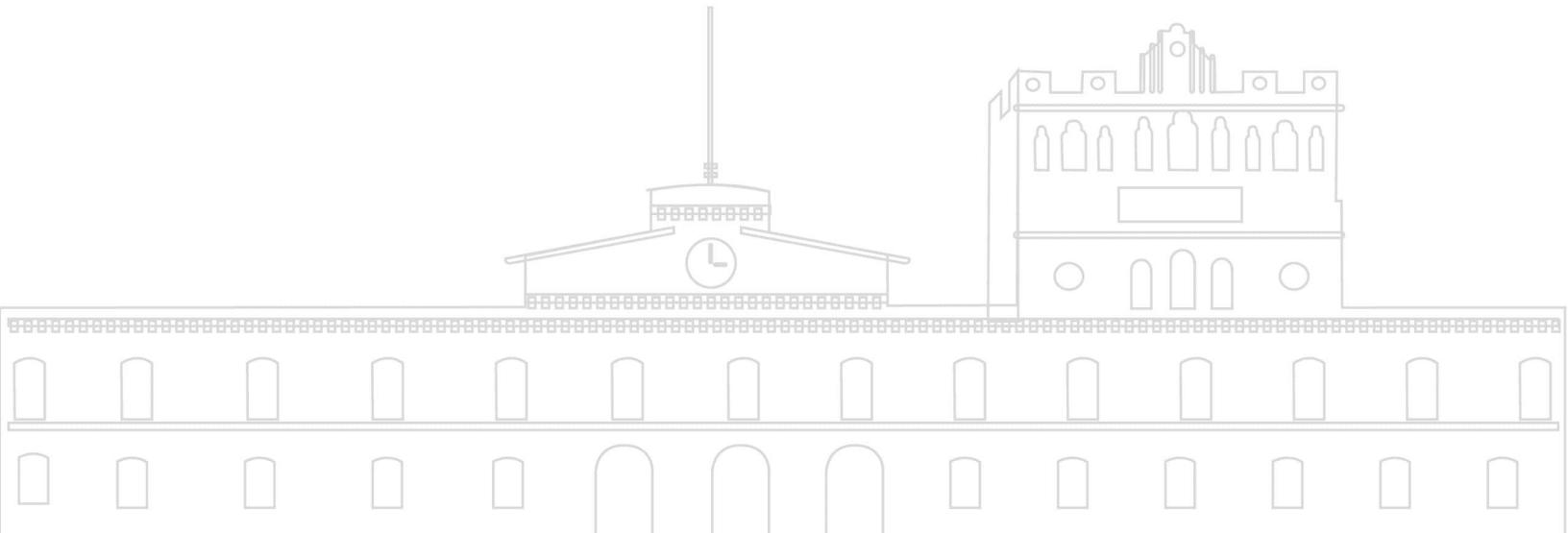


REPORTE DE PRÁCTICA NO. 0

NOMBRE DE LA PRÁCTICA: Práctica 0

ALUMNO:

José Eduardo Valles Aguilera



1. Introducción

Realizar una búsqueda de información en referencias bibliográficas físicas o digitales para desarrollar el marco teórico y presentar dos ejemplos de su constitución sobre la base de datos de Gestión de Flotillas de los siguientes conceptos:

- Procedimientos almacenados (Procedure)
- Funciones (Function)
- Estructuras de control condicionales y repetitivas
- Disparadores (Triggers)

2. Marco teórico

Procedimientos almacenados (Procedure)

Un procedimiento almacenado es un conjunto de instrucciones SQL que se almacena en la base de datos y puede ser ejecutado cuando se necesite. Estos procedimientos permiten encapsular lógica de negocio en la base de datos, facilitando la reutilización y el mantenimiento del código. Se crean utilizando la sentencia CREATE PROCEDURE y se invocan con CALL

Hernández, J. J. S. (s. f.). Unidad 12. Triggers, procedimientos y funciones en MySQL.
<https://josejuansanchez.org/bd/unidad-12-teoria/index.htmlprocedimientos>

Funciones (Function)

Una función es similar a un procedimiento almacenado, pero devuelve un valor único y puede ser utilizada dentro de sentencias SQL, como en un SELECT. Se crean con la sentencia CREATE FUNCTION.

