

Documentación taller #6

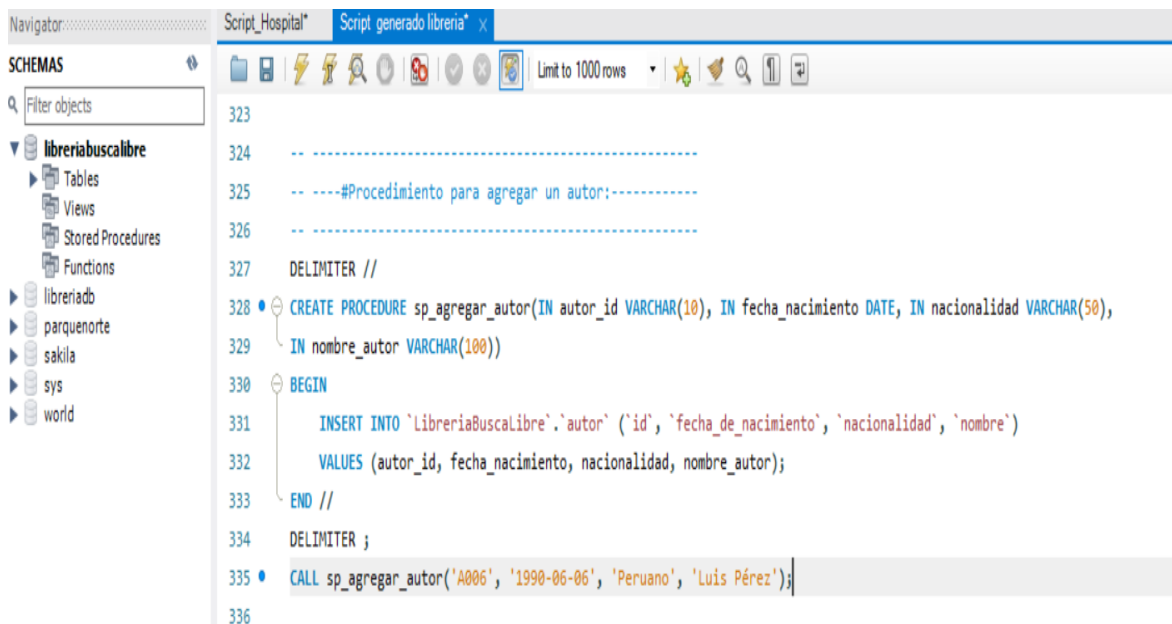
Para este taller debemos continuar con lo planteado en el taller 5 y hacer uso de lo aprendido anteriormente para poder implementar el CRUD en las bases de datos.

Los requerimientos son los siguientes:

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

Para solucionar el primer punto elabore los cuatro procedimientos almacenados que cumplen las funciones de agregar, actualizar, consultar y borrar datos de la tabla **autor**:

