# ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

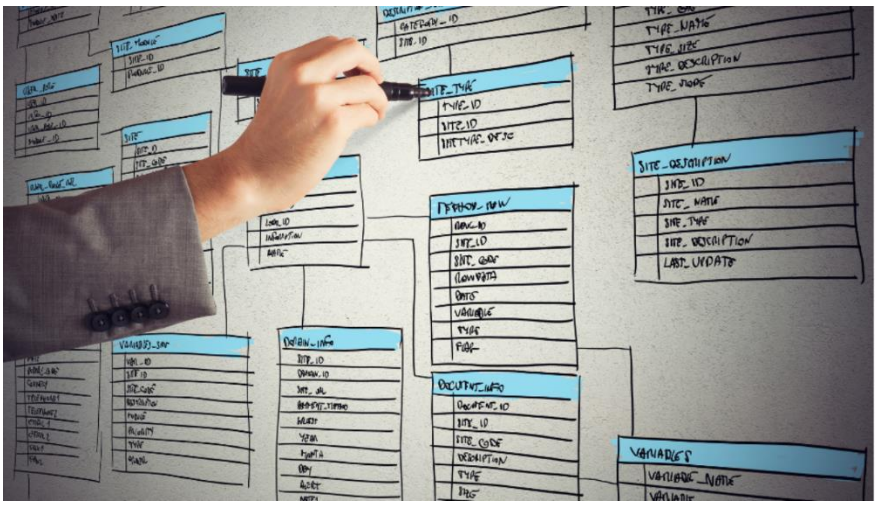
## BASE DE DATOS

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

## TAREA

# TÍTULO:

## INVESTIGACIÓN Y PRACTICA




Estudiante

Kevin Muñoz


## 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
);
```

```sql
CREATE TABLE cliente (
    ClienteID INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100),
    Estatura DECIMAL(5,2),
    FechaNacimiento DATE,
    Sueldo DECIMAL(10,2)    ,
    Edad INT
);
```

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

```sql
DELIMITER $$

CREATE PROCEDURE SeleccionarClientes()
BEGIN
    SELECT * FROM cliente;
END$$

DELIMITER ;
```

En cada procedimiento almacenado, el delimitador debe cambiarse, para este caso "$$", para evitar confusiones entre los delimitadores de las instrucciones en MySQL, se sabe que el ";" se usa para separar las instrucciones, pero dentro de un procedimiento almacenado, se pueden necesitar varios ";" para separar las diferentes sentencias que van a componer el cuerpo de ese procedimiento. Gracias a esto se puede incluir el ";" dentro del cuerpo sin finalizar la creación. Al final se restaura el delimitador predeterminado.

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

```sql
CALL SeleccionarClientes();
```

Result Grid | Filter Rows: | Export: | Wra

| ClienteID | Nombre | Estatura | FechaNacimiento | Sueldo |
|---|---|---|---|---|

Con CALL llamamos al procedimiento que se ha creado

## 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 InserciónCliente(
    IN c_Nombre VARCHAR(100),
    IN c_Estatura DECIMAL(5,2),
    IN c_FechaNacimiento DATE,
    IN c_Sueldo DECIMAL(10,2),
    IN c_Edad INT
)
BEGIN
    INSERT INTO cliente (Nombre, Estatura, FechaNacimiento, Sueldo, Edad)
    VALUES (c_Nombre, c_Estatura, c_FechaNacimiento, c_Sueldo, c_Edad);
END$$

DELIMITER ;
```

Se crea el procedimiento InserciónCliente, donde se toman 5 parámetros, e inserta los valores en “cliente” para crear un nuevo registro con los datos.

- Ejecutar - LLAMAR el procedimiento

```
CALL InserciónCliente('Pedro Pérez', 1.75, '1985-04-23', 4500.00, 25);
CALL InserciónCliente('Juan Gonzales', 1.80, '1999-03-20', 3500.00, 25);
```

Se insertan dos clientes, usando CALL, donde se pasarán los parámetros necesarios que fueron definidos cuando se creó el procedimiento.

### 2. Procedimiento de Actualización (UPDATE)

Actualizar la edad de un cliente específico:

```
DELIMITER $$

CREATE PROCEDURE ActualizarEdadCliente(
    IN c_ClienteID INT,
    IN c_NuevaEdad INT
)
BEGIN
    UPDATE cliente
    SET Edad = c_NuevaEdad
    WHERE ClienteID = c_ClienteID;
END$$

DELIMITER ;
```

Se crea un procedimiento para actualizar la edad de un cliente, dentro del cuerpo del cuerpo (BEGIN - END) se usan los comandos UPDATE, SET, WHERE, para indicar el proceso que se ejecutará al llamar al procedimiento

```
CALL ActualizarEdadCliente(1, 39);
```

Al llamar al procedimiento, dentro del paréntesis el primero número es el ID donde está el cliente a actualizar, el segundo número es la nueva edad.

**3. Procedimiento de Eliminación (DELETE)**

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

```
DELIMITER $$

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

DELIMITER ;
```

El procedimiento creado, define la eliminación de un cliente mediante su ID, donde el comando WHERE juega el papel importante ya que eliminará el cliente cuyo ClienteID coincida con el valor que se proporcione en c_ClienteID

```
CALL EliminarCliente(2);
```

Dentro del paréntesis se pone el ID del cliente a eliminar

## Introducción a Condiciones en Procedimientos Almacenados

**Uso de Condicionales (IF)**

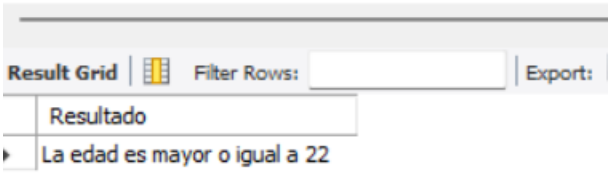
El uso de condicionales dentro de los procedimientos es fundamental para tomar decisiones basadas en los datos.

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

```
DELIMITER $$

CREATE PROCEDURE VerificarEdadCliente(
    IN c_ClienteID INT
)
BEGIN
    DECLARE v_Edad INT;

    SELECT Edad INTO v_Edad
    FROM cliente
    WHERE ClienteID = c_ClienteID;

    IF v_Edad >= 22 THEN
        SELECT 'La edad es mayor o igual a 22' AS Resultado;
    ELSE
        SELECT 'La edad es menor a 22' AS Resultado;
    END IF;
END $$

DELIMITER ;
```

Se crea el procedimiento VerificarEdadCliente para ver si la edad de un cliente es mayor o igual a 22 años, en la que se declara una variable v_Edad que almacenará la edad del cliente que se obtenga. Luego dentro del bloque SELECT se hace la consulta a la tabla cliente, donde buscará el valor de Edad para el ClienteID que coincida con el valor de c_ClienteID.

```
117     CALL VerificarEdadCliente(1);
118
```




Se ejecuta CALL, dentro del paréntesis se pondrá el ID del cliente que se desee buscar.

## 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**:

```
CREATE TABLE ordenes (
    OrdenID INT AUTO_INCREMENT PRIMARY KEY,
    ClienteID INT,
    FechaOrden DATE,
    MontoTotal DECIMAL(10,2),
    FOREIGN KEY (ClienteID) REFERENCES cliente(ClienteID)
);
```

- **Procedimientos de Órdenes -Insertar Orden**

```
DELIMITER $$

CREATE PROCEDURE InsertarOrden(
    IN o_ClienteID INT,
    IN o_FechaOrden DATE,
    IN o_MontoTotal DECIMAL(10,2)
)
BEGIN
    INSERT INTO ordenes (ClienteID, FechaOrden, MontoTotal)
    VALUES (o_ClienteID, o_FechaOrden, o_MontoTotal);
END$$

DELIMITER ;


CALL InsertarOrden(1, '2024-11-25', 700.00);
```

Se creó el procedimiento InsertarOrden para la inserción de registros, al momento de llamar al procedimiento se insertarán los valores que deseemos, los cuales corresponderán a los parámetros definidos en IN.

- **Procedimientos  Actualizar Orden**

```
DELIMITER $$

CREATE PROCEDURE ActualizarOrden(
    IN o_OrdenID INT,
    IN o_FechaOrden DATE,
    IN o_MontoTotal DECIMAL(10,2)
)
BEGIN
    UPDATE ordenes
    SET FechaOrden = o_FechaOrden, MontoTotal = o_MontoTotal
    WHERE OrdenID = o_OrdenID;
END$$

DELIMITER ;

CALL ActualizarOrden(1, '2024-12-22', 1000.00);
```

Con el procedimiento ActualizarOrden, se podrá actualizar un registro mediante la búsqueda de un ID, y se procederá a ingresar la nueva fecha y el nuevo monto.

- **Procedimientos Eliminar Orden**

```
DELIMITER $$

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

DELIMITER ;

CALL EliminarOrden(1);
```

Con el procedimiento de EliminarOrden se podrá eliminar un registro usando el ID correspondiente de la tabla ordenes.

## Entrega Final

**Instrucciones de Entrega:**

1. **Objetivos**:

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

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