TRIGGERE (DECLANŞATOARE)

Un trigger (declanşator) este o procedură stocată specială, asociată unei tabel sau vederi, care se lansează în execuţie automat la iniţierea unei operaţii DML: **UPDATE, INSERT** sau **DELETE,** asupra tabelului sau vederii în cauză. Triggerele extind posibilităţile altor instrumente de verificare a integrităţii datelor, cum ar fi constrângeri, valori implicite şi reguli. Totuşi aceste instrumente declarative vor fi preferate în locul triggerelor ori de câte ori este posibil.

Un tabel (vedere) poate avea asociaţi oricât de mulţi declanşatori(triggere). Declanşatorii asociaţi aceleiaşi instrucţiuni DML vor fi executaţi succesiv, într-o ordine oarecare.

Un declanşator poate conţine instrucţiuni SQL complexe şi poate accesa și date din alte tabele.

Declanşatorii sunt implicit tranzacţii, ceea ce înseamnă că **dacă execuţia declanşatorului eşuează, dintr-un motiv sau altul, atunci şi operaţia care a declanşat triggerul se anulează.**

SQL Server nu permite declanşatorilor să returneze valori rezultat.

Declanşatorii pot fi de tip **AFTER,** ceea ce înseamnă că instrucţiunile din corpul declansatorului ajung să fie executate la terminarea operaţiei care a detreminat lansarea în execuţie a declansatoruli sau de tip **INSTEAD OF** ceea ce înseamnă că instrucţiunile din corpul declansatorului se execută în locul operaţiei care a iniţiat declansatorul.

Instrucţiunea **CREATE TRIGGER** poate fi definită cu oricare dintre comenzile **UPDATE, INSERT** sau **DELETE** pentru a crea triggere specializate pentru fiecare tip de modificare a unui tabel. Dacă se foloseşte comanda **UPDATE,** atunci se poate de asemenea folosi funcţia *UPDATE(denumire_coloana*), care returnează **true** sau **false,** pentru a testa dacă coloana dată ca argument a fost modificată sau nu.

Instrucţiunea CREATE TRIGGER

Creează un trigger, ataşat unei anumite tabele(vedrei) pentru a fi executat la iniţierea unui anumit tip de operaţie de modificare a tabelului.

Sintaxa:

```
CREATE TRIGGER denumire_trigger
ON { tabel / vedere } [WITH ENCRYPTION]
   { FOR | AFTER | INSTEAD OF ) { [DELETE] [,] [INSERT] [,]
[UPDATE]}}
AS
Corpul_triggerului
```

denumire_trigger - este denumirea triggerului. Opţional se poate specifica numele proprietarului triggerului.

tabela / vedere - este numele tabelei sau a vederii căreia i se ataşează triggerul (nu poate fi un nume de vedere în cazul opţiunii AFTER). Opţional se poate specifica numele proprietarului tabelei.

WTTH ENCRYPTION - triggerul este înregistrat cu textul în formă criptată.

AFTER - specifică faptul că triggerul se execută la terminarea operaţiei care a activat triggerul. Opţiunea a fost introdusă începând cu versiunea SQL

Server 2000 pentru a face distincţie faţă de triggerele INSTEAD OF.

AFTER este opţiunea implicită atunci când FOR este singurul cuvânt cheie specificat. Triggerele AFTER nu se pot ataşa vederilor.

INSTEAD OF - specifică faptul că triggerul se execută *în locul* instrucţiunii SQL care a activat triggerul, substituindu-se acesteia. Cel mult un singur trigger INSTEAD OF poate fi ataşat unui tabel sau vederi pentru fiecare tip de operaţie INSERT, UPDATE sau DELETE. Totuşi, este posibil să definim vederi pe baza altor vederi şi fiecare vedere poate avea propriile triggere INSTEAD OF.

{ [DELETE] [,] [INSERT] [,] [UPDATE] }- specifică tipul operaţiei (operaţiilor) prin care se activează triggerul. Trebuie specificat cel puţin un tip de operaţie sau mai multe operaţii separate prin virgule.

AS - marchează începutul corpului triggerului.

Corpul_triggerului - specifică condiţiile şi acţiunile triggerului. Un trigger poate conţine condiţii prin care se specifică criterii suplimentare pentru executarea acţiunilor din corpul triggerului. Triggerele pot conţine orice număr şi orice tip de instrucţiuni SQL cu excepţia frazei SELECT pentru returnare de date. Un trigger are rolul de a face verificări şi eventual modificări asupra baze datelor şi în nici un caz nu trebuie să returneze date utilizatorului care l-a activat.

