TSQL

10. Crear el/los objetos de base de datos que:  
Ante el intento de borrar un artículo verifique que no exista stock.  
Si es así, lo borre.  
En caso contrario, emita un mensaje de error

DROP TRIGGER BorrarProducto;

GO

CREATE TRIGGER BorrarProducto ON Producto INSTEAD OF DELETE

AS

BEGIN

IF EXISTS (SELECT 1

FROM STOCK s

join deleted d on d.prod\_codigo = s.stoc\_producto

WHERE s.stoc\_cantidad > 0

)

BEGIN

-- Si tiene stock, no permite el borrado

--ROLLBACK TRANSACTION (NO FUNCIONA)

THROW 50000, 'No se puede borrar el producto porque contiene stock.', 1

END

ELSE

BEGIN

DELETE FROM STOCK

WHERE stoc\_producto IN (SELECT d.prod\_codigo

FROM deleted d);

DELETE FROM Producto

WHERE prod\_codigo IN (SELECT d.prod\_codigo

FROM deleted d);

END

END

GO

select \* from stock;

DELETE from producto WHERE prod\_codigo= '00001508'

**TSQL**

11. Cree el/los objetos de base de datos necesarios para que:  
Dado un código de empleado se retorne la cantidad de empleados que tiene a (directa o indirectamente).  
Solo contar aquellos empleados (directos o indirectos) que tengan un código mayor que su jefe directo.

CREATE FUNCTION dbo.empleadosACargo (@empleado numeric(6,0))

returns int

AS

BEGIN

DECLARE @CantEmpleadoACargo int = 0

DECLARE @jefeAux numeric(6,0) = @empleado

DECLARE @codeEmplAux numeric(6,0)

IF NOT EXISTS (SELECT \* FROM EMPLEADO WHERE empl\_jefe = @empleado)

BEGIN

return @CantEmpleadoACargo

END

SET @CantEmpleadoACargo = (

SELECT count(\*)

from empleado where empl\_jefe = @empleado and empl\_codigo > @empleado)

DECLARE cursor\_empleado CURSOR FOR SELECT e.empl\_codigo

FROM empleado e

WHERE e.empl\_jefe = @empleado

OPEN cursor\_empleado

FETCH NEXT from cursor\_empleado

INTO @JefeAux

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @CantEmpleadoACargo = @CantEmpleadoACargo + dbo.empleadosACargo(@JefeAux)

FETCH NEXT FROM cursor\_empleado

INTO @JefeAux

END

CLOSE cursor\_empleado

DEALLOCATE cursor\_empleado

RETURN @CantEmpleadoACargo

END

GO

TSQL  
13. Cree el/los objetos de base de datos necesarios para implantar la siguiente regla: “Ningún jefe puede tener un salario mayor al 20% de las suma de los salarios de sus empleados totales (directos + indirectos)”.  
Se sabe que en la actualidad dicha regla se cumple y que la base de datos es accedida por n aplicaciones de diferentes tipos y tecnologías.

CREATE TRIGGER ejercicio13 on empleado for UPDATE, DELETE

AS

BEGIN

if EXISTS (select 1

FROM inserted i

WHERE dbo.ej13(i.empl\_jefe)\*0.2 < (SELECT empl\_salario

FROM empleado

WHERE empl\_codigo = i.empl\_jefe))

BEGIN

print 'NO SE PUEDE ACTUALIZAR LOS EMPLEADOS PORQUE SUPERAN EL SALARIO'

ROLLBACK

END

IF EXISTS (SELECT 1

FROM inserted

WHERE dbo.ej13(empl\_codigo)\*0.2 < empl\_salario)

BEGIN

print 'NO SE PUEDE ACTUALIZAR LOS EMPLEADOS PORQUE SUPERAN EL SALARIO'

ROLLBACK

END

IF EXISTS (select 1

from deleted i

WHERE dbo.ej13(i.empl\_jefe)\*0.2 < (SELECT empl\_salario

FROM empleado

WHERE empl\_codigo = i.empl\_jefe))

BEGIN

print 'NO SE PUEDE BORRAR LOS EMPLEADOS PORQUE SUPERAN EL SALARIO'

ROLLBACK

END

END

GO

CREATE FUNCTION ej13 (@empleado numeric(6))

RETURNS decimal(12,2)

AS

BEGIN

DECLARE @salario decimal(12,2), @emp numeric(6), @sal decimal(12,2)

