

Actividad #3 – Taller 6

¿Qué es un procedimiento?

Es un conjunto de sentencias SQL, las cuales son agrupadas y almacenadas por un nombre particular en una BD relacional para ser usada y ejecutada en el momento que lo necesitemos donde su objetivo es realizar una tarea predeterminada como: consultas, insertar datos, actualizar, eliminar, hacer cálculos, entre otros.

Sintaxis de un procedimiento

```
CREATE PROCEDURE procedure_name
[ (parameter1 datatype [, parameter2 datatype, ...]) ]
BEGIN
    -- Cuerpo del procedimiento (aquí se incluye la lógica que se desea implementar)
END;
```

- Palabra clave **CREATE**: Especifica que se está creando un procedimiento almacenado, seguido de la palabra **PROCEDURE** y el **nombre** de dicho procedimiento.
- La sección de parámetros se especifican entre () donde cada uno lleva un nombre y un tipo de dato.
- Palabra clave **BEGIN**: indica el inicio del cuerpo del procedimiento, el cual contiene la lógica específica que se desea implementar dentro del procedimiento.
- Palabra clave **END**: indica el final del procedimiento.
- Dentro de la sección BEGIN y END se agrega el código SQL.

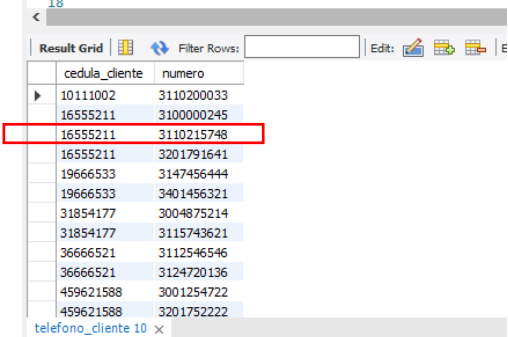
Primera actividad (Librería)

Procedimiento que permite agregar un nuevo teléfono a un cliente ya existente en la tabla teléfono cliente

```
DELIMITER //
CREATE PROCEDURE agregar_telefono (
IN cedula_cliente_proc VARCHAR(10),
IN numero_cliente_proc VARCHAR(15)
)
BEGIN
INSERT INTO telefono_cliente (cedula_cliente, numero)
VALUES (cedula_cliente_proc, numero_cliente_proc);
END //
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento agregar_telefono y consultar la tabla teléfono_cliente:

```
16 • CALL agregar_telefono ('16555211', '3110215748');
17 SELECT * FROM telefono_cliente
18
```



cedula_cliente	numero
10111002	3110200033
16555211	3100000245
16555211	3110215748
16555211	3201791641
19666533	3147456444
19666533	3401456321
31854177	3004875214
31854177	3115743621
36666521	3112546546
36666521	3124720136
459621588	3001254722
459621588	3201752222

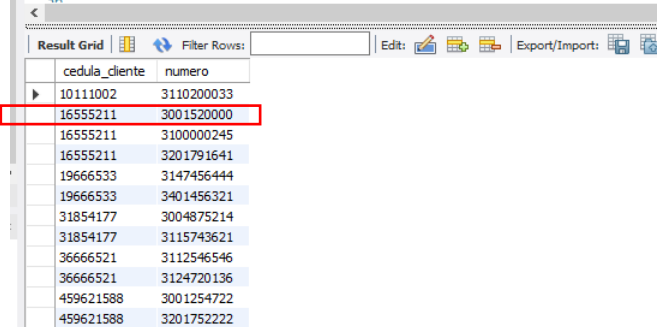
- En el cuerpo del procedimiento se utiliza **INSERT**, el cual nos está indicando que se va a insertar un nuevo registro en la tabla teléfono_cliente y tendrá los campos cedula_cliente y numero.
- Utilizamos **CALL** para llamar a un procedimiento almacenado, en este caso *agregar_telefono*.
- Palabra clave **IN** antes del nombre del parámetro, nos indica que es un parámetro de entrada, es decir, que se puede pasar un valor al procedimiento para ser utilizado en su interior (cuerpo del procedimiento).
- La sentencia **DELIMITER** es útil cuando debemos definir un bloque de código que contenga varias sentencias de SQL como los procedimientos almacenados.

Procedimiento que permite modificar el teléfono de un cliente ya existente, en la tabla teléfono cliente.