Instrucţiunile din corpul unui trigger pot face referire la două tabele speciale ale SQL Server numite **deleted** şi **inserted**.

• **deleted** şi **inserted** sunt tabele logice temporare având aceeaşi structură cu tabelului căreia îi este ataşat triggerul. Tabelul temporar **deleted** conţine valorile vechi ale rândurilor asupra cărora acţionează operaţia care a declanşat triggerul, iar tabelul temporar **inserted** conţine valorile noi ale aceloraşi rânduri. În cazul unui trigger INSTEAD OF ataşat unei vederi structura tabelelor deleted şi inserted este aceeaşi cu a vederii căreia îi este ataşat triggerul.

• pentru a regăsi valorile noi produse de o operaţie INSERT sau UPDATE, în triggerul ataşat se poate cupla tabelul inserted cu tabelul original. Se va ţine cont că la momentul execuţiei triggerului tabelul original conţine deja datele inserate sau modificate.

IF UPDATE *(coloana)* - testează dacă coloana specificată a fost modificată în urma unei operaţii INSERT sau UPDATE. Nu este folosită în cazul operaţiilor DELETE. Se pot testa mai multe coloane prin aceeaşi instrucţiune IF construind o expresie logică care conţine mai multe clauze *UPDATE(coloana). UPDATE(coloana)* se poale folosi oriunde în corpul unui trigger.

coloana - este numele coloanei pentru care se testează dacă a fost sau nu modificată.

Observaţii:

1 Triggerele sunt adesea folosite ca alternativă pentru a implementa condiţii de integritate referenţială în baza de date. Deşi SQL Server oferă facilităţi declarative pentru exprimarea integrităţii referenţiale, acestea sunt limitate la cazul restricţionat şi, mai mult. sunt limitate la tabelele bazei de date curente, fară a fi posibilă exprimarea de legături referenţiale între tabele din baze de date diferite.

2 Orice constrângere referitoare la tabelul căreia îi este ataşat un trigger este verificată înainte de execuţia triggerului. De aceea orice operaţie care ar viola o constrângere de cheie primară sau cheie străina este respinsă şi implicit triggerul asociat nu se va activa. Din acest motiv, dacă se doreşte implementarea unei constrângeri de integritate referenţială în variantă cascadată prin folosirea unui trigger, este necesar să se şteargă ori să se dezactiveze declaraţia de cheie străină pe care o înlocuieşte.

3 Orice trigger din SQL Server are acces la două tabele temporare, rezidente în memorie numite: deleted şi inserted. Datele din tabelele deleted şi inserted nu pot fi modificate, dar pot fi folosite pentru a testa efectul operaţiilor care au iniţiat triggerele, pentru condiţii de executare a diverse operaţii din cadrul triggerelor şi pentru a face discriminare între diferitele tipuri de operaţii (DELETE, INSERT sau UPDATE) care pot declanşa un trigger. Tabelul **deleted** conţine copiile randurilor afectate de o instrucţiune **DELETE** sau **UPDATE.** În timpul execuţiei unei asemenea operaţii rândurile afectate sunt şterse din tabelul asociat şi transferate în tabelul **deleted.** În mod normal, tabelul asociat şi tabelul **deleted** nu vor avea nici un rând în comun.

- În cazul unei operaţii **INSERT** tabelul **deleted** va fi vid. Tabelul **inserted** conţine copiile rândurilor afectate de instrucţiunea **INSERT.** În timpul execuţiei unei asemenea operaţii rândurile noi sunt adăugate simultan atât în tabelul **inserted,** cât şi în tabelul asociat.
- În cazul unei operaţii DELETE tabelul inserted va fi vid.
- Operaţia UPDATE poate fi privită ca o ştergere urmată de o inserare. Rândurile vechi sunt copiate în tabelul deleted, iar cele noi în tabelul asociat şi în tabelul inserted.

4 In trigger se poate determina tipul instrucţiunii declanşatoare astfel:

- dacă atât inserted cât şi deleted sunt nevide, atunci instrucţiunea este UPDATE;
- dacă inserted este vid şi deleted nevid, atunci instrucţiunea este DELETE;
- dacă inserted este nevid şi deleted vid, atunci instrucţiunea este INSERT.