Hernández, J. J. S. (s. f.-b). Unidad 12. Triggers, procedimientos y funciones en MySQL.
<https://josejuansanchez.org/bd/unidad-12-teoria/index.htmlfunciones>

Estructuras de control condicionales y repetitivas

Dentro de procedimientos y funciones, es común utilizar estructuras de control para dirigir el flujo de ejecución. Las estructuras condicionales (IF, CASE) permiten ejecutar bloques de código basados en condiciones específicas, mientras que las estructuras repetitivas (WHILE, LOOP, REPEAT) permiten la ejecución repetitiva de un bloque de código hasta que se cumpla una condición.

Temario: Procedimientos almacenados (estructuras repetitivas). (s. f.-b).
<https://www.tutorialesprogramacionya.com/mysqlia/temarios/descripcion.php?inicio=105cod=109punto=106>

Disparadores (Triggers)

Un disparador es un conjunto de instrucciones SQL que se ejecuta automáticamente en respuesta a ciertos eventos en una tabla, como inserciones, actualizaciones o eliminaciones. Se crean con la sentencia CREATE TRIGGER.

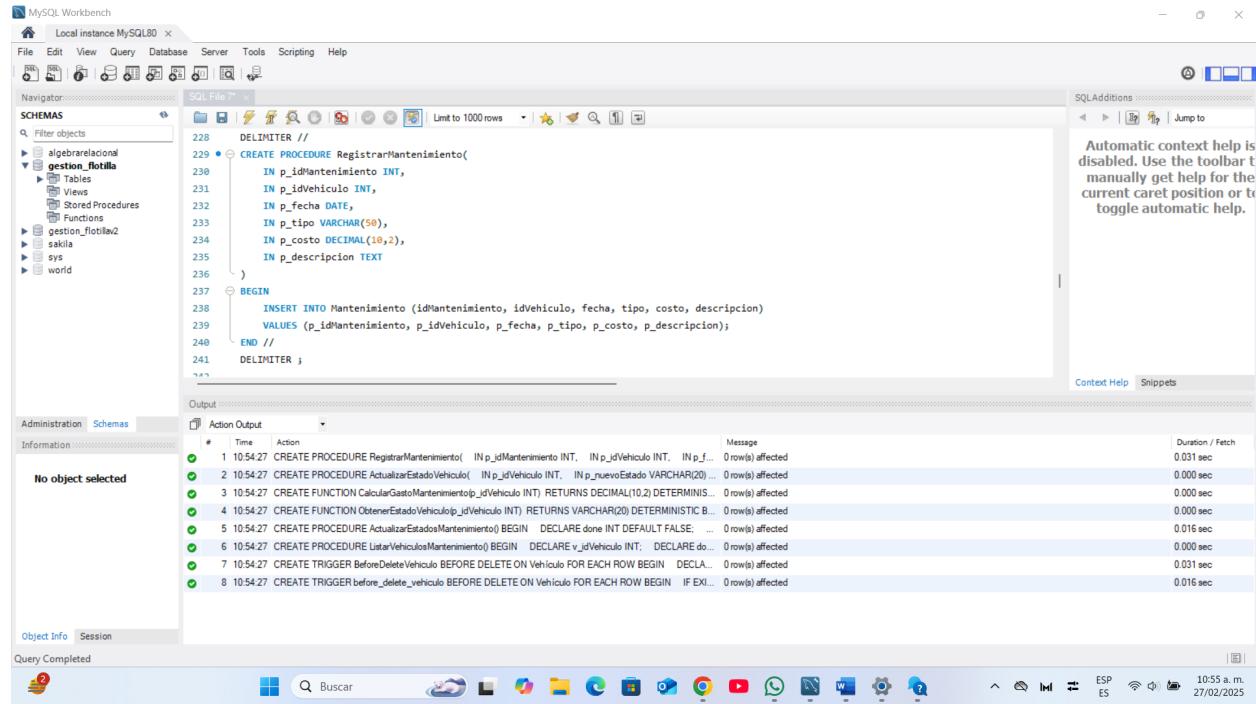
Hernández, J. J. S. (s. f.-c). Unidad 12. Triggers, procedimientos y funciones en MySQL.
<https://josejuansanchez.org/bd/unidad-12-teoria/index.htmltriggers>

