DECLANŞATOARE

Un declanşator (trigger) este o procedură care este executată în mod implicit când asupra tabelului asociat se execută o comandă **insert**, **update** sau **delete**. Declanşatoarele sunt medii prin care SQL oferă programatorilor de aplicaţii şi proiectantilor de BD să asigure integritatea BD.

Declanşatoarele sunt utile deoarece impun reguli fară a fi cuprinse în aplicaţiile utilizatorilor. Proiectantul declanşatorului specifică şi momentul când trebuie executat în raport cu instrucţiunile LMD.SQL Server ia în seama regulile şi valorile implicite înainte ca informaţiile să fie scrise în BD.

Acestea reprezintă pre-filtre care pot să impiedice manipulările de date, prin controlarea activităţilor din cadrul BD. Un declanşator poate fi şi un post-filtru care se execută după ce modificarea datelor a trecut de reguli.

Declanşatorii permit să executăm o procedură rezidentă ori de câte ori este executată o instrucţiune **INSERT, UPDATE** sau **DELETE** asupra unui tabel predefinit. Declanşatorul este o procedură specială stocată care:

- generează automat anumite valori ce derivă din valorile coloanelor;
- determină respectarea restricţiilor şi privilegiilor ;
- face posibilă jurnalizarea transparentă a evenimentelor;
- strânge informaţii statistice în legătură cu accesarea tabelelor.

Un declanşator este o procedură stocată specială care este executată de SQL Server la efectuarea unei operaţii de inserare, modificare sau stergere. Pentru că declanşatorii sunt executaţi după efectuarea operaţiei de inserare, modificare sau ştergere ei reprezintă un fel de ultim cuvânt la acestea. Dacă un declanşator respinge cererea, modificarea informaţiilor este refuzată, iar aplicaţia care a iniţiat operaţia deţine un mesaj de eroare. Cea mai simplă utilizare este determinată de impunerea regulilor de integritate în BD.

Observaţie: Deoarece declanşatoarele sunt executate după regulile de integritate dacă acestea nu sunt trecute atunci declanşatorul nu este executat. Pentru ca un declanşator să fie executat trebuie ca operaţia în cauză să nu fi eşuat.

Componentele unui declanşator

Un declanşator are trei componente:

- o instrucţiune de declarare –aceasta specifică instrucţiunile SQL care activează declanşatorul;
- o restricţie de declanşare –specifică condiţia care trebui să fie adevarată pentru ca declanşatorul să fie activat;
- acţiunea declanşatorului care specifică blocul de instrucţiuni care trebuie executate.

Pentru a crea un declanşator trebuie să fiţi posesorul BD. Atunci când se adaugă un declanşator pentru o coloana, linie sau tabel se schimbă uneori şi modul de accesare şi modul cum interacţionează alte obiecte cu acesta. La folosirea obiectelor declanşator se presupune respectarea condiţiilor:

- Declanşatoarele nu pot fi create pentru tabele temporare,
- Declanşatoarele trebuie să fie create numai pentru tabele din baza de date curentă,
- La eliminarea unui tabel, toate obicetele declanşator asociate cu acest tabel sunt, eliminate automat împreună cu tabelul.

Crearea unui declanşator se realizează cu instrucţiunea:

CREATE TRIGGER nume_declanşator
ON { *Nume_ tabel* | *Nume_ view* }
[WITH ENCRYPTION]
{
{ **{ FOR | AFTER | INSTEAD OF } { [INSERT] [,] [UPDATE] [,] [DELETE] }**
[WITH APPEND]
[NOT FOR REPLICATION]
AS

[{ IF UPDATE (*coloana* **)**

[**{ AND | OR } UPDATE (** *coloana* **)**] [*...n*]

| IF (COLUMNS_UPDATED ( ) { *bitwise_operator* } *updated_bitmask* **)**

{ operator de comparare } *coloana_bitmask* [*...n*]

}]

Instrucţiuni_SQL [...*n*]

}

}

Sau următoarea formă mai simplă

CREATE TRIGGER [posesor.]nume_declanşator

ON [posesor**.]**nume_tabela

FOR {INSERT, UPDATE, DELETE **}**

[WITH ENCRYPTION]

AS

Instrucţiuni **SQL**

Nume_declanşator este numele declanşatorului (trigger-ului). Un nume de declanşator trebuie să respecte regulile pentru identificatori şi trebuie să fie unic încadrul unei baze de date.

