CURSOARE SQL SERVER

Specificul bazelor de date relaţionale, şi al limbajului **SQL** în particular, este modul de operare asupra relaţiilor ca un tot unitar. Orice operaţie, se face asupra setului complet de tuple care satisface anumite condiţii. Cursoarele constituie un mod complementar de lucru prin care se permite accesul la tuple, una câte una, şi prelucrarea independentă a fiecărei tuple în parte. Prin introducerea cursoarelor se aduce o extensie utilă limbajului **SQL** care este astfel întregit cu toate facilităţile specifice limbajelor navigaţionale:

- poziţionare pe o anumită tuplă;
- modificări ale tuplei de la poziţia curentă;
- deplasare cursor înainte-înapoi;
- prelucrarea subsetului de tuple începând de la poziţia curentă ş.a.m.d.

De remarcat că nu este indicată folosirea cursoarelor în operaţiile obişnuite care se pot rezolva prin fraze **SQL.** Cursoarele au fost introduse ca o extensie şi nu ca alternativă la frazele **SELECT, UPDATE** ş.a.m.d. care sunt, în general, mult mai rapide.

O prelucrare bazată pe cursoare se desfăşoară în mai multe faze după cum urmează:

1. **Declararea unui cursor** - unui nume de cursor i se asociază setul de tuple rezultat, corespunzător unei fraze **SELECT;** de asemenea se specifică o serie de caracteristici ale cursorului.

2. **Deschiderea cursorului** - se execută fraza **SELECT** asociată, realizându-se ceea ce se numeşte *popularea cursorului.*

3. **Încărcarea cursorului** - se poziţionează cursorul în dreptul unei tuple şi se realizează accesul la conţinutul acesteia.

1. **Prelucrare** - se execută operaţiile specifice aplicaţiei.

4. **Închiderea cursorului** - se şterge setul de tuple cu care s-a populat cursorul, dar se menţine cursorul în sine împreună cu definiţia sa.

5. **Dealocarea cursorului** - se şterge cursorul împreună cu definiţia sa.

1) Instrucţiunea DECLARE CURSOR

Defineşte atributele unui cursor şi setul de tuple cu care se populează

DECLARE *denumire_cursor* CURSOR [FORWARD_ONLY | SCROLL]
[STATIC | DYNAMIC| READ_ONLY]
FOR *frază_select*
[FOR { READ ONLY | UPDATE [OF *listă_ coloane*] }]
unde:

- *nume_cursor* - este numele cursorului;
- *FORWARD_ONLY (tip implicit) returnează secvenţial rândurile furnizate de cursor.* Modificarile efectuate asupra datelor de bază sunt vizibile de indată ce se ajunge la ele.
- SCROLL - indică faptul că toate opţiunile de navigare-accesare date (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) sunt permise. În lipsa acestei opţiuni, NEXT este singura opţiune de navigare permisă.
- STATIC - defineşte un cursor derulabil, de tip READ_ONLY, care creează în tempdb, o copie temporară a setului de tuple corespunzător. Modificările ulterioare asupra datelor originale nu sunt reflectate în datele din cursor. Sunt interzise operaţiile de actualizare la nivelul cursorului. Cursoarele de tip *static* sunt denumite *instantanee* sau INSENSITIVE deoarece ele nu sunt sensibile la modificarile efectuate în sursa lor de date.
- DYNAMIC –sunt implicit derulabile. Modificarile efectuate asupra datelor de bază sunt vizibile de indată ce se ajunge la ele. Cursoarele de tip *dynamic* sunt denumite SENSITIVE deoarece ele sunt sensibile la modificarile efectuate în sursa lor de date.
- READ_ONLY – Optiune implicită. Asa cum sugerează şi numele său, această opţiune previne modificarea datelor din cadrul cursorului. Sunt interzise operaţiile de actualizare la nivelul cursorului.

- *frază_select* - defineşte setul de tuple corespunzător cursorului;
- *READ ONLY* - interzice operaţiile de actualizare date la nivelul cursorului;
- UPDATE [OF *listă_ coloane*]- indică coloanele care pot fi modificate în cursor. Numai coloanele specificate în lista OF *listă_ coloane* pot fi modificate. Dacă această listă lipseşte, atunci, implicit, toate coloanele sunt modificabile.

