ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

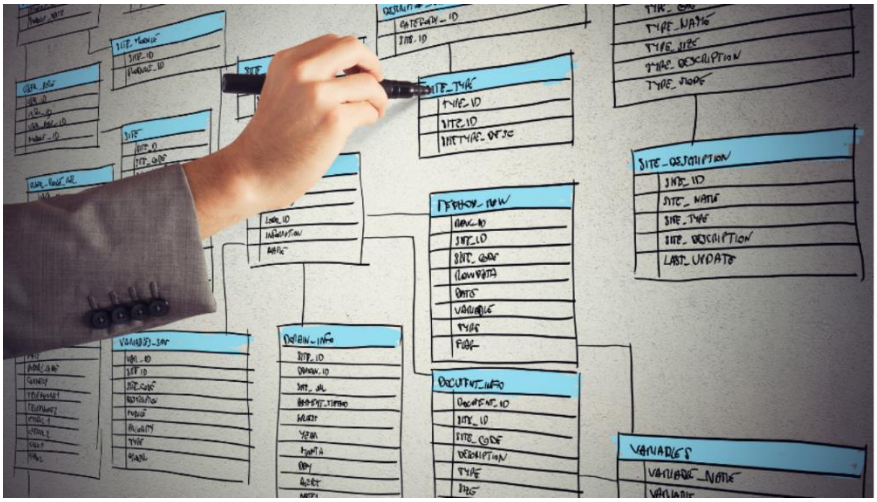
BASE DE DATOS

PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2024-B

TAREA

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

Cristian Tambaco

2024-B

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

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.

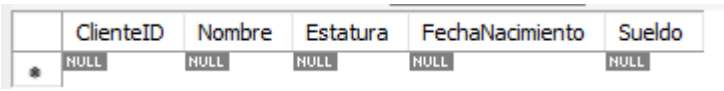
Crear la tabla de cliente:

```
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
);
```

Table: cliente

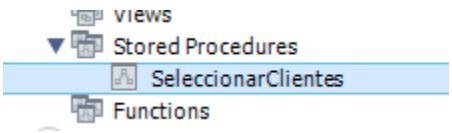
Columns:

ClienteID	int AI PK
Nombre	varchar(100)
Estatura	decimal(5,2)
FechaNacimiento	date
Sueldo	decimal(10,2)



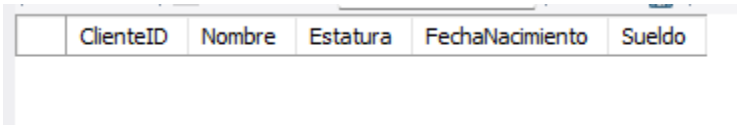
3. **Ejercicio 1: Crear un procedimiento simple que seleccione datos de la tabla cliente**

```
DELIMITER $$

CREATE PROCEDURE SeleccionarClientes()

BEGIN

   SELECT * FROM cliente;

END $$

DELIMITER ;
```



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

```
CALL SeleccionarClientes();
```

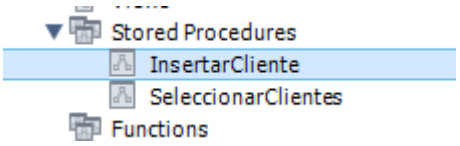


Inserción, Actualización y Eliminación de Datos

1. Procedimiento de Inserción (INSERT)

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

```
DELIMITER $$
CREATE PROCEDURE InsertarCliente(
   IN p_Nombre VARCHAR(100),
   IN p_Estatura DECIMAL(5,2),
   IN p_FechaNacimiento DATE,
   IN p_Sueldo DECIMAL(10,2)
)
BEGIN
   INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo)
   VALUES (p_Nombre, p_Estatura, p_FechaNacimiento, p_Sueldo);
END$$
DELIMITER ;
```



- Ejecutar - LLAMAR el procedimiento

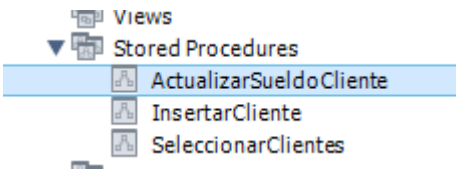
```
CALL InsertarCliente('Alexander', 1.68, '1983-02-14', 2200.50);
```

ClienteID	Nombre	Estatura	FechaNacimiento	Sueldo
1	Alexander	1.68	1983-02-14	2200.50
NULL	NULL	NULL	NULL	NULL

