

REPORTE DE PRÁCTICA NO. 0

PRÁCTICA 0

ALUMNO:

Daniel Monroy Garnica



1. Introducción

En esta practica haciendo uso de la biblioteca digital y otras fuentes de información, realizamos una investigación para poder descubrir el uso de diferentes técnicas para el uso practico del uso de los datos dentro de la base de datos y asimilar una forma parecida a la de programar en otro tipo de lenguaje.

2. Marco teórico

Procedimientos almacenados

Es un conjunto de comandos SQL que pueden guardarse en el servidor. Una vez que se hace, los clientes no necesitan lanzar cada comando individual, sino que pueden en su lugar llamar al procedimiento almacenado como un único comando

Funciones

Es un conjunto de instrucciones SQL que se almacena asociado a una base de datos. Es un objeto que se crea con la sentencia `CREATE FUNCTION` y se invoca con la sentencia `SELECT` o dentro de una expresión. Una función puede tener cero o muchos parámetros de entrada y siempre devuelve un valor, asociado al nombre de la función.

Estructuras de control condicionales y repetitivas

Las estructuras de control permiten, como su nombre lo indica, controlar el flujo de las instrucciones dentro de un procedimiento o una función. Algunas de las estructuras pueden llevar una etiqueta (`BEGIN`, `LOOP`, `REPEAT` y `WHILE`).

Disparadores

Los triggers o disparadores son objetos que se asocian con tablas y se almacenan en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (`INSERT`), borrado (`DELETE`) o actualización (`UPDATE`), ya que modifican los datos de una tabla.

3. Herramientas empleadas

1. MySQL Workbench es una herramienta visual y un entorno integrado de desarrollo diseñado para trabajar con bases de datos MySQL. Desarrollado por Oracle, este software se ha convertido en la elección preferida de desarrolladores y administradores de bases de datos gracias a su interfaz intuitiva y sus robustas funcionalidades.

4. Desarrollo

Procedimientos almacenados

1. En este primero se obtiene los vehículos que están en mantenimiento y luego usamos CALL para llamar al procedimiento

```
1 DELIMITER $$
2 CREATE PROCEDURE ObtenerVehiculosEnMantenimiento()
3 BEGIN
4     SELECT id_Vehiculo, Marca, Modelo, Estado, Ult_dia_Uso
5     FROM Auto
6     WHERE Estado = 'En_mantenimiento';
7 END$$
8 DELIMITER ;
9
10 CALL ObtenerVehiculosEnMantenimiento();
```

Como resultado obtenemos los datos de los vehículos en mantenimiento

	id_Vehiculo	Marca	Modelo	Estado	Ult_dia_Uso
▶	1	Nissan	Versa 2022	En mantenimiento	2025-02-14
	3	Toyota	RAV4 HEV	En mantenimiento	2025-01-22

2. Aquí usaremos un procedimiento para poder realizar el registro de un mantenimiento nuevo

```
1 DELIMITER $$
2 CREATE PROCEDURE RegistrarMantenimiento(
3     IN mantenimiento INT,
4     IN vehiculo_id INT,
5     IN fecha DATE,
6     IN tipo VARCHAR(80),
7     IN costo FLOAT
8 )
9 BEGIN
10     INSERT INTO Mantenimiento (id_mantenimiento, fecha_Mantenimiento,
11     tipo_Mantenimiento, costo_Total, id_Vehiculo)
12     VALUES (mantenimiento, fecha, tipo, costo, vehiculo_id);
13 END$$
14 DELIMITER ;
15
16 CALL RegistrarMantenimiento(10,3, '2025-02-22', 'Parabrisas', 4000.00);
```

Como resultado vemos como se inserto en la BD.

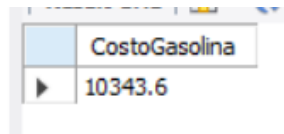
```
✔ 78 09:39:35 CALL RegistrarMantenimiento(10,3, '2025-02-22', 'Parabrisas', 4000.00)
```

Funciones

1. Ahora se va usar una función para sumar todos los gastos de gasolina y dar un costo total de toda la gasolina

```
1 DELIMITER $$
2 CREATE FUNCTION ObtenerCostoGasolina(id_vehiculo INT)
3 RETURNS FLOAT
4 DETERMINISTIC
5 BEGIN
6     DECLARE costo_total FLOAT;
7
8     SELECT SUM(Costo)
9     INTO costo_total
10    FROM Gasolina
11   WHERE id_Vehiculo = id_vehiculo;
12
13    RETURN costo_total;
14 END$$
15 DELIMITER ;
16
17 SELECT ObtenerCostoGasolina(1) AS CostoGasolina;
```

Como resultado vemos la suma de todo lo que van a gastar los autos solo por gasolina.



CostoGasolina
10343.6

2. En esta función va a decir cuál es el precio más alto de un mantenimiento

```
1 DELIMITER $$
2 CREATE FUNCTION ObtenerMantenimientoMasCaro(id_vehiculo INT)
3 RETURNS FLOAT
4 DETERMINISTIC
5 BEGIN
6     DECLARE costo_maximo FLOAT;
7
8     SELECT MAX(costo_Total)
9     INTO costo_maximo
10    FROM Mantenimiento
11   WHERE id_Vehiculo = id_vehiculo;
12
13    RETURN costo_maximo;
14 END$$
15 DELIMITER ;
```

Como resultado vemos cuál fue el precio del mantenimiento más caro.

