Universitatea
Politehnica
București

Facultatea de
Automatică și
Calculatoare

Catedra de
Calculatoare

Laborator 5
Cursoare PL/SQL

Autori

Conf. Dr. Ing. Alexandru Boicea
As. Drd. Ing. Ciprian-Octavian Truică

Cuprins

- Cursoare implicite
- Cursoare explicite
- Cursoare cu parametri
- Tipuri de variabile REF CURSOR, SYS_REFCURSOR și RECORD
- Eficiența cursoarelor

Cursoare

- Un cursor este o construcție PL/SQL prin care se pot controla și accesa informațiile dintr-o zonă de memorie alocată pentru a stoca datele procesate prin instrucțiuni SQL;
- Există două tipuri de cursoare:
  - Implicite – create implicit de catre sistemul de gestiune pentru toate comenzile DML, chiar și pentru interogările care returnează o singură înregistrare;
  - Explicite – create de dezvoltatori pentru controlul comenzilor DML care returnează mai multe înregistrări.

Cursoare implicite

- În PL/SQL se creează implicit un cursor pentru fiecare instrucțiune DML (SELECT, INSERT, DELETE, UPDATE);
- Pentru un utilizator acest lucru este total transparent și nu poate să controleze modul de execuție al unui cursor implicit, acesta poate numai să prelucreze datele returnate de cursorul implicit;
- Trebuie să se acorde mare atenție la modul în care se creează o cerere DML pentru că pot să apară probleme la execuția unui bloc. Trebuie să se acorde atenție și la modul în care sunt puse condițiile în clauza WHERE.
- De exemplu, dacă nu se folosește o variabilă de tip colecție(vector, index-by table, nested table) cu un BULK COLLECT INTO într-un SELECT care returnează mai multe înregistrări și se va folosi o variabilă simplă atunci blocul o să dea o eroare deoarece se încearcă să se insereze mai multe valori intr-o singură variabilă.

Cursoare implicite

- Ex. 1. Exemplu greșit de folosire a unui cursor implicit (returnează mai mute valori care nu pot să fie inserate în variabila salariu):

```
set serveroutput on;
declare
  salariu number;
begin
  select sal into salariu from emp where deptno = 20;
end;
```

- Ex. 2. Exemplu corect de folosire a unui cursor implicit (clauza where conține o condiție care garantează că se va întoarce o singură valoare):

```
set serveroutput on;
declare
  salariu number;
begin
  select sal into salariu from emp where empno = 7499;
end;
```

Cursoare implicite

- Ex. 3. Exemplu corect de folosire a unui cursor implicit care returnează mai multe înregistrări (se folosește un nested table în care se inserează salariile angajaților din departamentul 20)

```
set serveroutput on;
declare
  type nestedTable is table of emp.sal%type;
  salariu nestedTable;
begin
  select sal bulk collect into salariu from emp where deptno = 20;
end;
```

Cursoare explicite

- Cursoarele explicite sunt declarate de catre utilizator și sunt folosite pentru prelucrarea rezultatelor interogărilor care returnează linii multiple;
- Limitări:
  - Există un parametru de sistem OPEN_CURSOR care setează numărul maxim de cursoare care se pot deschide pe parcursul unei sesiuni:
    - pentru a vedea sau modifica acest parametru trebuie să ai privilegiu de DBA și se folosesc comenzile:
      - select name, value from v$parameter where name like 'open_cursors';
      - *ALTER SYSTEM SET open_cursors = 400 SCOPE=BOTH;*
  - O altă limitare a numărului de cursoare este dată de memoria disponibilă pentru gestionarea lor.

Cursoare explicite – secțiunea DECLARE

- În secțiunea **DECLARE** se definește numele cursorului și structura interogării care va fi efectuată cu el;
- În această secțiune, interogarea este compilată dar nu este executată, semnalându-se eventualele erori de compilare;
- Sintaxa este:

```
DECLARE
    CURSOR cursor_name [(<parameter> <parameter_type>)] IS
        SELECT column_manes FROM table_names
        WHERE conditions
        [FOR UPDATE [OF col1_name[, OF col2_name[, …]]]];
    variable cursor_name%rowtype;
```

Cursoare explicite – secțiunea DECLARE

- **cursore_name** – numele atribuit cursorului
- **column_names** – numele coloanelor returnate de cursor
- **table_names** – numele tabelelor folosite de interogare
- **conditions** – condițiile de filtrare sau de join
- **parameter** – numele parametrilor cursorului și este opțional
- **parameter_type** – tipul parametrilor
- **variable** – este o variabilă de tipul unei linii din cursor
- **for update** – este opțiunea de blocare a liniilor (*lock*) selectate de cursor în baza de date

Observatii:

- Interogarea SELECT poate să fie de orice tip, poate să conțină joinuri și poate să conțină subcereri;
- Există o convenție în Oracle care recomandă ca numele cursoarelor să înceapă cu litera **c**; folosind un astfel de nume va fi clar că numele din secțiunea de declarare se referă la un cursor.

Cursoare explicite – secțiunea DECLARE

- O înregistrare care este bazată pe cursor (cursor-based) este o înregistrare a cărei structuri este de tipul unei linii a cursorului;
- Pentru a declara o variabilă bazată pe cursor se folosește atributul %ROWTYPE;
- Referirea unui element al variabilei de tip cursor se face adăugând ca prefix numele variabilei (asemănător cu referirea unei coloane din tabel):

  value := variable_name.element_name
- Numele atribuit unui cursor nu este o variabilă PL/SQL, ci un identificator, deci nu se pot atribui valori numelui unui cursor și nici folosi în expresii;
- Rezultatele interogării cursorului devine setul activ de date al cursorului.