SELECT @salario = 0

IF (SELECT count(\*) FROM empleado WHERE empl\_jefe = @empleado) = 0

RETURN @salario

DECLARE c1 cursor for select empl\_codigo

from empleado

where empl\_jefe = @empleado

OPEN c1

fetch next into @emp, @sal

WHILE @@FETCH\_STATUS = 0

BEGIN

select @salario = @salario + @sal + dbo.ej11(@emp)

fetch next into @emp

END

CLOSE c1

DEALLOCATE c1

RETURN @salario

END

GO

TSQL

12. Cree el/los objetos de base de datos necesarios para que nunca un producto

pueda ser compuesto por sí mismo. Se sabe que en la actualidad dicha regla se

cumple y que la base de datos es accedida por n aplicaciones de diferentes tipos

y tecnologías. No se conoce la cantidad de niveles de composición existentes.

CREATE TRIGGER ejerc12 on composicion AFTER INSERT, update

AS

BEGIN

if (select sum(dbo.compuesto\_por(comp\_producto, comp\_componente))

FROM inserted) > 0

BEGIN

print 'EL PRODUCTO ESTA COMPUESTO POR SI MISMO'

ROLLBACK

END

END

GO

CREATE FUNCTION compuesto\_por (@prod char(8), @comp char(8))

RETURNS INT

AS

BEGIN

DECLARE @componente char(8)

IF @prod = @comp

RETURN 1

DECLARE c1 cursor for SELECT comp\_componente

FROM composicion

WHERE comp\_producto = @comp

OPEN c1

fetch next from c1 into @componente

WHILE @@FETCH\_STATUS = 0

BEGIN

IF dbo.compuesto\_por(@prod, @componente) = 1

BEGIN

CLOSE c1

DEALLOCATE c1

RETURN 1

END

fetch next from c1 into @componente

END

CLOSE c1

DEALLOCATE c1

RETURN 0

END

GO

select \* from Composicion

update Composicion set comp\_componente = '00001104'

where comp\_producto = '00001104' and

comp\_componente = '00001109'

GO

TSQL

16. Desarrolle el/los elementos de base de datos necesarios para que ante una venta

automaticamante se descuenten del stock los articulos vendidos. Se descontaran

del deposito que mas producto poseea y se supone que el stock se almacena

tanto de productos simples como compuestos (si se acaba el stock de los

compuestos no se arman combos)

En caso que no alcance el stock de un deposito se descontara del siguiente y asi

hasta agotar los depositos posibles. En ultima instancia se dejara stock negativo

en el ultimo deposito que se desconto.

create trigger ejercicio16 on item\_factura for insert

AS

BEGIN

declare @producto char(8),

@cantidad decimal(12,2),

@deposito char(2),

@depo\_ant char(2)

declare c\_item CURSOR FOR SELECT item\_producto,

item\_cantidad

FROM inserted

OPEN c\_item fetch next

FROM c\_item into @producto,

@cantidad

while @@FETCH\_STATUS = 0

BEGIN

if exists (SELECT \*

FROM composicion

WHERE comp\_producto = @producto)

BEGIN

declare @componente char(8),

@cantcomp decimal(12,2)

declare @depo\_cantidad decimal(12,2)