3. Herramientas empleadas

1. MySQL Workbench. Es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, gestión y mantenimiento para el sistema de base de datos MySQL.

4. Desarrollo

Ejemplos de su constitución sobre la base de datos de Gestión de Flotillas



The screenshot shows the MySQL Workbench interface. In the center, a SQL editor window displays the code for creating a stored procedure named 'RegistrarMantenimiento'. The code includes parameters for ID, vehicle ID, date, type, cost, and description, and an INSERT statement into the 'Mantenimiento' table. Below the editor, the 'Output' pane shows the results of the execution, listing 8 events with their time, action, message, and duration. The status bar at the bottom right indicates the session is completed.

```
DELIMITER //
CREATE PROCEDURE RegistrarMantenimiento(
    IN p_idMantenimiento INT,
    IN p_idVehiculo INT,
    IN p_fecha DATE,
    IN p_tipo VARCHAR(50),
    IN p_costo DECIMAL(10,2),
    IN p_descripcion TEXT
)
BEGIN
    INSERT INTO Mantenimiento (idMantenimiento, idVehiculo, fecha, tipo, costo, descripcion)
    VALUES (p_idMantenimiento, p_idVehiculo, p_fecha, p_tipo, p_costo, p_descripcion);
END //
DELIMITER ;
```

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_f...	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20) ...	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINIS...	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC B...	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; DECLARE do...	0 row(s) affected	0.000 sec
7	10:54:27	CREATE TRIGGER BeforeDeleteVehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN DECLAR...	0 row(s) affected	0.031 sec
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF EX...	0 row(s) affected	0.016 sec

Figure 1: Registrar un nuevo mantenimiento

The screenshot shows the MySQL Workbench interface with the 'ActualizarEstadoVehiculo' stored procedure selected in the SQL editor. The procedure updates the 'estado' field in the 'Vehiculo' table based on the provided 'p_nuevoEstado' parameter. The output pane displays the execution history of the stored procedures and functions created in the current session.

```

243  DELIMITER //
244  CREATE PROCEDURE ActualizarEstadoVehiculo(
245      IN p_idVehiculo INT,
246      IN p_nuevoEstado VARCHAR(20)
247  )
248  BEGIN
249      UPDATE Vehiculo
250      SET estado = p_nuevoEstado
251      WHERE idVehiculo = p_idVehiculo;
252  END //
253  DELIMITER ;

```

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_f...)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20) ...)	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINIS...	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC B...	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; DECLARE do...	0 row(s) affected	0.000 sec
7	10:54:27	CREATE TRIGGER BeforeDeleteVehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN DECLAR...	0 row(s) affected	0.031 sec
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF EX...	0 row(s) affected	0.016 sec

Figure 2: Actualizar el estado de un vehículo

The screenshot shows the MySQL Workbench interface with the 'CalcularGastoMantenimiento' function selected in the SQL editor. The function calculates the total maintenance cost for a given vehicle ID. The output pane displays the execution history of the stored procedures and functions created in the current session.

```

255  DELIMITER //
256  CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT)
257  RETURNS DECIMAL(10,2)
258  DETERMINISTIC
259  BEGIN
260      DECLARE total DECIMAL(10,2);
261      SELECT SUM(costo) INTO total
262      FROM Mantenimiento
263      WHERE idVehiculo = p_idVehiculo;
264      RETURN IFNULL(total, 0);
265  END //
266  DELIMITER ;

```

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_f...)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20) ...)	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINIS...	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC B...	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; DECLARE do...	0 row(s) affected	0.000 sec
7	10:54:27	CREATE TRIGGER BeforeDeleteVehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN DECLAR...	0 row(s) affected	0.031 sec
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF EX...	0 row(s) affected	0.016 sec

Figure 3: Calcular el gasto total en mantenimiento de un vehículo

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `gestion_flotilla`.
- SQL File:** A new file named `SQL File 7*` is open, containing the following SQL code:


```

268  DELIMITER //
269  • CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT)
270    RETURNS VARCHAR(20)
271    DETERMINISTIC
272  BEGIN
273    DECLARE v_estado VARCHAR(20);
274    SELECT estado INTO v_estado
275    FROM Vehiculo
276    WHERE idVehiculo = p_idVehiculo;
277    RETURN v_estado;
278  END //
279  DELIMITER ;
      
```
- Action Output:** The output pane shows the execution of several statements, including the creation of other procedures and functions, and triggers. The last few entries are:

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_fecha DATE)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20))	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINISTIC	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; DECLARE do...	0 row(s) affected	0.000 sec
7	10:54:27	CREATE TRIGGER BeforeDeleteVehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN DECLA...	0 row(s) affected	0.031 sec
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF EXI...	0 row(s) affected	0.016 sec
- Output:** The output pane also displays the results of the executed queries.

Figure 4: Obtener el estado de un vehículo

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `gestion_flotilla`.
- SQL File:** A new file named `SQL File 7*` is open, containing the following SQL code:


```

281  DELIMITER //
282  • CREATE PROCEDURE ActualizarEstadosMantenimiento()
283  BEGIN
284    DECLARE done INT DEFAULT FALSE;
285    DECLARE v_idVehiculo INT;
286    DECLARE cur CURSOR FOR
287      SELECT idVehiculo FROM Mantenimiento WHERE fecha >= CURDATE() - INTERVAL 30 DAY;
288    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
289
290    OPEN cur;
291    read_loop: LOOP
292      FETCH cur INTO v_idVehiculo;
293      IF done THEN
294        LEAVE read_loop;
295      END IF;
296
297      UPDATE Vehiculo
298      SET estado = 'mantenimiento'
299      WHERE idVehiculo = v_idVehiculo;
300    END LOOP;
301    CLOSE cur;
302  END //
303  DELIMITER ;
      
```
- Action Output:** The output pane shows the execution of three statements:

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_fecha DATE)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20))	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINISTIC	0 row(s) affected	0.000 sec
- Output:** The output pane also displays the results of the executed queries.

Figure 5: Actualizar el estado de los vehículos según el mantenimiento reciente

```

DELIMITER //
CREATE PROCEDURE ListarVehiculosMantenimiento()
BEGIN
    DECLARE v_idVehiculo INT;
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT idVehiculo FROM Vehiculo WHERE estado = 'mantenimiento';
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO v_idVehiculo;
        IF done THEN
            LEAVE read_loop;
        END IF;

        SELECT * FROM Vehiculo WHERE idVehiculo = v_idVehiculo;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

```

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_fecha DATE)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20))	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINISTIC	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; ...	0 row(s) affected	0.000 sec

Figure 6: Lista de vehículos en mantenimiento

```

DELIMITER //
CREATE TRIGGER BeforeDeleteVehiculo
BEFORE DELETE ON Vehiculo
FOR EACH ROW
BEGIN
    DECLARE count_mantenimiento INT;
    SELECT COUNT(*) INTO count_mantenimiento
    FROM Mantenimiento
    WHERE idVehiculo = OLD.idVehiculo;

    IF count_mantenimiento > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el vehículo, tiene registros de mantenimiento';
    END IF;
END //
DELIMITER ;