2) Instrucţiunea OPEN

Deschide un cursor şi populează cursorul prin executarea frazei SELECT specificată în declaraţia cursorului.

Sintaxa:

OPEN *nume_cursor* unde:

nume_cursor - este numele unui cursor declarat anterior;

Observaţie:

După deschiderea unui cursor variabila sistem **@@CURSOR_ROWS** conţine numărul de tuple încărcate de ultima operaţie **OPEN.**

3) Instrucţiunea FETCH

Accesează o tuplă dintr-un cursor şi încarcă conţinutul acesteia într-un set de variabile.

Sintaxa:

FETCH [NEXT | PRIOR | FIRST | LAST |
ABSOLUTE { n | @nvar } | RELATIVE { n | @nvar }]
FROM *nume_cursor* [INTO *lista_variabile*]

Unde:

- **NEXT** - mută poziţia curentă la tupla următoare şi încarcă conţinutul acesteia. Dacă **FETCH NEXT** este prima operaţie de încărcare a unui cursor, atunci va încărca prima tuplă a cursorului. **NEXT** este opţiunea implicită de încărcare.
- **PRIOR -** mută poziţia curentă la tupla precedentă şi încarcă conţinutul acesteia. Dacă FETCH PRIOR este prima operaţie de încărcare a unui cursor, atunci nu se încarcă nimic şi cursorul rămâne poziţionat pe prima tuplă.
- **FIRST -** mută poziţia curentă la prima tuplă şi încarcă conţinutul acesteia.
- **LAST -** mută poziţia curentă la ultima tuplă şi încarcă conţinutul acesteia.
- **ABSOLUTE** {n | @nvar} - poziţionare absolută pe tupla din poziţia n sau @nvar de la început dacă n sau @nvar este pozitiv. Dacă n sau @nvar este negativ poziţionarea se face faţă de sfârşitul cursorului (valoarea -1 poziţionează pe ultima tuplă din cursor!). Dacă n sau @nvar este 0 nu se încarcă nimic.
- **RELATIVE** [n | @nvar] - poziţionare relativă faţă de tupla curentă. Cu n sau @nvar având valoarea 0 se încarcă tupla curentă, valoarea 1 corespunde opţiunii NEXT, iar -l corespunde opţiunii PRIOR.
- **nume_cursor -** este numele unui cursor declarat anterior.
- **INTO** @listă_variabile - permite încărcarea conţinutului tuplei curente într-un set de variabile locale. Fiecare variabilă din listă este asociată, în ordine, câte unui atribut din relaţia cu care s-a încărcat cursorul. Corespondenţa trebuie să fie unu la unu ca număr şi ca tip de dată până la nivelul conversiilor implicite suportate de SQL Server.

4) Variabila sistem @@FETCH_STATUS

Returnează starea ultimei operaţii FETCH executată asupra unui cursor. Valorile posibile ale variabilei @@FETCH_STATUS şi semnificaţia acestora este dată în următorul tabel:

@@FETCH_STATUS	Descriere
0	Instrucţiunea FETCH s-a executat cu succes.
-1	Instrucţiunea FETCH a eşuat sau cursor în afara setului rezultat.
-2	Lipsă tuplă încarcată

Observaţii:

1. Deoarece @@FETCH_STATUS este o variabilă globală testarea sa trebuie făcută imediat după operaţia a cărei stare vrem să o aflăm, înaintea oricărei alte operaţii FETCH care modifică la rândul ei variabila @@FETCH_STATUS.

2. Variabila @@FETCH_STATUS poate avea valoarea -2 în cazul unui cursor fără opţiunea INSENSITIVE, dacă între timp un utilizator concurent a şters tupla pe care s-a încercat poziţionarea prin ultima operaţie FETCH.

5) Instrucţiunea CLOSE

Închide un cursor deschis şi eliberează setul rezultat asociat. Un cursor închis poate fi redeschis din nou printr-o instrucţiune **OPEN** care calculează o nouă valoare actualizată a setului rezultat. Instrucţiunea **CLOSE** se poate executa numai asupra unui cursor deschis.

Sintaxa:

CLOSE *nume_cursor*

unde:

nume_cursor - este numele unui cursor deschis anterior;

6) Instrucţiunea DEALLOCATE

Şterge o referinţă la un cursor. La ştergerea ultimei referinţe către un cursor toate structurile de date şi resursele asociate cursorului vor fi eliberate.

Sintaxa:

DEALLOCATE *nume_cursor*

unde:

nume_cursor - este numele unui cursor declarat anterior;

Observaţii:

1. Instrucţiunea **DEALLOCATE** elimină asocierea dintre un cursor şi numele său sau variabila cursor care referă cursorul. Dacă **numele** cursor sau variabila este singura care referă cursorul, atunci cursorul este dealocat şi toate resursele folosite de acesta sunt eliberate.

2. Putem asocia o variabilă cursor cu un cursor în două moduri:

   a. Prin atribuirea unui nume cursor unei variabile cursor. De exemplu:

```
DECLARE @cursor_furnizor CURSOR
DECLARE cursor_furnizor CURSOR FOR SELECT * FROM tFurnizori
SET @cursor_furnizor = cursor_furnizor
```

   b. Prin asocierea unei definiţii de cursor direct unei variabile cursor. De exemplu:

```
DECLARE @cursor_furnizor CURSOR
SET @cursor_furnizor= CURSOR FOR SELECT * FROM tFurnizori
```

3. Variabila cursor poate fi folosită in locul numelui cursorului în instrucţiunile Open, Fetch, Close, Deallocate

4. Un cursor este menţinut şi se transmite de la un batch la altul (sau de la o procedură la alta!) până la închiderea şi dealocarea sa. În schimb o variabilă cursor există doar în batch-ul în care a fost creată şi dispare la terminarea acesteia.

Exemple de utilizare a cursoarelor

l. Fie tabelul tStudenti cu urmatorul continut:

```
select codStud, nume, codSpec from tStudenti
```

```
codStud    nume                                               codSpec
---------- -------------------------------------------------- ----------
S01        Florin                                             Info
S02        Mihai                                              Info
S03        Alexandra                                          Info
S04        Andreea                                            Info
S05        Iuliana                                            Info
S11        Flavius                                            Mate
S12        Marian                                             Mate
S13        Adrian                                             Mate
S14        Victoria                                           Mate
S21        Raul                                               Bio
S22        Corina                                             Bio
S31        Maria                                              AM
S41        Ioana                                              EF
S42        Codrut                                             EF
S43        Sanziana                                           EF

(15 row(s) affected)
```

Următoarea secvenţă afişează cod student, nume student si cod specializare în ordinea inversă a apariţiei lor fizice în tabelul tStudenti. De remarcat că setul rezultat asociat cursorului este parcurs de la sfarsit catre inceput, ceea ce impune folosirea unui cursor cu opţiunea **SCROLL.**

```
DECLARE @CodStud char(10),@nume varchar(30),@CodSpec char(10)
Declare @i int=0
DECLARE cursorStudenti CURSOR SCROLL
    FOR SELECT CodStud,nume,codSpec FROM tStudenti
OPEN cursorStudenti
FETCH LAST FROM cursorStudenti INTO @codStud,@nume,@codSpec
WHILE @@FETCH_STATUS=0
BEGIN
set @i+=1
 Print 'Studentul '+convert(char(3),@i)+': '+rtrim(@CodStud)+'  ' +
convert(char(15),@Nume) +' '+ @CodSpec
 FETCH PRIOR FROM cursorStudenti INTO @codStud, @nume,@codSpec
END
CLOSE cursorStudenti
DEALLOCATE cursorStudenti
```

```
Studentul 1  : S43  Sanziana       EF
Studentul 2  : S42  Codrut         EF
Studentul 3  : S41  Ioana          EF
Studentul 4  : S31  Maria          AM
Studentul 5  : S22  Corina         Bio
Studentul 6  : S21  Raul           Bio
Studentul 7  : S14  Victoria       Mate
Studentul 8  : S13  Adrian         Mate
Studentul 9  : S12  Marian         Mate
Studentul 10 : S11  Flavius        Mate
Studentul 11 : S05  Iuliana        Info
Studentul 12 : S04  Andreea        Info
```

```
Studentul 13 : S03  Alexandra       Info
Studentul 14 : S02  Mihai           Info
Studentul 15 : S01  Florin          Info
```