declare c\_comp CURSOR FOR SELECT comp\_componente,

comp\_cantidad\*@cantidad

FROM Composicion

WHERE comp\_producto = @producto

OPEN c\_comp fetch next

FROM c\_comp into @componente,

@cantcomp

while @@FETCH\_STATUS = 0

BEGIN

declare c\_deposito CURSOR FOR SELECT stoc\_deposito,

stoc\_Cantidad

FROM stock

WHERE @componente = stoc\_producto and

stoc\_cantidad > 0

ORDER BY stoc\_cantidad desc

OPEN c\_deposito FETCH NEXT

FROM c\_deposito into @deposito,

TSQL 16

@depo\_cantidad

while @@FETCH\_STATUS = 0 and

@cantidad > 0

BEGIN

IF @depo\_cantidad >= @cantidad

BEGIN UPDATE stock set stoc\_cantidad = stoc\_cantidad - @cantidad

WHERE stoc\_producto = @componente and stoc\_deposito = @deposito

SELECT @cantidad = 0

END

ELSE

BEGIN UPDATE stock set stoc\_cantidad = stoc\_cantidad - @depo\_cantidad

where stoc\_producto = @componente and stoc\_deposito = @deposito

SELECT @depo\_ant = @deposito

SELECT @cantidad = @cantidad - @depo\_cantidad

END

fetch next

FROM c\_deposito into @deposito,

@depo\_cantidad

END

update stock set stoc\_cantidad = stoc\_cantidad - @depo\_cantidad

WHERE stoc\_producto = @componente and stoc\_deposito = @depo\_ant

CLOSE c\_deposito

DEALLOCATE c\_deposito

END

END

ELSE

BEGIN

DECLARE c\_deposito CURSOR FOR SELECT stoc\_deposito,

stoc\_Cantidad

FROM stock

WHERE @producto = stoc\_producto and

stoc\_cantidad > 0

ORDER BY stoc\_cantidad DESC

OPEN c\_deposito fetch next

FROM c\_deposito into @deposito,

@depo\_cantidad

while @@FETCH\_STATUS = 0 and

@cantidad > 0

BEGIN

IF @depo\_cantidad >= @cantidad

BEGIN

UPDATE stock set stoc\_cantidad = stoc\_cantidad - @cantidad

WHERE stoc\_producto = @producto and

stoc\_deposito = @deposito

SELECT @cantidad = 0

END

ELSE

BEGIN update stock set stoc\_cantidad = stoc\_cantidad - @depo\_cantidad

WHERE stoc\_producto = @producto and

stoc\_deposito = @deposito

TSQL - 16

SELECT @depo\_ant = @deposito

SELECT @cantidad = @cantidad - @depo\_cantidad

END

FETCH NEXT FROM c\_deposito into @deposito,

@depo\_cantidad

END

update stock set stoc\_cantidad = stoc\_cantidad - @depo\_cantidad

WHERE stoc\_producto = @producto and

stoc\_deposito = @depo\_ant

CLOSE c\_deposito

DEALLOCATE c\_deposito

END

FETCH NEXT FROM c\_item into @producto,

@cantidad

END

close c\_item

DEALLOCATE c\_item

END

GO

TSQL

17. Sabiendo que el punto de reposicion del stock es la menor cantidad de ese objeto

que se debe almacenar en el deposito y que el stock maximo es la maxima

cantidad de ese producto en ese deposito, cree el/los objetos de base de datos

necesarios para que dicha regla de negocio se cumpla automaticamente. No se

conoce la forma de acceso a los datos ni el procedimiento por el cual se

incrementa o descuenta stock

create trigger ejercic17 on stock for insert, update

as

begin

if exists (select \*

from inserted

where stoc\_cantidad > stoc\_stock\_maximo or

stoc\_Cantidad < stoc\_punto\_reposicion)

begin

print 'NO SE PUEDEN INGRESAR LOS ELEMENTOS. NO CUMPLE LA REGLA DE STOCK'

ROLLBACK

end

end

GO

TSQL

18. Sabiendo que el limite de credito de un cliente es el monto maximo que se le

puede facturar mensualmente, cree el/los objetos de base de datos necesarios

para que dicha regla de negocio se cumpla automaticamente. No se conoce la

forma de acceso a los datos ni el procedimiento por el cual se emiten las facturas

create trigger ejercicio18 on factura FOR insert

AS

BEGIN

IF exists

(SELECT \* FROM inserted i

join cliente on clie\_codigo = fact\_cliente

WHERE clie\_limite\_Credito <

(SELECT sum(fact\_total)

FROM factura

WHERE i.fact\_cliente = fact\_cliente and

year(i.fact\_fecha) = year(fact\_fecha) and

month(i.fact\_fecha) = month(fact\_fecha)))

BEGIN

print 'SUPERA EL LIMITE DE CREDITO'

ROLLBACK

END

END

GO

CREATE TRIGGER ejer18 on factura INSTEAD OF insert

AS

BEGIN

IF exists

(SELECT \* FROM inserted i

join cliente on clie\_codigo = fact\_cliente

WHERE clie\_limite\_Credito <

(SELECT sum(fact\_total) + I.FACT\_TOTAL

FROM factura

WHERE i.fact\_cliente = fact\_cliente and

year(i.fact\_fecha) = year(fact\_fecha) and

month(i.fact\_fecha) = month(fact\_fecha)))

BEGIN

print 'ALGUNAS FACTURAS NO SE INGRESARON. SE SUPERO EL LIMITE DE CREDITO'

END

INSERT FACTURA SELECT I.\*

FROM inserted i

join cliente on clie\_codigo = fact\_cliente

WHERE clie\_limite\_Credito >=

(SELECT sum(fact\_total)+I.FACT\_TOTAL

FROM factura

WHERE i.fact\_cliente = fact\_cliente and

year(i.fact\_fecha) = year(fact\_fecha) and

month(i.fact\_fecha) = month(fact\_fecha))

END

GO