Nume_ tabel | Nume_ view este numele tabelului sau vederii în care declanşatorul este executat.

INSERT, UPDATE, DELETE acestea definesc scopul declanşatorului şi specifică operaţiile care iniţiază declanşatorul.

WITH ENCRYPTION –această opţiune este utilizată pentru a împiedica ca utilizatorii să citescă definiţia declanşatorului după încărcarea lui pe server. SQL Server stochează definiţia unui declanşator în fişierul **Syscomment**. Criptează intrările în tabela **syscomments** care conţine textul comenzii CREATE TRIGGER.

AFTER specifică faptul că declanşatorul este executat doar când toate operaţiile din enunţul SQL al declanşatorului au fost execuate cu succes. Toate acţiunile referenţiale în cascadă şi verificările de constrângere deasemenea trebuie să reusească înainte de executarea acestui trigger. Folosirea clauzei AFTER este identică cu folosirea clauzei

FOR utilizată mai mult în ultimele versiuni de SQL Server. Declanşatorii AFTER nu pot fi definiţi pentru vederi.

INSTEAD OF Specifică faptul că declanşatorul este executat în locul enunţului SQL declanşat, astfel suprapunând operaţiile din enunţul declanşatorului.

Acest tip de declanşator poate fi definit pentru un tabel sau pentru o vedere când se realizează; INSERT, UPDATE sau DELETE.

Este posibil să definim vederi unde fiecare vedere să aibă propriul declanşator INSTEAD OF.

{ [DELETE] [**,**] [INSERT] [**,**] [UPDATE] } Sunt cuvinte cheie care specifică pentru ce operaţie are loc declanşatorul. Este obligatorie apariţia cel puţin a unei opţiuni.

Pentru declanşatori INSTEAD OF, opţiunea DELETE nu este permisă pentru tabelele care au legături referenţiale ce specifică o acţiune încascadă pentru opţiunea ON DELETE. Asemănător, opţiunea UPDATE nu este permisă pe tabele care au legături referenţiale ce specifică o acţiune încascadă pentru opţiunea ON UPDATE.

Instrucţiuni_sql Reprezintă condiţia (condiţiile) şi acţiunea (acţiunile) declanşatorului. Acţiunile declanşatorului sunt scrise în Transact-SQL şi pot cuprinde oricâte instrucţini sau comenzi Transact-SQL.

Câteva tabele speciale, de care am mai amintit, sunt folosite în instrucţiunea CREATE TRIGGER şi anume: **inserted** şi **deleted**.

IF UPDATE (*coloana*)Testează acţiunea unei comenzi INSERT sau UPDATE pentru coloana specificată şi nu este folosit pentru DELETE. Mai multe coloane pot fi specificate. Deoarece numele tabelului este specificat după clauza ON, nu este nevoie să includem numele tabelului înainte de numele coloanei în clauza IF UPDATE. Pentru a testa o adăugare sau o modificare pentru mai multe coloane, specificăm separat clauza UPDATE(*coloana)* după ce am scris-o pe prima. IF UPDATE va returna valoarea TRUE în acţiunea INSERT deoarece coloanele vor fi explicit sau implicit (NULL) inserate.

Nota Clauza IF UPDATE **(***coloana***)** functionează identic cu IF, IF...ELSE sau WHILE poate folosi şi blocul BEGIN...END.

UPDATE(*coloana*) poate fi folosit oriunde în corpul declanşatorului.