Result Grid	Filter
MantenimientoCaro	
7000	

Estructuras de control condicionales y repetitivas

1. En la siguiente vamos a ver con un IF cuales son los carros que usan mas y menos de 30 litros de gasolina

```

1 SELECT
2     gasolina.id_Vehiculo ,
3     Marca ,
4     Modelo ,
5     litros_Consumidos ,
6     tipo_Gasolina ,
7     IF(litros_Consumidos > 30, 'El vehículo usa mucha gasolina',
8     'El vehículo usa poca gasolina') AS Estado_Gasolina
9 FROM Gasolina
10 JOIN Auto ON Gasolina.id_Vehiculo = Auto.id_Vehiculo;

```

Como resultado vemos que autos usan poca o mucha gasolina.

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	id_Vehiculo	Marca	Modelo	litros_Consumidos	tipo_Gasolina	Estado_Gasolina
▶	1	Nissan	Versa 2022	50	Magma	El vehículo usa mucha gasolina
	2	Volkswagen	Nivus 2024	35	Premier	El vehículo usa mucha gasolina
	3	Toyota	RAV4 HEV	25	Magma	El vehículo usa poca gasolina
	4	Chevrolet	Bright drop 2025	15	Premier	El vehículo usa poca gasolina
	5	KIA	Picanto	35	Magma	El vehículo usa mucha gasolina

2. En este caso usaremos un LOOP y un IF para ir sumando los autos que están en mantenimiento.

```

1 DELIMITER $$
2 CREATE PROCEDURE ContarVehiculosEnMantenimiento()
3 BEGIN
4     DECLARE vehiculo_id INT;
5     DECLARE total_en_mantenimiento INT DEFAULT 0;
6     DECLARE done INT DEFAULT FALSE;
7     DECLARE vehiculo_cursor CURSOR FOR
8         SELECT id_Vehiculo FROM Auto WHERE Estado = 'En mantenimiento';
9     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
10    OPEN vehiculo_cursor;
11    read_loop: LOOP
12        FETCH vehiculo_cursor INTO vehiculo_id;
13        IF done THEN
14            LEAVE read_loop;
15        END IF;
16        SET total_en_mantenimiento = total_en_mantenimiento + 1;
17    END LOOP;
18    CLOSE vehiculo_cursor;
19    SELECT total_en_mantenimiento AS VehiculosEnMantenimiento;
20 END$$

```

```

21 DELIMITER ;
22
23 CALL ContarVehiculosEnMantenimiento();

```

Como resultado vemos cuantos autos están en mantenimiento.

Result Grid Filter Rows:	
	VehiculosEnMantenimiento
▶	2

Disparadores

1. Con este Trigger lo que se busca es que si al actualizar un documento este se le da una fecha que ya paso, dará un mensaje de error

```

1 DELIMITER $$
2 CREATE TRIGGER verificar_fecha_documento
3 BEFORE UPDATE ON Documento
4 FOR EACH ROW
5 BEGIN
6     IF NEW.fecha_Renovacion < CURDATE() THEN
7         SIGNAL SQLSTATE '45000'
8         SET MESSAGE_TEXT = 'La fecha de renovacion ya ha pasado';
9     END IF;
10 END $$
11 DELIMITER ;
12
13 update documento set fecha_renovacion = '2025-02-25' where id_Documento = 1;

```

Como resultado vemos que no se ha actualizado porque la fecha ya paso.

82 09:39:35 update documento set fecha_renovacion = '2025-02-25' where id_Documento = 1 Error Code: 1644. La fecha de renovación ya ha pasado

2. Con este Trigger se busca que en caso de insertar un nuevo registro de un auto, no se pueda dar que su ultimo día de uso es un día que aun no ocurre.

```

1 DELIMITER $$
2 CREATE TRIGGER VerificarFechaUltimoUso
3 BEFORE INSERT ON Auto
4 FOR EACH ROW
5 BEGIN
6     IF NEW.Ult_dia_Uso > CURDATE() THEN
7         SIGNAL SQLSTATE '45000'
8         SET MESSAGE_TEXT = 'La fecha del ltimo uso no puede ser en el futuro';
9     END IF;
10 END$$
11 DELIMITER ;
12
13 insert into auto values (9,'Nissan', 'Versa_2022','En_mantenimiento','2025-03-01')

```

Como resultado vemos que no se inserto el registro ya que la fecha aun no pasa.

5. Conclusiones

Con esta practica aprendí que hay varias formas de poder manejar los datos en una base de datos tanto para cuando quiero encontrar algo en especifico como para cuando quiero poner restricciones sobre lo que quiero buscar, actualizar o insertar.

Referencias Bibliográficas

References

- [1] Giménez, J. A. (2019). B Buenas prácticas en el diseño de bases de datos. *ARANDU UTIC*, 6(1), 193-210.
- [2] MySQL Basico. (s.f.). IES VIRGEN DEL ESPINO. <http://www.v-espino.com/chema/-daw1/tutoriales/mysql.pdf>