1. Procedimiento para agregar un autor:



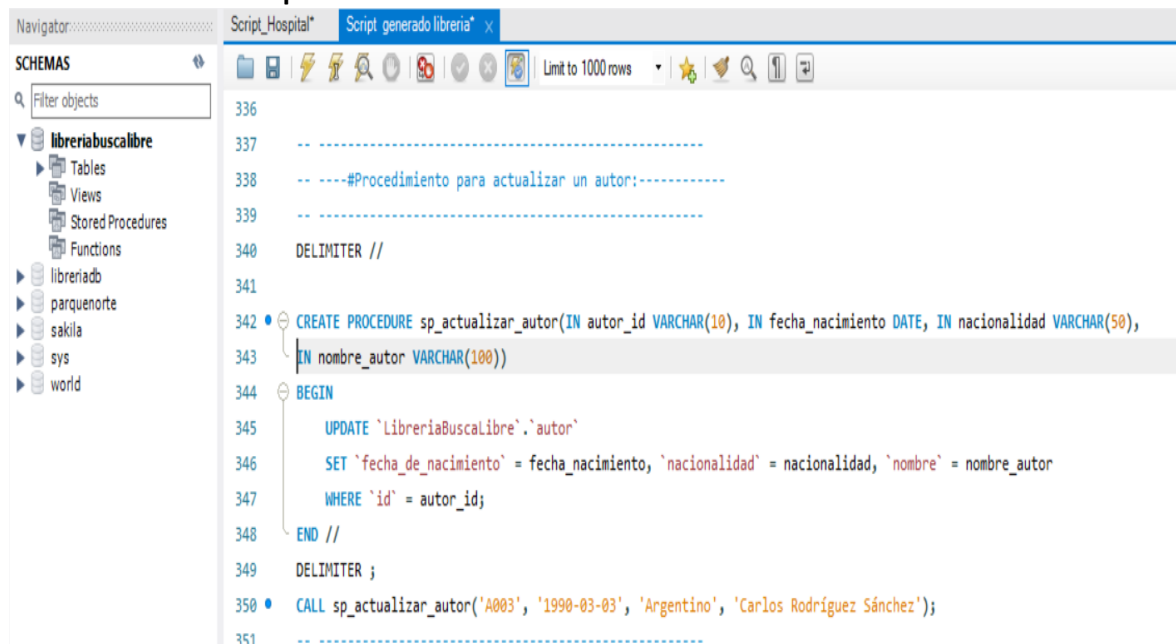
The screenshot shows a SQL IDE interface with a 'Script_Hospital*' window. The 'Script generado libreria*' tab is active, displaying a SQL script for creating a stored procedure to add an author. The script is as follows:

```
323
324  -- -----
325  -- ----#Procedimiento para agregar un autor:-----
326  -- -----
327  DELIMITER //
328  CREATE PROCEDURE sp_agregar_autor(IN autor_id VARCHAR(10), IN fecha_nacimiento DATE, IN nacionalidad VARCHAR(50),
329  IN nombre_autor VARCHAR(100))
330  BEGIN
331      INSERT INTO `LibreriaBuscaLibre`.`autor` (`id`, `fecha_de_nacimiento`, `nacionalidad`, `nombre`)
332      VALUES (autor_id, fecha_nacimiento, nacionalidad, nombre_autor);
333  END //
334  DELIMITER ;
335  CALL sp_agregar_autor('A006', '1990-06-06', 'Peruano', 'Luis Pérez');
336
```

En la anterior imagen podemos observar un procedimiento que se llama `sp_agregar_autor` y tiene cuatro parámetros de entrada: `autor_id` de tipo `VARCHAR(10)`, `fecha_nacimiento` de tipo `DATE`, `nacionalidad` de tipo `VARCHAR(50)` y `nombre_autor` de tipo `VARCHAR(100)`.

El cuerpo del procedimiento consta de una sola instrucción SQL que inserta una nueva fila en la tabla `autor` de la base de datos. La instrucción `INSERT INTO` especifica la tabla `autor` y los nombres de columna en los que se insertarán los valores. Luego, los valores se pasan a la instrucción `VALUES` en el mismo orden en que se enumeran las columnas. Los valores son los parámetros de entrada del procedimiento.