Coloana este numele unei coloane pentru testarea acţiunii INSERT sau UPDATE. Această coloană poate avea orice tip de date SQL Server.Oricum coloanele calculate nu pot fi utilizate în acest context.

IF **(**COLOANAS_UPDATED**())** Testează, dacă într-un declanşator INSERT sau UPDATE, coloanele menţionate au fost inserate respectiv modificate.

COLOANAS_UPDATED returnează un tip **varbinary** (tip binar variabil) ce indică ce coloane din tabel au fost inserate sau modificate.

Funcţia COLOANAS_UPDATED returnează biţii în ordine de la stanga la dreapta, cu cel mai puţin semnificativ bit în partea cea mai din stânga. Cel mai din stânga bit reprezentând prima coloana din tabel, următorul bit reprezentând cea de-a două coloană etc.

COLOANAS_UPDATED returnează mai mulţi octeţi pentru declanşatorul care a fost creat conţinând mai mult de 8 coloane, cu cel mai puţin semnificativ byte în partea din stânga.

COLOANAS_UPDATED va returna TRUE pentru toate coloanele din acţiunea INSERT deoarece coloanele au fost inserate cu valori explicite sau cu valori implicite (NULL).

COLOANAS_UPDATED poate fi folosit oriunde în corpul declanşatorului.

bitwise_operator Operaţor pe bit folosit în comparări.

Updated_bitmask Este o mască de tip întreg pentru acele coloanele care tocmai au fost modificate sau inserate. De exemplu, tabela **t1** conţine coloanele **C1**, **C2**, **C3**, **C4**, şi **C5**. Pentru a verifica dacă coloanele **C2**, **C3**, şi **C4** sunt toate modificate (tabela **t1** având un declanşator UPDATE), specifică valoarea 14 (01110). Pentru a verifica dacă doar coloana **C2** este modificată, specifică valoarea 2(00010).

Operaţor* de comparare**.Este unul din operatorii de comparare. Folosiţi semnul egal (=) pentru a verifica dacă toate coloanele specificate în ***updated_bitmask sunt realmente modificate.Folosiţi semnul mai mare (>) pentru a verifica dacă oricare sau câteva din coloanele specificate în *updated_bitmask* sunt modificate.

Coloana_bitmask Este o mască de tip întreg pentru acele coloane pentru care verificăm dacă ele au fost modificate sau inserate.

SQL Server permite declanşatori multiplii ce pot fi creaţi pentru fiecare eveniment de modificare de date (DELETE, INSERT, or UPDATE). De exemplu, dacă este executat CREATE TRIGGER FOR UPDATE pentru un tabel care deja are un declanşator UPDATE, atunci este creat un declanşator adiţional de tip UPDATE.

Exemplul 1 Declanşator ce afisează pe ecran un mesaj când se adaugă sau se modifică date din tabela Catalog

```
IF EXISTS (SELECT name
           FROM sysobjects
           WHERE name = 'declanş_mesaj ' AND type = 'TR')
   DROP TRIGGER  declanş_mesaj
GO
CREATE TRIGGER declanş_mesaj
  ON Catalog
  FOR INSERT, UPDATE
  AS RAISERROR ('S-a executat un INSERT sau un UPDATE', 16, 1)
  GO
INSERT INTO Catalog VALUES ('101','5',7,'11-17-2006')
GO
UPDATE Catalog
SET Nota=5
WHERE NrLeg='101' and Cod_disciplina='5'
```

Exemplul 2 **Acest declanşator afisează pe ecran anumite mesaje pentru oricine încearcă să adauge sau să modifice date din tabelul Catalog**

```
IF EXISTS (SELECT name
           FROM sysobjects
           WHERE name = ' declan1' AND type = 'TR')
   DROP TRIGGER declan1
GO
      CREATE TRIGGER declan1
      ON Catalog
      FOR INSERT, UPDATE
      AS
      DECLARE
        @@nota   integer,
        @@Nr_leg varchar(5)
      SELECT @@Nr_leg = i.NrLeg, @@nota = i.Nota
      FROM Catalog C, inserted i
       WHERE C.NrLeg = i.NrLeg
        IF (@@nota <5 )and(@@Nr_leg like '101')
```

```
  BEGIN
  RAISERROR ('Studentul cu NrLeg=101 are nota <5', 16, 1)

  END
  ELSE
  BEGIN
   RAISERROR ('Studentul cu NrLeg=%s are note de  %d ',
             16, 1, @@Nr_leg, @@nota)
  END

go
INSERT INTO Catalog VALUES ('101','4',6,'11-06-2006')
go
INSERT INTO Catalog VALUES ('101','4',2,'12-07-2006')
Go
UPDATE Catalog
SET Nota=5
WHERE NrLeg='101' and Cod_materie='4'
go
```

Exemplul 3 Declanşator pentru ştergeri şi modificări întabelul Catalog

```
IF EXISTS (SELECT name
     FROM sysobjects
           WHERE name = 'dec1' AND type = 'TR')
   DROP TRIGGER dec1
GO

CREATE TRIGGER dec1
ON Catalog
FOR INSERT, UPDATE
AS
DECLARE
  @@nota   integer,
  @@Nr_leg varchar(5)
SELECT @@Nr_leg = i.NrLeg,  @@nota = i.Nota
  FROM Catalog C, inserted i
  WHERE C.NrLeg = i.NrLeg
```

```
IF (@@nota <5 )AND(@@Nr_leg like '101')
BEGIN
   RAISERROR ('Studentul cu NrLeg=111 are Catalog <6', 16, 1)
  ROLLBACK TRANSACTION
  END
  ELSE
  BEGIN
  RAISERROR ('Studentul cu NrLeg=%s are Catalog de  %d ',
          16, 1, @@Nr_leg, @@nota)
  ROLLBACK TRANSACTION
  END

go
INSERT INTO Catalog VALUES ('101','5',6,'11-10-2006')
Go
INSERT INTO Catalog VALUES ('101','5',2,'12-01-2006')
go
UPDATE Catalog SET Nota=7 WHERE NrLeg='101' AND
Cod_disciplina='4'
go
```

Exemplul 4 Declanşator ce nu permite introducerea unor date nevalide în Catalog

```
IF EXISTS (SELECT name
     FROM sysobjects
           WHERE name = ' dec2' AND type = 'TR')
   DROP TRIGGER dec2
GO

CREATE TRIGGER dec2
ON Catalog
FOR INSERT, UPDATE
AS
DECLARE @nota integer
SELECT @nota=Catalog.nota from Catalog, inserted WHERE
Catalog.NrLeg=inserted.NrLeg
IF(@nota<=0) or (@nota>10)
BEGIN
```

```
         RAISERROR ('Nota invalida', 16, 1)
   ROLLBACK TRANSACTION
END

      go
      INSERT INTO Catalog VALUES ('102','4',-2,'12-01-2006')
      go
      SELECT * FROM Catalog WHERE NrLeg='102'
```

Exemplul 5 Declanşator ce adaugă 1 punct la orice notă din tabelul Catalog

```
      IF EXISTS (SELECT name
                FROM sysobjects
                WHERE name = ' dec3' AND type = 'TR')
         DROP TRIGGER dec3
      GO

      CREATE  TRIGGER dec3
      ON Catalog
      AFTER INSERT
      AS
         UPDATE Catalog
         SET Catalog.nota = Catalog.nota + 1
         FROM inserted
         WHERE Catalog.NrLeg = inserted.NrLeg
      go
      INSERT INTO Catalog VALUES ('103','4',4,'12-01-2006')
      go
      SELECT * FROM Catalog WHERE NrLeg='105'
```

