REPORTE DE PRÁCTICA NO. 1.5.

Práctica 0

ALUMNO: Armando Oswaldo Pérez Romero
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En esta práctica, nos enfocamos en la investigación y obtención de fuentes bibliográficas relacionadas con conceptos avanzados en el manejo de bases de datos, tales como procedimientos almacenados, funciones, estructuras de control condicionales y repetitivas, y disparadores (triggers). Estos conceptos son fundamentales para comprender y aplicar técnicas de programación dentro de los sistemas de gestión de bases de datos (SGBD), ya que permiten optimizar, automatizar y garantizar la integridad de los datos en entornos empresariales y aplicaciones complejas.

El objetivo principal de esta investigación es profundizar en el entendimiento teórico de estos temas, respaldado por fuentes bibliográficas confiables y actualizadas. Una vez consolidado el marco teórico, se procederá a la aplicación práctica de estos conceptos en un contexto real, utilizando como base la base de datos de Gestión de Flotillas de Automóviles que fue desarrollada previamente.

2. Marco Teórico

SQL

El lenguaje de consulta estructurado (SQL, por sus siglas en inglés) es un lenguaje estándar empleado para administrar y manejar bases de datos relacionales. Mediante SQL, los usuarios tienen la posibilidad de hacer varias operaciones en las bases de datos, tales como añadir, actualizar, eliminar y consultar información.

Este lenguaje es fundamental para interactuar con sistemas de gestión de bases de datos (SGBD) relacionales, como MySQL, PostgreSQL, SQL Server y Oracle, entre otros.

Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual.

Procedimientos almacenados (Procedure)

Los procedimientos almacenados son bloques de código SQL que se almacenan en la base de datos y se ejecutan en el servidor. Estos procedimientos permiten encapsular lógica de negocio, mejorar el rendimiento y reutilizar código. Se pueden invocar desde aplicaciones externas o desde otros procedimientos almacenados.

Los procedimientos almacenados son especialmente útiles para realizar operaciones complejas que involucran múltiples consultas y transacciones, ya que reducen la sobrecarga de red y mejoran la seguridad al limitar el acceso directo a las tablas.

Celko, J. (2005). SQL for Smarties: Advanced SQL Programming. Morgan Kaufmann.

Funciones (Function)

Las funciones en SQL son objetos que permiten realizar cálculos o manipulaciones de datos y devolver un valor único. A diferencia de los procedimientos almacenados, las funciones deben retornar un valor y pueden ser utilizadas en expresiones SQL, como en cláusulas SELECT, WHERE o ORDER BY.

Existen dos tipos principales de funciones: las funciones escalares, que devuelven un único valor, y las funciones de tabla, que retornan un conjunto de filas. Las funciones son esenciales para modularizar el código y simplificar consultas complejas.

Gulutzan, P., Pelzer, T. (2002). SQL-99 Complete, Really. CMP Books.

Estructuras de control condicionales y repetitivas

Las estructuras de control en SQL permiten manejar el flujo de ejecución de los procedimientos almacenados y funciones. Las estructuras condicionales, como IF-ELSE y CASE, permiten ejecutar bloques de código dependiendo de ciertas condiciones. Por otro lado, las estructuras repetitivas, como WHILE y FOR, permiten ejecutar un bloque de código múltiples veces hasta que se cumpla una condición específica.

Estas estructuras son fundamentales para implementar lógica de negocio compleja directamente en la base de datos, lo que mejora la eficiencia y reduce la necesidad de procesamiento en la capa de aplicación.

Ben-Gan, I. (2016). T-SQL Fundamentals. Microsoft Press.

Disparadores (Triggers)

Los disparadores, o triggers, son objetos de base de datos que se ejecutan automáticamente en respuesta a eventos específicos, como inserciones, actualizaciones o eliminaciones en una tabla. Los triggers se utilizan para imponer reglas de negocio, mantener la integridad de los datos y auditar cambios en la base de datos.

Existen dos tipos principales de triggers: los triggers de fila, que se ejecutan por cada fila afectada, y los triggers de sentencia, que se ejecutan una vez por operación. Su uso debe ser cuidadoso, ya que pueden afectar el rendimiento de la base de datos si no se diseñan adecuadamente.

Mullins, C. S. (2002). Database Administration: The Complete Guide to Practices and Procedures. Addison-Wesley.