2. Procedimiento para actualizar un autor:

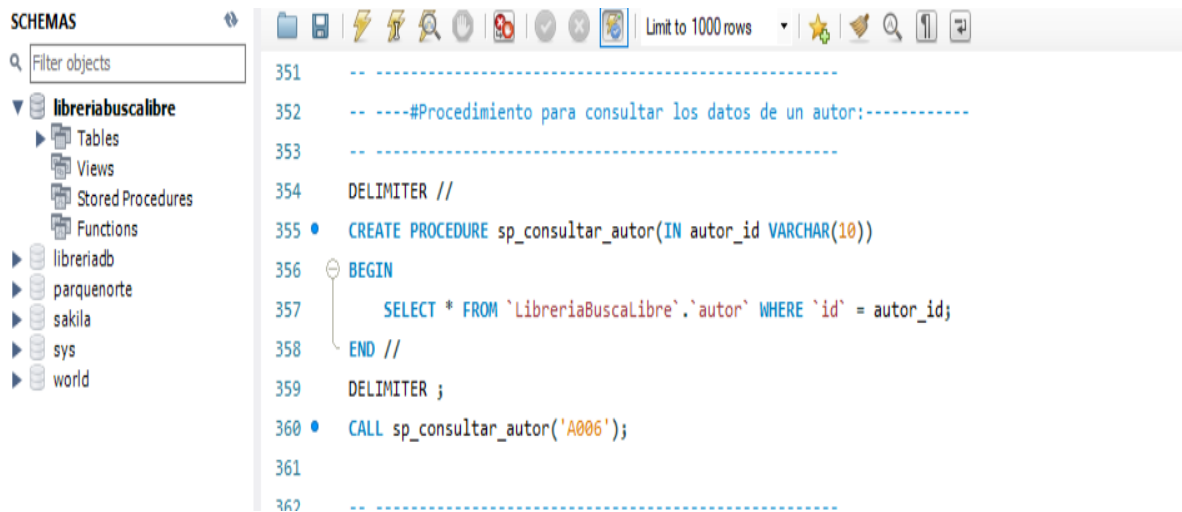


```
336
337  -----
338  -- ----#Procedimiento para actualizar un autor:-----
339  -----
340  DELIMITER //
341
342  CREATE PROCEDURE sp_actualizar_autor(IN autor_id VARCHAR(10), IN fecha_nacimiento DATE, IN nacionalidad VARCHAR(50),
343  IN nombre_autor VARCHAR(100))
344  BEGIN
345      UPDATE `LibreriaBuscaLibre`.`autor`
346      SET `fecha_de_nacimiento` = fecha_nacimiento, `nacionalidad` = nacionalidad, `nombre` = nombre_autor
347      WHERE `id` = autor_id;
348  END //
349  DELIMITER ;
350  CALL sp_actualizar_autor('A003', '1990-03-03', 'Argentino', 'Carlos Rodríguez Sánchez');
351  -----
```

En la imagen anterior podemos observar procedimiento que se llama `sp_actualizar_autor` y tiene cuatro parámetros de entrada: `autor_id` de tipo `VARCHAR(10)`, `fecha_nacimiento` de tipo `DATE`, `nacionalidad` de tipo `VARCHAR(50)` y `nombre_autor` de tipo `VARCHAR(100)`.

El cuerpo del procedimiento consta de una sola instrucción SQL que actualiza una fila existente en la tabla `autor` de la base de datos. La instrucción `UPDATE` especifica la tabla `autor` y las columnas que se actualizarán con los nuevos valores. La cláusula `SET` establece los valores de las columnas a los nuevos valores que se pasan como parámetros de entrada del procedimiento. La cláusula `WHERE` se utiliza para especificar qué fila debe actualizarse, en este caso, la fila cuyo `id` coincide con el parámetro `autor_id`.

3. Procedimiento para consultar los datos de un autor:



The screenshot shows a SQL IDE interface. On the left, a 'SCHEMAS' pane lists databases including 'libreriaBuscalibre'. The main editor displays SQL code for a stored procedure named 'sp_consultar_autor'. The code is as follows:

```
351  -- -----
352  -- ----#Procedimiento para consultar los datos de un autor:-----
353  -- -----
354  DELIMITER //
355  • CREATE PROCEDURE sp_consultar_autor(IN autor_id VARCHAR(10))
356  BEGIN
357      SELECT * FROM `LibreriaBuscaLibre`.`autor` WHERE `id` = autor_id;
358  END //
359  DELIMITER ;
360  • CALL sp_consultar_autor('A006');
361
362  -- -----
```

En la imagen anterior podemos observar procedimiento se llama `sp_consultar_autor` y tiene un parámetro de entrada: `autor_id` de tipo `VARCHAR(10)`.

El cuerpo del procedimiento consta de una sola instrucción SQL que busca una fila específica en la tabla `autor` de la base de datos. La instrucción `SELECT` se utiliza para recuperar todos los datos de la fila que cumple la condición especificada en la cláusula `WHERE`. En este caso, la cláusula `WHERE` se utiliza para buscar la fila cuyo `id` coincide con el parámetro `autor_id`.