Exemplul 6 Declanşator inlocuieste o operaţie INSERT cu operaţia UPDATE din enunţul SQL al declanşatorului astfel dacă se doreste inserarea unei Catalog pentru un student la o anumită materie se va face o modificarea adică se adaugă 1 punct la nota introdusă anterior la acea materie pentru studentul dat în tabelul Catalog

```
      IF EXISTS (SELECT name FROM sysobjects
            WHERE name = 'dec3' AND type = 'TR')
         DROP TRIGGER dec3
```

```
GO

CREATE  TRIGGER dec3
ON Catalog
INSTEAD OF INSERT
AS
   UPDATE Catalog
   SET Catalog.nota = Catalog.nota+1
   FROM inserted
   WHERE Catalog.NrLeg = inserted.NrLeg
go
INSERT INTO Catalog VALUES ('102','4',4,'12-01-2006')
go
SELECT * FROM Catalog WHERE NrLeg='102'
```

Exemplul7 Crearea unui declanşator ce afisează un mesaj dacă s-a modificat Nota şi un alt mesaj încaz contrar

```
IF EXISTS (SELECT name FROM sysobjects
      WHERE name = 'dec4' ÂND type = 'TR')
   DROP TRIGGER dec4
GO

CREATE  TRIGGER dec4
ON Catalog
FOR UPDATE
AS
IF UPDATE(Nota)
  RAISERROR ('S-a modificat campul Nota',16,1)
ELSE
  RAISERROR ('S-a modificat alt camp al tabelei CATALOG',16,1)
go
UPDATE Catalog SET Nota=10 WHERE NrLeg='105'
go
SELECT * FROM Catalog WHERE NrLeg='105'
go
UPDATE Catalog SET Cod_materie='3' WHERE NrLeg='105'
go
```

```
UPDATE Catalog SET Cod_materie='3', Nota=8 WHERE NrLeg='105'
```

Exemplul 8 Crearea unui declanşator care afisează un mesaj dacă s-au modificat Nota şi Cod_disciplina şi '***' încaz contrar**

```
IF EXISTS (SELECT name
           FROM sysobjects
           WHERE name = 'dec5' AND type = 'TR')
           DROP TRIGGER dec5
  GO

CREATE  TRIGGER dec5
  ON Catalog
  FOR UPDATE
  AS
  IF UPDATE(Nota) AND UPDATE(Cod_diciplina)
  RAISERROR('S-au modificat atributele Nota şi
  Cod_disciplina',16,1)
  ELSE
  RAISERROR ('*****',16,1)
go
UPDATE Catalog SET Nota=10 WHERE NrLeg='105'
go
SELECT * FROM Catalog WHERE NrLeg='105'
go
UPDATE Catalog SET Cod_diciplina ='3', Nota=8 WHERE NrLeg='105'
```

Exemplul 9 ***Crearea unui declanşator care afisează un mesaj dacă s-au modificat atributele Nota sau Cod_disciplina şi '*.*.*' în caz contrar.***

```
  IF EXISTS (SELECT name
     FROM sysobjects
     WHERE name = 'dec6' AND type = 'TR')
     DROP TRIGGER dec6
GO
CREATE  TRIGGER dec6
ON Catalog
FOR UPDATE
```

```
 AS
 IF (COLOANAS_UPDATED()&6)>0
   RAISERROR ('S-au modificat atrib. Nota sau Cod_materie',16,1)
   ELSE
   RAISERROR ('*.*.*',16,1)
   go
   UPDATE Catalog SET Nota=10 WHERE NrLeg='105'
   go
   SELECT * FROM Catalog WHERE NrLeg='105'
   go
   UPDATE Catalog SET Cod_materie='3', Nota=8 WHERE NrLeg='105'
   go
   UPDATE Catalog SET Data='3-3-2002' WHERE NrLeg='105'
```

Exemplul 9 ***Crearea unui declanşator care verifica dacă au fost modificate printr-o comanda UPDATE coloanele 3, 6 şi 9 din tabelul Student***

```
   IF EXISTS (SELECT name FROM sysobjects
   WHERE name = 'dec7' AND type = 'TR')
   DROP TRIGGER dec7
   GO

   CREATE TRIGGER dec7
   ON Student
   FOR UPDATE AS
   IF ( (SUBSTRING(COLOANAS_UPDATED(),1,1)=power(2,(3-1))--coloana
   3+ power(2,(6-1))) --coloana 6
 AND (SUBSTRING(COLOANAS_UPDATED(),2,1)=power(2,(1-1)))--coloana 9
 (coloana 1 din cel de-al II-lea octet )
PRINT 'Coloanele 3, 6 şi 9 au fost modificate'
GO

 UPDATE Student
 SET Initiala='O', Data_nastere='12-12-1980', NrLeg_sot='105'
 WHERE NrLeg='105'
 GO
 SELECT * FROM Student WHERE NrLeg='105'
```

Exemplul 10 Crearea unui declanşator pentru DELETE

```
IF EXISTS (SELECT name FROM sysobjects
      WHERE name = 'sterge_stud' ÂND type = 'TR')
   DROP TRIGGER sterge_stud
GO

CREATE TRIGGER sterge_stud
ON Student
FOR DELETE
AS
DECLARE @NL varchar(5)
SELECT @NL=deleted.NrLeg
    FROM deleted
    IF (cast(@NL as integer)>0)
    PRINT 'S-a Sters'
    ELSE
    BEGIN
    PRINT 'Acest NrLeg nu exista'
  ROLLBACK TRANSACTION
END
GO

  DELETE FROM Student WHERE NrLeg='17'
```

Modificarea unui declanşator se face cu comanda ALTER TRIGGER care are sintaxa:

ALTER TRIGGER *nume_declan*
ON **(** *table* | view **)**
[WITH ENCRYPTION]
{
{ **(** FOR | AFTER | INSTEAD OF **)** { [DELETE] [**,**] [INSERT] [**,**] [UPDATE] }
[NOT FOR REPLICATION]
AS
sql_statement [*...n*]
}
|
{ **(** FOR | AFTER | INSTEAD OF **)** { [INSERT] [**,**] [UPDATE] }

[NOT FOR REPLICATION]
AS
{ IF UPDATE **(** *coloana* **)**
[{ AND | OR } UPDATE **(** *coloana* **)**]
[*...n*]
| IF **(** COLOANAS_UPDATED **( )** { *bitwise_operaţor* } *updated_bitmask* **)**
{ *comparison_operaţor* } *coloana_bitmask* [*...n*]
}
sql_statement [*...n*]
}
}

Ştergerea unui declanşator se face cu comanda:

DROP TRIGGER { *nume_declan* } [**,***...n*]

Un declanşator poate să conţină oricâte instrucţiuni cu condiţia să fie încadrate într-un bloc Begin…END. La executarea unui declanşator SQL serverul accesează o tabelă specială cu datele ce au determinat rularea declanşatorului.Aceasta tabelă este **INSERTED** pentru operaţiile INSERT şi UPDATE şi respectiv **DELETED** pentru operaţiile DELETE şi UPDATE. Deoarece un declanşator este executat întotdeauna după operaţie, liniile din INSERTED reprezintă întotdeauna o dublură a uneia sau a mai multor înregistrări din tabela de bază.

Restricţii de utilizare a declanşatorilor

SQL-Serverul impune anumite restricţii asupra tipurilor de instrucţiuni care pot fi executate în cadrul unui declanşator. Majoritatea restricţiilor sunt determinate de imposibilitatea de anulare a consecinţelor unei operaţii insert, update sau delete.

În corpul unui declanşator nu poate apare nici una din instrucţiunile:

- CREATE DATABASE, CREATE TABLE INDEX, PROCEDURE, DEFAULT, RULE, TRIGGER şi VIEW.