```
DELIMITER //
CREATE PROCEDURE modificar_telefono(
IN cedula_cliente_proc VARCHAR(10),
IN numero_cliente_proc VARCHAR(15),
IN numero_nuevo VARCHAR(15)
)
BEGIN
UPDATE telefono_cliente
SET numero = numero_nuevo
WHERE cedula_cliente = cedula_cliente_proc AND numero = numero_cliente_proc;
END //
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento modificar_telefono y consultar la tabla teléfono_cliente:

```
34 • CALL modificar_telefono ('16555211', '3110215748', '3001520000');
35 SELECT * FROM telefono_cliente
```



cedula_cliente	numero
10111002	3110200033
16555211	3001520000
16555211	3100000245
16555211	3201791641
19666533	3147456444
19666533	3401456321
31854177	3004875214
31854177	3115743621
36666521	3112546546
36666521	3124720136
459621588	3001254722
459621588	3201752222

Adicional a las palabras claves y sentencias que utilizamos en el procedimiento anterior, se utilizan las siguientes:

- La palabra clave **UPDATE** se utiliza para modificar (actualizar) los datos existentes de una tabla.
- **SET** se utiliza para asignar un valor específico a una columna de una tabla.
- **WHERE** permite filtrar las filas de una tabla que se seleccionarán en una consulta.

Procedimiento que permite consultar un teléfono de cliente por medio del id

```
DELIMITER //
CREATE PROCEDURE consultar_telefono(
IN cedula_cliente_proc VARCHAR(10)
)
BEGIN
SELECT numero
FROM telefono_cliente
WHERE cedula_cliente = cedula_cliente_proc;
END //
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento consultar_telefono y consultar la tabla teléfono_cliente:

```

48 • CALL consultar_telefono ('16555211');
49 SELECT * FROM telefono_cliente

```

cedula_cliente	numero
10111002	3110200033
16555211	3001520000
16555211	3100000245
16555211	3201791641
19666533	3147456444
19666533	3401456321
31854177	3004875214
31854177	3115743621
36666521	3112546546
36666521	3124720136
459621588	3001254722
459621588	3201752222
54777544	3147894444
99547111	3120097775
NULL	NULL

Procedimiento que permite eliminar el teléfono de un cliente

```

DELIMITER //
CREATE PROCEDURE eliminar_telefono(
IN cedula_cliente_proc VARCHAR(10),
IN numero_cliente_proc VARCHAR(15)
)
BEGIN
DELETE FROM telefono_cliente
WHERE cedula_cliente = cedula_cliente_proc AND numero = numero_cliente_proc;
END //
//DELIMITER ;

```

Este es el resultado al llamar el procedimiento eliminar_telefono y consultar la tabla teléfono_cliente: (Se eliminó correctamente)

```

62 • CALL eliminar_telefono('16555211', '3001520000');
63 • SELECT * FROM telefono_cliente;
64

```

cedula_cliente	numero
10111002	3110200033
16555211	3100000245
16555211	3201791641
19666533	3147456444
19666533	3401456321
31854177	3004875214
31854177	3115743621
36666521	3112546546
36666521	3124720136
459621588	3001254722
459621588	3201752222
54777544	3147894444
99547111	3120097775
NULL	NULL

Adicional a las palabras claves y sentencias que utilizamos en los procedimientos anteriores, se utiliza la siguiente:

Palabra clave **DELETE** permite eliminar filas de una tabla.

¿Qué es un trigger?

También llamado disparador, es un objeto de base de datos el cual se activa automáticamente, dando respuesta a determinados eventos como: insertar, actualizar o eliminar datos en una tabla específica.

Sintaxis de un trigger