4. Procedimiento para borrar un autor:

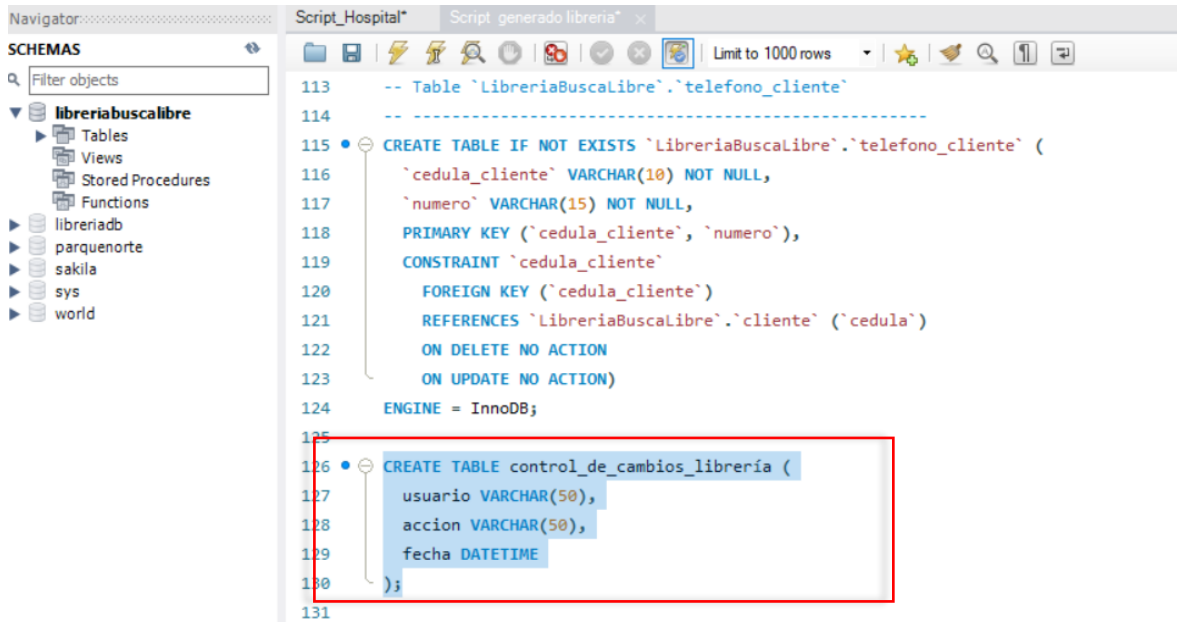


The screenshot shows a SQL IDE interface with SQL code for a stored procedure named 'sp_borrar_autor'. The code is as follows:

```
362  -- -----
363  -- ----#Procedimiento para borrar un autor:-----
364  -- -----
365  DELIMITER //
366  • CREATE PROCEDURE sp_borrar_autor(IN autor_id VARCHAR(10))
367  BEGIN
368      DELETE FROM `LibreriaBuscaLibre`.`autor` WHERE `id` = autor_id;
369  END //
370  DELIMITER ;
371  • CALL sp_borrar_autor('A006');
```

Este procedimiento se llama `sp_borrar_autor` y tiene un parámetro de entrada: `autor_id` de tipo `VARCHAR(10)`. El cuerpo del procedimiento consta de una sola instrucción SQL que elimina una fila específica de la tabla `autor` de la base de datos. La instrucción `DELETE FROM` se utiliza para eliminar una o varias filas de una tabla. En este caso, la cláusula `WHERE` se utiliza para especificar qué fila se debe eliminar, en este caso, la fila cuyo `id` coincide con el parámetro `autor_id`.

Para el siguiente paso cree una nueva tabla llamada "control_de_cambios_librería" con las columnas "usuario", "accion" y "fecha".