3. Herramientas Utilizadas

1. Navegador. Permite el acceso a la Web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser vistos.

2. Overleaf (Editor LaTeX Online). Es un editor de texto colaborativo en línea para crear y editar documentos en LaTeX, un sistema de composición de textos ampliamente utilizado en la academia

3. MySQL Workbench. es una herramienta visual de diseño y administración de bases de datos desarrollada por Oracle. Está específicamente diseñada para trabajar con bases de datos MySQL.

4. Ejemplos

1. Este procedimiento muestra los autos que están activos (estado = 1).

```
DELIMITER //
CREATE PROCEDURE MostrarAutosActivos()
BEGIN
    SELECT idAuto, Marca, Modelo
    FROM Automovil
    WHERE estado = 1;
END //
DELIMITER ;
```

Uso:

```
CALL MostrarAutosActivos();
```

idAuto	Marca	Modelo
1	Toyota	Corolla
2	Ford	Focus
3	Honda	Civic
4	Chevrolet	Spark
5	Nissan	Sentra

2. Este procedimiento inserta un nuevo automóvil en la tabla Automovil.

```
DELIMITER //
CREATE PROCEDURE InsertarAutomovil(
    IN marca VARCHAR(50),
    IN modelo VARCHAR(50),
    IN estado INT,
    IN idDocumentacion INT
)
BEGIN
    INSERT INTO Automovil (Marca, Modelo, estado, idDocumentacion)
    VALUES (marca, modelo, estado, idDocumentacion);
END //
DELIMITER ;
```

Uso:

```
CALL InsertarAutomovil('Toyota', 'Corolla', 1, 1);
```

idAuto	placa	Marca	Modelo	estado	idDocumentacion
1	ABC-1234	Toyota	Corolla	1	1
2	XYZ-5678	Ford	Focus	1	2
3	LMN-9101	Honda	Civic	1	3
4	JKL-2345	Chevrolet	Spark	1	4
5	DEF-6789	Nissan	Sentra	1	5

3. Esta función cuenta cuántos mantenimientos ha tenido un automóvil específico.

```
DELIMITER //
CREATE FUNCTION ContarMantenimientos(auto_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total
    FROM Mantenimiento
    WHERE idAuto = auto_id;
    RETURN total;
END //
DELIMITER ;
```

Uso:

```
SELECT ContarMantenimientos(1);
```

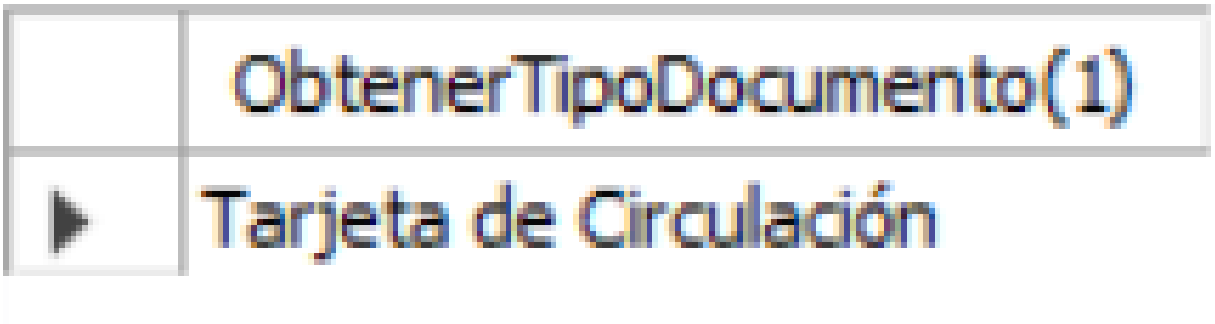
ContarMantenimientos(1)
1

4. Esta función devuelve el tipo de documento asociado a un automóvil.

```
DELIMITER //
CREATE FUNCTION ObtenerTipoDocumento(auto_id INT)
RETURNS VARCHAR(80)
DETERMINISTIC
BEGIN
    DECLARE tipo VARCHAR(80);
    SELECT tipoDocumento INTO tipo
    FROM Documentacion
    WHERE idDocumentacion = (SELECT idDocumentacion FROM Automovil WHERE idAuto
        = auto_id);
    RETURN tipo;
END //
DELIMITER ;
```