```
CREATE TRIGGER nombre_trigger
{BEFORE | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE}
ON nombre_tabla
[FOR EACH ROW]
[WHEN condicion]
BEGIN
-- cuerpo del trigger
END;
```

Adicional a las sentencias y conceptos mencionados anteriormente, también utilizaremos los siguientes:

- Palabra clave **CREATE**: Especifica que se está creando un trigger, seguido de la palabra **TRIGGER** y el **nombre** de dicho disparador.
- **{BEFORE | AFTER | INSTEAD OF}** nos permite especificar cuándo se activará el trigger.
- **{INSERT | UPDATE | DELETE}** sirve para especificar en qué tipo de operación se activará el trigger.
- **FOR EACH ROW** se utiliza para especificar que el cuerpo de un trigger se ejecute una vez para cada fila, lo cual permite realizar acciones específicas como actualizar, insertar o eliminar.

Creación de la tabla control de cambios librería

```
CREATE TABLE control_de_cambios_libreria(
usuario VARCHAR(45),
accion VARCHAR(45),
fecha DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Creación trigger que permite agregar un nuevo teléfono al cliente

El trigger agregar_telefono se dispara al insertar un nuevo registro en la tabla telefono_cliente y se refleja dicho registro en la tabla control_cambios_libreria, donde el campo *acción* muestra si se registró el teléfono.

```
DELIMITER //
CREATE TRIGGER agregar_telefono AFTER INSERT ON telefono_cliente
FOR EACH ROW
BEGIN
    INSERT INTO control_de_cambios_libreria VALUES(user(), 'Se agregó', now());
END;
// DELIMITER ;
```

Creación trigger que permite eliminar un teléfono al cliente

El trigger eliminar_telefono se dispara al eliminar un registro en la tabla telefono_cliente y se refleja dicho registro en la tabla control_cambios_libreria, donde el campo *acción* muestra si se eliminó el teléfono.

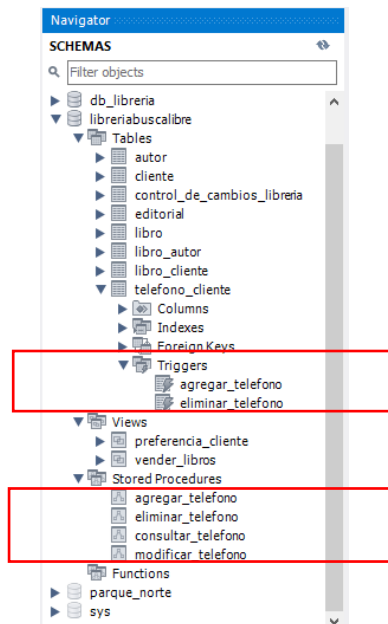
```
DELIMITER //
CREATE TRIGGER eliminar_telefono AFTER DELETE ON telefono_cliente
FOR EACH ROW
BEGIN
    INSERT INTO control_de_cambios_libreria VALUES(user(), 'Se eliminó', now());
END;
// DELIMITER ;
```

Resultado en la tabla control_cambios_libreria

75 • `SELECT * FROM control_de_cambios_libreria;`

usuario	accion	fecha
root@localhost	Se agregó	2023-02-14 18:20:26
root@localhost	Se eliminó	2023-02-14 18:22:02
root@localhost	Se agregó	2023-02-14 20:59:50
root@localhost	Se eliminó	2023-02-14 21:10:40

Después de crear los procedimientos y triggers, los podremos visualizar en el panel de navegación en la parte superior izquierda:

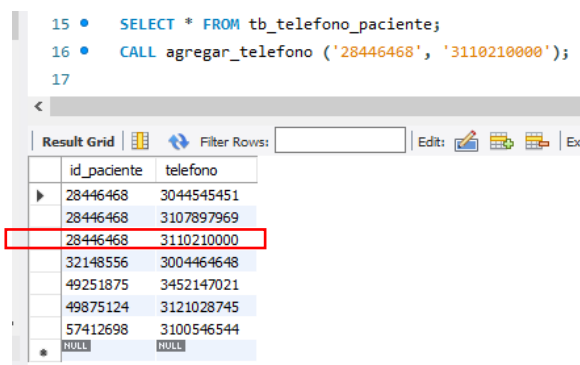


Segunda actividad – hospital.