Instrucţiunea ALTER TRIGGER

Modifică definiția unui declanșator.

Sintaxa:

```
ALTER TRIGGER [schema.]denumire_trigger
ON { tabel / vedere } [WITH ENCRYPTION]
   { FOR | AFTER | INSTEAD OF ) { [DELETE] [,] [INSERT] [,]
[UPDATE]}}
AS
Corpul_triggerului
```

Instrucţiunea DROP TRIGGER

şterge una sau mai multe triggere din baza de date curentă.

Sintaxa:

```
DROP TRIGGER [schema.]denumire_trigger
```

unde:

denumire_trigger - este numele triggerului care se şterge.

Proprietăţi ale triggerelor

1. Instrucţiunea CREATE TRIGGER trebuie să fie prima într-un lot(bacth) şi se aplică unui singur tabel.

2. Deşi un trigger poate să facă referire la obiecte din afara bazei de date curente, el poate fi creat numai în baza de date curentă.

3. Acelaşi trigger poate fi asociat cu mai multe tipuri de acţiuni asupra unui tabel; practic orice combinaţie dintre INSERT, DELETE şi UPDATE este permisă.

4. Orice operaţie de tip SET, pentru setarea opţiunilor de sistem, poate fi specificată în corpul unui trigger. Setările rămân valabile până la ieşirea din trigger după care revin la valorile iniţiale.

5. Vederilor li se pot asocia doar triggere instead of.

6. O instrucţiune TRUNCATE TABLE nu activează eventualele triggere de tip DELETE asociate tabelei curente.

Triggere multiple

SQL Server permite crearea mai multor triggere, cu nume diferite, pentru acelaşi tip de operaţie (DELETE, INSERT sau UPDATE) şi acelaşi tabel. Orice trigger creat cu un nume diferit de cele existente (pentru un tip de operaţie şi un tabel) se adaugă la acestea. Dacă se încearcă crearea unui trigger cu un nume care există deja, atunci se produce o eroare.

Triggere imbricate

Implicit, **SQL Server** permite apelurile imbricate de triggere (recursiv sau nu) până la 32 nivele de adâncime. Prin această limitare se garantează faptul că în cazul unui lanţ, potenţial infinit, de apeluri recursive acesta va fi terminat prin intervenţia sistemului în momentul depăşirii nivelului de imbricare maxim admis. Prin setarea corespunzătoare a opţiunilor de sistem se poate bloca propagarea activărilor de triggere prin intermediul altor triggere.

Exemple de trigerre

1) VALIDARE CNP

Vom crea un trigger care să interzică inserarea in tabelul tStudenti de rânduri (studenți) cu CNP eronat.
Mai intâi creăm o functie pentru validarea CNP-ului, ținând cont de următoarele:

Prima cifra din cifra din CNP reprezinta cifra sexului (M/F) astfel:
1/2 - cetateni romani nascuti intre 1 ian 1900 si 31 dec 1999
3/4 - cetateni romani nascuti intre 1 ian 1800 si 31 dec 1899
5/6 - cetateni romani nascuti intre 1 ian 2000 si 31 dec 2099
7/8 - rezidenti
Persoanele de cetatenie straina se identifica cu cifra "9"
Următoarele şase cifre corespund datei naşterii, în secvenţa an, lună, zi.
Cifrele 8 şi 9 din CNP arată judeţul natal.

Următoarele trei cifre formează un număr de ordine, cuprins între 001 şi 999.

