**Ejercicio Procedimientos almacenados y Funciones**

Integrantes

David Cruz

Edwin Abaunza

Kennen Cortez

Luis Mariño

Mahily Gutiérrez

**FICHA** 3197815

Bases de datos Relacionales

Martha Cuervo

Centro de Servicios Financieros

Bogotá D.C

**Procedimientos Almacenados**

**1. Procedimiento: control de la minuta**

Este procedimiento registra una nueva minuta diaria en el sistema, especificando la fecha y hora del recibido y entrega, las novedades, el nombre del responsable, descripción, el ambiente implicado, el ID del usuario responsable y el ID del guardia. Esto permite hacer seguimiento al uso del ambiente durante el día.

SQL: -- Procedimiento 1: control de la minuta

DELIMITER //

CREATE PROCEDURE control\_minuta(

IN fecha\_recibido DATETIME,

IN fecha\_entrega DATETIME,

IN novedad TEXT,

IN responsable VARCHAR(250),

IN descripcion TEXT,

IN ambiente\_id INT,

IN documento\_usuario INT,

IN id\_guardia INT

)

BEGIN

INSERT INTO registro\_minuta(

fecha\_hora\_recibo, fecha\_hora\_entrega, novedad,

responsable, descripcion\_min, ambiente\_id,

Usuario\_id\_usuario, guarda\_seguridad\_Usuario\_id\_usuario

)

VALUES (

fecha\_recibido, fecha\_entrega, novedad,

responsable, descripcion, ambiente\_id,

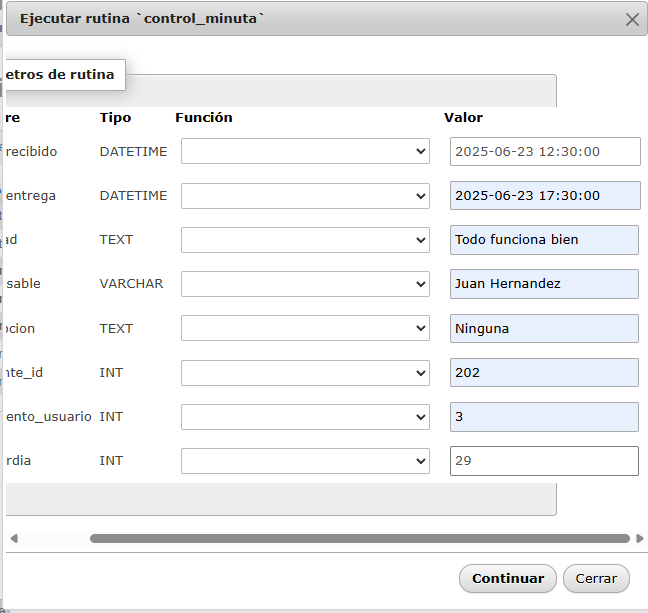
documento\_usuario, id\_guardia

);

END //

DELIMITER ;

**EJECUCIÓN**:





**2. Procedimiento: consultar asistencia de un aprendiz**

Este procedimiento permite consultar el historial de asistencias registradas para un aprendiz específico, mostrando la fecha y el estado de cada asistencia. Es útil para seguimiento disciplinario.

SQL:

-- Procedimiento 2: consultar asistencia de un aprendiz

DELIMITER //

CREATE PROCEDURE consultar\_asistencia\_aprendiz(

IN id\_aprendiz INT

)

BEGIN

SELECT fecha\_asistencia, estado\_asistencia

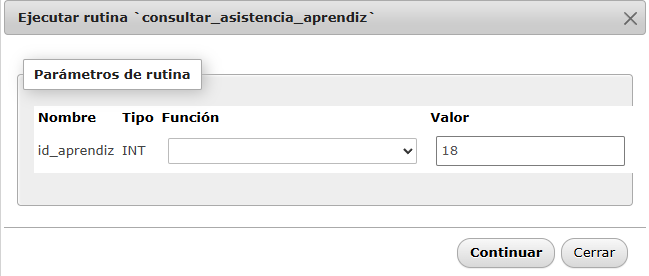
FROM registro\_asistencia

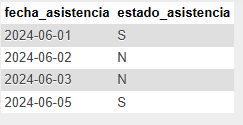
WHERE aprendiz\_Usuario\_id\_usuario = id\_aprendiz;

END //

DELIMITER ;

**EJECUCIÓN**:





**3. Procedimiento: eliminar incidente**

Este procedimiento elimina un incidente registrado previamente en el sistema a partir del ID del incidente. Se utiliza cuando se desea limpiar registros inválidos o duplicados.

SQL:

-- Procedimiento 3: eliminar incidente

DELIMITER //

CREATE PROCEDURE eliminar\_incidente(

IN incidente\_id INT

)

BEGIN

DELETE FROM registro\_incidente

WHERE id\_incidente = incidente\_id;

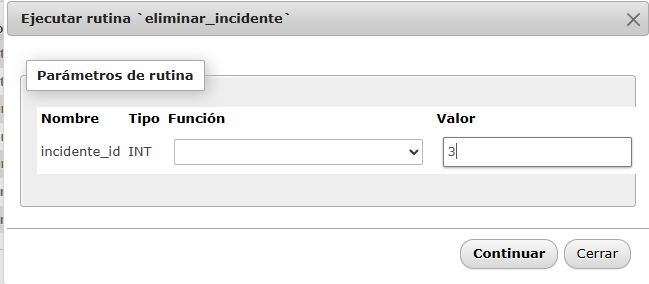
END //

DELIMITER ;

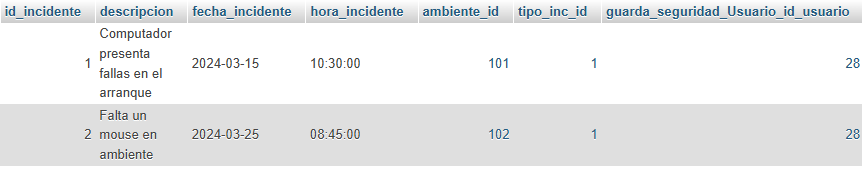
EJECUCIÓN:

Antes





Después:



**4. Procedimiento:** **actualizar estado de un recurso**

Este procedimiento actualiza el estado actual de un recurso (como un computador o proyector), por ejemplo, pasarlo de "Disponible" a "En Mantenimiento" o “Dañado”.

SQL:

-- Procedimiento 4: actualizar estado de un recurso

DELIMITER //

CREATE PROCEDURE actualizar\_estado\_recurso(

IN recurso\_id INT,

IN nuevo\_estado VARCHAR(20)

)

BEGIN

UPDATE recursos

SET estado = nuevo\_estado

WHERE id\_recurso = recurso\_id;

END //

DELIMITER ;

EJECUCIÓN:

Antes:





Despues:



**5. Procedimiento:** **consultar recursos por ambiente**

Este procedimiento permite consultar todos los recursos que pertenecen a un ambiente específico, mostrando su serial, nombre, tipo, estado y observación. Es útil para inventario.

SQL:

-- Procedimiento 5: consultar recursos por ambiente

DELIMITER //

CREATE PROCEDURE consultar\_recursos\_ambiente(

IN id\_ambiente INT

)

BEGIN

SELECT r.serial\_recurso, r.nombre\_recurso, tr.recurso\_tipo, r.estado, r.observacion

FROM recursos r

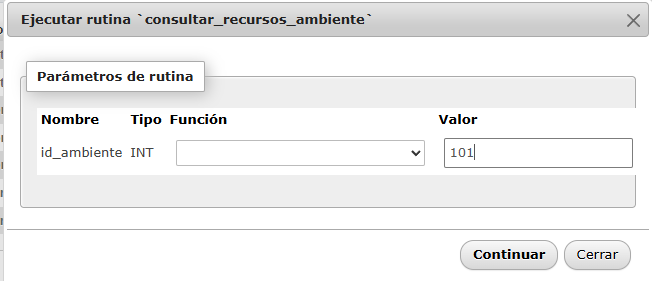
JOIN tipo\_recurso tr ON r.tipo\_recurso = tr.id\_tipo\_recurso

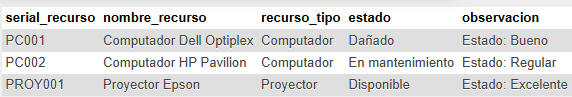
WHERE r.ambiente\_id = id\_ambiente;

END //

DELIMITER ;

EJECUCIÓN:





**6. Procedimiento: actualizar responsable de minuta**

Este procedimiento actualiza tanto el nombre del responsable como su ID en una minuta ya registrada. Esto se hace cuando hay un cambio de turno o corrección de datos.