Procedimiento que permite agregar un nuevo teléfono a un paciente ya existente en la tabla teléfono paciente

```
DELIMITER //  
CREATE PROCEDURE agregar_telefono (  
IN id_paciente_proc VARCHAR(10),  
IN telefono_paciente_proc VARCHAR(15)  
)  
BEGIN  
INSERT INTO tb_telefono_paciente (id_paciente, telefono)  
VALUES (id_paciente_proc, telefono_paciente_proc);  
END //  
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento agregar_telefono y consultar la tabla tb_telefono_paciente:



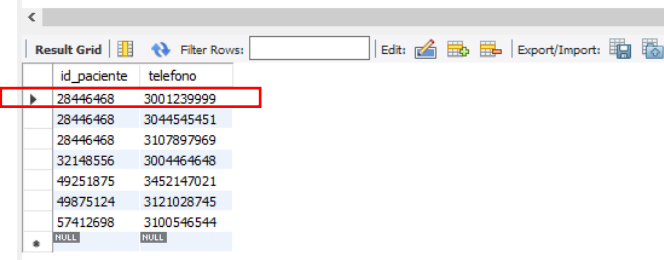
- En el cuerpo del procedimiento se utiliza **INSERT**, el cual nos está indicando que se va a insertar un nuevo registro en la tabla `tb_telefono_paciente` y tendrá los campos `id_paciente` y `telefono`.
- Utilizamos **CALL** para llamar a un procedimiento almacenado, en este caso *agregar_telefono*.
- Palabra clave **IN** antes del nombre del parámetro, nos indica que es un parámetro de entrada, es decir, que se puede pasar un valor al procedimiento para ser utilizado en su interior (cuerpo del procedimiento).
- La sentencia **DELIMITER** es útil cuando debemos definir un bloque de código que contenga varias sentencias de SQL como los procedimientos almacenados.

Procedimiento que permite modificar el teléfono de un paciente ya existente, en la tabla `telefono paciente`.

```
DELIMITER //
CREATE PROCEDURE modificar_telefono(
  IN id_paciente_proc VARCHAR(10),
  IN telefono_paciente_proc VARCHAR(15),
  IN telefono_nuevo VARCHAR(15)
)
BEGIN
  UPDATE tb_telefono_paciente
  SET telefono = telefono_nuevo
  WHERE id_paciente = id_paciente_proc AND telefono = telefono_paciente_proc;
END //
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento `modificar_telefono` y consultar la tabla `tb_telefono_paciente`:

```
31 • CALL modificar_telefono ('28446468', '3110210000', '3001239999');
32 • SELECT * FROM tb_telefono_paciente;
33
```



id_paciente	telefono
28446468	3001239999
28446468	3044545451
28446468	3107897969
32148556	3004464648
49251875	3452147021
49875124	3121028745
57412698	3100546544
TOTAL	7/7

Adicional a las palabras claves y sentencias que utilizamos en el procedimiento anterior, se utilizan las siguientes:

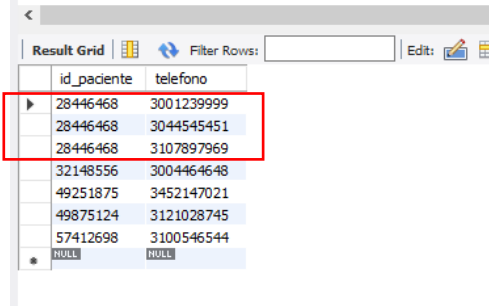
- La palabra clave **UPDATE** se utiliza para modificar (actualizar) los datos existentes de una tabla.
- **SET** se utiliza para asignar un valor específico a una columna de una tabla.
- **WHERE** permite filtrar las filas de una tabla que se seleccionarán en una consulta.

Procedimiento que permite consultar un teléfono de paciente por medio del id

```
CREATE PROCEDURE consultar_telefono(  
IN id_paciente_proc VARCHAR(10)  
)  
BEGIN  
SELECT telefono  
FROM tb_telefono_paciente  
WHERE id_paciente = id_paciente_proc;  
END //  
//DELIMITER ;
```

Este es el resultado al llamar el procedimiento consultar_telefono y consultar la tabla tb_telefono_paciente:

```
45 • CALL consultar_telefono ('28446468');  
46 • SELECT * FROM tb_telefono_paciente;  
47
```