Ultima cifră se numeste cifră de control și se calculează astfel: fiecare dintre primele 12 cifre sunt înmulţite cu cifra de pe aceeaşi poziţie din numărul 279146358279; rezultatele sunt însumate, apoi totul se împarte la 11. Dacă restul este 10, atunci cifra de control este 1, altfel cifra de control este egală cu restul obținut.

```
CREATE function dbo.ValidareCNP(@CNP char(13))
returns int
as
BEGIN
  declare @S int
  declare @i int
  declare @Pondere char(12)
  declare @cc int
  set @Pondere='279146358279'
  set @S=0
  set @i=1
  while @i<=12
    begin
      if substring(@CNP,@i,1) not between '0' and '9'
return 0
      set @S=@S+convert(int,substring(@CNP,@i,1))*
                convert(int,substring(@Pondere,@i,1))
      set @i=@i+1
     end
  set @cc=@s%11
  if @cc=10  set @cc=1
  if @cc!=convert(int, substring(@CNP,13,1)) return 0

  if convert(int,substring(@CNP,4,2))>12 return 0
  if convert(int,substring(@CNP,4,2))= 0 return 0
  if convert(int,substring(@CNP,4,2))>31 return 0
```

```
  if convert(int,substring(@CNP,4,2))= 0 return 0

  return 1
END
```

```
create trigger tg_validareCNP on scoala.[tStudenti]
 for insert,update
as
 BEGIN

  declare @n int=@@ROWCOUNT --numar randuri afectate de
comanda care a initiat trigger-ul

  if not update(CNP) return

  if @n>1
   begin
    print 'Aceasta varianta de trigger nu permite modificari in mai
multe randuri'
    rollback transaction -- anulam modificarile
    print ''Actualizarea tabelului a fost anulată''
    return
   end

  declare @CNP char(13)

  select @CNP=CNP from inserted     --inserted contine un singur
rand

  if dbo.ValidareCNP(@CNP)=0
     begin
  print 'EROARE CNP ' + @CNP
  rollback  -- anulam modificarile
  print 'Actualizarea tabelului a fost anulată'
end

END
```

2) Utilizarea cursoarelor în triggere

Vom rescrie triggerul tg_ validareCNP pentu a-l adapta comenzilor insert, delete, update ce afecteaza mai multe rânduri.

Vom crea un cursor pentru parcurgerea tabelului temporar inserted (adica rândurile nou adăugate sau modificate).

```
alter trigger scoala.tg_validareCNP on scoala.[tStudenti]
 for insert,update
as
BEGIN
 declare @n int=@@ROWCOUNT

 if not update(CNP) return

 declare @CNP char(13)
 Declare @OK int =1

 declare crs cursor for select CNP from inserted
 open crs

 fetch from crs into @CNP
 while @@FETCH_STATUS =0
  begin
    if dbo.validareCNP (@CNP) =0
      begin
set @Ok=0
       print 'CNP eronat ' + @CNP
      end
    fetch next from crs into @cnp
  end

  close crs
  deallocate crs

  Print 'Au fost verificate '+str(@n)+' CNP-uri'

  if @OK=0
    begin
  rollback

  print 'Toate actualizarile au fost anulate! ('+
str(@n,2)+' rânduri )'
    end
else
 print 'Prelucrare cu succes!'
end
```

Exemplu de utilizare

```
insert into tStudenti
(CodStud,nume,CNP,CodSpec,anStudiu,CodJud,localitate)
values('S06','Oana',
'6000506038311','Info',1,'AG','Pitesti'),

('S07','Mihaela','6010506036942','Info',1,'AG','Pitesti'),
('S08','Eugenia','6010710035028','Info',1,'AG','Pitesti')
```

```
In pagina Messages obtinem

CNP eronat 6010506036942
Au fost verificate          3 CNP-uri
Toate actualizarile au fost anulate! ( 3 rânduri )
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

3) Modificarea, prin intermediul triggerului, a datelor din alte tabele

- Sa se adauge tabelului tStudenti coloana media de tip numeric cu 2 zecimale

```
alter table scoala.tStudenti
add media decimal(4,2)
```

- Sa se actualizeze coloanal media cu media notelor studentului corespunzator

```
update tStudenti
 set media=med
from tStudenti as A inner join (select codStud,
avg(convert(decimal(4,2),nota)) as Med
from tNote  group by codStud) as B on A.CodStud=B.CodStud
```

```
select codStud,nume,media from tStudenti where media is not
null
```

codStud	nume	media
S01	Florin	7.00
S02	Mihai	7.50
S03	Alexandra	6.00
S04	Andreea	7.50
S11	Flavius	8.50
S12	Marian	6.00
S13	Adrian	9.00

- Sa se creeze un trigger care actualizează automat mediile studenților pentru care apar modificări, stergeri sau inserări de note în tabelul tNote