```

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:54:27	CREATE PROCEDURE RegistrarMantenimiento(IN p_idMantenimiento INT, IN p_idVehiculo INT, IN p_fecha DATE)	0 row(s) affected	0.031 sec
2	10:54:27	CREATE PROCEDURE ActualizarEstadoVehiculo(IN p_idVehiculo INT, IN p_nuevoEstado VARCHAR(20))	0 row(s) affected	0.000 sec
3	10:54:27	CREATE FUNCTION CalcularGastoMantenimiento(p_idVehiculo INT) RETURNS DECIMAL(10,2) DETERMINISTIC	0 row(s) affected	0.000 sec
4	10:54:27	CREATE FUNCTION ObtenerEstadoVehiculo(p_idVehiculo INT) RETURNS VARCHAR(20) DETERMINISTIC	0 row(s) affected	0.000 sec
5	10:54:27	CREATE PROCEDURE ActualizarEstadosMantenimiento() BEGIN DECLARE done INT DEFAULT FALSE; ...	0 row(s) affected	0.016 sec
6	10:54:27	CREATE PROCEDURE ListarVehiculosMantenimiento() BEGIN DECLARE v_idVehiculo INT; ...	0 row(s) affected	0.000 sec
7	10:54:27	CREATE TRIGGER BeforeDeleteVehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN DECLARE ...	0 row(s) affected	0.031 sec
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF EX... 0 row(s) affected	0.016 sec	0.016 sec

Figure 7: Evitar que se eliminen vehículos si tienen registros de mantenimiento

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `gestion_flotilla`.
- SQL Editor:** A trigger named `before_delete_vehiculo` is being created. The code is as follows:


```

344  DELIMITER //
345  • CREATE TRIGGER before_delete_vehiculo
346    BEFORE DELETE ON Vehiculo
347    FOR EACH ROW
348    BEGIN
349      IF EXISTS (SELECT 1 FROM Ruta WHERE idVehiculo = OLD.idVehiculo) THEN
350        SIGNAL SQLSTATE '45000'
351        SET MESSAGE_TEXT = 'No se puede eliminar el vehículo porque tiene rutas asignadas';
352      END IF;
353    END //
354  DELIMITER ;
      
```
- Output:** The action output shows the execution of various stored procedures and functions, including `RegistrarMantenimiento`, `ActualizarEstadoVehiculo`, `CalcularGastoMantenimiento`, `ObtenerEstadoVehiculo`, `ActualizarEstadosMantenimiento`, `ListarVehiculosMantenimiento`, and the trigger creation.
- System Bar:** The system bar at the bottom right shows the date and time as 11:16 a.m. on 27/02/2025.

Figure 8: Evitar que se eliminen vehículos si tienen rutas asignadas

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `gestion_flotilla`.
- SQL Editor:** A series of statements are run, including calls to `RegistrarMantenimiento`, `ActualizarEstadoVehiculo`, and `SELECT` queries for maintenance and vehicles.
- Result Grid:** A result grid displays vehicle information for vehicle ID 3, showing columns: `idVehiculo`, `marca`, `modelo`, `año`, `placa`, and `estado`. The data is as follows:

idVehiculo	marca	modelo	año	placa	estado
3	Honda	Civic	2021	DEF-456	mantenimiento
- Output:** The action output shows the execution of stored procedures and triggers, including `RegistrarMantenimiento`, `ActualizarEstadoVehiculo`, and the trigger `before_delete_vehiculo`.
- System Bar:** The system bar at the bottom right shows the date and time as 11:21 a.m. on 27/02/2025.

Figure 9: Verificando que los procesos almacenados funcionen correctamente

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

SCHEMAS: algebralacional, gestion_flotilla

SQL File 7

```

358 • SELECT * FROM Mantenimiento WHERE idVehiculo = 5;
359 • SELECT * FROM Vehiculo WHERE idVehiculo = 5;
360
361 • SELECT CalcularGastoMantenimiento();
362 • SELECT ObtenerEstadoVehiculo();
363

```

Result Grid | Filter Rows: Export: Wrap Cell Content: | Result Grid

CalcularGastoMantenimiento();
550.00

Result 13 x Result 14

Action Output

#	Time	Action	Message	Duration / Fetch
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF E...	0 row(s) affected	0.016 sec
9	11:21:28	CALL RegistrarMantenimiento(21, 5, 2025-08-01, 'Cambio de frenos', 250.00, Cambio de frenos delanteros y tr...	1 row(s) affected	0.032 sec
10	11:21:28	CALL ActualizarEstadoVehiculo(3, mantenimiento)	1 row(s) affected	0.031 sec
11	11:21:28	SELECT * FROM Mantenimiento WHERE idVehiculo = 5 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
12	11:21:28	SELECT * FROM Vehiculo WHERE idVehiculo = 5 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	11:23:33	SELECT CalcularGastoMantenimiento(); LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	11:23:33	SELECT ObtenerEstadoVehiculo(); LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

No object selected

Object Info Session

Query Completed

UV alto Ahora

Buscar

11:24 a. m. 27/02/2025

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

SCHEMAS: algebralacional, gestion_flotilla

SQL File 7

```

358 • SELECT * FROM Mantenimiento WHERE idVehiculo = 5;
359 • SELECT * FROM Vehiculo WHERE idVehiculo = 5;
360
361 • SELECT CalcularGastoMantenimiento();
362 • SELECT ObtenerEstadoVehiculo();
363

```

Result Grid | Filter Rows: Export: Wrap Cell Content: | Result Grid

ObtenerEstadoVehiculo();
mantenimiento

Result 13 Result 14

Action Output

#	Time	Action	Message	Duration / Fetch
8	10:54:27	CREATE TRIGGER before_delete_vehiculo BEFORE DELETE ON Vehiculo FOR EACH ROW BEGIN IF E...	0 row(s) affected	0.016 sec
9	11:21:28	CALL RegistrarMantenimiento(21, 5, 2025-08-01, 'Cambio de frenos', 250.00, Cambio de frenos delanteros y tr...	1 row(s) affected	0.032 sec
10	11:21:28	CALL ActualizarEstadoVehiculo(3, mantenimiento)	1 row(s) affected	0.031 sec
11	11:21:28	SELECT * FROM Mantenimiento WHERE idVehiculo = 5 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
12	11:21:28	SELECT * FROM Vehiculo WHERE idVehiculo = 5 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
13	11:23:33	SELECT CalcularGastoMantenimiento(); LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
14	11:23:33	SELECT ObtenerEstadoVehiculo(); LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

No object selected

Object Info Session

Query Completed

UV alto Ahora

Buscar

11:24 a. m. 27/02/2025

Figure 10: Verificando que las funciones funcionen correctamente

The screenshot shows the MySQL Workbench interface with a SQL file named "SQL File 7" open. The code executed is:

```

364 • CALL ActualizarEstadosMantenimiento();
365 •     SELECT idVehiculo, estado FROM Vehiculo WHERE estado = 'mantenimiento';
366 • CALL ListarVehiculosMantenimiento();
367 •     SELECT * FROM Vehiculo WHERE estado = 'mantenimiento';
368