The screenshot shows a database query result grid with two columns: 'id_paciente' and 'telefono'. The first three rows have 'id_paciente' values of 28446468 and 'telefono' values of 3001239999, 3044545451, and 3107897969. These three rows are highlighted with a red box. The remaining rows have 'id_paciente' values of 32148556, 49251875, 49875124, and 57412698, with corresponding 'telefono' values. The last row is a summary row with 'id_paciente' as 'NULL' and 'telefono' as 'NULL'.

id_paciente	telefono
28446468	3001239999
28446468	3044545451
28446468	3107897969
32148556	3004464648
49251875	3452147021
49875124	3121028745
57412698	3100546544
NULL	NULL

Procedimiento que permite eliminar el teléfono de un paciente

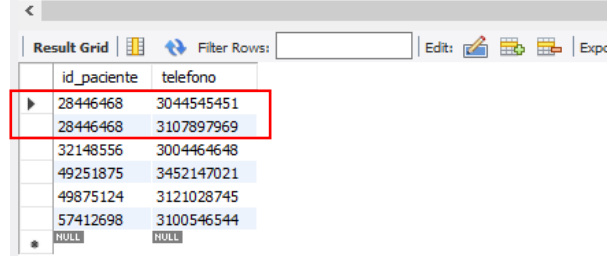
```
CREATE PROCEDURE eliminar_telefono(  
IN id_paciente_proc VARCHAR(10),  
IN telefono_paciente_proc VARCHAR(15)  
)  
BEGIN  
DELETE FROM tb_telefono_paciente  
WHERE id_paciente = id_paciente_proc AND telefono = telefono_paciente_proc;  
END //  
//DELIMITER ;
```

Adicional a las palabras claves y sentencias que utilizamos en los procedimientos anteriores, se utiliza la siguiente:

Palabra clave **DELETE** permite eliminar filas de una tabla.

Este es el resultado al llamar el procedimiento eliminar_telefono y consultar la tabla tb_telefono_paciente: (Se eliminó correctamente)

```
59 • CALL eliminar_telefono('28446468', '3001239999');
60 • SELECT * FROM tb_telefono_paciente;
61
```



id_paciente	telefono
28446468	3044545451
28446468	3107897969
32148556	3004464648
49251875	3452147021
49875124	3121028745
57412698	3100546544
NULL	NULL

Creación de la tabla control de cambios hospital

```
CREATE TABLE control_de_cambios_hospital(  
  usuario VARCHAR(45),  
  accion VARCHAR(45),  
  fecha DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

Creación trigger que permite agregar un nuevo teléfono al paciente

El trigger agregar_telefono se dispara al insertar un nuevo registro en la tabla tb_telefono_paciente y se refleja dicho registro en la tabla control_cambios_hospital, donde el campo *acción* muestra si se registró el teléfono.

```
DELIMITER //  
CREATE TRIGGER agregar_telefono AFTER INSERT ON tb_telefono_paciente  
  FOR EACH ROW  
  BEGIN  
    INSERT INTO control_de_cambios_hospital VALUES(user(), 'Se agregó', now());  
  END;  
// DELIMITER ;
```

Creación trigger que permite eliminar un teléfono al paciente

El trigger eliminar_telefono se dispara al eliminar un registro en la tabla teléfono_cliente y se refleja dicho registro en la tabla control_cambios_hospital, donde el campo *acción* muestra si se eliminó el teléfono.

```
DELIMITER //  
CREATE TRIGGER eliminar_telefono AFTER DELETE ON tb_telefono_paciente  
  FOR EACH ROW  
  BEGIN  
    INSERT INTO control_de_cambios_hospital VALUES(user(), 'Se eliminó', now());  
  END;  
// DELIMITER ;
```

Resultado en la tabla control_cambios_hospital

```
70 • SELECT * FROM control_de_cambios_hospital;
```

usuario	accion	fecha
root@localhost	Se agregó	2023-02-14 18:54:01
root@localhost	Se eliminó	2023-02-14 18:54:26
root@localhost	Se agregó	2023-02-14 21:16:26
root@localhost	Se eliminó	2023-02-14 21:28:07

Después de crear los procedimientos y triggers, los podremos visualizar en el panel de navegación en la parte superior izquierda:

