



PRACTICA

IMPLEMENTACIÓN DE

BITÁCORAS

Ing. Maximiliano Carsi Castrejón

DESCRIPCIÓN BREVE

Este documento trata sobre la implementación de logs a base de creación de Triggers para el registro de una bitácora de una base de datos.

Luis Eduardo Bahena Castillo

8°C IDyGS

INTRODUCCIÓN

Objetivo: Desarrollar habilidades en el diseño e implementación de sistemas de bitácoras para el seguimiento de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en una base de datos. Los estudiantes aprenderán a capturar y almacenar información relevante sobre las actividades realizadas en la base de datos, mejorando así la trazabilidad y seguridad de los datos.

Requisitos Previos:

- Conocimientos básicos de SQL y manejo de bases de datos.
- Acceso a un sistema de gestión de bases de datos como MySQL, PostgreSQL, o similar.

Descripción de la Base de Datos: Para esta práctica, propongo una base de datos simple de una librería, que incluye tablas para **Libros**, **Autores**, y **OperacionesLog**.

Script de Creación de Base de Datos y Tablas

```
CREATE DATABASE IF NOT EXISTS LibreriaDB;
USE LibreriaDB;
-- Creación de tabla Autores
CREATE TABLE IF NOT EXISTS Autores (
  id_autor INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255) NOT NULL,
  nacionalidad VARCHAR(100)
);
-- Creación de tabla Libros
CREATE TABLE IF NOT EXISTS Libros (
  id_libro INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  id_autor INT,
  año_publicacion INT,
  FOREIGN KEY (id_autor) REFERENCES Autores(id_autor)
);
-- Creación de tabla OperacionesLog
CREATE TABLE IF NOT EXISTS OperacionesLog (
  id_log INT AUTO_INCREMENT PRIMARY KEY,
  tipo_operacion VARCHAR(50) NOT NULL,
  tabla_afectada VARCHAR(100) NOT NULL,
  detalle_operacion TEXT NOT NULL,
  fecha_operacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Implementación de Bitácoras para Operaciones CRUD

1. Crear Triggers para Capturar Operaciones CRUD:

- Debes crear triggers para cada operación CRUD en las tablas **Libros** y **Autores**. Estos triggers insertarán un registro en la tabla **OperacionesLog** cada vez que se realice una operación CRUD (Alta, Baja, Modificación).
- Ejemplo de trigger para operaciones de inserción en la tabla **Libros**.

```
DELIMITER //  
CREATE TRIGGER AfterInsertLibros AFTER INSERT ON Libros  
FOR EACH ROW  
BEGIN  
    INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada,  
    detalle_operacion)  
    VALUES ('INSERT', 'Libros', CONCAT('Se insertó el libro: ',  
    NEW.titulo, ' del año: ', NEW.año_publicacion));  
END; //  
DELIMITER ;
```

2. Pruebas de los Logs:

- Realiza varias operaciones CRUD en las tablas **Libros** y **Autores**.
- Consulta la tabla **OperacionesLog** para verificar que las operaciones se están registrando correctamente.

Reporte de la práctica

- **Introducción:** Breve descripción del objetivo y la estructura de la base de datos.
- **Desarrollo:** Detalle de la implementación de los triggers y cualquier otra lógica utilizada para la captura de las operaciones CRUD.
- **Pruebas Realizadas:** Descripción de las pruebas CRUD realizadas y capturas de pantalla de la tabla OperacionesLog mostrando los registros de las operaciones.
- **Conclusiones:** Reflexiones sobre la importancia de implementar sistemas de bitácoras en bases de datos y cómo pueden ayudar en la trazabilidad y seguridad de los datos.

Entregables

- **Script SQL:** Incluyendo la creación de la base de datos, tablas, y triggers para la captura de operaciones CRUD.
- **Reporte de Práctica:** Documento que incluya todos los elementos descritos en la sección de reporte de práctica.

DESARROLLO

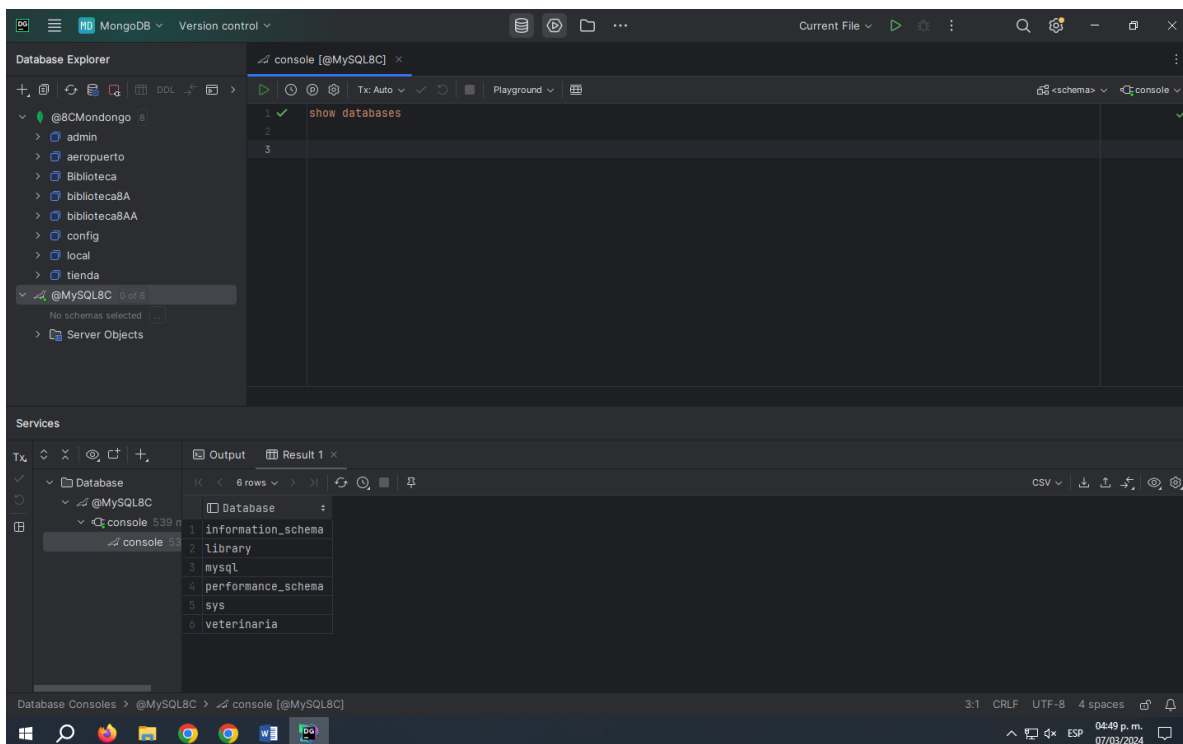
¿Qué es un log en base de datos? Un log en bitácoras de base de datos es un registro detallado y cronológico de todas las operaciones realizadas en la base de datos, incluyendo inserciones, actualizaciones y eliminaciones de datos.

¿Qué es un trigger en base de datos? Un trigger en bitácoras de base de datos es un procedimiento automatizado que se activa ante eventos específicos, como inserciones, actualizaciones o eliminaciones, registrando operaciones en la bitácora.

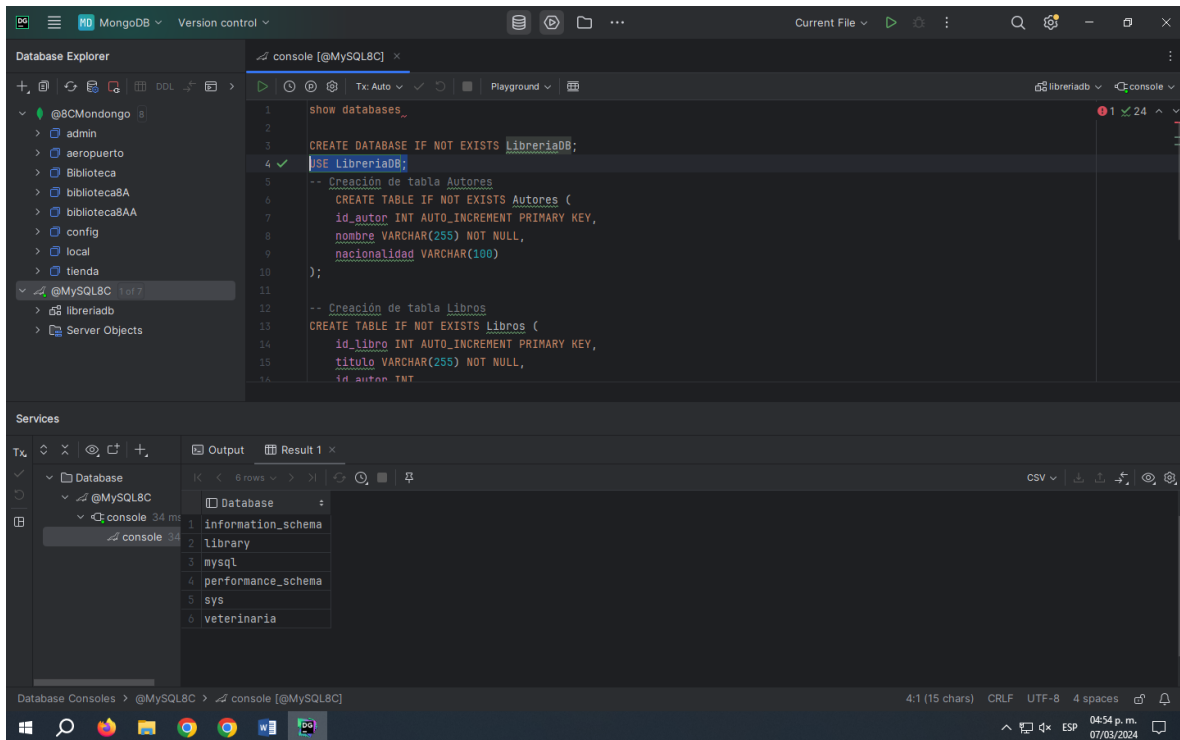
Antes de hacer la práctica ... ¡iiii IMPORTANTE !!!!!

Ya no será necesario detallar cada paso básico, puesto que hasta este momento ya se sabe todo lo que se va a realizar (Ejemplo como abrir aplicaciones, descargar o crear repositorios, descargar documentos, crear variables, conectar bases de datos entre otros).

Paso 1: Abrir la aplicación DataGrip para acceder a las bases de datos en MySQL (mismo puerto 3306 y mismo usuario-contraseña) y con comando **show databases** mostrar las bases de datos disponibles.



Paso 2: Ejecutar los comandos 3 y 4 de la siguiente imagen para crear la base de datos y utilizarla.



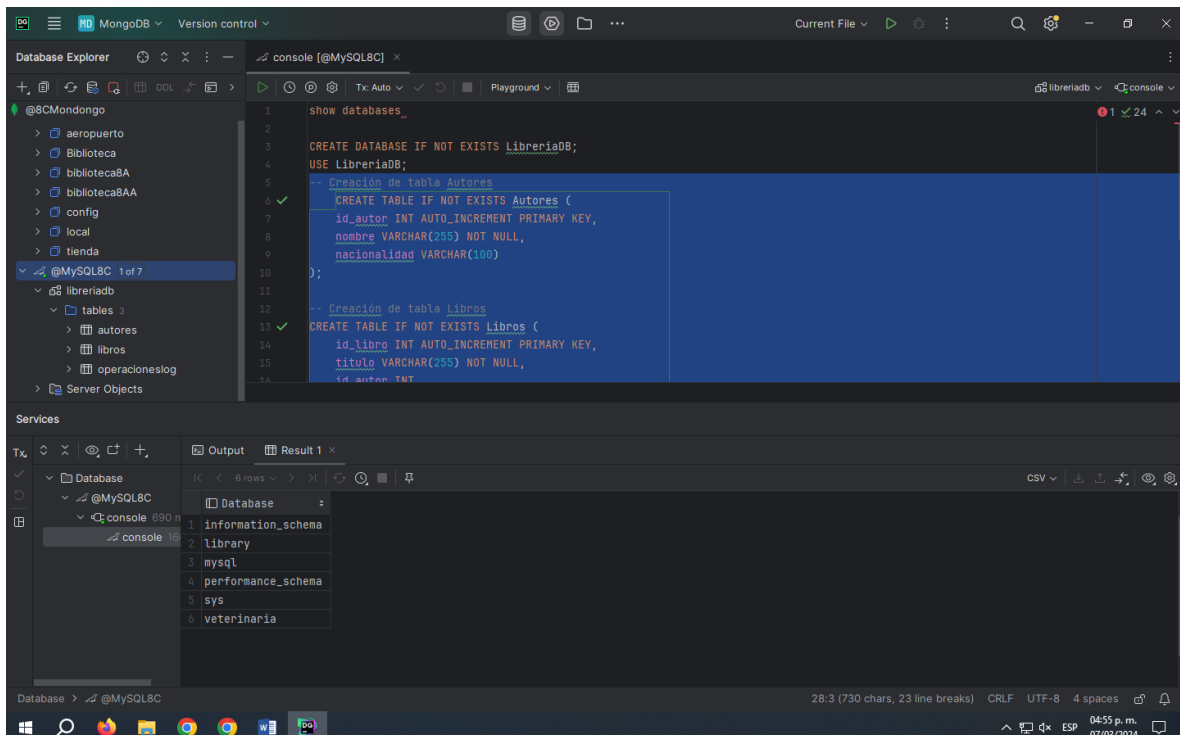
The screenshot shows the SQL Studio interface with the following SQL commands in the console:

```

1 show databases;
2
3 CREATE DATABASE IF NOT EXISTS LibreriaDB;
4 USE LibreriaDB;
5
6 -- Creación de tabla Autores
7 CREATE TABLE IF NOT EXISTS Autores (
8   id_autor INT AUTO_INCREMENT PRIMARY KEY,
9   nombre VARCHAR(255) NOT NULL,
10  nacionalidad VARCHAR(100)
11 );
12
13 -- Creación de tabla Libros
14 CREATE TABLE IF NOT EXISTS Libros (
15   id_libro INT AUTO_INCREMENT PRIMARY KEY,
16   titulo VARCHAR(255) NOT NULL,
17   id_autor INT
  
```

The Database Explorer on the left shows the connection to @MySQL8C and the database LibreriaDB. The Services panel at the bottom shows the output of the commands, listing the databases and the tables created.

Paso 3: Ejecutar el script de la tabla para crear las tablas **autores**, **libros** y **operacioneslog**, al final corroborar si se crearon correctamente en la conexión.



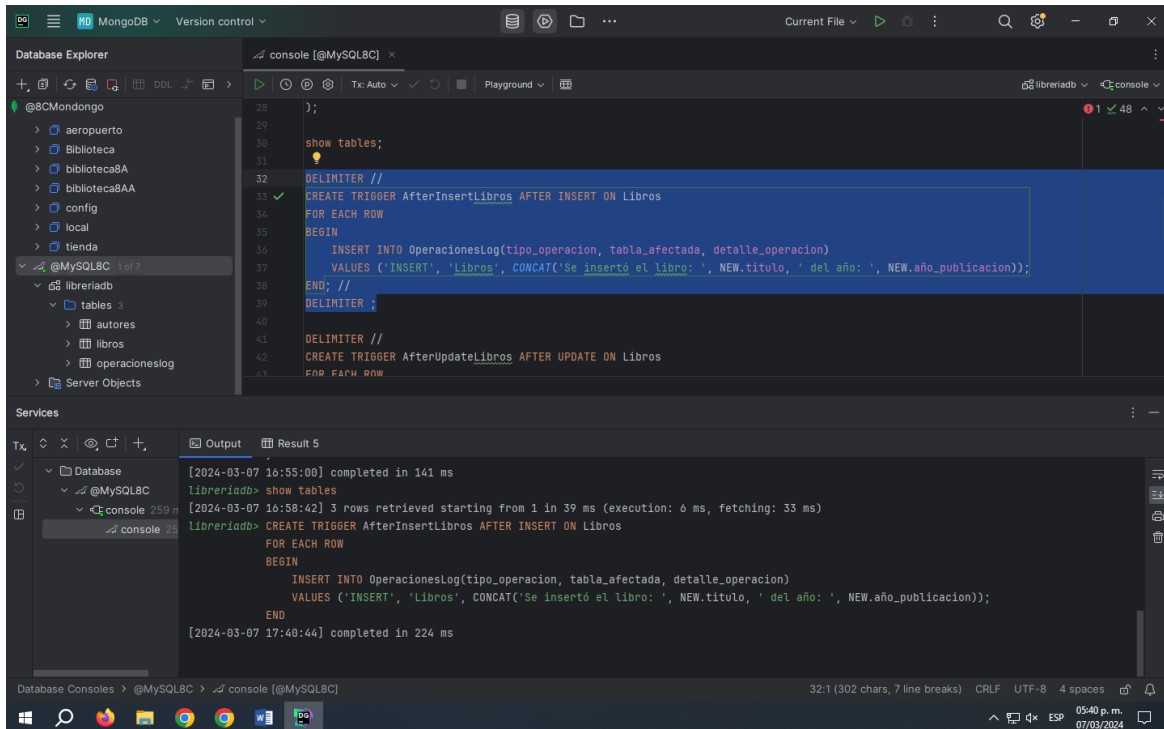
The screenshot shows the SQL Studio interface with the following SQL commands in the console:

```

1 show databases;
2
3 CREATE DATABASE IF NOT EXISTS LibreriaDB;
4 USE LibreriaDB;
5
6 -- Creación de tabla Autores
7 CREATE TABLE IF NOT EXISTS Autores (
8   id_autor INT AUTO_INCREMENT PRIMARY KEY,
9   nombre VARCHAR(255) NOT NULL,
10  nacionalidad VARCHAR(100)
11 );
12
13 -- Creación de tabla Libros
14 CREATE TABLE IF NOT EXISTS Libros (
15   id_libro INT AUTO_INCREMENT PRIMARY KEY,
16   titulo VARCHAR(255) NOT NULL,
17   id_autor INT
  
```

The Database Explorer on the left shows the connection to @MySQL8C and the database LibreriaDB. The Services panel at the bottom shows the output of the commands, listing the databases and the tables created.

Paso 4: De acuerdo al siguiente código **TIGGER** crear para saber la información después de haber ingresado un libro (se mostrará en la consola que se creó correctamente).



The screenshot shows a MySQL console window with the following SQL code being executed:

```

DELIMITER //
CREATE TRIGGER AfterInsertLibros AFTER INSERT ON Libros
FOR EACH ROW
BEGIN
  INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
  VALUES ('INSERT', 'Libros', CONCAT('Se insertó el libro: ', NEW.titulo, ' del año: ', NEW.año_publicacion));
END; //
DELIMITER ;

```

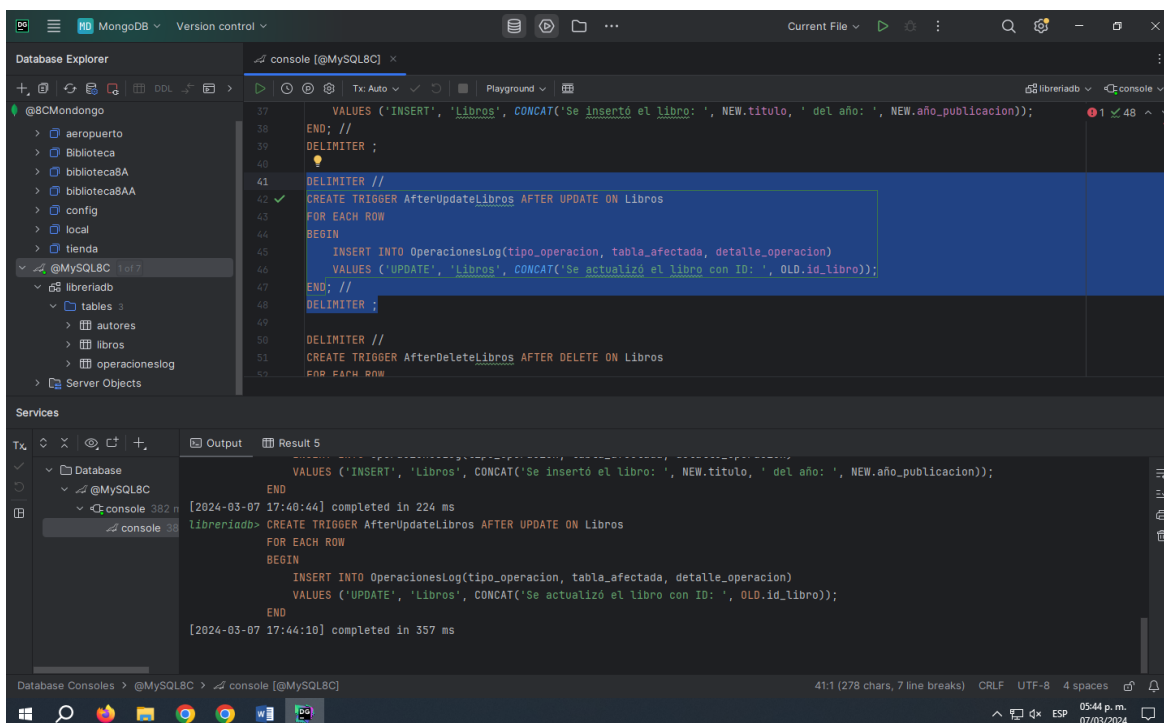
The console output shows the successful execution of the trigger creation:

```

[2024-03-07 16:55:00] completed in 141 ms
libreriadb> show tables
[2024-03-07 16:58:42] 3 rows retrieved starting from 1 in 39 ms (execution: 6 ms, fetching: 33 ms)
libreriadb> CREATE TRIGGER AfterInsertLibros AFTER INSERT ON Libros
FOR EACH ROW
BEGIN
  INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
  VALUES ('INSERT', 'Libros', CONCAT('Se insertó el libro: ', NEW.titulo, ' del año: ', NEW.año_publicacion));
END
[2024-03-07 17:40:44] completed in 224 ms

```

Paso 5: Ahora con el mismo código **TIGGER** crear para saber la información después de haber actualizado información de un libro (se mostrará en la consola que se creó correctamente).



The screenshot shows a MySQL console window with the following SQL code being executed:

```

DELIMITER //
CREATE TRIGGER AfterUpdateLibros AFTER UPDATE ON Libros
FOR EACH ROW
BEGIN
  INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
  VALUES ('UPDATE', 'Libros', CONCAT('Se actualizó el libro con ID: ', OLD.id_libro));
END; //
DELIMITER ;

```

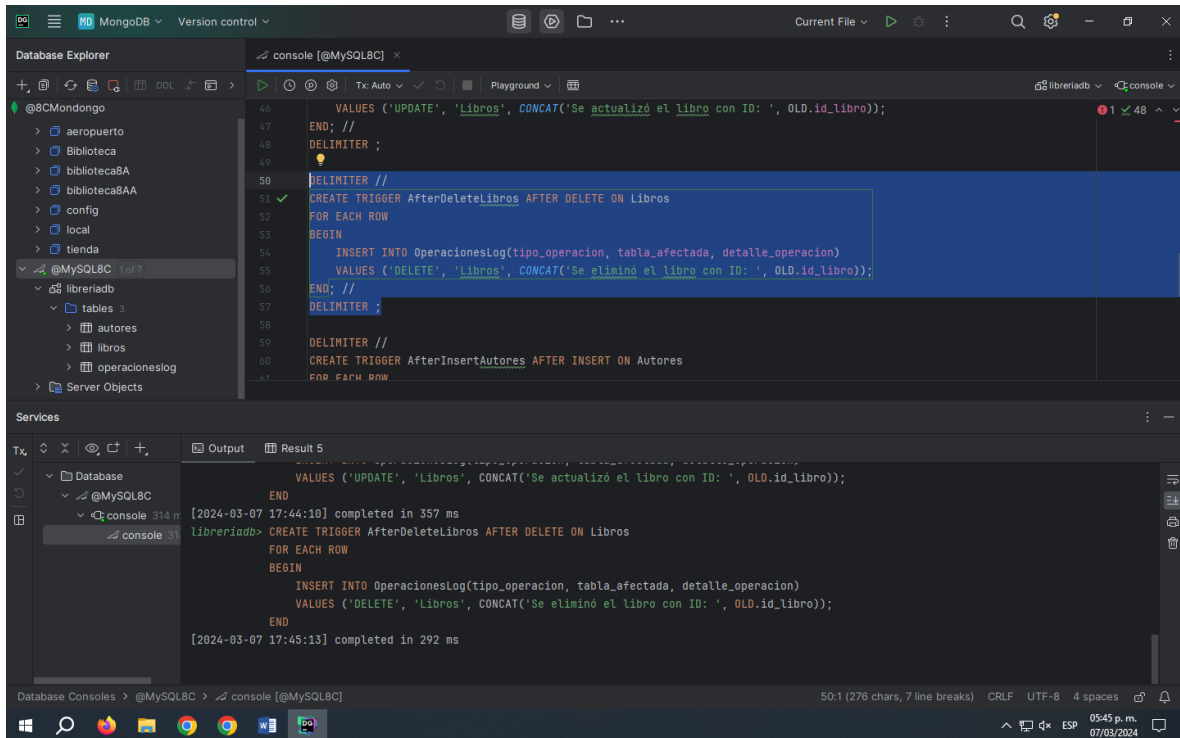
The console output shows the successful execution of the trigger creation:

```

[2024-03-07 17:40:44] completed in 224 ms
libreriadb> CREATE TRIGGER AfterUpdateLibros AFTER UPDATE ON Libros
FOR EACH ROW
BEGIN
  INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
  VALUES ('UPDATE', 'Libros', CONCAT('Se actualizó el libro con ID: ', OLD.id_libro));
END
[2024-03-07 17:44:10] completed in 357 ms

```

Paso 6: Siguiendo con el mismo código **TIGGER** crear para saber la información después de haber eliminado información de un libro (se mostrará en la consola que se creó correctamente).



The screenshot shows the SQL Studio interface with the following SQL code in the editor:

```

46 VALUES ('UPDATE', 'Libros', CONCAT('Se actualizó el libro con ID: ', OLD.id_libro));
47 END; //
48 DELIMITER ;
49
50 DELIMITER //
51 CREATE TRIGGER AfterDeleteLibros AFTER DELETE ON Libros
52 FOR EACH ROW
53 BEGIN
54     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
55     VALUES ('DELETE', 'Libros', CONCAT('Se eliminó el libro con ID: ', OLD.id_libro));
56 END; //
57 DELIMITER ;
58
59 DELIMITER //
60 CREATE TRIGGER AfterInsertAutores AFTER INSERT ON Autores
61 FOR EACH ROW

```

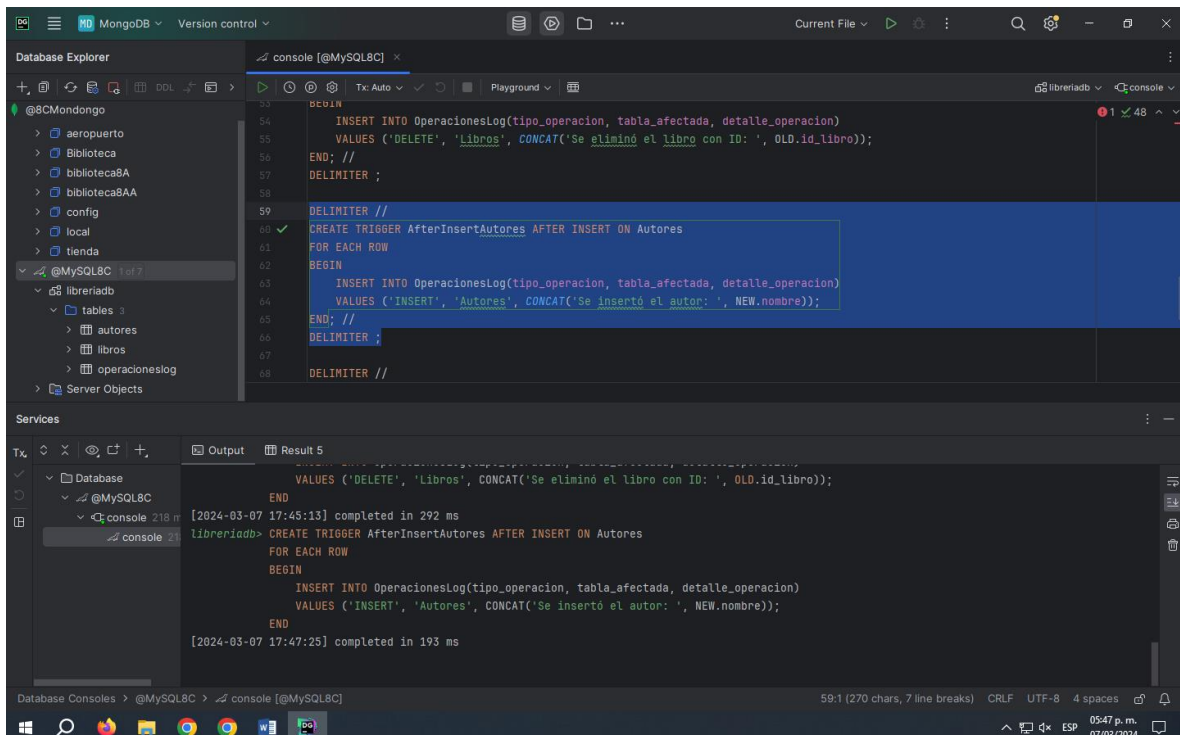
The Services panel shows the execution results:

```

Database
  MySQL8C
    console 314 m
      console 31
        [2024-03-07 17:44:10] completed in 357 ms
        libreriadb> CREATE TRIGGER AfterDeleteLibros AFTER DELETE ON Libros
        FOR EACH ROW
        BEGIN
        INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
        VALUES ('DELETE', 'Libros', CONCAT('Se eliminó el libro con ID: ', OLD.id_libro));
        END
        [2024-03-07 17:45:13] completed in 292 ms

```

Paso 7: Realizar los pasos anteriores 4, 5 y 6, pero ahora con la tabla de Autores



The screenshot shows the SQL Studio interface with the following SQL code in the editor:

```

53 BEGIN
54     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
55     VALUES ('DELETE', 'Libros', CONCAT('Se eliminó el libro con ID: ', OLD.id_libro));
56 END; //
57 DELIMITER ;
58
59 DELIMITER //
60 CREATE TRIGGER AfterInsertAutores AFTER INSERT ON Autores
61 FOR EACH ROW
62 BEGIN
63     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
64     VALUES ('INSERT', 'Autores', CONCAT('Se insertó el autor: ', NEW.nombre));
65 END; //
66 DELIMITER ;
67
68 DELIMITER //

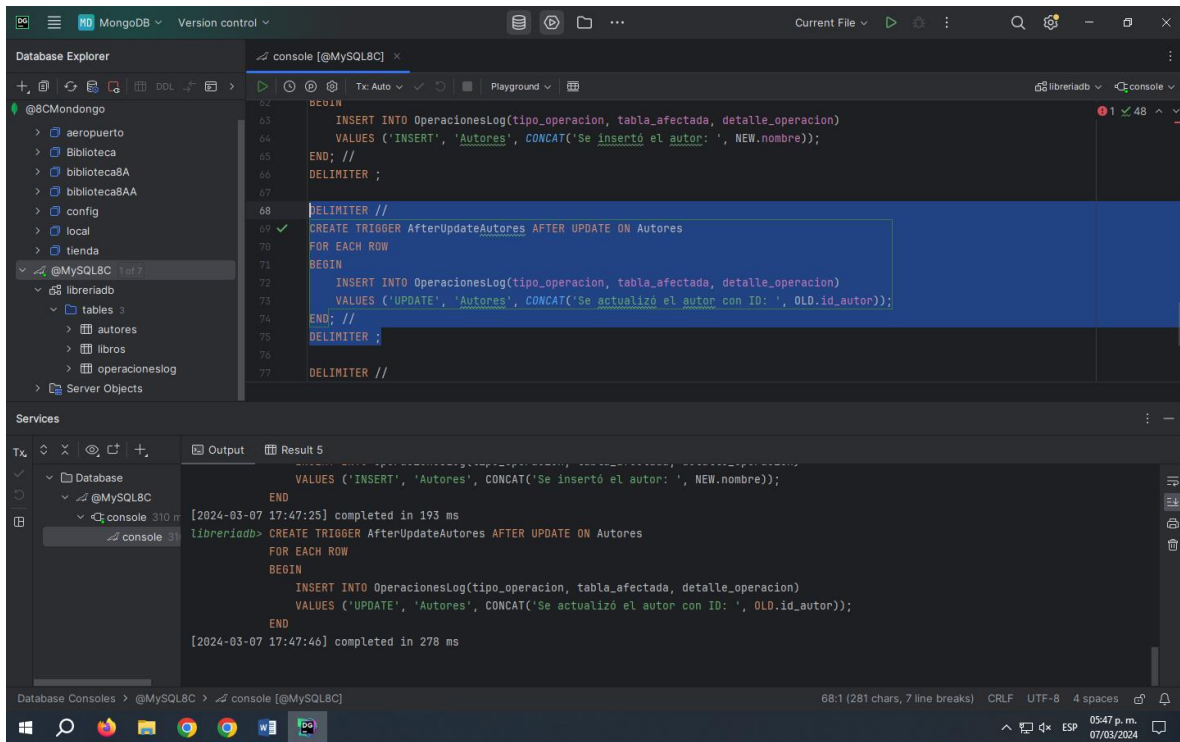
```

The Services panel shows the execution results:

```

Database
  MySQL8C
    console 218 m
      console 21
        [2024-03-07 17:45:13] completed in 292 ms
        libreriadb> CREATE TRIGGER AfterInsertAutores AFTER INSERT ON Autores
        FOR EACH ROW
        BEGIN
        INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
        VALUES ('INSERT', 'Autores', CONCAT('Se insertó el autor: ', NEW.nombre));
        END
        [2024-03-07 17:47:25] completed in 193 ms

```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Database Explorer' with the 'libreriaadb' database selected. The right pane shows the 'console [MySQL8C]' window with the following SQL scripts being executed:

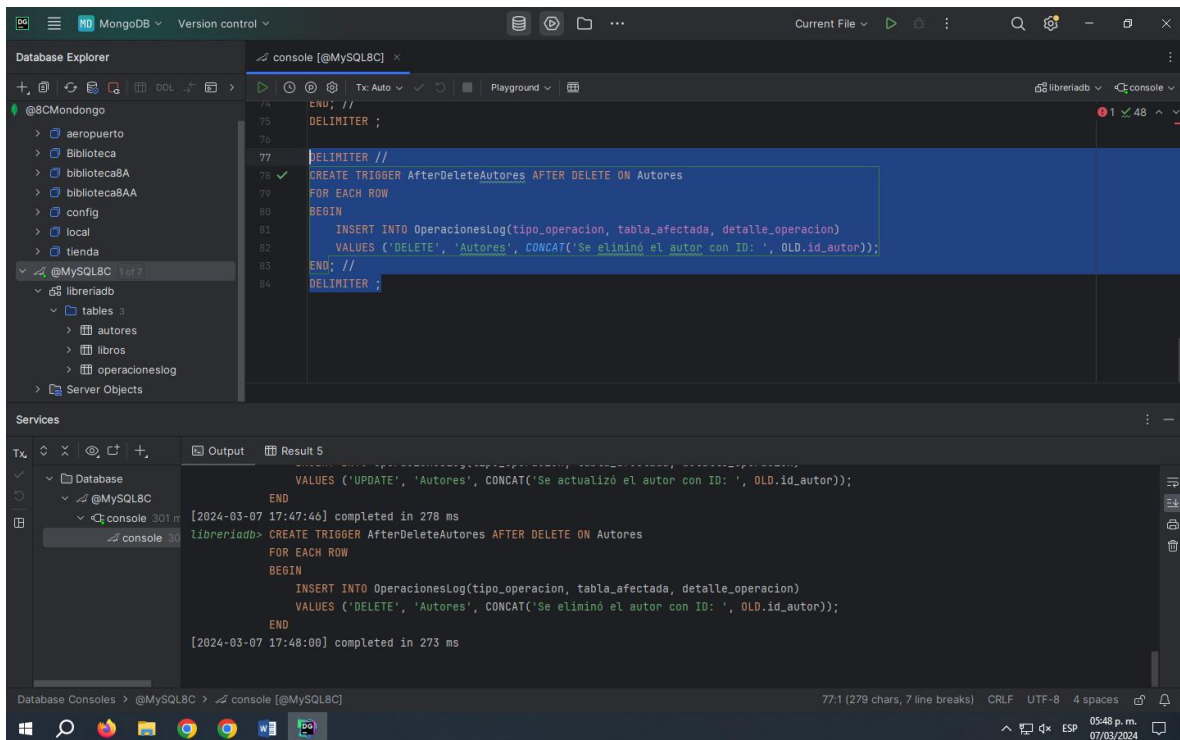
```

62 BEGIN
63     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
64     VALUES ('INSERT', 'Autores', CONCAT('Se insertó el autor: ', NEW.nombre));
65 END; //
66 DELIMITER ;
67
68 DELIMITER //
69 CREATE TRIGGER AfterUpdateAutores AFTER UPDATE ON Autores
70 FOR EACH ROW
71 BEGIN
72     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
73     VALUES ('UPDATE', 'Autores', CONCAT('Se actualizó el autor con ID: ', OLD.id_autor));
74 END; //
75 DELIMITER ;
76
77 DELIMITER //
  
```

The 'Output' window shows the execution results:

```

[2024-03-07 17:47:25] completed in 193 ms
libreriaadb> CREATE TRIGGER AfterUpdateAutores AFTER UPDATE ON Autores
FOR EACH ROW
BEGIN
INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
VALUES ('UPDATE', 'Autores', CONCAT('Se actualizó el autor con ID: ', OLD.id_autor));
END
[2024-03-07 17:47:46] completed in 278 ms
  
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Database Explorer' with the 'libreriaadb' database selected. The right pane shows the 'console [MySQL8C]' window with the following SQL scripts being executed:

```

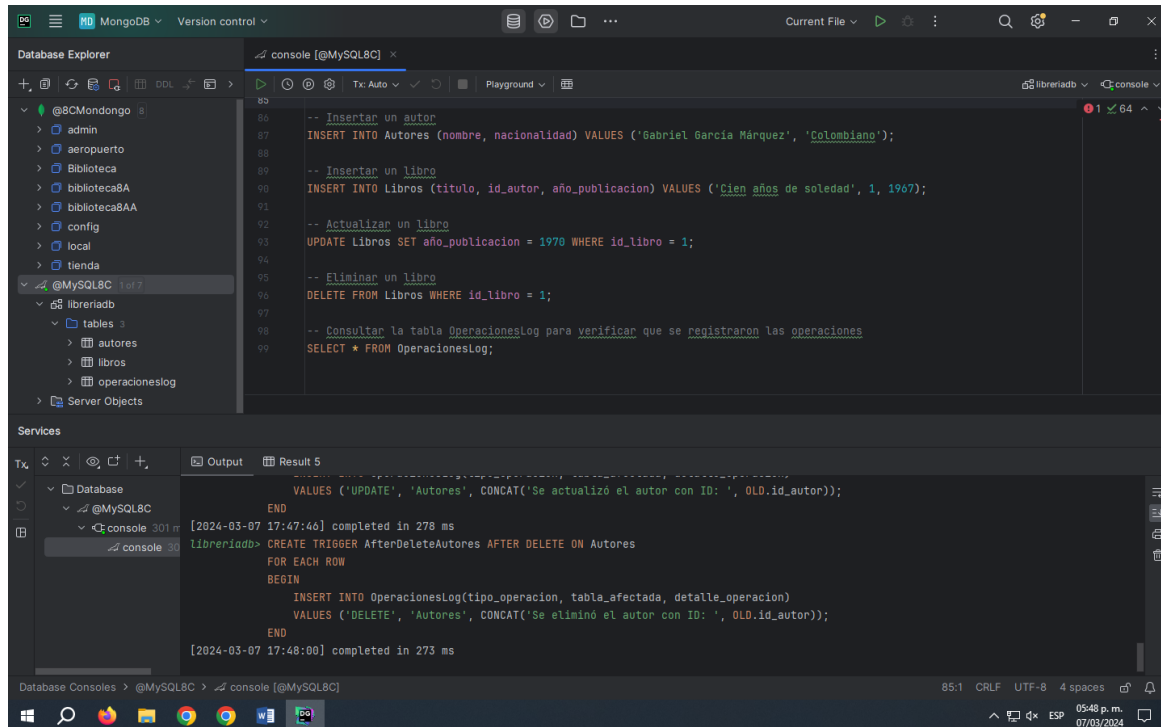
74 END; //
75 DELIMITER ;
76
77 DELIMITER //
78 CREATE TRIGGER AfterDeleteAutores AFTER DELETE ON Autores
79 FOR EACH ROW
80 BEGIN
81     INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
82     VALUES ('DELETE', 'Autores', CONCAT('Se eliminó el autor con ID: ', OLD.id_autor));
83 END; //
84 DELIMITER ;
  
```

The 'Output' window shows the execution results:

```

[2024-03-07 17:47:46] completed in 278 ms
libreriaadb> CREATE TRIGGER AfterDeleteAutores AFTER DELETE ON Autores
FOR EACH ROW
BEGIN
INSERT INTO OperacionesLog(tipo_operacion, tabla_afectada, detalle_operacion)
VALUES ('DELETE', 'Autores', CONCAT('Se eliminó el autor con ID: ', OLD.id_autor));
END
[2024-03-07 17:48:00] completed in 273 ms
  
```


Paso 8: Después de haber creado los **TRIGGERS**, realizar pruebas de log para determinar si se registran correctamente de acuerdo a los **TRIGGERS** creados anteriormente.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Database Explorer' with a tree view of the 'libreriadb' database, including tables like 'autores', 'libros', and 'operacioneslog'. The main editor window shows a SQL script with the following commands:

```

-- Insertar un autor
INSERT INTO Autores (nombre, nacionalidad) VALUES ('Gabriel García Márquez', 'Colombiano');

-- Insertar un libro
INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('Cien años de soledad', 1, 1967);

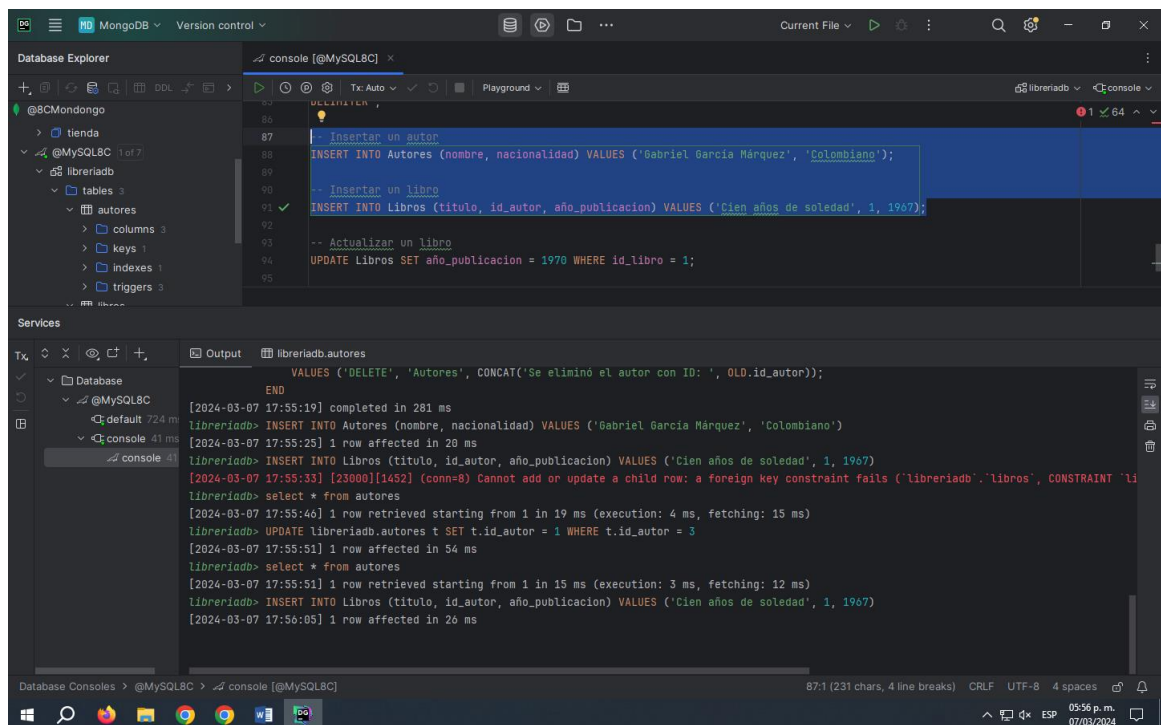
-- Actualizar un libro
UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1;

-- Eliminar un libro
DELETE FROM Libros WHERE id_libro = 1;

-- Consultar la tabla operacioneslog para verificar que se registraron las operaciones
SELECT * FROM OperacionesLog;
  
```

The 'Services' panel at the bottom shows the execution results of these commands, including timestamps and completion status.

Paso 9: Ahora empezar a ingresar un registro tanto de autor como de libro (comando **insert into**).



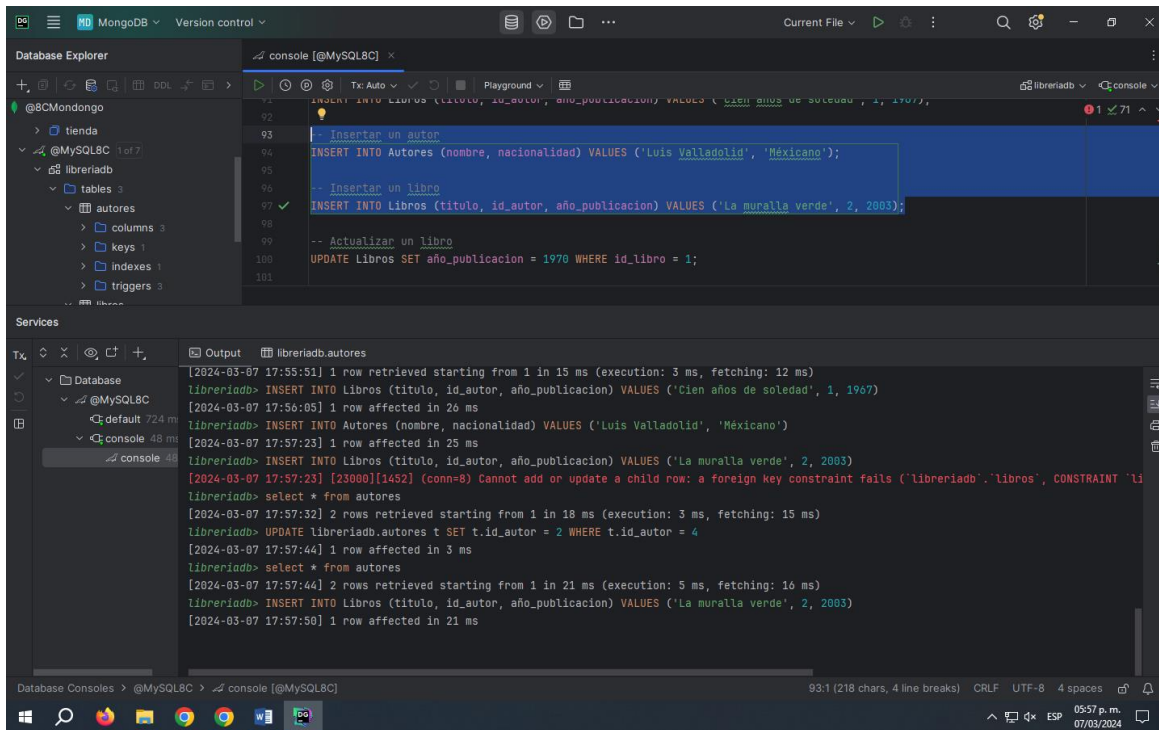
The screenshot shows the MySQL Workbench interface with the same SQL script as in the previous step. The 'Services' panel shows the execution results, including an error message:

```

[2024-03-07 17:55:19] completed in 281 ms
LibreriaDB> INSERT INTO Autores (nombre, nacionalidad) VALUES ('Gabriel García Márquez', 'Colombiano')
[2024-03-07 17:55:25] 1 row affected in 20 ms
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('Cien años de soledad', 1, 1967)
[2024-03-07 17:55:33] [23000][1452] (conn=8) Cannot add or update a child row: a foreign key constraint fails ('libreriadb','libros', CONSTRAINT 'li
LibreriaDB> select * from autores
[2024-03-07 17:55:40] 1 row retrieved starting from 1 in 19 ms (execution: 4 ms, fetching: 15 ms)
LibreriaDB> UPDATE libreriadb.autores t SET t.id_autor = 1 WHERE t.id_autor = 3
[2024-03-07 17:55:51] 1 row affected in 54 ms
LibreriaDB> select * from autores
[2024-03-07 17:55:51] 1 row retrieved starting from 1 in 15 ms (execution: 3 ms, fetching: 12 ms)
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('Cien años de soledad', 1, 1967)
[2024-03-07 17:56:05] 1 row affected in 26 ms
  
```

The error message indicates a foreign key constraint failure in the 'libros' table, likely due to the 'id_autor' value being 1, which does not exist in the 'autores' table.