```
create trigger tg_ActualizareNote on scoala.tNote
after insert, update, delete
as
begin
   declare @codStud char(10),@media decimal(4,2)cc

   if exists (select * from inserted)
     declare crs cursor for select distinct codStud from
inserted
   else
     declare crs cursor for select distinct CodStud from
deleted

   open crs

   fetch from crs into @codStud
   while @@FETCH_STATUS =0
     begin
      set @media=(select avg( convert(decimal(4,2),nota))
                  from tNote   where codStud=@codStud)

      update tStudenti set media=@media where
codStud=@codStud

  fetch next from crs into @codStud
     end
     close crs
     deallocate crs
 end
```

Utilizare :

```
select codStud, nume, media from tStudenti where
codStud='S03'
```

```
codStud      nume         media
S03          Alexandra     6.00
```

```
 select * from tNote where codStud='S03'
```

```
codCurs     codStud     dataExamen           nota
ASD         S03         2020-02-08 00:00:00     8
FP          S03         2020-02-05 00:00:00     4
```

Vom modifica nota studentului 'S03' de la cursul 'FP', din 4 în 5.

```
update tNote
 set nota=5
 where CodStud='S03' and codCurs='FP'
```

Comanda update lansează automat în execuție trigger-ul tg_ActualizareNote care aduce la zi coloana media

```
select codStud, nume, media from tStudenti where
codStud='S03'
```

```
codStud     nume        media
S03         Alexandra   6.50
```

4) Trigere instead of

Pentru exemplificare vom crea tabelul tRMU (tabel pentru Registru Matricol Unic al studentilor din Romania (un astfel de tabel l-ați încărcat și dvs. cu informații la admitere).

```
create table tRMU
  (CodStud char(10) constraint Pk_RMU primary key
          constraint Fk_RMU foreign key references
scoala.tStudenti(codStud),
  stareCivila varchar(20),
  cetatenie varchar(20),
  email varchar(100),
  liceuAbsolvit varchar(100)
  -- etc (multe coloane)
  )
```

Urmatoarea vedere colecteaza informațiile din tabelele tStudenti si tRMU

```
create view
vStudentiRMU( CodStud,nume,cnp,codSpec,anStudiu,codJud,loca
litate,
                        dataNasterii,
stareCivila,cetatenie,email,liceuAbsolvit)
as
select
A.CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate,
                        dataNasterii,
stareCivila,cetatenie,email,liceuAbsolvit
from scoala.tStudenti as A left join tRMU as B on
A.codStud=B.CodStud
```

Vom crea trei triggere associate vederii vStudentiRMU, care vor da posibilitatea actualizării tabelelor tStudenti si tRMU prin intermediul vederii vStudentiRMU

Triggerele sunt de tip instead of, adică aceste triggere se vor executa în locul comenzilor SQL (insert, update sau delete) care le activează

```
create trigger tg_StdRMU_insert on vStudentiRMU
instead of insert
as
Begin
  set nocount on
  /* 1) inseram in tabelul tStudenti informatiile
corespunzatoare
   din randurile memorate in tabelul temporar inserted
(asociat vederii) Pentru aceasta, folosim comanda insert cu
select prezentată la cursul Limbajul de manipulare a
datelor
   */

    insert into scoala.tStudenti

(CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate)
    select
CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate
from inserted

   /* 2) inseram in tabelul tRMU informatiile
corespunzatoare
   din randurile memorate in tabelul temporar inserted
   */

    insert into tRMU
       (CodStud, stareCivila, cetatenie, email,
liceuAbsolvit)
    select CodStud,
stareCivila,cetatenie,email,liceuAbsolvit
from inserted

 end
```

```
create trigger tg_StdRMU_delete on vStudentiRMU
instead of delete
as
Begin
  set nocount on  -- am dezactivat afisarea nr randuri
afectate de trigger

    /* 1) stergem din tabelul tRMU
    randurile indicate in tabelul temporar deleted
   */

  delete  tRMU
  from tRMU as A inner join deleted as B
       on A.CodStud=B.codStud
```

```
  /* 2) stergem din tabelul tStudenti
    randurile indicate in tabelul temporar deleted (asociat
vederii)
   */

  delete  scoala.tStudenti
  from scoala.tStudenti as A inner join deleted as B
       on A.CodStud=B.codStud

 end
```

```
create trigger tg_StdRMU_update on vStudentiRMU
instead of update
as
Begin
  set nocount on  -- am dezactivat afisarea nr randuri
afectate de trigger

   /* 1) stergem din tabelul tRMU
    randurile indicate in tabelul temporar deleted
   */

  delete  tRMU
  from tRMU as A inner join deleted as B
       on A.CodStud=B.codStud