```

The Result Grid displays the following data:

	idVehiculo	marca	modelo	anio	placa	estado
2	Nissan	Versa	2019	XY789		mantenimiento
3	Honda	Civic	2021	DEF456		mantenimiento
4	Ford	Focus	2018	GHL789		mantenimiento
5	Chevrolet	Malibú	2022	JKL012		mantenimiento
6	Hyundai	Elantra	2020	MNO345		mantenimiento
7	Mazda	3	2019	PQR678		mantenimiento
8	Volkswagen	Jetta	2021	STU901		mantenimiento
9	Kia	Forte	2018	VWX234		mantenimiento
10	Tesla	Model 3	2022	YZA567		mantenimiento
11	Subaru	Impreza	2020	BCD890		mantenimiento
12	BMW	Serie 3	2019	EFG123		mantenimiento

The Output pane shows the log of actions taken by the stored procedure:

#	Time	Action	Message	Duration / Fetch
30	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
31	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
32	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
33	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
34	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
35	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
36	11:28:24	SELECT * FROM Vehiculo WHERE estado = 'mantenimiento' LIMIT 0, 1000	19 row(s) returned	0.015 sec / 0.000 sec

Figure 11: Verificando que las estructuras de control anidadas y repetitivas funcionen correctamente

The screenshot shows the MySQL Workbench interface with a SQL file named "SQL File 7" open. The code executed is:

```

353 • END //;
354 • DELIMITER ;
355
356 •     CALL RegistrarMantenimiento(21, 5, '2025-08-01', 'Cambio de frenos', 250.00, 'Cambio de frenos delanteros y traseros');
357 •     CALL ActualizarEstadoVehiculo(3, 'mantenimiento');
358 •     SELECT * FROM Mantenimiento WHERE idVehiculo = 5;
359 •     SELECT * FROM Vehiculo WHERE idVehiculo = 3;
360
361 •     SELECT CalcularGastoMantenimiento();
362 •     SELECT ObtenerEstadoVehiculo();
363
364 •     CALL ActualizarEstadosMantenimiento();
365 •     SELECT idVehiculo, estado FROM Vehiculo WHERE estado = 'mantenimiento';
366 •     CALL ListarVehiculosMantenimiento();
367 •     SELECT * FROM Vehiculo WHERE estado = 'mantenimiento';
368
369 •     DELETE FROM Vehiculo WHERE idVehiculo = 1;
370 •     DELETE FROM Vehiculo WHERE idVehiculo = 3;

```

The Output pane shows the log of actions taken by the stored procedure, with two entries highlighted in red:

#	Time	Action	Message	Duration / Fetch
32	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
33	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
34	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
35	11:28:24	CALL ListarVehiculosMantenimiento()	1 row(s) returned	- / 0.000 sec
36	11:28:24	SELECT * FROM Vehiculo WHERE estado = 'mantenimiento' LIMIT 0, 1000	19 row(s) returned	0.015 sec / 0.000 sec
37	11:31:13	DELETE FROM Vehiculo WHERE idVehiculo = 1	Error Code: 1644. No se puede eliminar el vehículo, tiene registros de mantenimiento	0.000 sec
38	11:31:19	DELETE FROM Vehiculo WHERE idVehiculo = 3	Error Code: 1644. No se puede eliminar el vehículo, tiene registros de mantenimiento	0.000 sec

Figure 12: Verificando que los triggers funcionen correctamente

5. Conclusiones

Con esta práctica aprendí nuevas funciones de MySQL Workbench con las cuales ahora no sólo seré capaz de almacenar información sino evitar errores al momento de manipularla como borrar algo que no debía por accidente o verificar solamente ciertos registros con base en ciertas condiciones. Estoy seguro que esto será útil más adelante en el curso.

Referencias Bibliográficas

References

- [1] Hernández, J. J. S. (s. f.). *Unidad 12. Triggers, procedimientos y funciones en MySQL*. Recuperado de <https://josejuansanchez.org/bd/unidad-12-teoria/index.html>
- [2] Temario: Procedimientos almacenados (estructuras repetitivas). (s. f.). Recuperado de https://www.tutorialesprogramacionya.com/mysqlia/temarios/descripcion.php?inicio=105&cod=109&zpunto=106#google_vignette