**SQL:**

SQL:

-- Procedimiento 6: actualizar responsable de minuta

DELIMITER //

CREATE PROCEDURE actualizar\_responsable\_minuta(

IN id\_min INT,

IN nuevo\_responsable VARCHAR(30),

IN nuevo\_id\_usuario INT

)

BEGIN

UPDATE registro\_minuta

SET responsable = nuevo\_responsable,

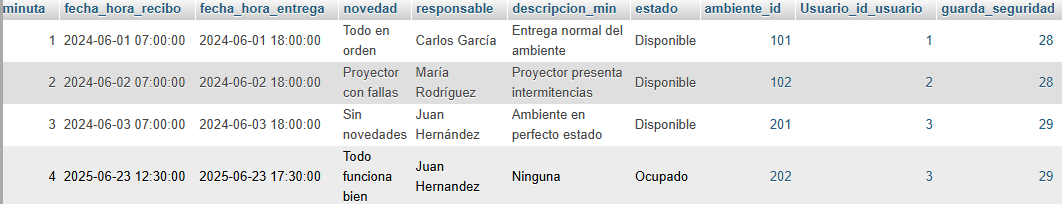
Usuario\_id\_usuario = nuevo\_id\_usuario

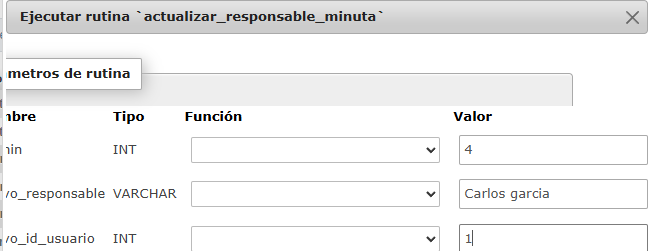
WHERE id\_minuta = id\_min;

END //

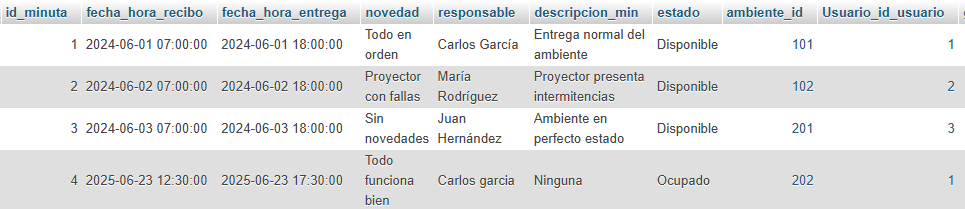
DELIMITER ;

EJECUCIÓN:

Antes: 



Después:



**7. Procedimiento: eliminar recurso**

Este procedimiento elimina un recurso del sistema mediante su ID. Se utiliza para depurar el inventario.

SQL:

-- Procedimiento 7: eliminar recurso

DELIMITER //

CREATE PROCEDURE eliminar\_recurso(

IN recurso\_id INT

)

BEGIN

DELETE FROM recursos

WHERE id\_recurso = recurso\_id;

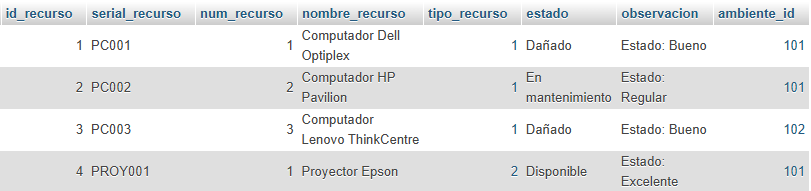
END //

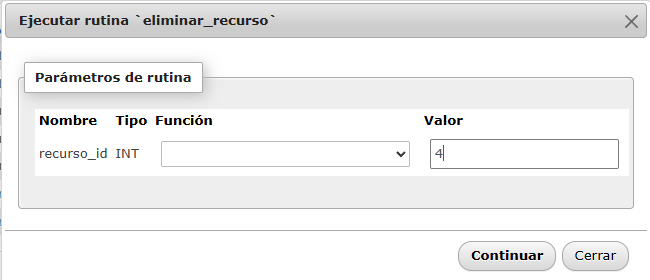
DELIMITER ;

DELIMITER //

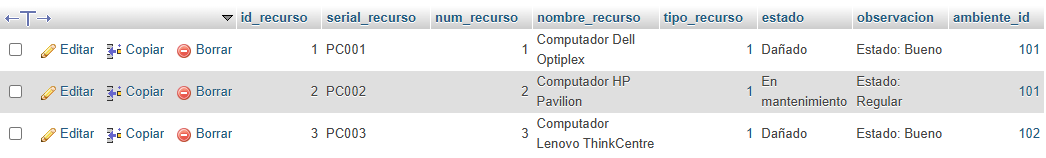
EJECUCIÓN:

Antes:





Después:



**8. Procedimiento: monitorear traslado de recurso**

Este procedimiento registra el traslado de un recurso de un ambiente a otro, registrando la fecha y observaciones, y a

ctualiza su ubicación actual. Esto permite controlar préstamos de dispositivos.