  /* 2) stergem din tabelul tStudenti
    randurile indicate in tabelul temporar deleted (asociat
vederii)
  */

  delete  scoala.tStudenti
  from scoala.tStudenti as A inner join deleted as B
       on A.CodStud=B.codStud

  /* 3) inseram in tabelul tStudenti informatiile
corespunzatoare
   din randurile memorate in tabelul temporar inserted
(asociat vederii)
   */

  insert into scoala.tStudenti

(CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate)
  select
CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate
from inserted

   /* 4) inseram in tabelul tRMU informatiile
corespunzatoare
   din randurile memorate in tabelul temporar inserted
   */
```

```
  insert into tRMU
      (CodStud, stareCivila, cetatenie, email,
liceuAbsolvit)
  select CodStud, stareCivila,cetatenie,email,liceuAbsolvit
from inserted



 end
```

Exemplificare :

```
insert into vStudentiRMU
      (CodStud,nume,cnp,codSpec,anStudiu,codJud,localitate,
       stareCivila,cetatenie,email,liceuAbsolvit)
 values
('S15', 'Mirela', '2990530038700', 'mate', 1, 'AG',
'Calinesti', 'necasatorit', 'Romana', 'mirela@yahoo.com',
'Col. Zinca Golescu'  ),

('S16', 'COSMIN', '1990424152477', 'mate', 1, 'VL',
'Dragasani', 'necasatorit', 'Romana', 'cosmin@yahoo.com',
'LICEUL TEHNOLOGIC BRATIANU'
)
```

In pagina Messages obtinem

```
Au fost verificate          2 CNP-uri
Prelucrare cu succes!

(2 row(s) affected)
```

```
select * from tStudenti
```

codStud	nume	CNP	codSpec	anStudiu	codJud	localitate	dataNasterii	media
S01	Florin	1990514123456	Info	1	AG	Pitesti	14/05/1999 00:00	7
S02	Mihai	1991130123456	Info	1	AG	Costesti	30/11/1999 00:00	7.5
S03	Alexandra	2980117123456	Info	2	AG	Pitesti	17/01/1998 00:00	6
S04	Andreea	2980824123456	Info	2	AG	Stefanesti	24/08/1998 00:00	7.5
S05	Iuliana	6000517031234	Info	1	AG	C.de Arges	17/05/2000 00:00	NULL
S11	Flavius	1990413123456	Mate	1	AG	Mioveni	13/04/1999 00:00	8.5
S12	Marian	1980919123456	Mate	2	AG	Pitesti	19/09/1998 00:00	6
S13	Adrian	1990718123456	Mate	1	AG	Costesti	18/07/1999 00:00	9
S14	Victoria	6010307033333	Mate	1	AG	Pitesti	07/03/2001 00:00	NULL
S15	Mirela	2990530038700	mate	1	AG	Calinesti	30/05/1999 00:00	NULL
S16	Cosmin	1990424152477	mate	1	VL	Dragasani	24/04/1999 00:00	NULL
S21	Raul	1991228123456	Bio	1	AG	Pitesti	28/12/1999 00:00	NULL
S22	Corina	2991121123456	Bio	1	AG	Stefanesti	21/11/1999 00:00	NULL
S31	Maria	2990426123456	AM	1	AG	Pitesti	26/04/1999 00:00	NULL

S41	Ioana	6000421033333	EF	1	AG	Pitesti	21/04/2000 00:00	NULL
S42	Codrut	6001212033333	EF	1	AG	Pitesti	12/12/2000 00:00	NULL

```
select * from tRMU
```

CodStud	stareCivila	cetatenie	email	liceuAbsolvit
S15	necasatorit	Romana	mirela@yahoo.com	Col. Zinca Golescu
S16	necasatorit	Romana	cosmin@yahoo.com	LICEUL TEHNOLOGIC BRATIANU

```
update vStudentiRMU
set nume='Cosmin Radu',starecivila='casatorit'
where codStud='S16'
```

```
Au fost verificate          1 CNP-uri
Prelucrare cu succes!

