# **Trabajo Practico sobre Transacciones en MySQL**

**Ejercicio 1: Transferencia de fondos**

Realiza una transacción que transfiera $100 desde la cuenta de origen (cuenta A) a la cuenta de destino (cuenta B).

CREATE TABLE cuentas (

numero\_cuenta VARCHAR (10) PRIMARY KEY,

saldo DECIMAL (10, 2)

);

INSERT INTO cuentas (numero\_cuenta, saldo) VALUES

('A', 1000.00),

('B', 750.00),

('C', 1200.00),

('D', 500.00),

('E', 2000.00);

select \* from cuentas;

**Ejercicio 2**

Crea un procedimiento almacenado llamado ActualizarInventario que tome dos parámetros de entrada:

producto\_id (VARCHAR, 10): El identificador del producto a actualizar.

cantidad\_a\_restar (INT): La cantidad que se restará del inventario.

En el procedimiento, utiliza una transacción para realizar la actualización del inventario:

* Obtene la cantidad actual del producto especificado.
* Verifica si la cantidad restada sería válida (mayor o igual a cero).
* Si la cantidad es válida, realiza la actualización del inventario restando la cantidad especificada.
* Si la cantidad restada sería negativa, muestra un mensaje al usuario indicando que la operación se cancela y realiza un rollback.

Llama al procedimiento ActualizarInventario con diferentes valores de producto\_id y cantidad\_a\_restar, incluyendo casos donde la cantidad restada resultaría en un inventario negativo.

Verifica que el procedimiento funcione correctamente y que se muestren mensajes de error y se realice un rollback cuando corresponda.

-- Actualización de cantidades de productos en el inventario

CREATE TABLE inventario (

producto\_id VARCHAR (10) PRIMARY KEY,

cantidad INT

);

INSERT INTO inventario (producto\_id, cantidad) VALUES

('Producto1', 50),

('Producto2', 30),

('Producto3', 70),

('Producto4', 20),

('Producto5', 60);

DELIMITER //

CREATE PROCEDURE ActualizarInventario (

IN p\_producto\_id VARCHAR (10),

IN cantidad\_a\_restar INT

)

BEGIN

DECLARE nueva\_cantidad INT;

START TRANSACTION;

-- Obtene la cantidad actual del producto

SELECT cantidad INTO nueva\_cantidad FROM inventario WHERE producto\_id = p\_producto\_id;

-- Verifica si la cantidad restada es válida

IF nueva\_cantidad - cantidad\_a\_restar >= 0 THEN

-- Realiza la actualización del inventario

UPDATE inventario SET cantidad = nueva\_cantidad - cantidad\_a\_restar WHERE producto\_id = p\_producto\_id;

COMMIT;

ELSE

-- La cantidad restada sería negativa, muestra un mensaje y realiza un rollback

select 'La cantidad restada resultaría en un inventario negativo. Operación cancelada.';

ROLLBACK;

END IF;

END;

//

DELIMITER ;

CALL ActualizarInventario('Producto1', 15);

CALL ActualizarInventario('Producto1', 1500);

**Ejercicio 3**

Crea un procedimiento almacenado llamado RegistrarCompra que tome dos parámetros de entrada:

cuenta (VARCHAR, 10): El número de cuenta del cliente.

monto (DECIMAL, 10, 2): El monto de la compra.

En el procedimiento, utiliza una transacción para realizar la actualización del saldo y el registro de la transacción:

* Obtene el saldo actual de la cuenta especificada.
* Verifica si el saldo es suficiente para la compra.
* Si el saldo es suficiente, resta el monto de la compra del saldo y registra la transacción en la tabla transacciones.
* Si el saldo no es suficiente, muestra un mensaje de error al cliente indicando que la compra se cancela y realiza un rollback.

Llama al procedimiento RegistrarCompra con diferentes valores de cuenta y monto, incluyendo casos donde el saldo no es suficiente para la compra.

Verifica que el procedimiento funcione correctamente y que se muestren mensajes de error y se realice un rollback cuando corresponda.

CREATE TABLE cuentas\_clientes (

numero\_cuenta VARCHAR (10) PRIMARY KEY,

saldo DECIMAL (10, 2)

);

INSERT INTO cuentas\_clientes (numero\_cuenta, saldo) VALUES

('Cuenta1', 1000.00),

('Cuenta2', 750.00),

('Cuenta3', 1200.00),

('Cuenta4', 500.00),

('Cuenta5', 2000.00);

CREATE TABLE transacciones (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero\_cuenta VARCHAR (10),

monto DECIMAL (10, 2)

);

select \* from cuentas\_clientes;