo	3045566778, 3205678901	2

- **Punto 1 de la segunda actividad (Se adjunta archivo *insertar_datos_libreria.sql*):**

Utilizando el ejercicio del hospital realice lo siguiente:

Convierta el MR en una base de datos en MySQL utilizando sentencias SQL o el diagrama EER.

6. Se convierte el modelo relacional en una base de datos en MySQL, usando sentencias SQL.



- **Punto 2 de la segunda actividad:**

Complete la información para las tablas realizadas con al menos 5 registros por tabla.

7. Se llenan las tablas con 5 registros cada una. Se adjunta archivo .sql con los comandos empleados para poblarlas.

```

1  INSERT INTO hospital.medico (id_medico, nombre, telefono, especialidad)
2  VALUES
3  ('MED001', 'Dr. Carlos Gómez', '3151234567', 'Cirugía general'),
4  ('MED002', 'Dra. Laura Martínez', '3109876543', 'Pediatría'),
5  ('MED003', 'Dr. Juan Rodríguez', '3008765432', 'Cardiología'),
6  ('MED004', 'Dra. María González', '3187654321', 'Ginecología'),
7  ('MED005', 'Dr. Andrés Pérez', '3206543210', 'Oftalmología');
8
9  • INSERT INTO hospital.enfermero (id_enfermero, id_medico, nombre, telefono)
10 VALUES
11 ('ENF001', 'MED001', 'María Rodríguez', '3151234567'),
12 ('ENF002', 'MED002', 'Juan Pérez', '3109876543'),
13 ('ENF003', 'MED001', 'Luisa Gómez', '3008765432'),
14 ('ENF004', 'MED003', 'Pedro Sánchez', '3187654321'),
15 ('ENF005', 'MED002', 'Ana Martínez', '3206543210');
16
17 • INSERT INTO hospital.medico_procedimiento (idmedico_procedimiento, medico_id, procedimiento_id)
18 VALUES
19 ('P01', 'MED001', 'PROC001'),
20 ('P02', 'MED002', 'PROC002'),
21 ('P03', 'MED003', 'PROC003'),
22 ('P04', 'MED004', 'PROC004'),
23 ('P05', 'MED005', 'PROC005');
24
25 • INSERT INTO hospital.procedimiento (id_procedimiento, tipo_procedimiento)
26 VALUES
27 ('PROC001', 'Cirugía cardíaca'),
28 ('PROC002', 'Extracción de apéndice'),

```

- **Punto 3 de la segunda actividad (Se adjunta archivo *consultas_hospital.sql* para poder replicar la consulta):**

“realice una consulta que me permita conocer qué medicamentos ha tomado cada paciente y la dosis suministrada.”

8. Se emplean comandos SQL para realizar la consulta requerida. El comando devuelve una lista que muestra qué medicamentos ha tomado cada paciente junto con la dosis suministrada de cada medicamento.

```

1  -- que medicamentos ha tomado cada paciente y la dosis suministrada
2  • select paciente.id_paciente, paciente.nombre, paciente_medimento.nombre_medimento, medicamento.dosis from paciente
3  join paciente_medimento on paciente.id_paciente = paciente_medimento.id_paciente
4  join medicamento on paciente_medimento.nombre_medimento = medicamento.nombre
5  ;

```

id_paciente	nombre	nombre_medimento	dosis
100112233445	Laura Gutiérrez	Paracetamol	500 mg
101234567890	Juan Ramírez	Ibuprofeno	400 mg
102345678901	María Pérez	Omeprazol	20 mg
103456789012	Carlos Rodríguez	Amoxicilina	500 mg
104567890123	Luisa Martínez	Loratadina	10 mg

“realice una consulta que me permita conocer qué enfermeros estuvieron en los procedimientos de los pacientes.”

