

INVESTIGAR QUE SON Procedimientos Almacenados en Bases de Datos

- Entender qué son los procedimientos almacenados y cómo funcionan.
- Aprender a crear procedimientos almacenados sencillos.
- PRACTICA - Realizar operaciones de **INSERT**, **SELECT**, **DELETE** y **UPDATE** usando procedimientos almacenados.
- **Revisión de Buenas Prácticas**

Introducción a los Procedimientos Almacenados **MSQL- PostgreSQL – Sql Server**

1. Concepto y Beneficios de los Procedimientos Almacenados

- **Explicación:** Los procedimientos almacenados son conjuntos de instrucciones SQL que se guardan y ejecutan en el servidor de base de datos. Permiten ejecutar operaciones complejas, con seguridad, rendimiento optimizado y reutilización de código.
- **Beneficios:**
 - Reutilización de código.
 - Mejora en la seguridad (al evitar inyecciones SQL).
 - Optimización en el rendimiento de consultas frecuentes.
 - Consistencia en las operaciones realizadas.

2. ESPECIFICAR LA Sintaxis Básica de un Procedimiento Almacenado

- **Explicación:** El delimitador se cambia temporalmente para permitir el uso de **;** dentro del procedimiento.

Crear la tabla de cliente:

```
CREATE TABLE cliente (  
    ClienteID INT AUTO_INCREMENT PRIMARY KEY, -- Campo para el ID único del cliente  
    Nombre VARCHAR(100), -- Campo para el nombre del cliente  
    Estatura DECIMAL(5,2), -- Campo para la estatura del cliente con dos decimales  
    FechaNacimiento DATE, -- Campo para la fecha de nacimiento del cliente  
    Sueldo DECIMAL(10,2) -- Campo para el sueldo del cliente con dos decimales  
);
```

Deber Procedimientos.sql*

```
1 CREATE TABLE cliente (  
2     ClienteID INTEGER PRIMARY KEY AUTOINCREMENT, -- Campo para el ID único del cliente  
3     Nombre VARCHAR(100), -- Campo para el nombre del cliente  
4     Estatura DECIMAL(5,2), -- Campo para la estatura del cliente con dos decim  
5     FechaNacimiento DATE, -- Campo para la fecha de nacimiento del cliente  
6     Sueldo DECIMAL(10,2) -- Campo para el sueldo del cliente con dos decimal  
7 );  
8 INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo)  
9 VALUES  
10 ('Juan Pérez', 1.75, '1990-05-10', 2500.50),  
11 ('María Gómez', 1.60, '1985-11-25', 3000.75),  
12 ('Luis Rodríguez', 1.80, '1992-08-15', 2800.00),  
13 ('Ana Fernández', 1.68, '1988-03-30', 3200.20),  
14 ('Carlos Martínez', 1.90, '1995-12-05', 2600.00);  
15
```

Execution finished without errors.
Result: query executed successfully. Took 0ms, 5 rows affected
At line 9:
INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo)

3. Ejercicio 1: Crear un procedimiento simple que seleccione datos de la tabla cliente

```
15 SELECT * FROM cliente;  
16 SELECT Nombre, Sueldo FROM cliente WHERE Sueldo > 3000;
```

	ClienteID	Nombre	Estatura	FechaNacimiento	Sueldo
1	1	Juan Pérez	1.75	1990-05-10	2500.5
2	2	María Gómez	1.6	1985-11-25	3000.75
3	3	Luis Rodríguez	1.8	1992-08-15	2800
4	4	Ana Fernández	1.68	1988-03-30	3200.2
5	5	Carlos Martínez	1.9	1995-12-05	2600

Execution finished without errors.
Result: 5 rows returned in 32ms
At line 15:
SELECT * FROM cliente;

```

15 SELECT * FROM cliente,
16 SELECT Nombre, Sueldo FROM cliente WHERE Sueldo>3000;

```

	Nombre	Sueldo
1	María Gómez	3000.75
2	Ana Fernández	3200.2

```

Execution finished without errors.
Result: 2 rows returned in 18ms
At line 16:
SELECT Nombre, Sueldo FROM cliente WHERE Sueldo>3000;

```

4. **Ejercicio:** Ejecutar - LLAMAR el procedimiento

Inserción, Actualización y Eliminación de Datos

1. Procedimiento de Inserción (INSERT)

- Crear un procedimiento que permita insertar un nuevo cliente en la tabla cliente

- Ejecutar - LLAMAR el procedimiento

```

Deber Procedimientos.sql*
7  -- SQLite nl admite procedimientos almacenados por ellos solo se mostrara el script
8  DELIMITER $$
9  -- Creamos el procedimiento
10 CREATE PROCEDURE InsertarCliente (
11     IN p_Nombre VARCHAR(100),
12     IN p_Estatura DECIMAL(5,2),
13     IN p_FechaNacimiento DATE,
14     IN p_Sueldo DECIMAL(10,2)
15 )
16 BEGIN
17     INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo)
18     VALUES (p_Nombre, p_Estatura, p_FechaNacimiento, p_Sueldo);
19 END$$
20 DELIMITER ;
21 -- Llamar al procedimiento
22 CALL InsertarCliente('Pedro López', 1.72, '1990-07-15', 2700.00);

```

2. Procedimiento de Actualización (UPDATE)

Actualizar la edad de un cliente específico:

```

DELIMITER $$
CREATE PROCEDURE ActualizarEdad(
    IN p_ClienteID INT,
    IN p_NEdad TINYINT
)
BEGIN
    UPDATE cliente
    SET Edad = p_NEdad
    WHERE ClienteID = p_ClienteID;
END$$
DELIMITER ;

CALL ActualizarEdad(1, 35);

```

3. Procedimiento de Eliminación (DELETE)

Eliminar un cliente de la base de datos usando su ClienteID:

```

CALL ActualizarEdad(1, 35);

-- Eliminar el cliente por su ID
DELIMITER $$
CREATE PROCEDURE EliminarCliente (
    IN p_ClienteID INT
)
BEGIN
    DELETE FROM cliente
    WHERE ClienteID = p_ClienteID;
END$$
DELIMITER ;

```

Introducción a Condiciones en Procedimientos Almacenados

Uso de Condicionales (IF)

El uso de condicionales dentro de los procedimientos es fundamental para tomar decisiones basadas en los datos.

Verifica si la edad de un cliente es mayor o igual a 22:

```

-- Condicional IF
DELIMITER $$

]CREATE PROCEDURE VerificarEdadCliente (
    IN p_ClienteID INT
)
]BEGIN
    DECLARE edad_cliente INT;
    SELECT Edad INTO edad_cliente
    FROM cliente
    WHERE ClienteID=p_ClienteID;
] IF edad_cliente>=22 THEN
    SELECT CONCAT('Cliente ID ', p_ClienteID, ' tiene una edad mayor o igual a 22: ', edad_cliente) AS Resultado;
ELSE
    SELECT CONCAT('Cliente ID ', p_ClienteID, ' tiene una edad menor a 22: ', edad_cliente) AS Resultado;
END IF;
END$$
DELIMITER ;

```

Creación de la Tabla de Órdenes CON RELACIÓN CON EL CLIENTE - FORANEA

Para almacenar las órdenes de los clientes, se debe crear la tabla **ordenes**:

- Procedimientos de Órdenes -Insertar Orden

```

-- Insertar Orden
DELIMITER $$

]CREATE PROCEDURE InsertarOrden (
    IN p_Peticion VARCHAR(255),
    IN p_ClienteID INT
)
]BEGIN
    INSERT INTO ordenes (Peticion, ClienteID)
    VALUES (p_Peticion, p_ClienteID);
END$$
DELIMITER ;
CALL InsertarOrden('Entrega de mercancia', 1);

```

- Procedimientos Actualizar Orden

```

-- Actualizar Orden
DELIMITER $$
CREATE PROCEDURE ActualizarOrden (
    IN p_OrdenID INT,
    IN p_NuevaPetición VARCHAR(255),
    IN p_NuevoClienteID INT
)
BEGIN
    UPDATE ordenes
    SET Petición = p_NuevaPetición,
        ClienteID = p_NuevoClienteID
    WHERE OrdenID = p_OrdenID;
END$$
DELIMITER ;
CALL ActualizarOrden(1, 'Devolucion de dinero', 2);

```

■ Procedimientos Eliminar Orden

```

-- Eliminar orden por su id
DELIMITER $$
CREATE PROCEDURE EliminarOrden(
    IN p_OrdenID INT
)
BEGIN
    DELETE FROM ordenes
    WHERE OrdenID = p_OrdenID;
END$$
DELIMITER ;
CALL EliminarOrden(4);

```

Entrega Final

Instrucciones de Entrega:

1. Objetivos:

Crear procedimientos almacenados para **insertar, actualizar, eliminar y consultar** registros en las tablas cliente y órdenes.

2. Archivo de Script:

Los estudiantes deben escribir y guardar el código SQL con todos los procedimientos mencionados.

3. Documento PDF:

Incluir las capturas de pantalla y explicaciones detalladas de los pasos realizados durante la tarea.

4. **Subida a GitHub:**

Subir el script .sql y el documento PDF a un repositorio en GitHub para su REVISIÓN