(1 row(s) affected)
```

```
select * from tStudenti
```

codStud	nume	CNP	codSpec	anStudiu	codJud	localitate	dataNasterii	media	
S01	Florin	1990514123456	Info	1	AG	Pitesti	14/05/1999 00:00		7
S02	Mihai	1991130123456	Info	1	AG	Costesti	30/11/1999 00:00		7.5
S03	Alexandra	2980117123456	Info	2	AG	Pitesti	17/01/1998 00:00		6
S04	Andreea	2980824123456	Info	2	AG	Stefanesti	24/08/1998 00:00		7.5
S05	Iuliana	6000517031234	Info	1	AG	C.de Arges	17/05/2000 00:00	NULL	
S11	Flavius	1990413123456	Mate	1	AG	Mioveni	13/04/1999 00:00		8.5
S12	Marian	1980919123456	Mate	2	AG	Pitesti	19/09/1998 00:00		6
S13	Adrian	1990718123456	Mate	1	AG	Costesti	18/07/1999 00:00		9
S14	Victoria	6010307033333	Mate	1	AG	Pitesti	07/03/2001 00:00	NULL	
S15	Mirela	2990530038700	mate	1	AG	Calinesti	30/05/1999 00:00	NULL	
S16	Cosmin Radu	1990424152477	mate	1	VL	Dragasani	24/04/1999 00:00	NULL	
S21	Raul	1991228123456	Bio	1	AG	Pitesti	28/12/1999 00:00	NULL	
S22	Corina	2991121123456	Bio	1	AG	Stefanesti	21/11/1999 00:00	NULL	
S31	Maria	2990426123456	AM	1	AG	Pitesti	26/04/1999 00:00	NULL	
S41	Ioana	6000421033333	EF	1	AG	Pitesti	21/04/2000 00:00	NULL	
S42	Codrut	6001212033333	EF	1	AG	Pitesti	12/12/2000 00:00	NULL	

```
select * from tRMU
```

CodStud	stareCivila	cetatenie	email	liceuAbsolvit
S15	necasatorit	Romana	mirela@yahoo.com	Col. Zinca Golescu
S16	casatorit	Romana	cosmin@yahoo.com	LICEUL TEHNOLOGIC BRATIANU

```
update vStudentiRMU
set CodStud='S17'
where CodStud='S16'
```

```
select * from tStudenti
```

codStud	nume	CNP	codSpec	anStudiu	codJud	localitate	dataNasterii	media	
S01	Florin	1990514123456	Info	1	AG	Pitesti	14/05/1999 00:00		7
S02	Mihai	1991130123456	Info	1	AG	Costesti	30/11/1999 00:00		7.5
S03	Alexandra	2980117123456	Info	2	AG	Pitesti	17/01/1998 00:00		6
S04	Andreea	2980824123456	Info	2	AG	Stefanesti	24/08/1998 00:00		7.5
S05	Iuliana	6000517031234	Info	1	AG	C.de Arges	17/05/2000 00:00	NULL	
S11	Flavius	1990413123456	Mate	1	AG	Mioveni	13/04/1999 00:00		8.5
S12	Marian	1980919123456	Mate	2	AG	Pitesti	19/09/1998 00:00		6
S13	Adrian	1990718123456	Mate	1	AG	Costesti	18/07/1999 00:00		9
S14	Victoria	6010307033333	Mate	1	AG	Pitesti	07/03/2001 00:00	NULL	
S15	Mirela	2990530038700	mate	1	AG	Calinesti	30/05/1999 00:00	NULL	
S17	Cosmin Radu	1990424152477	mate	1	VL	Dragasani	24/04/1999 00:00	NULL	
S21	Raul	1991228123456	Bio	1	AG	Pitesti	28/12/1999 00:00	NULL	
S22	Corina	2991121123456	Bio	1	AG	Stefanesti	21/11/1999 00:00	NULL	
S31	Maria	2990426123456	AM	1	AG	Pitesti	26/04/1999 00:00	NULL	
S41	Ioana	6000421033333	EF	1	AG	Pitesti	21/04/2000 00:00	NULL	
S42	Codrut	6001212033333	EF	1	AG	Pitesti	12/12/2000 00:00	NULL	

```
select * from tRMU
```

CodStud	stareCivila	cetatenie	email	liceuAbsolvit
S15	necasatorit	Romana	mirela@yahoo.com	Col. Zinca Golescu
S17	casatorit	Romana	cosmin@yahoo.com	LICEUL TEHNOLOGIC BRATIANU

```
delete vStudentiRMU where codStud in ('S15','S17')
```

```
(2 row(s) affected)
```