9. Se emplean comandos SQL para realizar la consulta requerida. Estas instrucciones devuelve una lista que muestra qué enfermeros estuvieron

involucrados en los procedimientos realizados en los pacientes, junto con el tipo de procedimiento realizado.

```
7 -- que enfermeros estuvieron en los procedimientos de los pacientes
8 • select enfermero.nombre, procedimiento.tipo_procedimiento from enfermero
9 join medico on enfermero.id_medico = medico.id_medico
10 join medico_procedimiento on medico.id_medico = medico_procedimiento.medico_id
11 join procedimiento on medico_procedimiento.procedimiento_id = id_procedimiento
12 ;
13
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
nombre	tipo_procedimiento			
▶ María Rodríguez	Cirugía cardíaca			
Juan Pérez	Extracción de apéndice			
Luisa Gómez	Cirugía cardíaca			
Pedro Sánchez	Colonoscopia			
Ana Martínez	Extracción de apéndice			

- **Punto 5 de la segunda actividad (Se adjunta archivo *vistas_hospital.sql*):**

“Realice las tres vistas que considere sean las más importantes y explique el motivo de su selección.”

10. Tres vistas importantes:

- **Primer vista:** Se crea la vista "medicamentos_pacientes", se considera importante porque proporciona una visión completa y detallada de los medicamentos administrados a cada paciente, lo que es crucial para garantizar la seguridad del paciente, mantener registros precisos y coordinar eficazmente la atención médica.

SCHEMAS

Filter objects

- Columns
- Indexes
- Foreign Keys
- Triggers
- Views**
 - costos_pacientes
 - medicamentos_pacientes**
 - medico_paciente
- Stored Procedures
- Functions
- jugos
- libreria
- libreria daniel morales

Administration Schemas

Information

View: medicamentos_pacientes

Columns:

id_paciente	varchar(30)
nombre	varchar(45)
nombre_medamento	varchar(45)
dosis	varchar(45)

1 • `SELECT * FROM hospital.medicamentos_pacientes;`

Result Grid

id_paciente	nombre	nombre_medamento	dosis
100112233445	Laura Gutiérrez	Paracetamol	500 mg
101234567890	Juan Ramírez	Ibuprofeno	400 mg
102345678901	María Pérez	Omeprazol	20 mg
103456789012	Carlos Rodríguez	Amoxicilina	500 mg
104567890123	Luisa Martínez	Loratadina	10 mg

- **Segunda vista:** Se crea la vista "costos_pacientes" porque proporciona información esencial sobre los costos asociados con los procedimientos médicos realizados para cada paciente, lo que es fundamental para la transparencia financiera, la gestión de cuentas de pacientes y la planificación financiera del hospital.

Filter objects

- Columns
- Indexes
- Foreign Keys
- Triggers
- Views**
 - costos_pacientes**
 - medicamentos_pacientes
 - medico_paciente
- Stored Procedures
- Functions
- jugos
- libreria
- libreria daniel morales

Administration Schemas

Information

View: medicamentos_pacientes

Columns:

id_paciente	varchar(30)
nombre	varchar(45)
nombre_medamento	varchar(45)
dosis	varchar(45)

1 • `SELECT * FROM hospital.costos_pacientes;`

Result Grid

nombre	valor_total	tipo_procedimiento	fecha
Laura Gutiérrez	150000	Cirugía cardíaca	2024-04-01
Juan Ramírez	200000	Extracción de apéndice	2024-04-02
María Pérez	175000	Colonoscopia	2024-04-03
Carlos Rodríguez	220000	Parto por cesárea	2024-04-04
Luisa Martínez	180000	Cirugía de cataratas	2024-04-05

- **Tercer vista:** Se crea la vista "medico_paciente" porque proporciona información esencial sobre qué médico está a cargo de la atención de cada paciente, lo que es fundamental para garantizar una coordinación efectiva de

la atención médica, un seguimiento preciso del historial médico y una asignación adecuada de responsabilidades dentro del hospital.

The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane displays a tree view of database objects. The 'Views' folder is expanded, and 'medico_paciente' is selected. Below this, the 'View: medicamentos_pacientes' section shows the columns: id_paciente (varchar(30)), nombre (varchar(45)), nombre_medimento (varchar(45)), and dosis (varchar(45)).

On the right, the SQL editor shows the query: `SELECT * FROM hospital.medico_paciente;`. Below the editor, the 'Result Grid' displays the data returned by the query. The data is as follows:

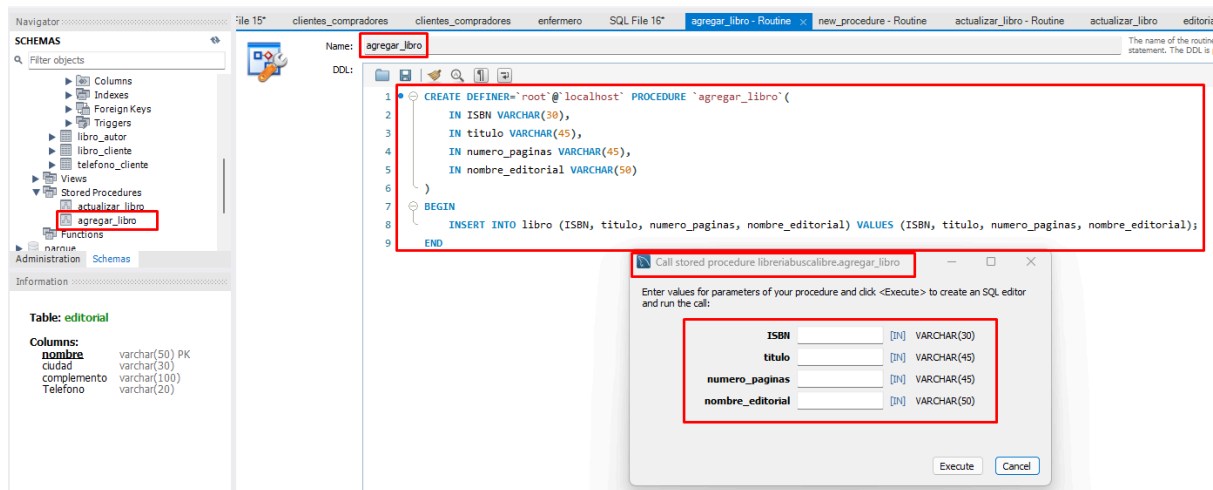
nombre	nombre_paciente
Dr. Carlos Gómez	Laura Gutiérrez
Dra. Laura Martínez	Juan Ramírez
Dr. Juan Rodríguez	María Pérez
Dra. María González	Carlos Rodríguez
Dr. Andrés Pérez	Luisa Martínez

- **Punto 1 de la tercera actividad:**

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

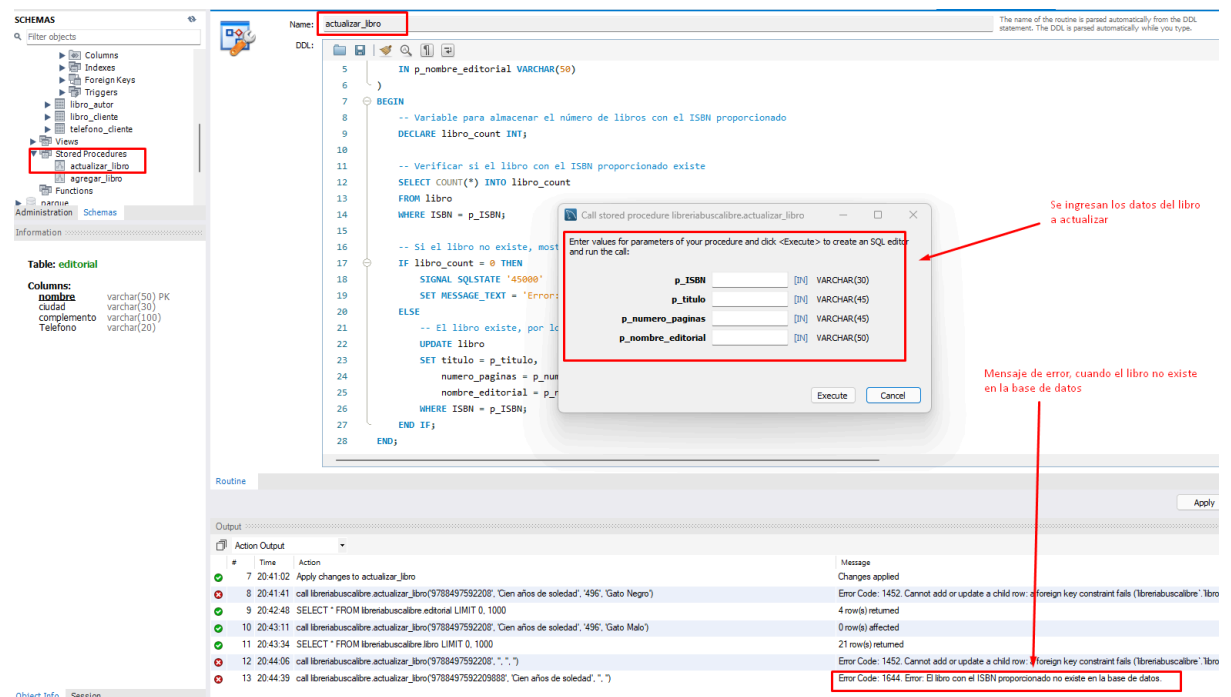
11. Procedimiento Agregar Libro (Se adjunta archivo procedimiento_agregar_libro.sql):

Este procedimiento, lo nombré “agregar_libro”, deberá tomar los detalles de un libro (ISBN, título, número de páginas, nombre de la editorial) como entrada y los inserta como un nuevo registro en la tabla libro.



