**Ejemplo en SQLServer.**

1. *-- --------------------------------------------------------------------*
2. *-- Creamos una base de datos si no existiese.*
3. *-- --------------------------------------------------------------------*
5. IF NOT EXISTS (**SELECT** \* **from** sys.databases **where** **name** = 'db\_test')
6. **BEGIN**
7. **CREATE** **DATABASE** db\_test;
8. **END**
10. *-- Establecemos la  base de datos predeterminada*
11. USE db\_test;

14. *-- --------------------------------------------------------------------*
15. *-- Creamos una tabla si no existiese.*
16. *-- Representa los datos de expedientes*
17. *-- --------------------------------------------------------------------*
19. IF NOT EXISTS (**SELECT** \* **FROM** sys.sysobjects **WHERE** **name**='expedientes' AND xtype='U')
20. **BEGIN**
21. **CREATE** **TABLE** expedientes (
22. code             **VARCHAR**(15)  NOT NULL,
23. state            **VARCHAR**(20)  **DEFAULT** 'INICIO',
24. stateChangedDate DATETIME,
25. **PRIMARY** **KEY** (code)
26. );
27. **END**;
29. *-- Insertamos algunos expedientes de ejemplo*
30. **DELETE** **FROM** expedientes **WHERE** code IN ('exp1','exp2', 'exp3');
31. **INSERT** **INTO** expedientes (code) **VALUES** ('exp1');
32. **INSERT** **INTO** expedientes (code) **VALUES** ('exp2');
33. **INSERT** **INTO** expedientes (code) **VALUES** ('exp3');
35. *-- Si no existe la tabla de cambios de esstado la creamos*
36. IF NOT EXISTS (**SELECT** \* **FROM** sys.sysobjects **WHERE** **name**='expStatusHistory' AND xtype='U')
37. **BEGIN**
38. **CREATE** **TABLE** expStatusHistory (
39. id    **INT**         IDENTITY,
40. code  **VARCHAR**(15) NOT NULL,
41. state **VARCHAR**(20) NOT NULL,
42. **date**  DATETIME   **DEFAULT** GetDate(),
43. **PRIMARY** **KEY**  (id)
44. );
45. **END**;

48. *-- Borramos el Trigger si existise*
49. IF OBJECT\_ID ('StatusChangeDateTrigger', 'TR') **IS** NOT NULL
50. **BEGIN**
51. **DROP** **TRIGGER** StatusChangeDateTrigger;
52. **END**;
54. GO *-- Necesario*
56. *-- Cremamos un Trigger sobre la tabla expedientes*
57. **CREATE** **TRIGGER** StatusChangeDateTrigger
58. **ON** expedientes
59. **AFTER** **UPDATE** **AS**
60. *-- ¿Ha cambiado el estado?*
61. IF **UPDATE**(state)
62. **BEGIN**
63. *-- Actualizamos el campo stateChangedDate a la fecha/hora actual*
64. **UPDATE** expedientes **SET** stateChangedDate=GetDate() **WHERE** code=(**SELECT** code **FROM** inserted);
66. *-- A modo de auditoría, añadimos un registro en la tabla expStatusHistory*
67. **INSERT** **INTO** expStatusHistory  (code, state) (**SELECT** code, state **FROM** deleted **WHERE** code=deleted.code);
69. *-- La tabla deleted contiene información sobre los valores ANTIGUOS mientras que la tabla inserted contiene los NUEVOS valores.*
70. *-- Ambas tablas son virtuales y tienen la misma estructura que la tabla a la que se asocia el Trigger.*
71. **END**;

---sentencias importantes

inne join http://es.wikipedia.org/wiki/Join

SELECT \*

FROM empleado

INNER JOIN departamento

ON empleado.IDDepartamento = departamento.IDDepartamento

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\*\*\*\*\*/

SELECT \*

FROM empleado, departamento

WHERE empleado.IDDepartamento = departamento.IDDepartamento

TRIGGER

<http://www.devjoker.com/contenidos/catss/278/Triggers-en-Transact-SQL.aspx>

 La sintaxis general de un trigger es la siguiente.

|  |
| --- |
| **CREATE** **TRIGGER** <Trigger\_Name, **sysname**, Trigger\_Name>  **ON** <Table\_Name, **sysname**, Table\_Name>  AFTER <Data\_Modification\_Statements, , **INSERT**,**DELETE**,**UPDATE**>  **AS**  **BEGIN**  -- SET NOCOUNT ON added to prevent extra result sets from  -- interfering with SELECT statements.  **SET** **NOCOUNT** **ON**;  -- Insert statements for trigger here  **END** |

  El siguiente ejemplo, graba un historico de saldos cada vez que se modifica un saldo de la tabla cuentas.

|  |
| --- |
| **CREATE** **TRIGGER** TR\_CUENTAS  **ON** CUENTAS  AFTER **UPDATE**  **AS**  **BEGIN**  -- SET NOCOUNT ON impide que se generen mensajes de texto  -- con cada instrucción  **SET** **NOCOUNT** **ON**;  **INSERT** **INTO** HCO\_SALDOS  (IDCUENTA, SALDO, FXSALDO)  **SELECT** IDCUENTA, SALDO, getdate()  **FROM** INSERTED  **END** |

La siguiente instrucción provocará que el trigger se ejecute:

|  |
| --- |
| **UPDATE** CUENTAS  **SET** SALDO = SALDO + 10  **WHERE** IDCUENTA = 1 |

 Podemos especificar a que columnas de la tabla debe afectar el trigger.

|  |
| --- |
| **ALTER** **TRIGGER** TR\_CUENTAS  **ON** CUENTAS  AFTER **UPDATE**  **AS**  **BEGIN**  -- SET NOCOUNT ON impide que se generen mensajes de texto  -- con cada instrucción  **SET** **NOCOUNT** **ON**;    **IF** **UPDATE**(SALDO) -- Solo si se actualiza SALDO  **BEGIN**  **INSERT** **INTO** HCO\_SALDOS  (IDCUENTA, SALDO, FXSALDO)  **SELECT** IDCUENTA, SALDO, getdate()  **FROM** INSERTED  **END**  **END** |

 Los trigger están dentro de la transacción original (Insert, Delete o Update) por lo cual si dentro de nuestro trigger hacemos un RollBack Tran, no solo estaremos echando atrás nuestro trigger sino también toda la transacción; en otras palabras si en un trigger ponemos un RollBack Tran, la transacción de Insert, Delete o Update volverá toda hacia atrás.

|  |
| --- |
| **ALTER** **TRIGGER** TR\_CUENTAS  **ON** CUENTAS  AFTER **UPDATE**  **AS**  **BEGIN**  -- SET NOCOUNT ON impide que se generen mensajes de texto  -- con cada instrucción  **SET** **NOCOUNT** **ON**;  **INSERT** **INTO** HCO\_SALDOS  (IDCUENTA, SALDO, FXSALDO)  **SELECT** IDCUENTA, SALDO, getdate()  **FROM** INSERTED    **ROLLBACK**  **END** |

    En este caso obtendremos el siguiente mensaje de error:

La transacción terminó en el desencadenador. Se anuló el lote.

    Podemos activar y desactivar Triggers a tarvés de las siguientes instrucciones.

|  |
| --- |
| -- Desactiva el trigger TR\_CUENTAS  **DISABLE** **TRIGGER** TR\_CUENTAS **ON** CUENTAS  GO  -- activa el trigger TR\_CUENTAS  **ENABLE** **TRIGGER** TR\_CUENTAS **ON** CUENTAS  GO  -- Desactiva todos los trigger de la tabla CUENTAS  **ALTER** **TABLE** CUENTAS **DISABLE** **TRIGGER** ALL  GO  -- Activa todos los trigger de la tabla CUENTAS  **ALTER** **TABLE** CUENTAS **ENABLE** **TRIGGER** ALL |

CREATE PROCEDURE DIC\_FORO  
  
AS  
SET NOCOUNT ON  
  
SELECT  
Campos  
FROM  
Tabla  
With(NoLock)  
  
SET NOCOUNT OFF  
GO  
  
Los Storeds admiten parámetros. Sirven para poder realizar querys más precisas.  
  
Por ejemplo:  
  
CREATE PROCEDURE DIC\_FORO  
(  
@IdUsuario int=0  
)  
AS  
SET NOCOUNT ON  
  
SELECT  
\*  
FROM  
Usuario  
With(NoLock)  
WHERE  
Usuario=@IdUsuario  
  
  
SET NOCOUNT OFF  
GO  
  
  
También podemos utilizar parámetros de salida, para retornar datos. Para ello declararemos el parámetro como Output.

**TRIGGERS PARTE 2**  
Los ejemplos de este documento se realizan en la Base de datos de Biblioteca

TRIGGER QUE EVITA QUE SE BORRE MAS DE UN USUARIO A LA VEZ  
create trigger tr\_borrarUsuario  
on Usuario  
after delete  
as  
if(select COUNT(\*) from deleted)>1  
begin  
RAISERROR('NO PUEDE BORRAR MAS DE UN USUARIO A LA VEZ',16,1)  
rollback transaction  
end  
  
PROBANDO EL TRIGGER  
select \* from usuario  
  
insert into usuario values('U05','Paez','Pedro')  
insert into usuario values('U06','Jara','Marta')  
insert into usuario values('U07','Ayala','Marta')  
  
delete from usuario where U\_nombre='Marta'  
  
  
PARA DESHABILITAR UN TRIGGER  
Alter table Usuario  
disable trigger tr\_BorrarUsuario  
  
PARA BORRAR UN TRIGGER  
drop trigger tr\_BorrarUsuario  
  
Que triggers afectan a una tabla  
exec sp\_depends copiaLibro  
exec sp\_depends historico  
  
PARA OBTENER LA DEFINICION DE UN TRIGGER  
exec sp\_helptext tr\_update\_historico  
  
--PARA VER LOS TRIGGERS QUE ESTAN ACTUANDO SOBRE UNA TABLA  
exec sp\_helptrigger prestamo  
  
TRIGGERS INSTEAD OF (EN VEZ DE)  
Estos triggers ejecutan un grupo de instrucciones automaticamente cuando se trata de insertar, actualizar o borrar una fila de una tabla, pero estas instrucciones se ejecutan en vez de la inserción, actualización o borrado.  
  
Crear un trigger que impida borrar filas de la tabla histórico  
create trigger tr\_borrarHistorico  
on Historico instead of delete  
as  
raiserror('NO SE PUEDE BORRAR FILAS DE ESTA TABLA',10,1)  
  
SELECT \* FROM HISTORICO  
  
delete from Historico where H\_multa<0

MODIFICACIÓN AL TRIGGER QUE CALCULA LA MULTA EN LA TABLA HISTORICO DE LA BASES DE DATOS BIBLIOTECA  
  
/\*Cada vez que se actualiza la fecha de devolución en una fila   
de la tabla HISTORICO, se calcula los dias de retraso y la multa,   
conociendo que la multa es de 0,50 dólares por dia.\*/  
  
alter trigger tr\_update\_historico  
on historico  
after update  
as  
declare  
@cod char(3),  
@nCopia int,  
@a date,--Variable para recuperar la fechaPrestamo  
@b date,--Variable para recuperar la fecha que devuelve  
@c int, --Nro de dias que se tarda en entregar  
@d int --Aqui se guarda el numero de dias de prestamo  
select  
@cod=inserted.L\_codigo,  
@nCopia=inserted.Cp\_nro,  
@a=INSERTED.H\_fechaPrestamo,  
@b=INSERTED.H\_fechaDevuelve,  
@d=Inserted.H\_nroDiasPrestamo  
from INSERTED  
select @c=datediff(dd,@a,@b)from inserted  
  
if @c>=@d  
begin  
update HISTORICO set H\_multa=(@c-@d)\*0.5  
where Historico.L\_codigo=@cod and Historico.Cp\_nro=@nCopia   
and Historico.H\_fechaPrestamo=@a  
end  
else  
begin  
update HISTORICO set H\_multa=0  
where Historico.L\_codigo=@cod and Historico.Cp\_nro=@nCopia   
and Historico.H\_fechaPrestamo=@a  
end  
  
TRIGGERS SOBRE LA BASE DE DATOS REPORTE VENTAS  
  
TRIGGER 1  
/\*Cada vez que se inserta una fila en la tabla ventas, en la tabla ventasMes se incrementa el valor de la venta a VM\_ventaTotalMes\*/  
  
create trigger tr\_Venta\_all  
on ventas  
for insert  
as  
declare  
@mes int,  
@anio int,  
@valor money,  
@vendedor int  
  
select  
@mes=MONTH(inserted.Vt\_fecha),  
@anio=YEAR(inserted.Vt\_fecha),  
@valor=inserted.Vt\_valor,  
@vendedor=inserted.V\_codigo  
from inserted  
if exists(select \* from VentasMes where V\_codigo=@vendedor and Vm\_mes=@mes and Vm\_año=@anio)  
begin  
update VentasMes set Vm\_ventaTotalMes=Vm\_ventaTotalMes+@valor where V\_codigo=@vendedor and Vm\_mes=@mes and Vm\_año=@anio  
end  
else begin  
insert into VentasMes values(@mes,@anio,@valor,@vendedor)  
end  
  
PROBAR EL TRIGGER  
select \* from ventas  
select \* from ventasmes  
  
delete from ventas  
delete from ventasmes  
  
insert into Ventas values('V13','2010-01-29',2000,100,'C02')  
insert into Ventas values('V16','2010-02-28',2000,100,'C02')  
  
  
TRIGGER 2  
/\*Cada vez que se cambia el valor de venta enuna fila de la tabla ventas, en la tabla ventasMes se cambia el valor de la venta en VM\_ventaTotalMes \*/  
  
create trigger tr\_Venta\_update  
on ventas  
for update  
as  
declare  
@mes int,  
@anio int,  
@valorNuevo money,  
@vendedor int,  
@valorViejo money  
select  
@mes=MONTH(inserted.Vt\_fecha),  
@anio=YEAR(inserted.Vt\_fecha),  
@valorNuevo=inserted.Vt\_valor,  
@vendedor=inserted.V\_codigo  
from inserted  
  
select @valorViejo=deleted.Vt\_valor from deleted   
update VentasMes set Vm\_ventaTotalMes=Vm\_ventaTotalMes+@valorNuevo-@valorViejo where V\_codigo=@vendedor and Vm\_mes=@mes and Vm\_año=@anio  
  
PROBAR EL TRIGGER  
update Ventas set Vt\_valor=4500 where Vt\_codigo='V13'  
  
  
TRIGGER 3  
/\* Cada vez que se borra una fila de la tabla Ventas, se disminuye el total de ventas del vendedor correspondiente en la tabla VentasMes \*/  
  
create trigger tr\_Venta\_delete  
on ventas  
for delete  
as  
declare  
@mes int,  
@anio int,  
@valor money,  
@vendedor int,  
@fecha date  
  
select  
@mes=MONTH(deleted.Vt\_fecha),  
@anio=YEAR(deleted.Vt\_fecha),  
@valor=deleted.Vt\_valor,  
@vendedor=deleted.V\_codigo,  
@fecha=deleted.Vt\_fecha  
from deleted  
  
if exists(select \* from Ventas where V\_codigo=@vendedor and MONTH(@fecha)=@mes and @anio=YEAR(@fecha))  
begin    
update VentasMes set Vm\_ventaTotalMes=Vm\_ventaTotalMes-@valor where V\_codigo=@vendedor and Vm\_mes=@mes and Vm\_año=@anio  
end  
else begin  
delete VentasMes where V\_codigo=@vendedor and Vm\_mes=@mes and Vm\_año=@anio  
end  
  
PROBAR EL TRIGGER   
select \* from clientes  
select \* from Vendedor  
select \* from ventas  
select \* from ventasMes  
  
insert into ventas values('001','2011-09-11',3000,100,'C05')  
insert into ventas values('002','2011-09-17',2600,100,'C01')  
insert into ventas values('003','2011-09-12',5000,171,'C01')