CONCEPTOS DE BASES DE DATOS II

Programador Universitario y Lic. en Informática
Fac. de C. Exactas y Tecnología. – UNT

Trabajo Práctico Nro. 7 - Ciclo 2022

Ejercicio 1: Realice las siguientes modificaciones en las tablas indicadas:

- ```
  ALTER TABLE medicamento ADD COLUMN stock integer;
  ```

- ```
  UPDATE medicamento SET stock = 100;
  ```

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

- ```
  ALTER TABLE factura ADD COLUMN pagada char;
  ALTER TABLE factura ADD COLUMN saldo numeric(10, 2);
  ```

- ```
  UPDATE factura SET pagada = 'N';
  ```

- ```
  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
                -- subconsulta con el cálculo de todos los pagos por factura
      WHERE f.id_factura = sub.id_factura;

      RAISE NOTICE 'Facturas actualizadas exitosamente';
  END;
  $BODY$ LANGUAGE plpgsql;
  ```

Ejercicio 2: Realice los siguientes triggers, analizando cuidadosamente qué acción (INSERT, UPDATE o DELETE), sobre qué tabla y cuando (BEFORE o AFTER) se deben activar los mismos:

a)

```
CREATE OR REPLACE FUNCTION medicamento_stock_por_tratamiento()
RETURNS TRIGGER AS $stock_por_tratamiento$
BEGIN
    UPDATE medicamento SET stock = stock - NEW.dosis
    WHERE id_medicamento = NEW.id_medicamento;
    RETURN NEW;
END
$stock_por_tratamiento$ LANGUAGE plpgsql

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

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 compras FOR EACH ROW
EXECUTE PROCEDURE medicamento_stock_por_compras();
```

c)

```
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 med_bajo_stock(
            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 med_bajo_stock 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 med_bajo_stock
            VALUES (NEW.id_medicamento, NEW.nombre, NEW.presentacion, NEW.stock,
v_precio, v_proveedor);
        ELSE
            UPDATE med_bajo_stock
            SET stock = NEW.stock
            WHERE id = NEW.id_medicamento;
        END IF;
    END IF;
    RETURN NEW;
END
$stock_bajo$ LANGUAGE plpgsql;

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

d)

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

    IF NEW.stock > 50 THEN
            DELETE FROM med_bajo_stock
            WHERE id = NEW.id_medicamento;
    ELSE
            UPDATE med_bajo_stock
            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();
```

e)

```
CREATE OR REPLACE FUNCTION factura_actualizar_saldo()
RETURNS TRIGGER AS $pago_nuevo$
BEGIN
    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;
$pago_nuevo$ LANGUAGE plpgsql

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

f)

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

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

g)

```
CREATE OR REPLACE FUNCTION comision_nueva()
RETURNS TRIGGER AS $comision_agrega$
DECLARE
    v_id_laboratorio smallint;
    v_precio numeric(8,2);
BEGIN
    CREATE TABLE IF NOT EXISTS comisiones(
            id_laboratorio SMALLINT,
            laboratorio VARCHAR(100),
            fecha DATE,
            precio NUMERIC(8,2),
            monto_total NUMERIC(10,2),
            comision NUMERIC(8,2)
    );

    v_id_laboratorio := (SELECT id_laboratorio FROM medicamento WHERE id_medicamento =
NEW.id_medicamento);
    v_precio := (SELECT precio FROM medicamento WHERE id_medicamento =
NEW.id_medicamento);

    IF NOT EXISTS(SELECT * FROM comisiones WHERE id_laboratorio = v_id_laboratorio) THEN
            INSERT INTO comisiones
            VALUES(v_id_laboratorio, (SELECT laboratorio FROM laboratorios WHERE
id_laboratorio = v_id_laboratorio), NEW.fecha_indicacion, v_precio, v_precio, v_precio * 0.01);

    ELSE
            UPDATE comisiones
            SET fecha = NEW.fecha_indicacion,
                    precio = v_precio,
                    monto_total = monto_total + v_precio,
                    comision = (monto_total + v_precio) * 0.01
            WHERE id_laboratorio = v_id_laboratorio;
    END IF;
    RETURN NEW;
END;
$comision_agrega$ LANGUAGE plpgsql

CREATE TRIGGER comision_agrega AFTER INSERT ON tratamiento FOR EACH ROW
EXECUTE PROCEDURE comision_nueva();
```

h)

```
CREATE OR REPLACE FUNCTION comision_baja()
RETURNS TRIGGER AS $comision_borra$
DECLARE
    v_id_laboratorio smallint;
    v_precio numeric(8,2);
BEGIN
    CREATE TABLE IF NOT EXISTS comisiones(
            id_laboratorio SMALLINT,
            laboratorio VARCHAR(100),
            fecha DATE,
            precio NUMERIC(8,2),
            monto_total NUMERIC(10,2),
            comision NUMERIC(8,2)
    );

    v_id_laboratorio := (SELECT id_laboratorio FROM medicamento WHERE id_medicamento =
OLD.id_medicamento);
    v_precio := (SELECT precio FROM medicamento WHERE id_medicamento =
OLD.id_medicamento);

    IF EXISTS(SELECT * FROM comisiones WHERE id_laboratorio = v_id_laboratorio) THEN
            UPDATE comisiones
            SET fecha = OLD.fecha_indicacion,
                        precio = v_precio,
                        monto_total = monto_total - v_precio,
                        comision = (monto_total - v_precio) * 0.01
            WHERE id_laboratorio = v_id_laboratorio;
    END IF;
    RETURN NEW;
END;
$comision_borra$ LANGUAGE plpgsql

CREATE TRIGGER comision_borra AFTER DELETE ON tratamiento FOR EACH ROW
EXECUTE PROCEDURE comision_baja();
```

Trabajo Práctico Nro. 7 - Ciclo 2022

Ejercicio 3: Realice las siguientes auditorías por trigger.

a)

```
CREATE TABLE audita_medicamento(
    id SERIAL,
    usuario VARCHAR(100),
    fecha DATE,
    id_medicamento INTEGER,
    medicamento VARCHAR(100),
    presentacion VARCHAR(100),
    precio NUMERIC(8,2),
    operacion CHAR,
    estado VARCHAR(10),
    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(), NEW.id_medicamento, NEW.nombre, NEW.presentacion,
NEW.precio, 'I','alta');
            RETURN NEW;
    END IF;
    IF (TG_OP = 'DELETE') THEN
            INSERT INTO audita_medicamento VALUES
            (DEFAULT, USER, NOW(), OLD.id_medicamento, OLD.nombre, OLD.presentacion,
OLD.precio, 'D','baja');
            RETURN OLD;
    END IF;
    IF (TG_OP = 'UPDATE') THEN
            INSERT INTO audita_medicamento VALUES
            (DEFAULT, USER, NOW(), OLD.id_medicamento, OLD.nombre, OLD.presentacion,
OLD.precio, 'U','antes');
            INSERT INTO audita_medicamento VALUES
            (DEFAULT, USER, NOW(), NEW.id_medicamento, NEW.nombre, NEW.presentacion,
NEW.precio, 'U','despues');
            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();
```

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

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

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

CREATE TABLE diagnostico_borrado
(
  id_paciente INTEGER,
  id_empleado INTEGER,
  fecha DATE,
  descripcion VARCHAR(100),
  id_patologia SMALLINT
);
```

```
CREATE OR REPLACE FUNCTION tablas_auditoria()
RETURNS TRIGGER AS $audita_tablas$
BEGIN
    IF (TG_RELNAME = 'consulta') THEN
            INSERT INTO consulta_borrada VALUES (OLD.*);
    END IF;
    IF (TG_RELNAME = 'estudiorealizado') THEN
            INSERT INTO estudio_borrado VALUES (OLD.*);
    END IF;
    IF (TG_RELNAME = 'diagnostico') THEN
            INSERT INTO diagnostico_borrado VALUES (OLD.*);
    END IF;

    INSERT INTO audita_tablas VALUES
            (DEFAULT, USER, NOW(), OLD.id_paciente, OLD.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 estudiorealizado
FOR EACH ROW EXECUTE PROCEDURE tablas_auditoria();

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