**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,

@description VARCHAR (15),

@manu\_code CHAR(3),

@unit\_price DECIMAL(6,2),

@unit\_code SMALLINT,

@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