Paso 10: Volver a ingresar un registro tanto de autor como de libro (comando insert into).



```

-- Insertar un autor
INSERT INTO Autores (nombre, nacionalidad) VALUES ('Luis Valladolid', 'México');

-- Insertar un libro
INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003);

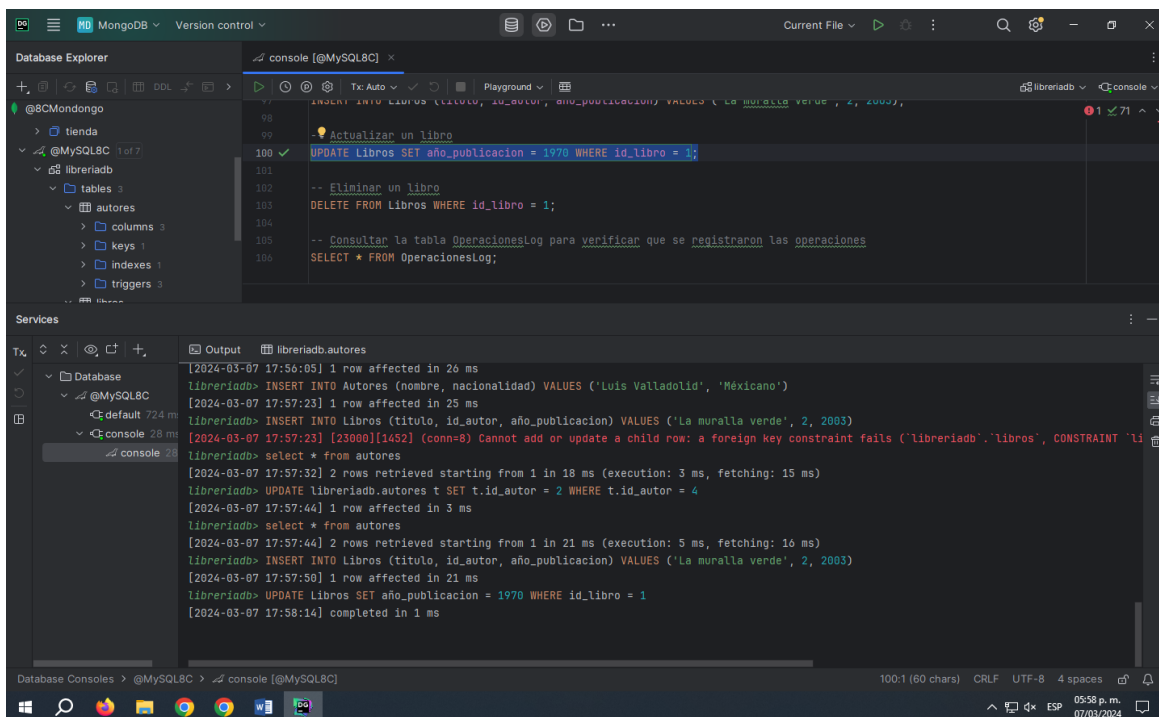
-- Actualizar un libro
UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1;
  
```

Output:

```

[2024-03-07 17:55:51] 1 row retrieved starting from 1 in 15 ms (execution: 3 ms, fetching: 12 ms)
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('Cien años de soledad', 1, 1967)
[2024-03-07 17:56:05] 1 row affected in 26 ms
LibreriaDB> INSERT INTO Autores (nombre, nacionalidad) VALUES ('Luis Valladolid', 'México')
[2024-03-07 17:57:23] 1 row affected in 25 ms
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:23] [23000][1452] (conn=8) Cannot add or update a child row: a foreign key constraint fails ('LibreriaDB'. 'Libros', CONSTRAINT 'LibreriaDB'
LibreriaDB> select * from autores
[2024-03-07 17:57:32] 2 rows retrieved starting from 1 in 18 ms (execution: 3 ms, fetching: 15 ms)
LibreriaDB> UPDATE LibreriaDB.autores t SET t.id_autor = 2 WHERE t.id_autor = 4
[2024-03-07 17:57:44] 1 row affected in 3 ms
LibreriaDB> select * from autores
[2024-03-07 17:57:44] 2 rows retrieved starting from 1 in 21 ms (execution: 5 ms, fetching: 16 ms)
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:50] 1 row affected in 21 ms
  
```

Paso 11: Con comando update en la tabla libros actualizar el año de publicación de un libro.



```

-- Actualizar un libro
UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1;

-- Eliminar un libro
DELETE FROM Libros WHERE id_libro = 1;

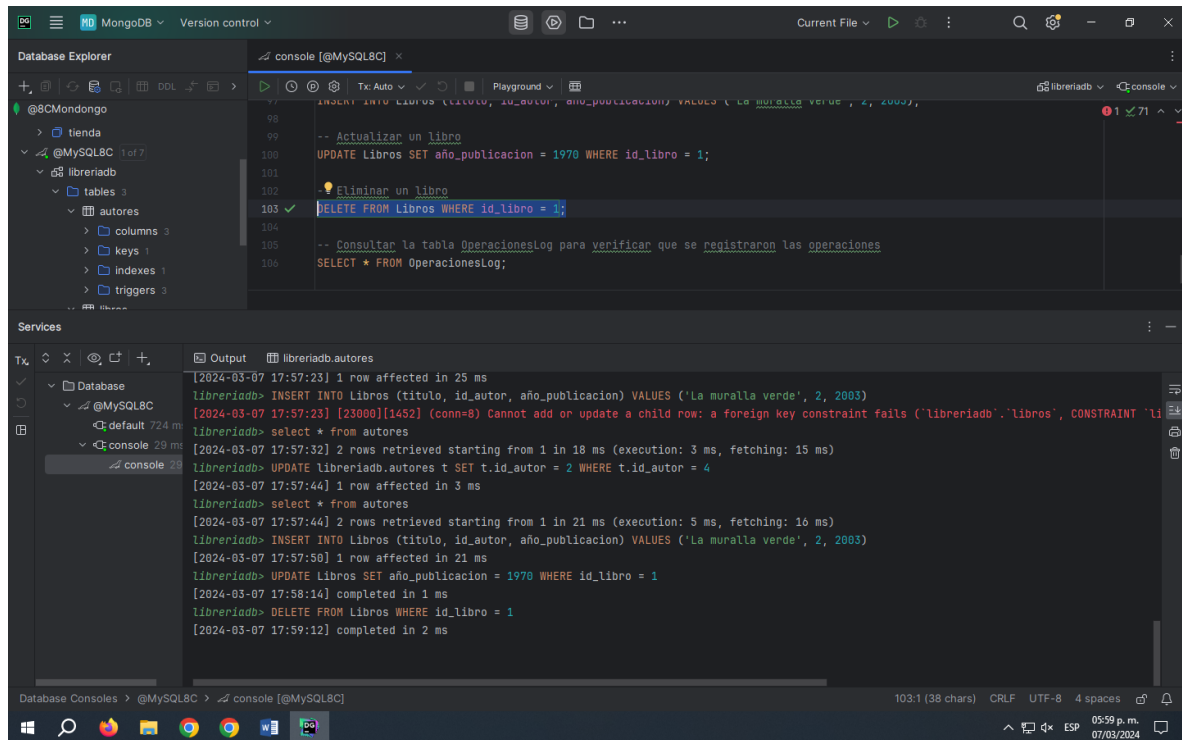
-- Consultar la tabla OperacionesLog para verificar que se registraron las operaciones
SELECT * FROM OperacionesLog;
  
```

Output:

```

[2024-03-07 17:56:05] 1 row affected in 26 ms
LibreriaDB> INSERT INTO Autores (nombre, nacionalidad) VALUES ('Luis Valladolid', 'México')
[2024-03-07 17:57:23] 1 row affected in 25 ms
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:23] [23000][1452] (conn=8) Cannot add or update a child row: a foreign key constraint fails ('LibreriaDB'. 'Libros', CONSTRAINT 'LibreriaDB'
LibreriaDB> select * from autores
[2024-03-07 17:57:32] 2 rows retrieved starting from 1 in 18 ms (execution: 3 ms, fetching: 15 ms)
LibreriaDB> UPDATE LibreriaDB.autores t SET t.id_autor = 2 WHERE t.id_autor = 4
[2024-03-07 17:57:44] 1 row affected in 3 ms
LibreriaDB> select * from autores
[2024-03-07 17:57:44] 2 rows retrieved starting from 1 in 21 ms (execution: 5 ms, fetching: 16 ms)
LibreriaDB> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:50] 1 row affected in 21 ms
LibreriaDB> UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1
[2024-03-07 17:58:14] completed in 1 ms
  
```

Paso 12: Con comando **delete** en la tabla **libros** eliminar el registro del libro con **id 1**.



```

-- Actualizar un libro
UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1;

-- Eliminar un libro
DELETE FROM Libros WHERE id_libro = 1;

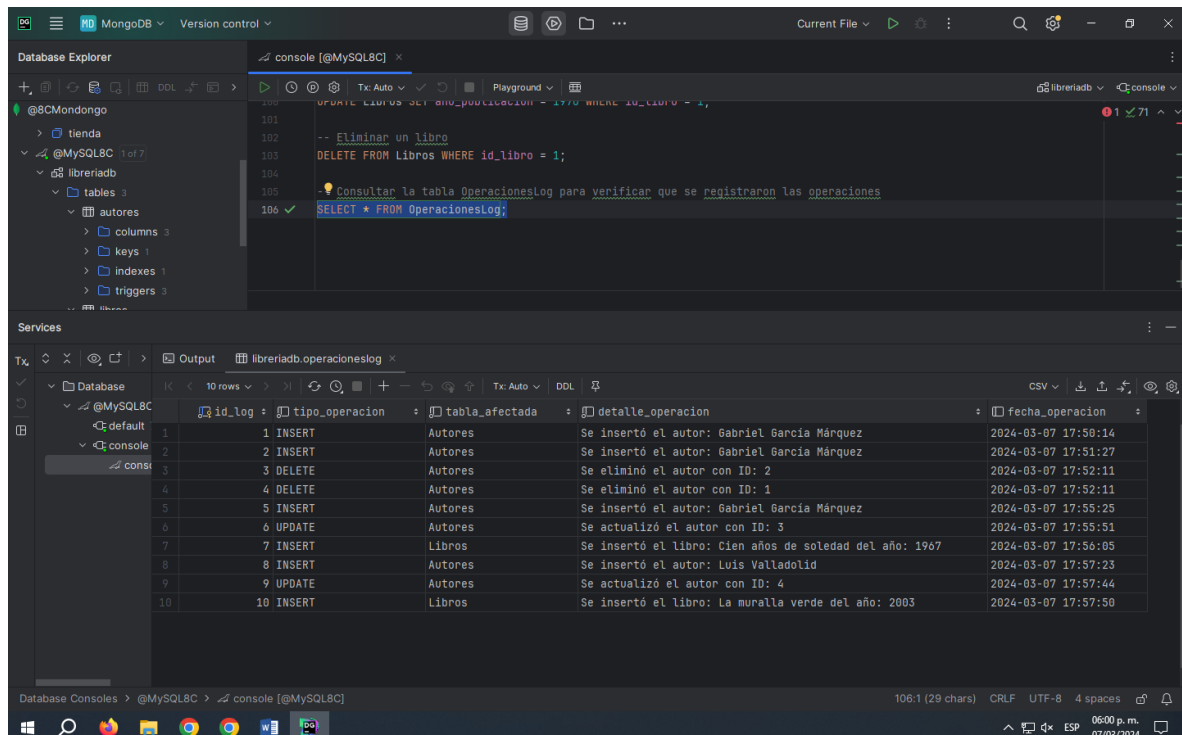
-- Consultar la tabla OperacionesLog para verificar que se registraron las operaciones
SELECT * FROM OperacionesLog;
  
```

Output:

```

[2024-03-07 17:57:23] 1 row affected in 25 ms
libreriaadb> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:32] [23000][1452] (conn=8) Cannot add or update a child row: a foreign key constraint fails ('libreriaadb`.`Libros', CONSTRAINT 'libreriaadb`.`fk_libros_autor' FOREIGN KEY (id_autor) REFERENCES autores (id_autor))
libreriaadb> select * from autores
[2024-03-07 17:57:32] 2 rows retrieved starting from 1 in 18 ms (execution: 3 ms, fetching: 15 ms)
libreriaadb> UPDATE libreriaadb.autores t SET t.id_autor = 2 WHERE t.id_autor = 4
[2024-03-07 17:57:44] 1 row affected in 3 ms
libreriaadb> select * from autores
[2024-03-07 17:57:44] 2 rows retrieved starting from 1 in 21 ms (execution: 5 ms, fetching: 16 ms)
libreriaadb> INSERT INTO Libros (titulo, id_autor, año_publicacion) VALUES ('La muralla verde', 2, 2003)
[2024-03-07 17:57:50] 1 row affected in 21 ms
libreriaadb> UPDATE Libros SET año_publicacion = 1970 WHERE id_libro = 1
[2024-03-07 17:58:14] completed in 1 ms
libreriaadb> DELETE FROM Libros WHERE id_libro = 1
[2024-03-07 17:59:12] completed in 2 ms
  
```

Paso 13: Finalmente, con comando **select * from** en la tabla **operacionlogs** se mostrará el historial de todos los cambios que se han realizado.



```

SELECT * FROM OperacionesLog;
  
```

id_log	tipo_operacion	tabla_afectada	detalle_operacion	fecha_operacion
1	INSERT	Autores	Se insertó el autor: Gabriel García Márquez	2024-03-07 17:50:14
2	INSERT	Autores	Se insertó el autor: Gabriel García Márquez	2024-03-07 17:51:27
3	DELETE	Autores	Se eliminó el autor con ID: 2	2024-03-07 17:52:11
4	DELETE	Autores	Se eliminó el autor con ID: 1	2024-03-07 17:52:11
5	INSERT	Autores	Se insertó el autor: Gabriel García Márquez	2024-03-07 17:55:25
6	UPDATE	Autores	Se actualizó el autor con ID: 3	2024-03-07 17:55:51
7	INSERT	Libros	Se insertó el libro: Cien años de soledad del año: 1967	2024-03-07 17:56:05
8	INSERT	Autores	Se insertó el autor: Luis Valladolid	2024-03-07 17:57:23
9	UPDATE	Autores	Se actualizó el autor con ID: 4	2024-03-07 17:57:44
10	INSERT	Libros	Se insertó el libro: La muralla verde del año: 2003	2024-03-07 17:57:50

CONCLUSIÓN

Las bitácoras de bases de datos con logs son una herramienta fundamental para el seguimiento y la auditoría de las operaciones realizadas en una base de datos. Estas bitácoras registran de manera detallada todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) realizadas en las tablas de la base de datos, proporcionando un registro histórico de cambios que pueden ser cruciales para la seguridad, la integridad y la trazabilidad de los datos.

La implementación de triggers para capturar operaciones CRUD garantiza que cada vez que se realice una modificación en los datos, se registre automáticamente en la bitácora de la base de datos. Esto permite a los administradores y desarrolladores monitorear el flujo de datos, identificar posibles problemas de rendimiento o integridad de datos, así como rastrear actividades sospechosas o maliciosas.

Además de ser una herramienta invaluable para la detección de errores y el mantenimiento de la integridad de los datos, las bitácoras de bases de datos con logs también son esenciales para cumplir con requisitos de cumplimiento normativo y para la resolución de problemas en entornos de producción. En resumen, las bitácoras de bases de datos con logs son una práctica recomendada para garantizar la transparencia, la seguridad y la confiabilidad de los sistemas de bases de datos en cualquier entorno operativo.