The screenshot shows a database management tool with a 'Navigator' pane on the left and a 'Script' editor on the right. The 'Navigator' pane shows a tree view of databases, including 'libreriaBuscalibre'. The 'Script' editor shows a SQL script for creating a table. The script is as follows:

```

113 -- Table `LibreriaBuscaLibre`.`telefono_cliente`
114 -----
115 CREATE TABLE IF NOT EXISTS `LibreriaBuscaLibre`.`telefono_cliente` (
116   `cedula_cliente` VARCHAR(10) NOT NULL,
117   `numero` VARCHAR(15) NOT NULL,
118   PRIMARY KEY (`cedula_cliente`, `numero`),
119   CONSTRAINT `cedula_cliente`
120     FOREIGN KEY (`cedula_cliente`)
121     REFERENCES `LibreriaBuscaLibre`.`cliente` (`cedula`)
122     ON DELETE NO ACTION
123     ON UPDATE NO ACTION)
124 ENGINE = InnoDB;
125
126 CREATE TABLE control_de_cambios_librería (
127   usuario VARCHAR(50),
128   accion VARCHAR(50),
129   fecha DATETIME
130 );
131

```

Después de esto cree los dos Trigger:

1. Trigger para registrar la acción de agregar un registro en la tabla autor:

```

-----
-- ----#CREACION DE LOS TRIGGERS:-----
-----

-- ----#Trigger para registrar la acción de agregar un registro en la tabla autor:-----
-----

CREATE TRIGGER agregar_autor_trigger
AFTER INSERT ON autor
FOR EACH ROW
INSERT INTO control_de_cambios_librería (usuario, accion, fecha)
VALUES (USER(), 'Agregó un registro en la tabla autor', NOW());

```

Este trigger se llama agregar_autor_trigger y se activa automáticamente después de que se inserta una fila en la tabla autor. La cláusula AFTER INSERT ON se utiliza para especificar el evento que activa el trigger y la tabla en la que se debe verificar el evento. La cláusula FOR EACH ROW indica que el trigger se activa para cada fila que se inserta en la tabla.

El cuerpo del trigger consta de una sola instrucción SQL que inserta una nueva fila en la tabla control_de_cambios_librería. La instrucción INSERT INTO especifica la tabla control_de_cambios_librería y los nombres de columna en los que se insertarán los valores. Los valores son el nombre de usuario del que realiza la operación, un mensaje que indica que se agregó un registro en la tabla autor y la fecha y hora en que se realiza la operación.

2. Trigger para registrar la acción de agregar un registro en la tabla autor:

```
386  -- -----
387  -- ----#Trigger para registrar la acción de eliminar un registro en la tabla autor:-----
388  -- -----
389
390 • CREATE TRIGGER eliminar_autor_trigger
391     AFTER DELETE ON autor
392     FOR EACH ROW
393     INSERT INTO control_de_cambios_librería (usuario, accion, fecha)
394     VALUES (USER(), 'Eliminó un registro en la tabla autor', NOW());
395
```

Este trigger se llama `eliminar_autor_trigger` y se activa automáticamente después de que se elimina una fila de la tabla `autor`. La cláusula `AFTER DELETE ON` se utiliza para especificar el evento que activa el trigger y la tabla en la que se debe verificar el evento. La cláusula `FOR EACH ROW` indica que el trigger se activa para cada fila que se elimina de la tabla.

El cuerpo del trigger consta de una sola instrucción SQL que inserta una nueva fila en la tabla `control_de_cambios_librería`. La instrucción `INSERT INTO` especifica la tabla `control_de_cambios_librería` y los nombres de columna en los que se insertarán los valores. Los valores son el nombre de usuario del que realiza la operación, un mensaje que indica que se eliminó un registro en la tabla `autor` y la fecha y hora en que se realiza la operación.

Después de esto procedí a realizar el mismo procedimiento, pero con la base de datos de Hospital específicamente con la tabla de **medicamento**.

