**Práctica de Triggers I**

1. Dada la tabla Products de la base de datos stores7 se requiere crear una tabla ***Products_historia_precios*** y crear un trigger que registre los cambios de precios que se hayan producido en la tabla ***Products***.

   Tabla ***Products_historia_precios***

   - Stock_historia_Id  Identity (PK)
   - Stock_num
   - Manu_code
   - fechaHora  (grabar fecha y hora del evento)
   - usuario (grabar usuario que realiza el cambio de precios)
   - unit_price_old
   - unit_price_new
   - estado char  default 'A'  check (estado IN ('A','I')

```
CREATE TABLE products_historia_precios (
        stock_historia_id int IDENTITY(1,1) PRIMARY KEY,
        stock_num smallint,
        manu_code char(3),
        fechaHora datetime,
        usuario varchar(20),
        unit_price_old decimal(6,2),
        unit_price_new decimal(6,2),
        estado char DEFAULT 'A' CHECK(estado IN('A','I')),
);

Opción 1
```

```
CREATE TRIGGER cambio_precios_TR ON products
AFTER UPDATE AS
BEGIN
       DECLARE @unit_price_old decimal(6,2)
       DECLARE @unit_price_new decimal(6,2)
       DECLARE @stock_num smallint
       DECLARE @manu_code char(3)

       DECLARE precios_stock CURSOR FOR
       SELECT i.stock_num,i.manu_code, i.unit_price, d.unit_price
                 FROM inserted i JOIN deleted d
             ON (i.stock_num = d.stock_num and i.manu_code = d.manu_code)
       WHERE i.unit_price != d.unit_price

       OPEN precios_stock

       FETCH NEXT FROM precios_stock
       INTO @stock_num, @manu_code, @unit_price_new, @unit_price_old

       WHILE @@FETCH_STATUS = 0
       BEGIN
              INSERT INTO products_historia_precios
                           (stock_num, manu_code, unit_price_new, unit_price_old,
fechaHora, usuario)
             VALUES
                           (@stock_num, @manu_code, @unit_price_new, @unit_price_old,
GETDATE(), SYSTEM_USER)

FETCH NEXT FROM precios_stock
```

```
        INTO @stock_num, @manu_code, @unit_price_new, @unit_price_old

        END

        CLOSE precios_stock
        DEALLOCATE precios_stock
END;
```

Opción 2

```
CREATE TRIGGER cambio_precios_TR ON products
AFTER UPDATE AS
BEGIN

   INSERT INTO products_historia_precios
   (stock_num, manu_code, unit_price_new, unit_price_old, fechaHora, usuario)
   SELECT i.stock_num,i.manu_code, i.unit_price, d.unit_price , GETDATE(), CURRENT_USER
   FROM inserted i JOIN deleted d
                    ON (i.stock_num = d.stock_num and i.manu_code = d.manu_code)
        WHERE i.unit_price != d.unit_price

END;
```

2. Crear un trigger sobre la tabla ***Products_historia_precios*** que ante un delete sobre la misma realice en su lugar un update del campo estado de 'A' a 'I' (inactivo).

```
CREATE TRIGGER delete_stock_historia ON products_historia_precios
INSTEAD OF DELETE AS
BEGIN
        DECLARE @stock_historia_id int

        DECLARE stock_historia_borrado CURSOR FOR
             SELECT stock_historia_id FROM deleted

        OPEN stock_historia_borrado

        FETCH NEXT FROM stock_historia_borrado
        INTO @stock_historia_id

        WHILE @@FETCH_STATUS = 0
        BEGIN
                 UPDATE products_historia_precios
                      SET estado = 'I' WHERE stock_historia_id = @stock_historia_id

                FETCH NEXT FROM stock_historia_borrado
                 INTO @stock_historia_id
        END
        CLOSE stock_historia_borrado
        DEALLOCATE stock_historia_borrado
END;
```

Opcion 2

```
CREATE TRIGGER deleteField
ON products_historia_precios
INSTEAD OF DELETE
AS
BEGIN
        UPDATE products_historia_precios set estado = 'I'
```

```
        where stock_historia_id in (select stock_historia_id from deleted);
END
```

3. Validar que sólo se puedan hacer inserts en la tabla Products en un horario entre las 8:00 AM y 8:00 PM. En caso contrario enviar un error por pantalla.

   OPCION 1

```
CREATE TRIGGER inserts_stock ON products
INSTEAD OF INSERT
AS
BEGIN
    IF(DATEPART(HOUR, GETDATE()) BETWEEN 8 AND 20)
    BEGIN
            INSERT INTO products
                    (stock_num, manu_code, unit_price, unit_code)
                        SELECT stock_num, manu_code, unit_price, unit_code
                          FROM inserted
    END
    ELSE
    BEGIN
           RAISERROR('Maestro que hace a esta hora laburando?', 16, 1)
    END
END;
```

   OPCION 2

```
CREATE or ALTER TRIGGER inserts_stock ON products
AFTER INSERT AS
BEGIN
       IF(DATEPART(HOUR, GETDATE()) NOT BETWEEN 8 AND 20)
       BEGIN
           Rollback
           THROW 50000,'Maestro que hace a esta hora laburando?',1
       END
END;
```

4. Crear un trigger que ante un borrado sobre la tabla ***ORDERS*** realice un borrado en cascada sobre la tabla ***ITEMS***, validando que sólo se borre 1 orden de compra.
   Si detecta que están queriendo borrar más de una orden de compra, informará un error y abortará la operación.

```
CREATE TRIGGER delete_orders_and_items ON orders
INSTEAD OF DELETE AS
BEGIN
    DECLARE @order_num smallint

    IF((SELECT COUNT(*) FROM deleted) > 1)
    BEGIN
           THROW 50000, 'No se pueden eliminar mas de una orden a la vez', 1
    END
    ELSE
    BEGIN
           SELECT @order_num = order_num FROM deleted;

           DELETE FROM items   WHERE order_num = @order_num;
           DELETE FROM orders WHERE order_num = @order_num;
```

```
    END
END;
```

5. Crear un trigger de insert sobre la tabla ***ítems*** que al detectar que el código de fabricante (manu_code) del producto a comprar no existe en la tabla ***manufact***, inserte una fila en dicha tabla con el manu_code ingresado, en el campo manu_name la descripción 'Manu Orden 999' donde 999 corresponde al nro. de la orden de compra a la que pertenece el ítem y en el campo lead_time el valor 1.

```
CREATE TRIGGER insert_items ON items
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @manu_code char(3)
    DECLARE @order_num smallint

    DECLARE items_insertados CURSOR FOR
    SELECT manu_code, order_num FROM inserted

    OPEN items_insertados

    FETCH NEXT FROM items_insertados
    INTO @manu_code, @order_num

    WHILE @@FETCH_STATUS = 0
    BEGIN
           IF NOT EXISTS (SELECT * FROM manufact WHERE manu_code = @manu_code)
           BEGIN
                  INSERT INTO manufact(manu_code, manu_name, lead_time)
                  VALUES(@manu_code, 'Manu orden ' + trim(str(@order_num)), 1)
           END

           FETCH NEXT FROM items_insertados
           INTO @manu_code, @order_num

    END

    CLOSE items_insertados
    DEALLOCATE items_insertados

    INSERT INTO items(item_num, order_num, manu_code, stock_num, quantity, unit_price)
                SELECT item_num, order_num, manu_code, stock_num, quantity, unit_price
                  FROM  inserted

END;
```

6. Crear tres triggers (Insert, Update y Delete) sobre la tabla ***Products*** para replicar todas las operaciones en la tabla ***Products _replica***, la misma deberá tener la misma estructura de la tabla ***Products***.

```
CREATE TABLE Products_replica(
       stock_num smallint,
       manu_code char(3),
       unit_price decimal(6,2),
       unit_code smallint,
       constraint pk_products_replica
          primary key (stock_num, manu_code));

CREATE TRIGGER replica_insert ON products
AFTER INSERT
AS
```

```
BEGIN
	INSERT INTO Products_replica
	(stock_num, manu_code, unit_price, unit_code)
	SELECT stock_num, manu_code, unit_price, unit_code FROM inserted
END;

CREATE TRIGGER replica_delete ON products
AFTER DELETE
AS
BEGIN
	DELETE sr FROM Products_replica sr
	JOIN deleted d ON (sr.stock_num = d.stock_num AND sr.manu_code = d.manu_code)
END;
```

```
CREATE TRIGGER replica_update ON products
AFTER UPDATE AS
BEGIN
	UPDATE sr SET sr.unit_price = i.unit_price,
		      sr.unit_code = i.unit_code
	FROM Products_replica sr JOIN inserted i
		ON (sr.stock_num = i.stock_num AND sr.manu_code = i.manu_code)
END;
```

7. Crear la vista ***Productos_x_fabricante*** que tenga los siguientes atributos:
   ***Stock_num, description, manu_code, manu_name, unit_price***

   Crear un trigger de Insert sobre la vista anterior que ante un insert, inserte una fila en la tabla Products, pero si el manu_code no existe en la tabla manufact, inserte además una fila en dicha tabla con el campo lead_time en 1.

```
CREATE VIEW productos_x_fabricante AS
SELECT p.stock_num, p.manu_code, tp.description, m.manu_name, p.unit_price
 FROM products p JOIN manufact m ON p.manu_code = m.manu_code
		 JOIN product_types tp on p.stock_num = tp.stock_num;
```

```
CREATE TRIGGER insert_productos_x_fabricante_TR ON productos_x_fabricante
INSTEAD OF INSERT AS
BEGIN
	DECLARE @stock_num smallint
	DECLARE @manu_code char(3)
	DECLARE @description varchar(15)
	DECLARE @manu_name varchar(15)
	DECLARE @unit_price decimal(6,2)

	DECLARE insert_cursor CURSOR FOR
		SELECT stock_num, manu_code, description, manu_name, unit_price
		  FROM inserted

	OPEN insert_cursor

	FETCH NEXT FROM insert_cursor
	INTO @stock_num, @manu_code, @description, @manu_name, @unit_price

	WHILE @@FETCH_STATUS = 0
	BEGIN
		IF NOT EXISTS (SELECT 1 FROM manufact WHERE manu_code = @manu_code)
		BEGIN
			INSERT INTO manufact(manu_code, manu_name, lead_time)
			VALUES(@manu_code, @manu_name, 1)
		END
```

```
            INSERT INTO products (stock_num, manu_code, unit_price)
            VALUES(@stock_num, @manu_code, @unit_price)

            FETCH NEXT FROM insert_cursor
            INTO @stock_num, @manu_code, @description, @manu_name, @unit_price
        END
        CLOSE insert_cursor
        DEALLOCATE insert_cursor
END;
```