2. Fie interogarea

```
select A.codStud,Nume,avg(convert(decimal(4,2),nota)) as Media
from tStudenti as A inner join tNote as B on A.codStud=B.codStud
group by A.codStud,Nume order by media desc
```

```
codStud    Nume                                              Media
---------- ------------------------------------------------- ---------
S13        Adrian                                            9.000000
S11        Flavius                                           8.500000
S02        Mihai                                             7.500000
S04        Andreea                                           7.500000
S01        Florin                                            7.000000
S03        Alexandra                                         6.000000
S12        Marian                                            6.000000
```

Dorim sa formăm echipe din cate 2 studenti, tinand cont de mediile lor, astfel:
echipa 1: primul student cu ultimul student din lista,
echipa a 2-a: al doilea student cu penultimul student din lista, etc

```
DECLARE crs CURSOR SCROLL
FOR select A.codStud,Nume,avg(convert(decimal(4,2),nota)) as Media
from tStudenti as A inner join tNote as B on A.codStud=B.codStud
group by A.codStud,Nume order by media desc

open crs

declare @Codstd1 char(5),@Nume1 char(15),@Media1 decimal(4,2),
        @Codstd2 char(5),@Nume2 char(15),@Media2 decimal(4,2)
declare @p int, @q int
set @p=1
set @q=@@CURSOR_ROWS
while @p<@q
 begin
   fetch absolute @p from crs into @CodStd1,@Nume1,@Media1
   fetch absolute @q from crs into @CodStd2,@Nume2,@Media2

   print 'Echipa ' + str(@p,3) + ': ' +
          @Codstd1 + ' ' + @Nume1 + str(@Media1,5,2) +
              '   +   ' +
          @Codstd2 + ' ' + @Nume2 + str(@Media2,5,2)

   set @P+=1
   set @q-=1
 end
 close crs

 deallocate crs
```

```
Echipa   1: S13   Adrian          9.00   +   S12   Marian          6.00
Echipa   2: S11   Flavius         8.50   +   S03   Alexandra       6.00
Echipa   3: S02   Mihai           7.50   +   S01   Florin          7.00
```

Sincronizarea cursoarelor cu comenzile DELETE şi UPDATE

DELETE FROM *denumire_tabel* WHERE CURRENT OF *denumire_cursor*

Efectuează ştergerea poziţionată. Stergerea se referă la tupla aflată la pozitia curentă a cursorului. CURRENT OF *denumire_cursor* precizează cursorul deschis utilizat de delete, cursor ce trebuie să permită actualizarea (precizată prin clauza *for update*)

UPDATE *denumire_tabel*
SET coloana=expresie, …,coloana=expresie
WHERE CURRENT OF *denumire_cursor*

Efectuează actualizarea poziţionată. Actualizarea se referă la tupla corespunzătoare poziţiei curente a cursorului. CURRENT OF *denumire_cursor* precizează cursorul deschis utilizat de update, cursor ce trebuie să permită actualizarea (precizată prin clauza *for update*)

Exercitiul 3

Scriptul urmator realizeaza stergerea angajatilor cu salariul mai mare decat 10000, majorarea salariilor angajatilor departamentului d1 cu 20% si a salariilor celorlalti angajati cu 10%

```
declare @codDep as char(10)
declare @sal numeric(5,0)

declare cursorModiSal cursor
  for select codDep,salariu from tAngajati for update

Open cursorModiSal

Fetch next from cursorModiSal into @codDep, @sal
while @@fetch_status=0
 BEGIN

   if @sal>10000
     begin
      delete from tAngajati
       where current of cursorModiSal
      goto urmatorul
     end
   if @codDep='d1'
     update tAngajati set salariu=salariu*1.20
```

```
     where current of cursorModiSal
   else
     update tAngajati set salariu=salariu*1.10
     where current of cursorModiSal

  urmatorul:
    Fetch next from cursorModiSal into @codDep, @sal

 END

close cursorModiSal
deallocate cursorModiSal
```