**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 de tanto el recibido como la entrega, las novedades, responsable, descripción, estado, etc. Esto con la finalidad de conocer los datos de la minuta al momento de la entrega.

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(30),

IN descripcion TEXT,

IN estado varchar(15)

IN ambiente\_id INT,

IN documento\_usuario INT

)

BEGIN

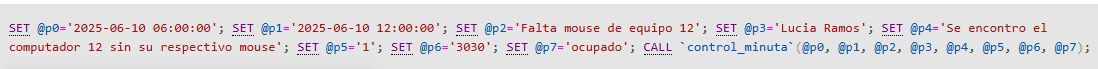
INSERT INTO registro\_minuta(fecha\_hora\_recibido, fecha\_hora\_entrega, novedad, responsable, descripcion\_amb, ambiente\_id, docu\_id)

VALUES (fecha\_recibido, fecha\_entrega, novedad, responsable, descripcion, ambiente\_id, documento\_usuario);

END //

DELIMITER ;

EJECUCIÓN:





1. **Procedimiento registrar ingreso aprendiz**:  
   Este procedimiento guarda la hora de ingreso de un aprendiz, asociándola con su documento y programa. Su finalidad es generar un historial de asistencia y permitir validaciones automáticas por jornada.

SQL:

-- Procedimiento 2: registrar ingreso aprendiz

DELIMITER //

CREATE PROCEDURE registrar\_ingreso\_aprendiz(

IN fecha DATE,

IN estado ENUM('N', 'E', 'S'),

IN id\_jornada INT,

IN id\_aprendiz INT

)

BEGIN

INSERT INTO registro\_asistencia(fecha\_asistencia, estado\_asistencia, jorn\_id, apr\_id)

VALUES (fecha, estado, id\_jornada, id\_aprendiz);

END //

DELIMITER ;

1. **Procedimiento registrar salida aprendiz**:  
   Este procedimiento registra la hora de salida del aprendiz, con el objetivo de dejar constancia del tiempo que permaneció dentro de las instalaciones, útil para seguridad y control interno.

SQL:

-- Procedimiento 3: registrar salida aprendiz

-- Se asume que el estado de salida es 'S'

DELIMITER //

CREATE PROCEDURE registrar\_salida\_aprendiz(

IN id\_asistencia INT

)

BEGIN

UPDATE registro\_asistencia

SET estado\_asistencia = 'S'

WHERE id\_asistencia = id\_asistencia;

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento registrar incidente**:  
   Este procedimiento permite almacenar un reporte de incidente que ocurrió en cualquier ambiente o zona del centro. Incluye descripción, ubicación, responsable y fecha, con el objetivo de generar alertas o seguimientos.

SQL:

-- Procedimiento 4: registrar incidente

DELIMITER //

CREATE PROCEDURE registrar\_incidente(

IN descripcion TEXT,

IN fecha DATE,

IN hora TIME,

IN id\_ambiente INT,

IN id\_tipo INT

)

BEGIN

INSERT INTO registro\_incidente(descripcion, fecha\_incidente, hora\_incidente, ambiente\_id, tipo\_inc\_id)

VALUES (descripcion, fecha, hora, id\_ambiente, id\_tipo);

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento asignar recursos a ambiente**:  
   Este procedimiento vincula uno o varios recursos tecnológicos o físicos a un ambiente específico. Su finalidad es mantener actualizado el inventario de cada ambiente y facilitar el control en minutas.

SQL:

-- Procedimiento 5: asignar recursos a ambiente

DELIMITER //

CREATE PROCEDURE asignar\_recurso\_ambiente(

IN serial VARCHAR(30),

IN numero TINYINT,

IN estado VARCHAR(20),

IN observacion TEXT,

IN ambiente INT,

IN tipo INT

)

BEGIN

INSERT INTO recursos(serial\_recurso, num\_recurso, estado, observacion, ambiente\_id, recurso\_id)

VALUES (serial, numero, estado, observacion, ambiente, tipo);

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento actualizar estado recurso**:  
   Este procedimiento cambiara el estado de un recurso determinado dependiendo de las condiciones que se pueda encontrar, por ejemplo, pasar un recurso de operativo o en funcionamiento a dañado, eso según inspecciones o reportes de los incidentes.

SQL:

-- Procedimiento 6: actualizar estado recurso

DELIMITER //

CREATE PROCEDURE actualizar\_estado\_recurso(

IN serial VARCHAR(30),

IN nuevo\_estado VARCHAR(20)

)

BEGIN

UPDATE recursos

SET estado = nuevo\_estado

WHERE serial\_recurso = serial;

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento asistencia aprendiz**:  
   Este procedimiento registra una entrada en la tabla de asistencias indicando la fecha, el estado, es decir, si asistió, llego tarde o no asistió, el aprendiz y la jornada

SQL:

-- Procedimiento 7: asistencia aprendiz

DELIMITER //

CREATE PROCEDURE registrar\_asistencia\_aprendiz(

IN fecha DATE,

IN estado ENUM('N','E','S'),

IN jornada\_id INT,

IN aprendiz\_id INT

)

BEGIN

INSERT INTO registro\_asistencia(fecha\_asistencia, estado\_asistencia, jorn\_id, apr\_id)

VALUES (fecha, estado, jornada\_id, aprendiz\_id);

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento mostrar ambientes disponibles**:  
   Este procedimiento devuelve los ambientes que estén en estado disponible para hacerle uso o asignar a un responsable especifico

SQL:

-- Procedimiento 8: mostrar ambientes disponibles

DELIMITER //

CREATE PROCEDURE ambientes\_disponibles()

BEGIN

SELECT \* FROM ambiente WHERE estado = 'disponible';

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento cambiar responsable minuta**:  
   Este procedimiento permite actualizar el responsable de una minuta registrada por un cambio de turno o el reemplazo de un instructor que se la haga imposible estar en la formación de los aprendices

SQL:

-- Procedimiento 9: cambiar responsable minuta

DELIMITER //

CREATE PROCEDURE actualizar\_responsable\_minuta(

IN id\_minuta INT,

IN nuevo\_responsable VARCHAR(30)

)

BEGIN

UPDATE registro\_minuta

SET responsable = nuevo\_responsable

WHERE id\_minuta = id\_minuta;

END //

DELIMITER ;

EJECUCIÓN:

1. **Procedimiento reporte de recursos de ambiente**:  
   Este procedimiento genera un reporte detallado de los recursos existentes en una ambiente, incluyendo tipo, estado y observaciones.

SQL:

-- Procedimiento 10: reporte de recursos de ambiente

DELIMITER //

CREATE PROCEDURE reporte\_recursos\_ambiente(

IN id\_ambiente INT

)

BEGIN

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

FROM recursos r

JOIN tipo\_recurso tr ON r.recurso\_id = tr.id\_recurso

WHERE r.ambiente\_id = id\_ambiente;

END //

DELIMITER ;

EJECUCIÓN:

**Funciones**

1. **Función contar dispositivos en minuta**:  
   Esta función calcula cuántos dispositivos fueron entregados y cuántos fueron recibidos en una minuta específica. Su finalidad es detectar diferencias que puedan indicar pérdidas o robos.

SQL:

-- Función 1: contar dispositivos en minuta

DELIMITER //

CREATE FUNCTION contar\_recursos\_en\_ambiente(id\_ambiente INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE total INT;

SELECT COUNT(\*) INTO total

FROM recursos

WHERE ambiente\_id = id\_ambiente;

RETURN total;

END //

DELIMITER ;

EJECUCIÓN:

1. **Función calcular tiempo permanencia**:  
   Esta función calcula el tiempo exacto que un estudiante, estuvo dentro del centro, usando la hora de entrada y salida. Esto apoya el control disciplinario y de seguridad.

SQL:

-- Función 2: calcular tiempo permanencia (días)

DELIMITER //

CREATE FUNCTION calcular\_dias\_permanencia(id\_aprendiz INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE total INT;

SELECT COUNT(\*) INTO total FROM registro\_asistencia WHERE apr\_id = id\_aprendiz;

RETURN total;

END //

DELIMITER ;

EJECUCIÓN:

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 3: 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 4: 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 5: 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 6: 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 cantidad de incidentes en ambiente**:  
   Esta función realiza el conteo de cuantos incidentes se han registrado en un ambiente determinado. Permite analizar puntos críticos.

SQL:

-- Función 7: cantidad de incidentes en ambiente

DELIMITER //

CREATE FUNCTION cantidad\_incidentes(id\_ambiente INT)

RETURNS INT

READS SQL DATA

BEGIN

DECLARE cantidad INT;

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

RETURN cantidad;

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 8: consultar programa del aprendiz

DELIMITER //

CREATE FUNCTION programa\_aprendiz(id\_aprendiz INT)

RETURNS VARCHAR(50)

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 9: buscar correo usuario

DELIMITER //

CREATE FUNCTION correo\_usuario(doc INT)

RETURNS VARCHAR(50)

READS SQL DATA

BEGIN

DECLARE correo VARCHAR(50);

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 10: buscar coordinación de programa

DELIMITER //

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

RETURNS VARCHAR(50)

READS SQL DATA

BEGIN

DECLARE nombre\_coord VARCHAR(50);

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: