# CONCEPTOS DE BASES DE DATOS II

Programador Universitario – Lic. en Informática – Ing. en Informática
Facultad de Ciencias Exactas y Tecnología – UNT

## Trabajo Práctico Nro. 8 - Ciclo 2023

**Triggers**

**Ejercicio nro. 1:**

- ALTER TABLE tratamiento ALTER COLUMN dosis SET DATA TYPE integer USING dosis::integer;

- ```
  CREATE OR REPLACE FUNCTION factura_modificar_saldo()
  RETURNS void AS $BODY$
  BEGIN
          UPDATE factura f
          SET saldo = monto - pagado
          FROM (SELECT id_factura, SUM(monto) AS pagado
                  FROM pago
                  GROUP BY id_factura) AS sub
          WHERE f.id_factura = sub.id_factura;
          RAISE NOTICE 'Facturas actualizadas exitosamente';
  END;
  $BODY$ LANGUAGE plpgsql;
  ```

**Ejercicio nro. 2:**

a)

```
CREATE OR REPLACE FUNCTION medicamento_stock_por_tratamiento()
RETURNS TRIGGER AS $stock_por_tratamiento$
BEGIN
        IF NEW.dosis > (SELECT stock FROM medicamento
                        WHERE id_medicamento = NEW.id_medicamento) THEN
                RAISE EXCEPTION 'No hay stock suficiente para iniciar el tratamiento';
        ELSE
                UPDATE medicamento SET stock = stock - NEW.dosis
                WHERE id_medicamento = NEW.id_medicamento;
                RETURN NEW;
        END IF;
END
$stock_por_tratamiento$ LANGUAGE plpgsql

CREATE TRIGGER stock_por_tratamiento BEFORE INSERT ON tratamiento FOR EACH ROW
EXECUTE PROCEDURE medicamento_stock_por_tratamiento();
```
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

b)

```
CREATE OR REPLACE FUNCTION medicamento_stock_por_compras()
RETURNS TRIGGER AS $stock_por_compras$
BEGIN
        UPDATE medicamento
        SET stock = stock + NEW.cantidad
        WHERE id_medicamento = NEW.id_medicamento;
        RETURN NEW;
END; $stock_por_compras$ LANGUAGE plpgsql

CREATE TRIGGER stock_por_compras AFTER INSERT ON compra FOR EACH ROW
EXECUTE PROCEDURE medicamento_stock_por_compras();
```

c)

```
CREATE OR REPLACE FUNCTION factura_actualizar_saldo()
RETURNS TRIGGER AS $pago_nuevo$
BEGIN
        IF NEW.monto > (SELECT saldo FROM factura WHERE id_factura = NEW.id_factura) THEN
                RAISE EXCEPTION 'No se puede pagar más de lo que se debe';
        ELSE
                UPDATE factura
                SET saldo = saldo - NEW.monto
                WHERE id_factura = NEW.id_factura;
                IF (SELECT saldo FROM factura WHERE id_factura = NEW.id_factura) = 0 THEN
                        UPDATE factura SET pagada = 'S' WHERE id_factura = NEW.id_factura;
                ELSE
                        UPDATE factura SET pagada = 'N' WHERE id_factura = NEW.id_factura;
                END IF;
                RETURN NEW;
        END IF;
END; $pago_nuevo$ LANGUAGE plpgsql

CREATE TRIGGER pago_nuevo BEFORE INSERT ON pago FOR EACH ROW
EXECUTE PROCEDURE factura_actualizar_saldo();
```

d)

```
CREATE OR REPLACE FUNCTION factura_borra_pago()
RETURNS TRIGGER AS $pago_elimina$
BEGIN
        UPDATE factura
        SET saldo = saldo + OLD.monto, pagada = 'N'
        WHERE id_factura = OLD.id_factura;
        RETURN NEW;
END; $pago_elimina$ LANGUAGE plpgsql
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

```
CREATE TRIGGER pago_elimina AFTER DELETE ON pago FOR EACH ROW
EXECUTE PROCEDURE factura_borra_pago();
```

e)

```
CREATE OR REPLACE FUNCTION medicamento_bajo_stock()
RETURNS TRIGGER AS $stock_bajo$
DECLARE
        v_precio numeric(8,2);
        v_proveedor varchar(100);
BEGIN
        CREATE TABLE IF NOT EXISTS medicamento_reponer(
                id INTEGER,
                nombre VARCHAR(100),
                presentacion VARCHAR(100),
                stock INTEGER,
                precio NUMERIC(8,2),
                proveedor VARCHAR(100) );

        IF NEW.stock < 50 THEN
                IF NOT EXISTS(SELECT * FROM medicamento_reponer
                                WHERE id = NEW.id_medicamento) THEN
                        v_precio := (SELECT precio_unitario
                                FROM compras
                                WHERE id_medicamento = NEW.id_medicamento
                                ORDER BY fecha DESC
                                LIMIT 1);
                        v_proveedor := (SELECT proveedor
                                        FROM proveedores
                                        INNER JOIN compras USING(id_proveedor)
                                        WHERE id_medicamento = NEW.id_medicamento
                                        ORDER BY fecha DESC
                                        LIMIT 1);
                        INSERT INTO medicamento_reponer
                        VALUES (NEW.id_medicamento, NEW.nombre,
                        NEW.presentacion, NEW.stock, v_precio, v_proveedor);
                ELSE
                        UPDATE medicamento_reponer
                        SET stock = NEW.stock
                        WHERE id = NEW.id_medicamento;
                END IF;
        END IF;
        RETURN NEW;
END;
$stock_bajo$ LANGUAGE plpgsql;
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

```
CREATE TRIGGER stock_bajo AFTER UPDATE OF stock ON medicamento
FOR EACH ROW WHEN (NEW.stock < OLD.stock) EXECUTE PROCEDURE
medicamento_bajo_stock();
```

f)

```
CREATE OR REPLACE FUNCTION medicamento_aumenta_stock( )
RETURNS TRIGGER AS $stock_aumento$
BEGIN
        CREATE TABLE IF NOT EXISTS medicamento_reponer(
                id INTEGER,
                nombre VARCHAR(100),
                presentacion VARCHAR(100),
                stock INTEGER,
                precio NUMERIC(8,2),
                proveedor VARCHAR(100) );

        IF NEW.stock > 50 THEN
                DELETE FROM medicamento_reponer
                WHERE id = NEW.id_medicamento;
        ELSE
                UPDATE medicamento_reponer
                SET stock = NEW.stock
                WHERE id = NEW.id_medicamento;
        END IF;
        RETURN NEW;
END;
$stock_aumento$ LANGUAGE plpgsql

CREATE TRIGGER stock_aumento AFTER UPDATE OF stock ON medicamento
FOR EACH ROW WHEN (NEW.stock > OLD.stock) EXECUTE PROCEDURE
medicamento_aumenta_stock();
```

**Ejercicio nro. 3:**

Realice las siguientes auditorías por trigger.

a)

```
CREATE TABLE audita_medicamento(
        id SERIAL,
        usuario VARCHAR(100),
        fecha DATE,
        operacion CHAR,
        estado VARCHAR(10),
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

```
        id_medicamento INTEGER,
        id_clasificacion SMALLINT,
        id_laboratorio SMALLINT,
        nombre VARCHAR(50),
        presentacion VARCHAR(50),
        precio NUMERIC(8,2),
        stock INTEGER,
        PRIMARY KEY(id) );

CREATE OR REPLACE FUNCTION medicamento_auditoria()
RETURNS TRIGGER AS $audita_medicamento$
BEGIN
        IF (TG_OP = 'INSERT') THEN
                INSERT INTO audita_medicamento VALUES
                (DEFAULT, USER, NOW(), 'I', 'alta', NEW.id_medicamento, NEW.id_clasificacion,
                NEW.id_laboratorio, NEW.nombre, NEW.presentacion,
                NEW.precio, NEW.stock);
                RETURN NEW;
        END IF;
        IF (TG_OP = 'DELETE') THEN
                INSERT INTO audita_medicamento VALUES
                (DEFAULT, USER, NOW(), 'D', 'baja', OLD.id_medicamento, OLD.id_clasificacion,
                OLD.id_laboratorio, OLD.nombre, OLD.presentacion, OLD.precio, OLD.stock);
                RETURN OLD;
        END IF;
        IF (TG_OP = 'UPDATE') THEN
                INSERT INTO audita_medicamento VALUES
                (DEFAULT, USER, NOW(), 'U', 'antes', OLD.id_medicamento,
                OLD.id_clasificacion, OLD.id_laboratorio, OLD.nombre, OLD.presentacion,
                OLD.precio, OLD.stock);

                INSERT INTO audita_medicamento VALUES
                (DEFAULT, USER, NOW(), 'U', 'despues', NEW.id_medicamento,
                NEW.id_clasificacion, NEW.id_laboratorio, NEW.nombre, NEW.presentacion,
                NEW.precio, NEW.stock);
                RETURN NEW;
        END IF;
END
$audita_medicamento$ LANGUAGE plpgsql

CREATE TRIGGER audita_medicamento
AFTER INSERT OR DELETE OR UPDATE ON medicamento
FOR EACH ROW EXECUTE PROCEDURE medicamento_auditoria();
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

b)

```
CREATE TABLE audita_empleado(
        id SERIAL,
        usuario VARCHAR(100),
        fecha DATE,
        id_empleado INTEGER,
        dni VARCHAR(8),
        nombre VARCHAR(100),
        apellido VARCHAR(100),
        sueldo_v NUMERIC(9,2),
        sueldo_n NUMERIC(9,2),
        diferencia NUMERIC(9,2),
        estado VARCHAR(10),
        PRIMARY KEY(id)         );

CREATE OR REPLACE FUNCTION empleado_audita_sueldo()
RETURNS TRIGGER AS $audita_sueldo$
BEGIN
        INSERT INTO audita_empleado VALUES
                (DEFAULT, USER, NOW(), OLD.id_empleado,
                 (SELECT dni FROM personas WHERE id_persona = OLD.id_empleado),
                 (SELECT nombre FROM personas WHERE id_persona = OLD.id_empleado),
                 (SELECT apellido FROM personas WHERE id_persona = OLD.id_empleado),
                 OLD.sueldo, NEW.sueldo, ABS(NEW.sueldo - OLD.sueldo),
                 CASE
                        WHEN (NEW.sueldo > OLD.sueldo) THEN 'aumento'
                        ELSE 'descuento'
                 END);
        RETURN NEW;
END; $audita_sueldo$ LANGUAGE plpgsql

CREATE TRIGGER audita_sueldo AFTER UPDATE OF sueldo ON empleado
FOR EACH ROW WHEN (OLD.sueldo <> NEW.sueldo) EXECUTE PROCEDURE
empleado_audita_sueldo();
```

c)

```
CREATE TABLE audita_tablas_sistema(
        id SERIAL,
        usuario VARCHAR(100),
        fecha DATE,
        id_paciente INTEGER,
        fecha_tabla DATE,
        tabla VARCHAR(50),
        PRIMARY KEY(id) );
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

```
CREATE TABLE consulta_borrada(
        id_paciente INTEGER,
        id_empleado INTEGER,
        fecha DATE,
        id_consultorio SMALLINT,
        hora TIME WITHOUT TIME ZONE,
        resultado VARCHAR(100) );

CREATE TABLE estudio_borrado(
        id_paciente INTEGER,
        id_estudio SMALLINT,
        fecha DATE,
        id_equipo SMALLINT,
        id_empleado INTEGER,
        resultado VARCHAR(100),
        observacion VARCHAR(100),
        precio NUMERIC(10,2) );

CREATE TABLE tratamiento_borrado(
        id_paciente INTEGER,
        id_medicamento INTEGER,
        fecha DATE,
        prescribe INTEGER,
        nombre VARCHAR(50),
        descripcion VARCHAR(100),
        dosis integer,
        costo NUMERIC(10,2) );

CREATE OR REPLACE FUNCTION tablas_auditoria()
RETURNS TRIGGER AS $audita_tablas$
DECLARE v_fecha DATE;
BEGIN
        IF (TG_RELNAME = 'consulta') THEN
                INSERT INTO consulta_borrada VALUES (OLD.*);
                v_fecha := OLD.fecha;
        END IF;
        IF (TG_RELNAME = 'estudio_realizado') THEN
                INSERT INTO estudio_borrado VALUES (OLD.*);
                v_fecha := OLD.fecha;
        END IF;
        IF (TG_RELNAME = 'tratamiento') THEN
                INSERT INTO tratamiento_borrado VALUES (OLD.*);
                v_fecha := OLD.fecha_indicacion;
        END IF;
```

## Trabajo Práctico Nro. 8 - Ciclo 2023

```
        INSERT INTO audita_tablas_sistema VALUES
        (DEFAULT, USER, NOW(), OLD.id_paciente, v_fecha, TG_RELNAME);

        RETURN OLD;
END
$audita_tablas$ LANGUAGE plpgsql

CREATE TRIGGER audita_tablas AFTER DELETE ON consulta
FOR EACH ROW EXECUTE PROCEDURE tablas_auditoria();

CREATE TRIGGER audita_tablas AFTER DELETE ON estudio_realizado
FOR EACH ROW EXECUTE PROCEDURE tablas_auditoria();

CREATE TRIGGER audita_tablas AFTER DELETE ON tratamiento
FOR EACH ROW EXECUTE PROCEDURE tablas_auditoria();
```