Uso:

```
   SELECT ObtenerTipoDocumento(1);
```



5. Este bloque de código verifica si un automóvil necesita mantenimiento basado en su estado.

```
DELIMITER //
CREATE PROCEDURE VerificarMantenimiento(auto_id INT)
BEGIN
    DECLARE estadoAuto INT;
    SELECT estado INTO estadoAuto
    FROM Automovil
    WHERE idAuto = auto_id;

    IF estadoAuto = 0 THEN
        SELECT 'El autom vil necesita mantenimiento' AS Mensaje;
    ELSE
        SELECT 'El autom vil  est  en buen estado' AS Mensaje;
    END IF;
END //
DELIMITER ;
```

Uso:

```
CALL VerificarMantenimiento(1);
```

	Mensaje
▶	El automóvil está en buen estado

6. Este bloque de código muestra todos los mantenimientos de un automóvil específico.

```
DELIMITER //
CREATE PROCEDURE MostrarMantenimientos(auto_id INT)
BEGIN
    SELECT idMantenimiento, fechaMantenimiento, tipoMantenimiento, costoTotal
    FROM Mantenimiento
    WHERE idAuto = auto_id;
END //
DELIMITER ;
```

Uso:

```
CALL MostrarMantenimientos(1);
```

	idMantenimiento	fechaMantenimiento	tipoMantenimiento	costoTotal
▶	1	2025-01-10	Cambio de Aceite	1500

7. Este trigger evita que se inserte un mantenimiento si el costo es negativo.

```
1
2           DELIMITER //
3           CREATE TRIGGER DenegarMantenimientoNegativo
4           BEFORE INSERT ON Mantenimiento
5           FOR EACH ROW
6           BEGIN
7               IF NEW.costoTotal < 0 THEN
8                   SIGNAL SQLSTATE '45000'
9                   SET MESSAGE_TEXT = 'El costo del mantenimiento no puede ser negativo';
10              END IF;
11          END //
12          DELIMITER ;
```

8. Este trigger evita que se actualice el estado de un automóvil si el nuevo estado no es 0 o 1.

```
1
2               DELIMITER //
3               CREATE TRIGGER DenegarEstadoInvalido
4               BEFORE UPDATE ON Automovil
5               FOR EACH ROW
6               BEGIN
7                   IF NEW.estado NOT IN (0, 1) THEN
8                       SIGNAL SQLSTATE '45000'
9                       SET MESSAGE_TEXT = 'El estado del autom vil debe ser 0 (inactivo)
                            o 1 (activo)';
10                  END IF;
11              END //
12              DELIMITER ;
```

```
94 •    SHOW TRIGGERS;
95
```

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

Trigger	Event	Table	Statement	Timing	Created
DenegarEstadoInvalido	UPDATE	automovil	BEGIN IF NEW.estado NOT IN (0, 1) THEN ...	BEFORE	2025-02-27 11:27:49.45
DenegarMantenimientoNegativo	INSERT	mantenimiento	BEGIN IF NEW.costoTotal < 0 THEN SIG...	BEFORE	2025-02-27 11:26:22.79

5. Conclusiones

En esta práctica, se exploraron y aplicaron conceptos avanzados en el manejo de bases de datos, específicamente procedimientos almacenados, funciones, estructuras de control condicionales y repetitivas, y disparadores (triggers). Estos temas son fundamentales para optimizar, automatizar y garantizar la integridad de los datos en un sistema de gestión de bases de datos relacionales.

A través de la implementación de ejemplos prácticos en la base de datos de Gestión de Flotillas de Automóviles, se logró comprender cómo estos elementos interactúan y se complementan para resolver problemas comunes en la administración de datos. Los conceptos aprendidos son aplicables en entornos reales, donde la automatización y la validación de datos son esenciales para garantizar la confiabilidad de los sistemas de información.
En conclusión, el dominio de estos temas es crucial para cualquier profesional que trabaje con bases de datos, ya que proporciona las herramientas necesarias para optimizar procesos, mejorar el rendimiento y mantener la consistencia de los datos en aplicaciones empresariales y sistemas complejos.

Referencias Bibliográficas

References

[1] Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual.

[2] Celko, J. (2005). SQL for Smarties: Advanced SQL Programming. Morgan Kaufmann.

[3] Gulutzan, P., Pelzer, T. (2002). SQL-99 Complete, Really. CMP Books.

[4] Mullins, C. S. (2002). Database Administration: The Complete Guide to Practices and Procedures. Addison-Wesley.