2. Procedimiento de Actualización (UPDATE)

Actualizar la edad de un cliente específico:

```
DELIMITER $$
CREATE PROCEDURE ActualizarSueldoCliente(
   IN p_ClienteID INT,
   IN p_NuevoSueldo DECIMAL(10,2)
)
BEGIN
   UPDATE cliente
   SET Sueldo = p_NuevoSueldo
   WHERE ClienteID = p_ClienteID;
END$$
DELIMITER ;
```




```
-- - Ejecutar - LLAMAR el procedimiento

CALL ActualizarSueldoCliente(1, 5000.00);
```

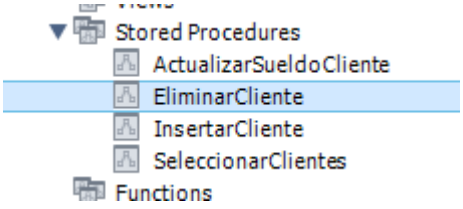
ClienteID	Nombre	Estatura	FechaNacimiento	Sueldo
1	Alexander	1.68	1983-02-14	5000.00
NULL	NULL	NULL	NULL	NULL

3. Procedimiento de Eliminación (DELETE)

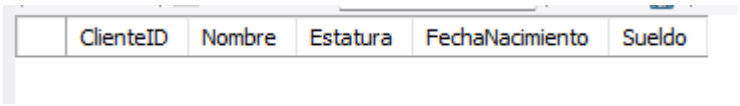
Eliminar un cliente de la base de datos usando su ClienteID:

```
DELIMITER $$
```

```
CREATE PROCEDURE EliminarCliente(
   IN p_ClienteID INT
)
BEGIN
   DELETE FROM cliente
   WHERE ClienteID = p_ClienteID;
END$$
DELIMITER ;
```




```
-- - Ejecutar - LLAMAR el procedimiento

CALL EliminarCliente(1);

-- - Ejecutar - LLAMAR el procedimiento
CALL SeleccionarClientes();
```



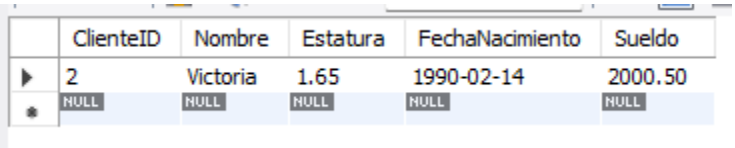
Introducción a Condiciones en Procedimientos Almacenados

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:

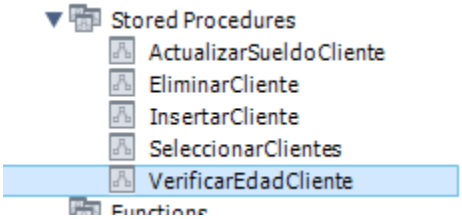
```
-- - Ejecutar - LLAMAR el procedimiento
CALL InsertarCliente('Victoria', 1.65, '1990-02-14', 2000.50);
```



ClienteID	Nombre	Estatura	FechaNacimiento	Sueldo
2	Victoria	1.65	1990-02-14	2000.50
NULL	NULL	NULL	NULL	NULL

```
-- Procedimiento de verificación
DELIMITER $$
CREATE PROCEDURE VerificarEdadCliente(
    IN p_ClienteID INT
)
BEGIN
    DECLARE v_Edad INT;
    DECLARE v_FechaNacimiento DATE;

    -- Obtener la fecha de nacimiento del cliente
    SELECT FechaNacimiento INTO v_FechaNacimiento
    FROM cliente
    WHERE ClienteID = p_ClienteID;

    -- Calcular la edad
    SET v_Edad = TIMESTAMPDIFF(YEAR, v_FechaNacimiento, CURDATE());

    -- Verificar si la edad es mayor o igual a 22
    IF v_Edad >= 22 THEN
        SELECT 'Es mayor a 22' AS Mensaje;
    ELSE
        SELECT 'Es menor a 22' AS Mensaje;
```

```
    END IF;
END$$
DELIMITER ;
```




```
-- - Ejecutar - LLAMAR el procedimiento
CALL VerificarEdadCliente(2);
```

	Mensaje
▶	Es mayor a 22

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**:

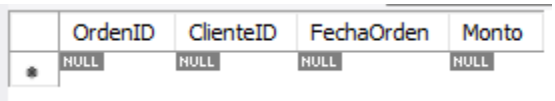
```
-- Crear la tabla ordenes
CREATE TABLE ordenes (
    OrdenID INT PRIMARY KEY AUTO_INCREMENT,
        ClienteID INT,
        FechaOrden DATE,
    Monto DECIMAL(10,2),
        FOREIGN KEY (ClienteID) REFERENCES cliente(ClienteID)
);
```

Table: ordenes

Columns:

OrdenID	int AI PK
FechaOrden	date
Monto	decimal(10,2)
ClienteID	int

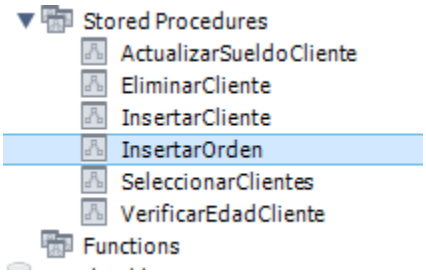
SELECT * FROM ordenes;



	OrdenID	ClienteID	FechaOrden	Monto
*	NULL	NULL	NULL	NULL

- Procedimientos de Órdenes -Insertar Orden

```
-- Procedimientos de Órdenes -Insertar Orden
DELIMITER $$
CREATE PROCEDURE InsertarOrden(
   IN p_ClienteID INT,
   IN p_FechaOrden DATE,
   IN p_Monto DECIMAL(10,2)
)
BEGIN
   INSERT INTO ordenes (ClienteID, FechaOrden, Monto)
   VALUES (p_ClienteID, p_FechaOrden, p_Monto);
END$$
DELIMITER ;
```

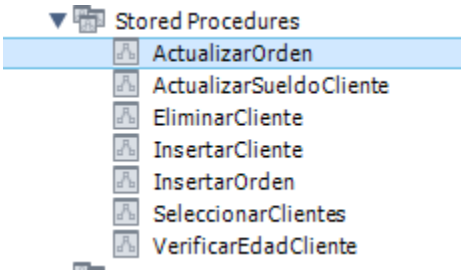



```
-- - Ejecutar - LLAMAR el procedimiento
CALL InsertarOrden(2, '2024-12-01', 115.00);
```

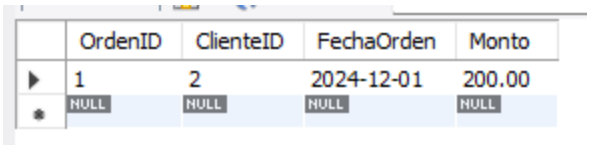
OrdenID	ClienteID	FechaOrden	Monto
1	2	2024-12-01	115.00
NULL	NULL	NULL	NULL

- Procedimientos Actualizar Orden

```
-- Procedimientos Actualizar Orden


DELIMITER $$
CREATE PROCEDURE ActualizarOrden(
   IN p_OrdenID INT,
   IN p_NuevoMonto DECIMAL(10,2)
)
BEGIN
   UPDATE ordenes
   SET Monto = p_NuevoMonto
   WHERE OrdenID = p_OrdenID;
END$$
DELIMITER ;
```



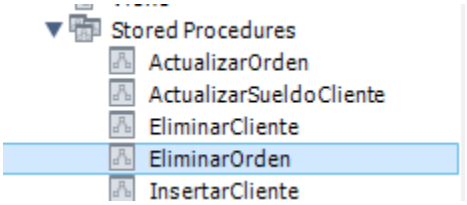

```
-- - Ejecutar - LLAMAR el procedimiento
CALL ActualizarOrden(1, 200.00);
```



OrdenID	ClienteID	FechaOrden	Monto
1	2	2024-12-01	200.00
NULL	NULL	NULL	NULL

- Procedimientos Eliminar Orden

```
DELIMITER $$
CREATE PROCEDURE EliminarOrden(
   IN p_OrdenID INT
)
BEGIN
   DELETE FROM ordenes
   WHERE OrdenID = p_OrdenID;
END$$
DELIMITER ;
```




```
-- - Ejecutar - LLAMAR el procedimiento
CALL EliminarOrden(1);
```

OrdenID	ClienteID	FechaOrden	Monto
NULL	NULL	NULL	NULL

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

Link:

https://github.com/CristianTambaco/Tarea_Procedimiento_Almacenado.git