SQL:

-- Procedimiento 8: reporte de recursos de ambiente

CREATE PROCEDURE monitorear\_traslado\_recurso(

IN recurso\_id INT,

IN ambiente\_origen INT,

IN ambiente\_destino INT,

IN observacion TEXT

)

BEGIN

INSERT INTO traslado\_recurso (

recurso\_id, ambiente\_origen\_id, ambiente\_destino\_id,

fecha\_traslado, observacion

)

VALUES (

recurso\_id, ambiente\_origen, ambiente\_destino,

NOW(), observacion

);

UPDATE recursos

SET ambiente\_id = ambiente\_destino

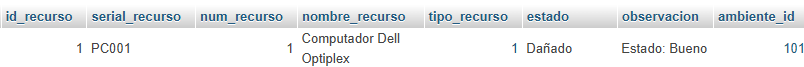
WHERE id\_recurso = recurso\_id;

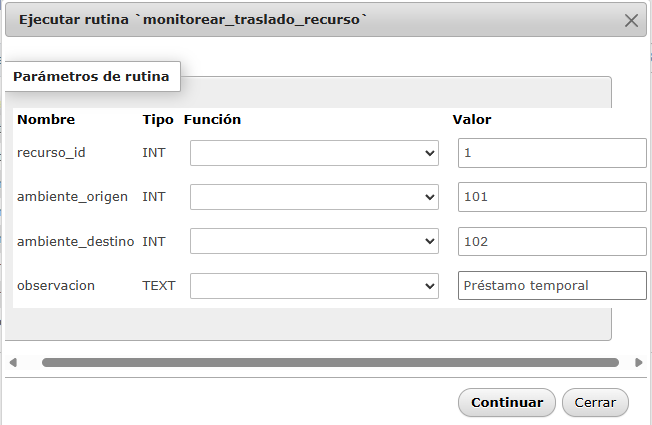
END //

DELIMITER ;

EJECUCIÓN:

Antes:





Después:

****

****

**Funciones**

1. **Función obtener estado ambiente**:  
   Esta función retorna el estado actual de un ambiente (disponible, en uso, mantenimiento), basándose en los horarios y reportes. Es clave para asignación de espacios.

SQL:

-- Función 1: obtener estado ambiente

DELIMITER //

CREATE FUNCTION estado\_ambiente(id INT)

RETURNS VARCHAR(30)

READS SQL DATA

BEGIN

DECLARE estado\_actual VARCHAR(30);

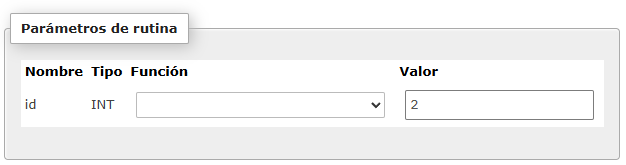
SELECT estado INTO estado\_actual FROM ambiente WHERE id\_ambiente = id;

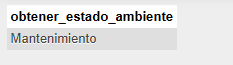
RETURN estado\_actual;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función verificar recursos por ambiente**:  
   Esta función devuelve la cantidad y tipo de recursos que están asignados a un ambiente determinado, permitiendo validar si el inventario es correcto al generar minutas o asignaciones.

SQL:

-- Función 2: verificar recursos por ambiente

DELIMITER //

CREATE FUNCTION total\_recursos\_ambiente(id INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE total INT;

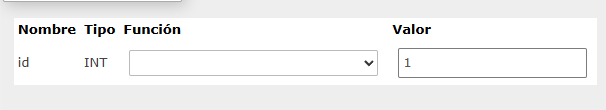
SELECT COUNT(\*) INTO total FROM recursos WHERE ambiente\_id = id;

RETURN total;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función obtener incidentes por ambiente**:  
   Esta función devuelve todos los incidentes reportados en un ambiente específico, junto con su fecha, descripción y estado. Apoya a mantenimiento o seguridad.

SQL:

-- Función 3: obtener incidentes por ambiente

DELIMITER //

CREATE FUNCTION contar\_incidentes\_ambiente(id INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE total INT;

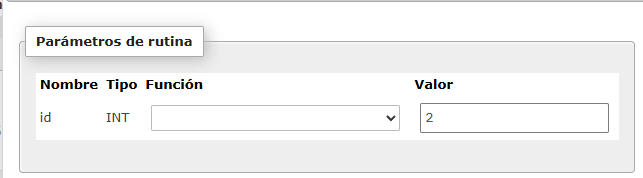
SELECT COUNT(\*) INTO total FROM registro\_incidente WHERE ambiente\_id = id;

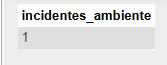
RETURN total;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función contar aprendices programa**:  
   Esta función cuenta la cantidad de aprendices que están registrados en un programa específico. Es útil para determinar el tamaño de cada ficha.

SQL:

-- Función 4: contar aprendices programa

DELIMITER //

CREATE FUNCTION contar\_aprendices\_programa(id\_programa INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE total INT;

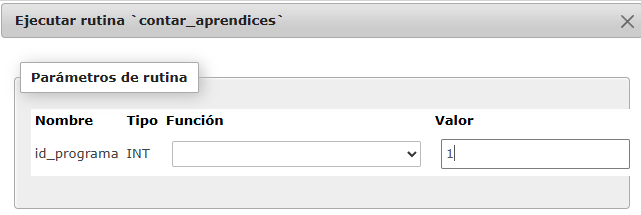
SELECT COUNT(\*) INTO total FROM aprendiz WHERE programa\_id = id\_programa;

RETURN total;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función consultar el programa del aprendiz**:  
   Esta función se implementaría para obtener el programa de un aprendiz especifico según su id, Esto es ideal para realizar seguimiento a los aprendices que estén condicionados.

SQL:

-- Función 5: consultar programa del aprendiz

DELIMITER //

CREATE FUNCTION programa\_aprendiz(id\_aprendiz INT)

RETURNS VARCHAR(30)

READS SQL DATA

BEGIN

DECLARE nombre\_programa VARCHAR(50);

SELECT p.nombre\_programa INTO nombre\_programa

FROM aprendiz a JOIN programas p ON a.programa\_id = p.id\_programas

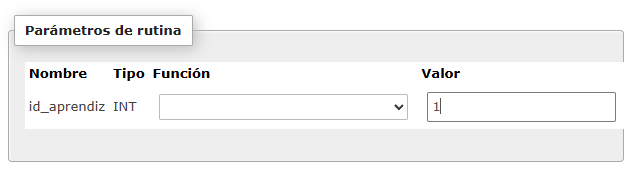
WHERE a.id\_aprendiz = id\_aprendiz;

RETURN nombre\_programa;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función buscar correo de usuario**:  
   Esta función se utilizaría para consultar el correo de los usuarios buscándolo a partir de su número de documento

SQL:

-- Función 6: buscar correo usuario

DELIMITER //

CREATE FUNCTION correo\_usuario(doc INT)

RETURNS VARCHAR(30)

READS SQL DATA

BEGIN

DECLARE correo VARCHAR(30);

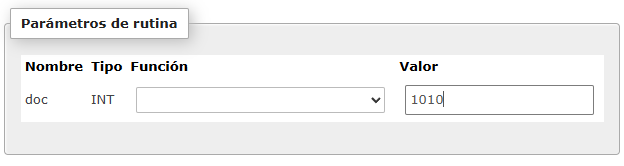
SELECT u.correo INTO correo FROM Usuario u WHERE u.documento = doc;

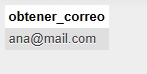
RETURN correo;

END //

DELIMITER ;

EJECUCIÓN:





1. **Función buscar coordinación responsable de programa**:  
   Esta función permite consultar la coordinación académica asociada al programa en el que está inscrito un aprendiz. A partir del ID del aprendiz, la función accede a su programa de formación y retorna el nombre de la coordinación correspondiente.

SQL:

-- Función 7: buscar coordinación de programa

DELIMITER //

CREATE FUNCTION buscar\_coordinacion\_aprendiz(id\_aprendiz INT)

RETURNS VARCHAR(30)

READS SQL DATA

BEGIN

DECLARE nombre\_coord VARCHAR(30);

SELECT c.nombre\_coordinacion INTO nombre\_coord

FROM aprendiz a

JOIN programas p ON a.programa\_id = p.id\_programas

JOIN coordinacion c ON p.coordinacion\_id = c.id\_coordinacion

WHERE a.id\_aprendiz = id\_aprendiz;

RETURN nombre\_coord;

END //

DELIMITER ;

EJECUCIÓN:

