Práctica de Triggers II

1. Se pide: Crear un trigger que valide que ante un insert de una o más filas en la tabla ítems, realice la siguiente validación:
   - Si la orden de compra a la que pertenecen los ítems ingresados corresponde a clientes del estado de California, se deberá validar que estas órdenes puedan tener como máximo 5 registros en la tabla ítem.
   - Si se insertan más ítems de los definidos, el resto de los ítems se deberán insertar en la tabla ***items_error*** la cual contiene la misma estructura que la tabla ítems más un atributo fecha que deberá contener la fecha del día en que se trató de insertar.

   Ej. Si la Orden de Compra tiene 3 items y se realiza un insert masivo de 3 ítems más, el trigger deberá insertar los 2 primeros en la tabla ítems y el restante en la tabla ítems_error.
   Supuesto: En el caso de un insert masivo los items son de la misma orden.

```
create trigger Tr_temaA
on items
instead of insert
AS
BEGIN
  declare @stock_num  smallint, @order_num smallint, @item_num smallint,
          @quantity smallint
  declare @unit_price decimal(8,2)
  declare @manu_code char(3),@state char(2)

  declare c_items cursor for select i.item_num, i.order_num, stock_num,
                                    manu_code, quantity, unit_price, state
                               from inserted i JOIN orders o
                                           ON (i.order_num=o.order_num)
                                               JOIN customer c
                                    ON (o.customer_num=c.customer_num);
   open c_items
   fetch from c_items
         into @item_num,@order_num,@stock_num,@manu_code,
                @quantity, @unit_price, @state

   while @@fetch_status=0
   BEGIN
       if @state='CA'
       begin
            if (select COUNT(*) FROM items where order_num=@order_num) < 5
            begin
                 INSERT INTO items (i.item_num, i.order_num, stock_num,
                                    manu_code, quantity, unit_price)
                     VALUES(@item_num,@order_num,@stock_num,@manu_code,
                                 @quantity,@unit_price)

             end
             else
             begin
                 INSERT INTO items_error
                         VALUES(@item_num, @order_num, @stock_num,
                                @manu_code, @quantity, @unit_price,
                                getDate())
             end
        end
        else
        begin
             INSERT INTO items VALUES(@item_num, @order_num, @stock_num,
                                      @manu_code, @quantity, @unit_price)
        end
```

```
    fetch from c_items into @item_num, @order_num, @stock_num,
                            @manu_code, @quantity, @unit_price, @state;
    END
   close c_items
   deallocate c_items
END

----pruebas del trigger

CREATE TABLE [dbo].[items_error](
   [item_num] [smallint] NOT NULL,
   [order_num] [smallint] NOT NULL,
   [stock_num] [smallint] NOT NULL,
   [manu_code] [char](3) NOT NULL,
   [quantity] [smallint] NULL DEFAULT ((1)),
   [unit_price] [decimal](8, 2) NULL,
   [fecha] [datetime] NULL
)

select * from items where order_num in(
select order_num from orders o, customer c
where o.customer_num=c.customer_num
and c.state='CA')

insert into items values (14,1003,9,'ANZ',1,10)
insert into items values (15,1003,9,'ANZ',1,10)
insert into items values (16,1003,9,'ANZ',1,10)
insert into items values (17,1003,9,'ANZ',1,10)
insert into items values (18,1003,9,'ANZ',1,10)

select * from items_error
```

2. Triggers Dada la siguiente vista

```
CREATE VIEW ProdPorFabricante AS
            SELECT m.manu_code, m.manu_name, COUNT(*)
             FROM manufact m INNER JOIN products p
                      ON (m.manu_code = p.manu_code)
             GROUP BY manu_code, manu_name;
```

Crear un trigger que permita ante un insert en la vista ProdPorFabricante insertar una fila en la tabla manufact.
Observaciones: el atributo leadtime deberá insertarse con un valor default 10
El trigger deberá contemplar inserts de varias filas, por ej. ante un INSERT / SELECT.

```
CREATE TRIGGER insFabric ON ProdPorFabricante
INSTEAD OF INSERT AS
BEGIN
  INSERT INTO manufact (manu_code, manu_name, lead_time)
          select manu_code,manu_name,10
            from inserted;
END
```

3. Crear un trigger que ante un INSERT o UPDATE de una o más filas de la tabla Customer, realice la siguiente validación.

- La cuota de clientes correspondientes al estado de California es de 20, si se supera dicha cuota se deberán grabar el resto de los clientes en la tabla customer_pend.
- Validar que si de los clientes a modificar se modifica el Estado, no se puede superar dicha cuota.

Si por ejemplo el estado de CA cuenta con 18 clientes y se realiza un update o insert masivo de 5 clientes con estado de CA, el trigger deberá modificar los 2 primeros en la tabla customer y los restantes grabarlos en la tabla customer_pend.
La tabla customer_pend tendrá la misma estructura que la tabla customer con un atributo adicional fechaHora que deberá actualizarse con la fecha y hora del día.

```
CREATE TABLE customer_pend (
   [fecha_hora] [datetime] NOT NULL,
   [customer_num] [smallint] NOT NULL,
   [fname] [varchar](15) NULL,
   [lname] [varchar](15) NULL,
   [company] [varchar](20) NULL,
   [address1] [varchar](20) NULL,
   [address2] [varchar](20) NULL,
   [city] [varchar](15) NULL,
   [state] [char](2) NULL,
   [zipcode] [char](5) NULL,
   [phone] [varchar](18) NULL,
   [customer_num_referedBy] [smallint] NULL,
   [status] [char](1) NULL
)

DROP TRIGGER TR_customer

CREATE TRIGGER TR_customer
ON customer
INSTEAD OF INSERT, UPDATE
AS
BEGIN

   DECLARE @customer_num SMALLINT
   DECLARE @fname VARCHAR(15)
   DECLARE @lname VARCHAR(15)
   DECLARE @company VARCHAR(20)
   DECLARE @address1 VARCHAR(20)
   DECLARE @address2 VARCHAR(20)
   DECLARE @city VARCHAR(15)
   DECLARE @state CHAR(2), @old_state CHAR(2)
   DECLARE @zipcode CHAR(5)
   DECLARE @phone VARCHAR(18)
   DECLARE @customer_num_referedBy SMALLINT
   DECLARE @status CHAR(1)

   DECLARE nuevos CURSOR FOR (SELECT i.*, d.state FROM inserted i LEFT JOIN
deleted d
                                                        ON (i.customer_num
= d.customer_num))

   OPEN nuevos

   FETCH NEXT FROM nuevos INTO @customer_num, @fname, @lname, @company,
@address1, @address2, @city, @state, @zipcode, @phone,
@customer_num_referedBy, @status, @old_state

   WHILE @@FETCH_STATUS = 0
   BEGIN
```

```
        IF(@old_state IS NULL) -- si es NULL, se trata de una operacion
INSERT
        BEGIN
              IF(@state = 'CA') -- si es de CA, se verifica el cupo de 20
              BEGIN
                    IF((SELECT COUNT(*) FROM customer WHERE state = 'CA')
< 20) -- se verifica el cupo de 20
                    BEGIN
                          INSERT INTO customer
                          VALUES (@customer_num, @fname, @lname, @company,
@address1, @address2, @city, @state, @zipcode, @phone,
@customer_num_referedBy, @status)
                    END
                    ELSE
                    BEGIN
                          INSERT INTO customer_pend
                          VALUES (GETDATE(), @customer_num, @fname,
@lname, @company, @address1, @address2, @city, @state, @zipcode, @phone,
@customer_num_referedBy, @status)
                    END
              END
              ELSE -- si no es de CA, el insert es normal
              BEGIN
                    INSERT INTO customer
                    VALUES (@customer_num, @fname, @lname, @company,
@address1, @address2, @city, @state, @zipcode, @phone,
@customer_num_referedBy, @status)
              END
        END
        ELSE
        BEGIN
              IF(@state = 'CA') -- si es de CA, se verifica el cupo de 20
              BEGIN
                    IF((SELECT COUNT(*) FROM customer WHERE state = 'CA')
< 20) -- se verifica el cupo de 20
                    BEGIN
                          UPDATE customer
                          SET customer_num = @customer_num, fname =
@fname, lname = @lname, company = @company, address1 = @address1,
                                address2 = @address2, city = @city, state
= @state, zipcode = @zipcode, phone = @phone, customer_num_referedBy =
@customer_num_referedBy, status = @status
                          WHERE customer_num = @customer_num
                    END
                    ELSE
                    BEGIN
                          INSERT INTO customer_pend
                          VALUES (GETDATE(), @customer_num, @fname,
@lname, @company, @address1, @address2, @city, @old_state, @zipcode,
@phone, @customer_num_referedBy, @status)
                    END
              END
              ELSE -- si no es de CA, el UPDATE es normal
              BEGIN
                    UPDATE customer
                    SET customer_num = @customer_num, fname = @fname,
lname = @lname, company = @company, address1 = @address1,
                    address2 = @address2, city = @city, state = @state,
zipcode = @zipcode, phone = @phone, customer_num_referedBy =
@customer_num_referedBy, status = @status
                    WHERE customer_num = @customer_num
              END
        END
```

```
        FETCH NEXT FROM nuevos INTO @customer_num, @fname, @lname,
@company, @address1, @address2, @city, @state, @zipcode, @phone,
@customer_num_referedBy, @status, @old_state
  END

  CLOSE nuevos

  DEALLOCATE nuevos
END

■  prueba
select count(*) from customer where state='CA'

select customer_num, state
  from customer where customer_num between 123 and 126

update customer set state='CA' where customer_num between 122 and 126

select * from customer_updates_pend
```

4. **Dada la siguiente vista**

```
CREATE VIEW ProdPorFabricanteDet AS
    SELECT m.manu_code, m.manu_name, pt.stock_num, pt.description
      FROM manufact m LEFT OUTER JOIN products p ON m.manu_code = p.manu_code
                LEFT OUTER JOIN product_types pt ON p.stock_num = pt.stock_num;
```

Se pide: Crear un trigger que permita ante un DELETE en la vista ProdPorFabricante borrar los datos en la tabla manufact pero sólo de los fabricantes cuyo campo description sea NULO (o sea que no tienen stock).
Observaciones: El trigger deberá contemplar borrado de varias filas mediante un DELETE masivo. En ese caso sólo borrará de la tabla los fabricantes que no tengan productos en stock, borrando los demás.

```
CREATE TRIGGER delFabric
ON ProdPorFabricanteDet
INSTEAD OF DELETE AS
BEGIN
  DELETE FROM manufact WHERE manu_code IN
             (select manu_code from deleted where description IS NULL);
END
```

5. **Se pide crear un trigger que permita ante un delete de una <u>sola</u> fila en la vista ordenesPendientes valide:**
   - **Si el cliente asociado a la orden tiene sólo esa orden pendiente de pago (paid_date IS NULL), no permita realizar la Baja, informando el error.**
   - **Si la Orden tiene más de un ítem asociado, no permitir realizar la Baja, informando el error.**
   - **Ante cualquier otra condición borrar la Orden con sus ítems asociados, respetando la integridad referencial.**

Estructura de la vista: customer_num, fname, lname, Company, order_num, order_date WHERE paid_date IS NULL.

```
CREATE VIEW ordenesPendientes AS
SELECT c.customer_num, fname, lname, company,
       o.order_num, order_date
```

```
  FROM customer c JOIN orders o ON c.customer_num=o.customer_num
 WHERE paid_date IS NULL;

SELECT * FROM ordenesPendientes;
--

CREATE TRIGGER borrarOrden ON ordenesPendientes
instead of delete AS
BEGIN
    declare @cantidadOrdenesPendientes int
    declare @cantidadItems int

    select @cantidadOrdenesPendientes=COUNT(o.order_num)
      from orders o JOIN deleted d ON o.customer_num = d.customer_num
       and o.paid_date is null

    select @cantidadItems = COUNT(i.item_num)
      from items I join deleted d on i.order_num = d.order_num

    if(@cantidadItems > 1)
        THROW 50001, 'Error: La Orden posee mas de un item', 1

    if(@cantidadOrdenesPendientes = 1)
        THROW 50002,'Error: El cliente tiene solo 1 orden pendiente', 1

    delete from items where order_num=(select order_num from deleted)
    delete from orders where order_num=(select order_num from deleted)
END

-- PRUEBAS DE FUNCIONAMIENTO
-- EJECUTA CORRECTO
update orders set paid_date = null where customer_num=117

insert into orders (order_num,customer_num) VALUES (8984,117)
insert into items (order_num,item_num,stock_num,manu_code,quantity)
       VALUES (8984,1,1,'HRO',1)

DELETE FROM ordenesPendientes where order_num =8984

-- DA ERROR POR TENER SÒLO 1 ORDEN PENDIENTE DE PAGO

insert into orders (order_num,customer_num)values (8987,106)

DELETE FROM ordenesPendientes where order_num=8987



-- DA ERROR POR TENER MÀS DE UN ITEM ASOCIADO A LA ORDEN...

insert into orders (order_num,customer_num) VALUES (8985,101)
insert into items (order_num,item_num,stock_num,manu_code,quantity)
       VALUES (8985,1,1,'HRO',1)
insert into items (order_num,item_num,stock_num,manu_code,quantity)
       VALUES (8985,2,1,'SMT',1)

DELETE FROM ordenesPendientes where order_num=8985

select c.customer_num,count(order_num)
  from orders o right outer join customer c
    on o.customer_num=c.customer_num and paid_date is null
 group by c.customer_num
```