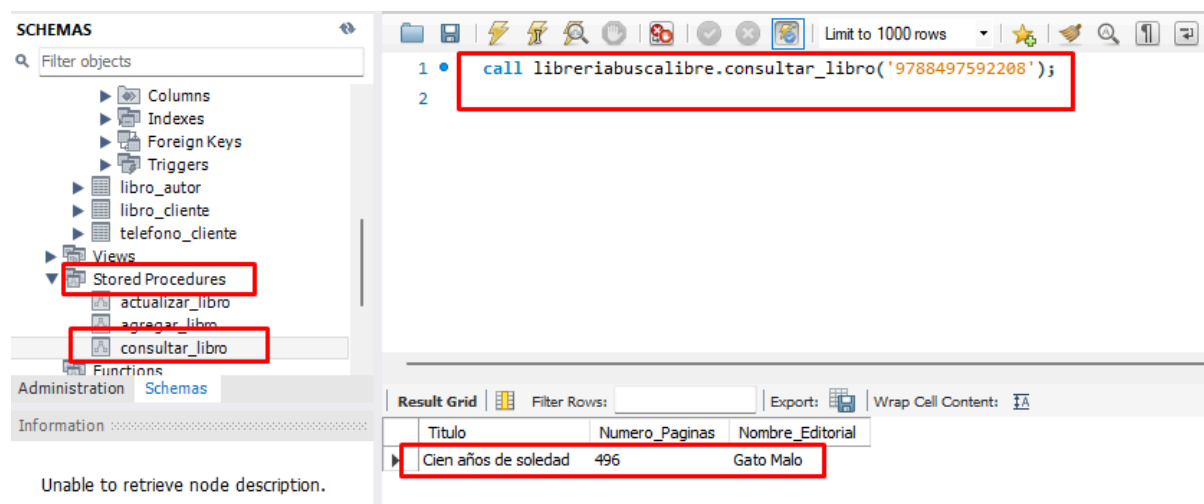
12. Procedimiento Actualizar Libro:

Este procedimiento, lo nombré “actualizar_libro”, deberá verificar primero si el libro existe en la base de datos. Si existe, debe actualizar la información con los nuevos valores proporcionados. Si no existe, enviará un mensaje de error indicando que el libro no se encuentra en la base de datos.



13. Procedimiento Consultar Libro:

Este procedimiento, lo nombré “consultar_libro”, deberá buscar un libro por su ISBN en la tabla libro y devolver su información si lo encuentra, o un mensaje indicando que el libro no existe en la base de datos.



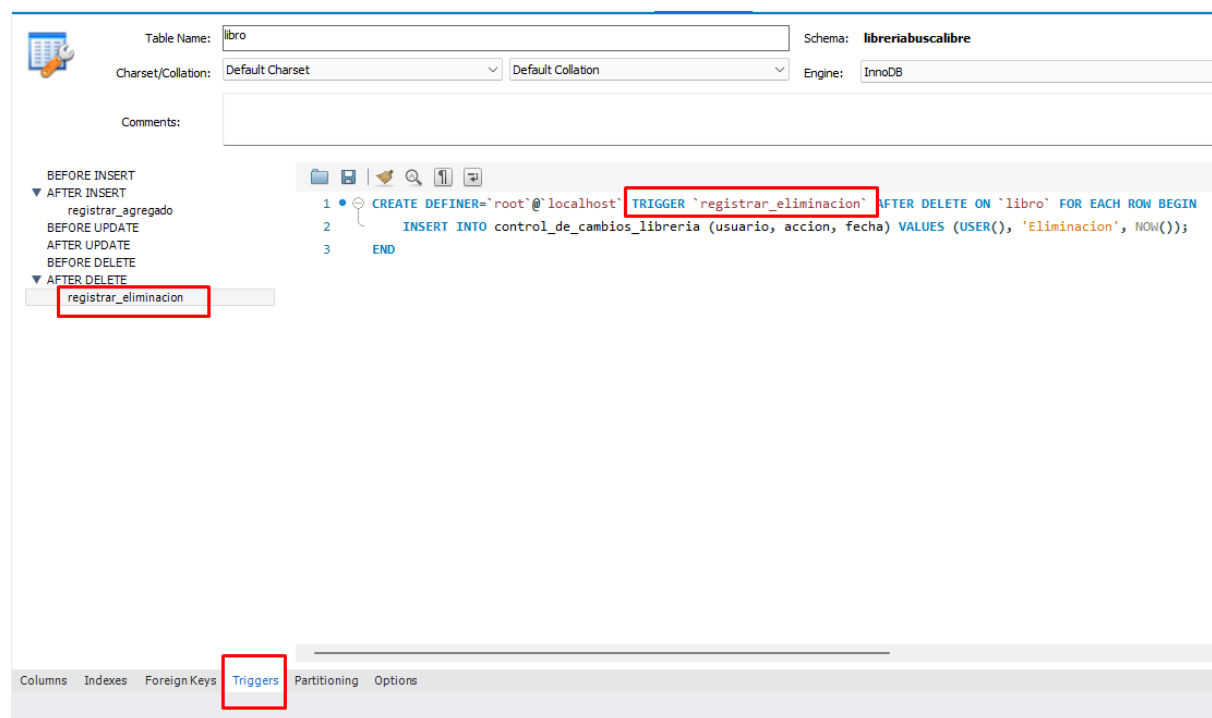
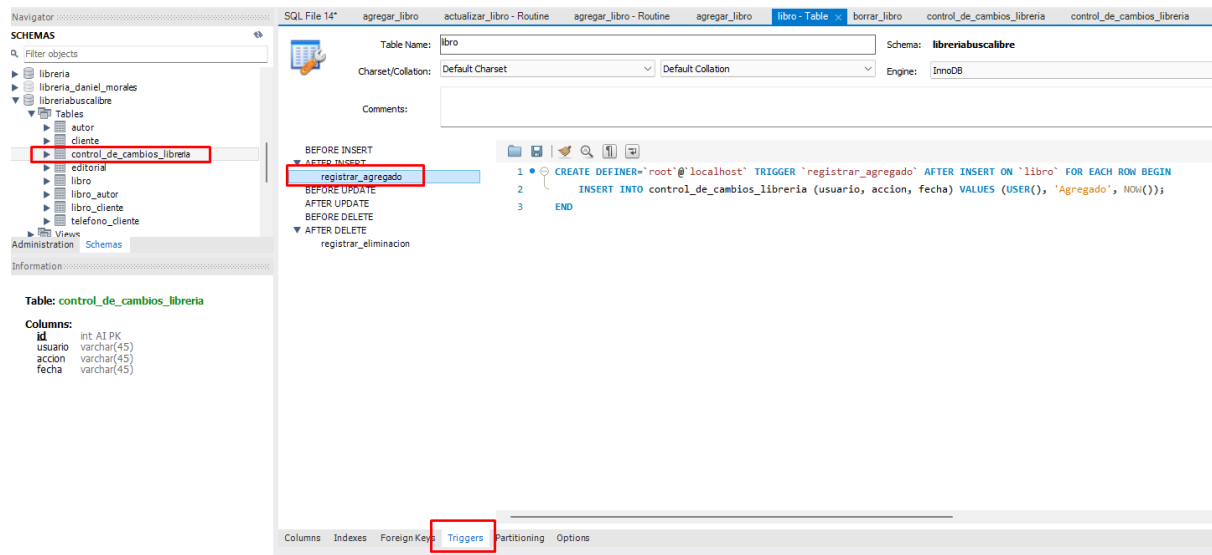
The screenshot displays a database management interface. On the left, the 'SCHEMAS' pane shows a tree view of database objects. Under the 'Views' folder, 'Stored Procedures' is expanded, and 'consultar_libro' is highlighted with a red box. The main editor shows a SQL query: `call libreriaibusalibre.consultar_libro('9788497592208');`, which is also highlighted with a red box. Below the editor, the 'Result Grid' shows the output of the query. The results are as follows:

Titulo	Numero_Paginas	Nombre_Editorial
Cien años de soledad	496	Gato Malo

14. Procedimiento Borrar Libro:

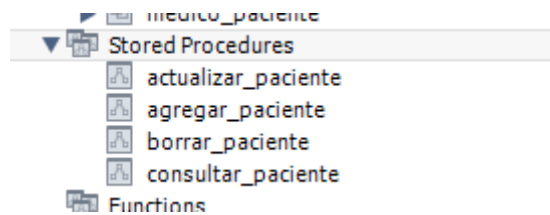
Este procedimiento, lo nombré “borrar_libro”, deberá verificar si un libro con el ISBN proporcionado existe en la base de datos y, en caso afirmativo, lo deberá borrar de la tabla libro. Si el libro no existe, no debe realizar ninguna acción.