1. Procedimiento para agregar un Medicamento:

```
376
377  -- -----
378  -- ----#Procedimiento para crear un medicamento:-----
379  -- -----
380  DELIMITER //
381 • CREATE PROCEDURE CrearMedicamento
382   (
383     IN id_Medicamento VARCHAR(10),
384     IN Nombre VARCHAR(50),
385     IN dosis VARCHAR(20)
386   )
387   BEGIN
388     INSERT INTO mydb.Medicamento (id_Medicamento, Nombre, dosis)
389     VALUES (id_Medicamento, Nombre, dosis);
390   END //
391   DELIMITER ;
392 • CALL CrearMedicamento('M006', 'Aspirina', '500 mg');
393
```

Este procedimiento se define con tres parámetros de entrada: "id_Medicamento", "Nombre" y "dosis".

El objetivo de este procedimiento es insertar un nuevo registro en la tabla "Medicamento" de la base de datos "mydb". La sentencia INSERT INTO se utiliza para agregar los valores proporcionados a la tabla. Los valores que se insertan en la tabla son los valores que se pasan como parámetros al procedimiento.

2. Procedimiento para Consultar un Medicamento:

```
394      -- -----
395      -- ----#Procedimiento para consultar un medicamento por su ID:-----
396      -- -----
397      DELIMITER //
398
399  • CREATE PROCEDURE ConsultarMedicamento
400      (
401          IN id_Medicamento VARCHAR(10)
402      )
403      BEGIN
404          SELECT * FROM mydb.Medicamento
405          WHERE id_Medicamento = id_Medicamento;
406      END //
407      DELIMITER ;
408  • CALL ConsultarMedicamento('M003');
409      -- -----
```

El objetivo de este procedimiento es buscar y devolver la información de un medicamento específico que tenga el ID de medicamento que se pasa como parámetro. La sentencia SELECT se utiliza para recuperar los datos de la tabla "Medicamento" de la base de datos "mydb" que coincidan con el valor del parámetro de entrada.

La cláusula WHERE se utiliza para filtrar los registros de la tabla y solo devolver aquellos que coincidan con el valor del parámetro. En este caso, la cláusula WHERE compara el valor de "id_Medicamento" de la tabla con el valor del parámetro "id_Medicamento" del procedimiento.

3. Procedimiento para Actualizar un Medicamento:

```
410  -- ----#Procedimiento para actualizar un medicamento:-----
411  -----
412  DELIMITER //
413  • CREATE PROCEDURE ActualizarMedicamento
414  (
415      IN id_Medicamento VARCHAR(10),
416      IN Nombre VARCHAR(50),
417      IN dosis VARCHAR(20)
418  )
419  BEGIN
420      UPDATE mydb.Medicamento
421      SET Nombre = Nombre, dosis = dosis
422      WHERE id_Medicamento = id_Medicamento;
423  END //
424  DELIMITER ;
425  • CALL ActualizarMedicamento('M005', 'Clonazepam', '2 mg');
426  ---
```

Este procedimiento se define con tres parámetros de entrada: "id_Medicamento", "Nombre" y "dosis".

El objetivo de este procedimiento es actualizar la información de un medicamento específico que tenga el ID de medicamento que se pasa como parámetro. La sentencia UPDATE se utiliza para modificar los datos de la tabla "Medicamento" de la base de datos "mydb" que coincidan con el valor del parámetro de entrada.

La cláusula SET se utiliza para especificar los valores que se actualizarán en la tabla. En este caso, los valores que se actualizarán son los valores que se pasan como parámetros al procedimiento. La cláusula WHERE se utiliza para filtrar los registros de la tabla y solo actualizar aquellos que coincidan con el valor del parámetro.