- toate instrucţiunile DROP;
- instrucţiunile de modificare a obiectelor din bază ALTER TABLE, ALTER DATABASE şi TRUNCATE TABLE;
- UPDATE STATISTICS;
- Operaţii de încărcare a BD LOAD DATABASE şi LOAD TRANZACTION;
- Toate instrucţiunile care realizează modificări fizice pe disc:DISK.
- Crearea de fişiere temporare fie implicita fie prin SELECT;
- Nu se poate crea un declanşator pentru o vedere ci numai pentru tabelă de bază care defineşte vederea.

  Manipularea unor coloane care conţin bloc nu determină execuţia unui declanşator.

Tipuri de declanşatoare

Atunci când se creează un declanşator se specifică de câte ori se execută. Există 2 tipuri de declanşatoare:

-Declanşator linie –care se execută ori de câte ori tabelul este afectat de instrucţiunea declanşatoare. Dacă o instrucţiune UPDATE actualizează 20 de linii, declanşatorul se execută de 20 de ori. Dacă instrucţiunea nu afecteză nici o linie declanşatorul nu este executat.

-Declanşatorul instrucţiune este executat o singură dată pentru instrucţiunea declanşatoare. De exemplu dacă este executată o instrucţiune UPDATE care actualizează 10 linii este executat o singură dată

Declanşatorele INSERT şi UPDATE sunt utile pentru ca impun restricţiile referenţiale şi asigură validarea datelor înainte de a le scrie în tabele. De obicei se foloseşte pentru a verifică dacă datele urmărite satisfac anumite criterii. La declanşatoare se apelează numai când criteriile de integritate sunt foarte complete. Declanşatorul de mai jos este activat ori de câte ori se înserează sau se modifică o înregistrare în tabelul vânzări.

```
CREATE TRIGGER ins_vanzari
ON vanzari
FOR INSERT,UPDATE
AS
DECLARE  @zidinluna tinyint
SELECT @zidinluna=DataPart(Day, iData_comenzi)
FROM vanzari v, Inserted i
WHERE  v.star_id=i.star_id
      AND v.nr_com=i.nr_com
      AND v.denumire=i.denumire
IF @zidinluna>15
Begin
ROLLBACK
END
GO
CREATE TRIGGER elimart
ON articole
FOR DELETE
AS
FOR EACH row
    WHERE (pretart)<100
    DELETE
   END
CREATE TRIGGER ins.art
ON articol FOR INSERT
FOR EACH row
BEGIN
INSERT INTO art
```

END

Se poate crea declanşatorul care să reacţioneze dacă se modifică doar o coloană. Instrucţiunea IF update poate fi utilizată într-un declanşator pentru a decide dacă se conţinuă execuţia acestuia.

IF UPDATE (preţ)AND (@@row COUNT=1)
BEGIN
-
-
END

In acest caz blocul se execută numai dacă s-a modificat coloana pret .Modificarea unei coloane nu înseamnă neapărat schimbarea valorii ei.

Declanşatoare pentru stergere

Se folosesc pentru a preveni ştergerea acolo unde acestea ar afecta integritatea datelor (Este exemplul cheilor străine pentru alte tabele. Al doilea este ştergerea unui articol în cascadă care să elimine înregistrările copie ale unei înregistrări particulare)

CREATE TRIGGER sterg
ON vanzări
FOR DELETE
AS
IF @@row COUNT>1 /*verifica nr de linii afectate impiedicând stergerea a mai mult de o linie*/
BEGIN
ROLLBACK
RAISERROR(‘puteti sterge o singură instrucţiune la un moment dat**’,16, 10)**
END
DECLARE @stare_id char(4)
SELECT @stare_id=s.stare-id

```
FROM vanzări v deleted s
IF EXIST (SELECT *
        FROM vanzări
        WHERE stare_id=@s.stare_id)
BEGIN
ROLLBACK
RAISERROR(‘nu poate fi sters’,16,10)
END
GO
```