PARCIAL 14-11-2018 · PARTE 1 - (c) QUERY COMPLEJO

- **Opción 1**

```
SELECT c.customer_num,
       c.fname + ' ' + c.lname AS nombreyapellido,
       s.sname,
       SUM(i.quantity * i.total_price) AS montoCliente,
       r.customer_num,
       r.nombreyapellido,
       r.montoReferente

FROM customer c JOIN state s ON (c.state = s.code)

                JOIN orders o ON (c.customer_num = o.customer_num)

                JOIN items i ON (i.order_num = o.order_num)

                LEFT JOIN (SELECT c.customer_num,

                                  c.fname + ' ' + c.lname AS nombreyapellido,

                                 SUM(i.quantity * i.total_price) montoReferente

                           FROM customer c

                               LEFT JOIN orders o ON (c.customer_num =

o.customer_num AND YEAR(o.order_date) = 2015)

                               LEFT JOIN items i ON (i.order_num = o.order_num)

                           GROUP BY c.customer_num,

                                    c.fname + ' ' + c.lname

                          ) r ON (c.customer_num_referedBy = r.customer_num)

WHERE o.order_date BETWEEN '2015-01-01' AND '2015-12-31'

GROUP BY c.customer_num,
         c.fname + ' ' + c.lname,
            s.sname,
            r.customer_num,
            r.nombreyapellido,
            r.montoReferente

HAVING SUM(i.quantity * i.total_price) > COALESCE(r.montoReferente, 0)

ORDER BY 3, 4 DESC
```

- **Opción 2**

```
SELECT a.customer_num,
       a.nombreyapellido,
       a.sname,
       a.montoCliente,
       B.nombreyapellido,
       B.montoReferente

FROM (SELECT c.customer_num,
             c.fname + ' ' + c.lname AS nombreyapellido,
             s.sname,
             SUM(i.quantity * i.total_price) AS montoCliente,
             c.customer_num_referedBy

      FROM customer c JOIN state s ON (c.state = s.code)
                      JOIN orders o ON (c.customer_num = o.customer_num)
                      JOIN items i ON (i.order_num = o.order_num)

      WHERE YEAR(o.order_date) = 2015

      GROUP BY c.customer_num
               c.fname + ' ' + c.lname,
               s.sname,
               c.customer_num_referedBy

      ) a

       LEFT JOIN (SELECT c.customer_num,
                         c.fname + ' ' + c.lname AS nombreyapellido,
                         SUM(i.quantity * i.total_price) AS montoReferente

                  FROM customer c LEFT JOIN orders o ON (c.customer_num =

o.customer_num AND YEAR(o.order_date) = 2015)

                                  LEFT JOIN items i ON (i.order_num =

o.order_num)

                  GROUP BY c.customer_num,

                           c.fname + ' ' + c.lname) b

      ON (a.customer_num_referedBy = B.customer_num)

WHERE a.monto_cliente > COALESCE(b.montoReferente, 0)

ORDER BY 3, 4 DESC
```

- **Opción 3**

```
 SELECT c1.customer_num,
        nombreyapellido,
        c1.sname,
        montoTotal,
        c2.fname + ' ' + c2.lname,
        SUM(i.quantity * i.total_price) montoTotalRef

FROM (SELECT c.customer_num,
             c.fname + ' ' + c.lname AS nombreyapellido,
             s.sname,
             c.customer_num_referedBy,
             SUM(i.quantity * i.total_price) AS montoTotal

      FROM customer c JOIN orders o ON (c.customer_num = o.customer_num)
                      JOIN items i ON (i.order_num = o.order_num)
                      JOIN state s ON (c.state = s.code)

      WHERE o.order_date BETWEEN '2015-01-01' AND '2015-12-31'

      GROUP BY c.customer_num
               c.fname,
               c.lname,
               s.sname,
               c.customer_num_referedBy
      ) c1

           LEFT JOIN customer c2 ON (c1.customer_num_referedBy =

c2.customer_num)

           LEFT JOIN orders o ON (c2.customer_num = o.customer_num AND

YEAR(o.order_date) = 2015)

           LEFT JOIN items i ON (i.order_num = o.order_num)

WHERE YEAR(o.order_date) = 2015

GROUP BY c1.customer_num,
         nombreyapellido,
         c1.sname,
         c1.montoTotal,
         c2.fname,
         c2.lname

HAVING montoTotal > COALESCE(SUM(i.quantity * i.total_price), 0)

ORDER BY 3, 4 DESC
```

PARCIAL 14-11-2018 · PARTE 2 - (d) STORED PROCEDURE

```
CREATE TABLE CuentaCorriente (
    id BIGINT IDENTITY,
    fechaMovimiento DATETIME,
    customer_num SMALLINT REFERENCES customer,
    order_num INT REFERENCES orders,
    importe DECIMAL(12,2)
)



CREATE PROCEDURE cargarEnCuentaCorriente
AS
BEGIN
    DECLARE @customer_num SMALLINT,
            @order_num SMALLINT,
            @order_date DATETIME,
            @total DECIMAL(10,2),
            @paid_date DATETIME
    DECLARE CURSOR cursor1 FOR
        SELECT c.customer_num,
               o.order_num,
               o.order_date,
               o.paid_date,
               SUM(i.quantity * i.total_price)
        FROM orders o JOIN items i ON (o.order_num = i.order_num)

    OPEN cursor1

    FETCH NEXT FROM cursor1
        INTO @customer_num, @order_num, @order_date, @paid_date, @total

    WHILE (@@FETCH_STATUS = 0) BEGIN
       INSERT INTO CuentaCorriente (fechaMovimiento, customer_num, order_num, importe)
            VALUES (@order_date, @customer_num, @order_num, @total)
        IF (@paid_date IS NOT NULL) BEGIN
       INSERT INTO CuentaCorriente (fechaMovimiento, customer_num, order_num, importe)
                VALUES (@paid_date, @customer_num, @order_num, @total)
        END
            FETCH NEXT FROM cursor1
                INTO @customer_num, @order_num, @order_date, @paid_date, @total
    END

    CLOSE cursor1

    DEALLOCATE cursor1

END
```

PARCIAL 14-11-2018 · PARTE 2 - (3) TRIGGER

```
CREATE TRIGGER triggerInsert ON ProductosV
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @stock_num SMALLINT,
            @description VARCHAR(15),
            @manu_code CHAR(3),
            @unit_price DECIMAL(8,2),
            @unit_code SMALLINT,
            @unit_descr VARCHAR(15)
    DECLARE CURSOR cursor2 FOR
        SELECT * FROM inserted
    OPEN cursor2
    FETCH NEXT FROM cursor2
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr
    BEGIN TRANSACTION
    WHILE (@@FETCH_STATUS = 0) BEGIN
        IF NOT EXISTS (SELECT * FROM manufact m
                       WHERE m.manu_code = @manu_code) BEGIN
            RAISERROR('No existe el fabricante' + CAST(@manu_code AS VARCHAR), 12, 1)
            ROLLBACK TRANSACTION
        END
        IF NOT EXISTS (SELECT * FROM units u
                       WHERE u.unit_code = @unit_code) BEGIN
            INSERT INTO units (unit_code, unit_descr)
                VALUES (@unit_code, @unit_descr)
        END
        IF NOT EXISTS (SELECT * FROM product_types tp
                       WHERE tp.stock_num = @stock_num) BEGIN
            INSERT INTO product_types (stock_num, description)
                VALUES (@stock_num, @description)
        END
        INSERT INTO products SELECT * FROM inserted
        COMMIT TRANSACTION
        FETCH NEXT FROM cursor2
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr
    END
    CLOSE cursor2
    DEALLOCATE cursor2
END
```

```
CREATE TRIGGER triggerUpdate
ON ProductosV
INSTEAD OF UPDATE
AS BEGIN
    DECLARE @stock_num SMALLINT,
            @description VARCHAR(15),
            @manu_code CHAR(3),
            @unit_price DECIMAL(8,2),
            @unit_code SMALLINT,
            @unit_descr VARCHAR(15)
    DECLARE CURSOR cursor3 FOR
        SELECT * FROM inserted
    OPEN cursor3
    FETCH NEXT FROM cursor3
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr
    BEGIN TRANSACTION
      WHILE (@@FETCH_STATUS = 0) BEGIN
          IF NOT EXISTS (SELECT * FROM units u
                       WHERE u.unit_code = @unit_code) BEGIN
            INSERT INTO units (unit_code, unit_descr)
                VALUES (@unit_code, @unit_descr)
        END
        IF EXISTS (SELECT * FROM products p JOIN manufact m ON (p.manu_code =
m.manu_code)
                    WHERE m.manu_code = @manu_code) BEGIN
            UPDATE products SET unit_price = @unit_price,
                                unit_code = @unit_code
            WHERE manu_code = @manu_code
              AND stock_num = @stock_num
        END
        ELSE BEGIN
            RAISERROR('No existe la combinación producto-fabricante', 12, 1)
            ROLLBACK TRANSACTION
        END
        COMMIT TRANSACTION
        FETCH NEXT FROM cursor3
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr
    END
    CLOSE cursor3
      DEALLOCATE cursor3
END
```

Otra solución: los dos triggers en uno

```
CREATE TRIGGER productosV_Tr ON ProductosV

INSTEAD OF INSERT, UPDATE

AS BEGIN

    DECLARE @stock_num    SMALLINT,         @unit_price   DECIMAL(6,2),
            @description  VARCHAR (15),     @unit_code    SMALLINT,
            @manu_code    CHAR(3),          @unit_descr   VARCHAR(15)

    DECLARE curinsertados CURSOR FOR
        SELECT stock_num, description, manu_code, unit_price, unit_code, unit_descr
        FROM inserted

    OPEN curinsertados

    FETCH NEXT FROM curinsertados
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr

    WHILE (@@FETCH_STATUS = 0)

    BEGIN

        BEGIN TRY

            BEGIN TRAN

            IF NOT EXISTS (SELECT TOP 1 NULL FROM deleted) BEGIN -- es un INSERT

                IF NOT EXISTS (SELECT 1 FROM manufact m
                               WHERE m.manu_code = @manu_code) BEGIN

                    RAISERROR('No existe el fabricante', 16, 1)

                END

                IF NOT EXISTS (SELECT 1 FROM units u
                               WHERE u.unit_code = @unit_code) BEGIN

                    INSERT INTO units (unit_code, unit_descr)
                        VALUES (@unit_code, @unit_descr)

                END

                IF NOT EXISTS (SELECT 1 FROM product_types pt
                               WHERE pt.stock_num = @stock_num) BEGIN

                    INSERT INTO product_types (stock_num, description)
                        VALUES (@stock_num, @description)

                END

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

            END
```

```
            ELSE BEGIN -- es un UPDATE

                IF NOT EXISTS (SELECT 1 FROM products p
                               WHERE p.stock_num = @stock_num
                                 AND p.manu_code = @manu_code) BEGIN

                    RAISERROR('No existe el Producto', 16, 2)

                END

                IF NOT EXISTS (SELECT 1 FROM units u
                               WHERE u.unit_code = @unit_code) BEGIN

                    INSERT INTO units (unit_code, unit_descr)
                        VALUES (@unit_code, @unit_descr)

                END

                UPDATE products
                    SET unit_price = @unit_price,
                        unit_code = @unit_code
                    WHERE stock_num = @stock_num
                      AND manu_code = @manu_code;

            END

            COMMIT TRANSACTION

        END TRY

        BEGIN CATCH

            ROLLBACK TRANSACTION

        END CATCH

        FETCH NEXT FROM curinsertados
       INTO @stock_num, @description, @manu_code, @unit_price, @unit_code, @unit_descr

    END

    CLOSE curinsertados

    DEALLOCATE curinsertados

END
```