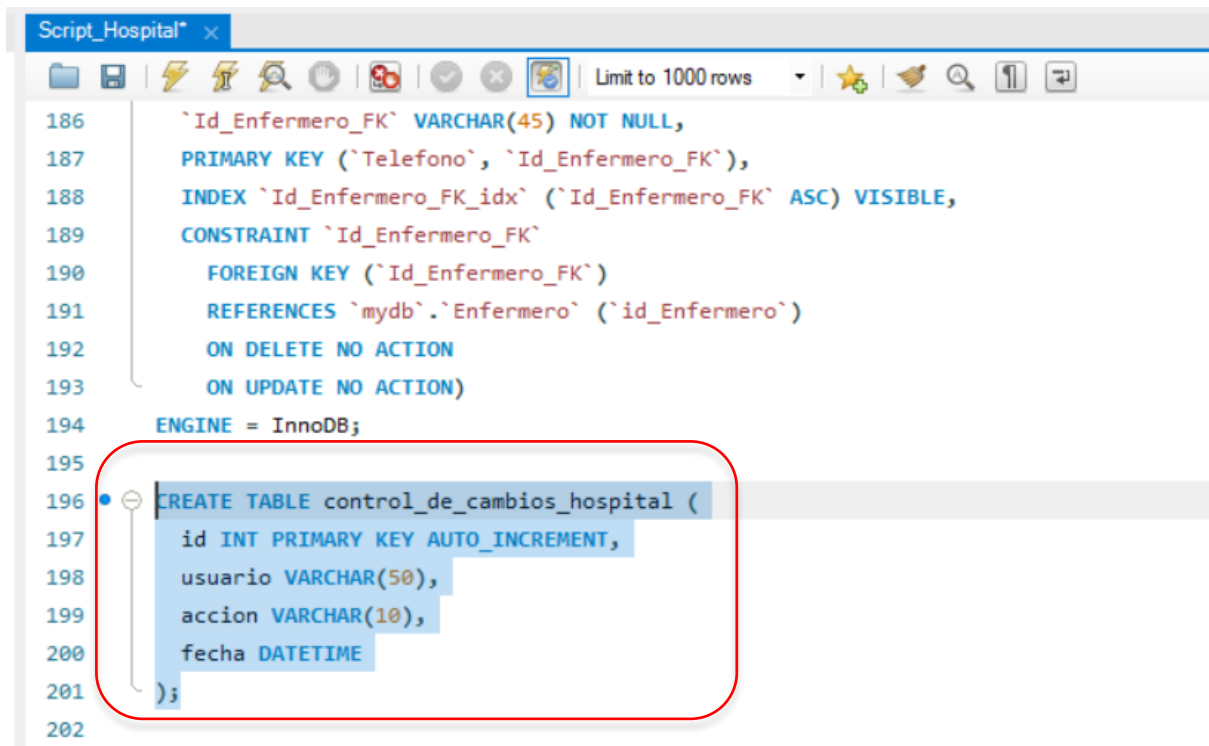
4. Procedimiento para Borrar un Medicamento:

```
427      -- -----
428      -- ----#Procedimiento para borrar un medicamento por su ID:-----
429      -- -----
430      DELIMITER //
431 • CREATE PROCEDURE BorrarMedicamento
432   (
433       IN id_Medicamento VARCHAR(10)
434   )
435   BEGIN
436       DELETE FROM mydb.Medicamento
437       WHERE id_Medicamento = id_Medicamento;
438   END //
439   DELIMITER ;
440 • CALL BorrarMedicamento('M001');
441
```

Este procedimiento almacenado llamado "BorrarMedicamento" tiene el objetivo de este procedimiento es eliminar un medicamento específico que tenga el ID de medicamento que se pasa como parámetro. La sentencia DELETE se utiliza para eliminar los datos de la tabla "Medicamento" de la base de datos "mydb" que coincidan con el valor del parámetro de entrada.

La cláusula WHERE se utiliza para filtrar los registros de la tabla y solo eliminar aquellos que coincidan con el valor del parámetro. En este caso, la cláusula WHERE compara el valor de "id_Medicamento" de la tabla con el valor del parámetro "id_Medicamento" del procedimiento.

Para el siguiente paso cree una nueva tabla llamada "control_de_cambios_hospital" con tres columnas (usuario, accion, fecha).



```
186     `Id_Enfermero_FK` VARCHAR(45) NOT NULL,  
187     PRIMARY KEY (`Telefono`, `Id_Enfermero_FK`),  
188     INDEX `Id_Enfermero_FK_idx` (`Id_Enfermero_FK` ASC) VISIBLE,  
189     CONSTRAINT `Id_Enfermero_FK`  
190         FOREIGN KEY (`Id_Enfermero_FK`)  
191         REFERENCES `mydb`.`Enfermero` (`id_Enfermero`)  
192         ON DELETE NO ACTION  
193         ON UPDATE NO ACTION)  
194     ENGINE = InnoDB;  
195  
196     CREATE TABLE control_de_cambios_hospital (  
197         id INT PRIMARY KEY AUTO_INCREMENT,  
198         usuario VARCHAR(50),  
199         accion VARCHAR(10),  
200         fecha DATETIME  
201     );  
202
```

Después de esto cree los dos Trigger:

1. Trigger para registrar la acción de agregar un registro en la tabla Medicamento:

```
-----  
-- ----#CREACION DE LOS TRIGGERS:-----  
-----  
-- ----#Trigger para registrar la acción de agregar un registro en la tabla Medicamento:-----  
-----  
  
DELIMITER //  
CREATE TRIGGER tr_control_cambios_agregar  
AFTER INSERT ON Medicamento  
FOR EACH ROW  
BEGIN  
    INSERT INTO control_de_cambios_hospital (usuario, accion, fecha) VALUES (USER(), 'agregar', NOW());  
END //  
DELIMITER ;
```

En la imagen anterior vemos un "trigger" llamado "tr_control_cambios_agregar" que se activará automáticamente después de insertar un nuevo registro en la tabla "Medicamento" de una base de datos MySQL.

El objetivo de este trigger es registrar la acción de agregar un nuevo medicamento en una tabla llamada "control_de_cambios_hospital". La sentencia INSERT se utiliza para agregar una nueva entrada en la tabla de control de cambios.

En este caso, el trigger está diseñado para registrar el nombre de usuario que realiza la acción, la acción en sí ("agregar" en este caso) y la fecha y hora en que se realiza la acción. La función USER() se utiliza para obtener el nombre de usuario actualmente conectado y la función NOW() se utiliza para obtener la fecha y hora actual.

2. Trigger para registrar la acción de eliminar un registro en la tabla Medicamento

```
-----  
-- ----#Trigger para registrar la acción de eliminar un registro en la tabla Medicamento:-----  
-----  
  
DELIMITER //  
CREATE TRIGGER tr_control_cambios_eliminar  
AFTER DELETE ON Medicamento  
FOR EACH ROW  
BEGIN  
    INSERT INTO control_de_cambios_hospital (usuario, accion, fecha) VALUES (USER(), 'eliminar', NOW());  
END //  
DELIMITER ;
```

En la imagen anterior vemos el segundo "trigger" llamado "tr_control_cambios_eliminar" que se activará automáticamente después de eliminar un registro en la tabla "Medicamento" de una base de datos MySQL.

El objetivo de este trigger es registrar la acción de eliminar un medicamento en una tabla llamada "control_de_cambios_hospital". La sentencia INSERT se utiliza para agregar una nueva entrada en la tabla de control de cambios.

En este caso, el trigger está diseñado para registrar el nombre de usuario que realiza la acción, la acción en sí ("eliminar" en este caso) y la fecha y hora en que se realiza la acción. La función USER() se utiliza para obtener el nombre de usuario actualmente conectado y la función NOW() se utiliza para obtener la fecha y hora actual.

Adjunto algunos de mis resultados probando la base de datos:

Creo un nuevo medicamento:

```
389     VALUES (id_Medicamento, Nombre, dosis);  
390     END //  
391     DELIMITER ;  
392 • CALL CrearMedicamento('M006', 'Aspirina', '500 mg');  
393
```

Aquí consulto el nuevo medicamento creado:

```
405     WHERE id_Medicamento = id_Medicamento;
406 END //
407 DELIMITER ;
408 • CALL ConsultarMedicamento('M006');
409 • drop procedure ConsultarMedicamento;
410 -----
411 -- ----#Procedimiento para actualizar un me
```

Result Grid | Filter Rows: | Export: | Wrap

	id_Medicamento	Nombre	dosis
▶	M001	Ibuprofeno	400 mg
	M002	Paracetamol	500 mg
	M003	Amoxicilina	500 mg
	M004	Omeprazol	20 mg
	M005	Diazepam	5 mg
	M006	Aspirina	500 mg