Luego creo los triggers, con los nombres registrar_agregado y registrar_eliminacion, se usan para realizar acciones automáticas (en este caso, registrar inserciones y eliminaciones) en la tabla de registro (control_de_cambios_libreria) cada vez que se inserta o elimina un registro en la tabla libro.



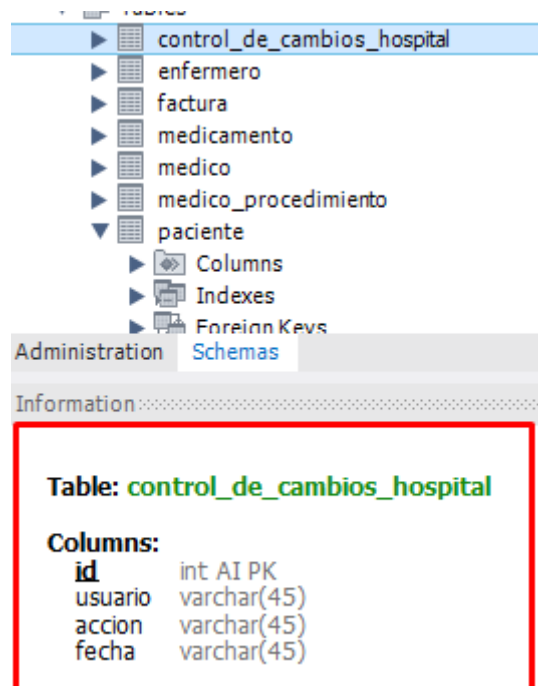
- *Elabore 4 procedimientos almacenados que me permitan agregar, actualizar, consultar y borrar, en una de las tablas del Hospital (segunda actividad).*

(Se adjunta archivo procedimientos_hospital.sql)



- *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.*

Se crea la tabla `control_de_cambios_hospital`, con los datos solicitados.



Se crea el trigger `borrar_paciente`

Name: borrar_paciente

DDL:

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `borrar_paciente`(  
2     IN p_id_paciente VARCHAR(30)  
3 )  
4 BEGIN  
5     DELETE FROM hospital.paciente WHERE id_paciente = p_id_paciente;  
6 END
```

Se crea el trigger registrar_agregado

Table Name: paciente Schema: hospital
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

BEFORE INSERT
▼ AFTER INSERT
 registrar_agregado
BEFORE UPDATE
AFTER UPDATE
BEFORE DELETE
▼ AFTER DELETE
 registrar_eliminacion

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `registrar_agregado` AFTER INSERT ON `paciente` FOR EACH ROW BEGIN  
2     INSERT INTO control_de_cambios_hospital (usuario, accion, fecha) VALUES (USER(), 'Agregado', NOW());  
3 END
```