Cursoare explicite – secțiunea OPEN

- În secțiunea OPEN se deschide cursorul și se face execuția efectivă a interogării, obținându-se ca rezultat liniile care vor constitui setul activ de date;
- Sintaxa este:

  OPEN cursor_name [(<parameter> <parameter_type>)];
- **cursor_name** - numele atribuit cursorului
- **parameter** – numele parametrilor formali ai cursorului
- **parameter_type** – tipul parametrilor

Cursoare explicite – secțiunea OPEN

- Cursorul se deschide cu valorile actuale pentru fiecare parametru formal specificat în declarația cursorului, în caz contrat parametrii care apar în declarația cursorului vor avea valori prestabilite;
- După ce s-a deschis cursorul se pot atribui parametrii actuali la parametrii formali folosind operatorul de atribuire;
- Atunci când se execută comanda OPEN, cursorul identifică numai acele linii care satisfac condiția de interogare, dar liniile nu sunt citite până când nu se face prelucrarea de către cursor;
- Cursorul se inițializează la prima linie a setului activ.

Cursoare explicite – secțiunea OPEN

• Fiecare cursor explicit are patru atribute care sunt setate automat de catre sistemul de gestiune și conțin informații despre starea lor:

Atribut	Semnificație
%ISOPEN	Returnează TRUE dacă cursorul este deschis și FALSE dacă cursorul este închis;
%FOUND	Returnează: -INVALID_CURSOR dacă cursorul este declarat dar nu este deschis sau dacă cursorul a fost închis; -NULL dacă cursorul este deschis dar nu a fost executat; -TRUE dacă atribuirea s-a efectuat cu succes; -FALSE dacă nu a fost returnat niciun rând.

Cursoare explicite – secțiunea OPEN

Atribut	Semnificație
%NOTFOUND	Returnează: -INVALID_CURSOR dacă cursorul este declarat dar nu este deschis sau dacă cursorul a fost închis; -NULL dacă cursorul este deschis dar nu a fost executat; -FALSE dacă atribuirea s-a efectuat cu succes; -TRUE dacă nu a fost returnat niciun rând.
%ROWCOUNT	Returnează: -INVALID_CURSOR dacă cursorul este declarat dar nu este deschis sau dacă cursorul a fost închis; - Numărul de linii selectate și atribuite.

Cursoare explicite – secțiunea FETCH

- În secțiunea **FETCH** se memorează valorile din linia curentă a cursorului într-o variabilă;
- Sintaxa este:

```
LOOP
        FETCH cursor_name INTO variable_names;
        EXIT WHEN condition;
        ...
END LOOP;
```

- **cursore_name** – numele atribuit cursorului
- **variable_names** – numele variabilelor în care se memorează valorile din linia curentă a cursorului
- **condition** – condiția de ieșire din ciclul LOOP
- Instrucțiunea **FETCH** preia liniile din setul activ una câte una și sortează setul activ, dacă este cazul;
- Cursorul avansează automat pe următoarea linie la executarea comenzii FETCH

Cursoare explicite – secțiunea CLOSE

- În secțiunea **CLOSE** se închide setul de linii returnate de catre cursor după deschidere;
- Sintaxa este:

  CLOSE cursor_name;

- **cursor_name** – numele cursorului
- După închiderea unui cursor rezultatul interogării se pierde;
- Pentru a accesa mulțimea datelor asociate unui cursor închis trebuie redeschis cursorul.

Cursoare explicite

- Structura completă a unui cursor explicit este următoarea:

```
DECLARE
        CURSOR cursor_name IS select ...;
        variable cursor_name%rowtype;
BEGIN
        OPEN cursor_name;
        LOOP
                FETCH cursor_name INTO variable;
                EXIT WHEN condition
                ...
        END LOOP;
        CLOSE cursor_name;
END
```

- În general, **condition** este **cursor_name%NOTFOUND**

Cursoare explicite

- Ex. 4. Să se declare un cursor care selectează denumirea departamentului, numele angajatului, salariul și data angajării pentru acei angajati care au venit în companie în 1981.

```
set serveroutput on;
declare
  cursor c_angajati is
    select d.dname, e.empno, e.ename, e.sal, e.hiredate
      from emp e inner join dept d on e.deptno=d.deptno
      where e.hiredate like '%81'
      order by hiredate;
    v_angajat c_angajati%rowtype;
begin
  open c_angajati;
  loop
    fetch c_angajati into v_angajat;
    exit when c_angajati%notfound;
    dbms_output.put_line(v_angajat.dname||' '||v_angajat.empno||' '||
      v_angajat.ename||' '||v_angajat.sal||' '||v_angajat.hiredate);
  end loop;
  close c_angajati;
end;
```

Cursoare explicite

- Ex. 5. Să se folosească un cursor pentru a face o listă cu veniturile managerilor din companie.

```
set serveroutput on;
declare
  cursor c_angajati is
    select d.dname, e.empno, e.ename, e.sal, e.comm
      from emp e inner join dept d on d.deptno = e.deptno
     where lower(e.job) like lower('Manager');
  angajat c_angajati%rowtype;
  venit number := 0;
```

Cursoare explicite

```
begin
  open c_angajati;
  dbms_output.put_line(rpad('ECUSON', 10)||rpad('NUME', 20)||
    rpad('DEPARTAMENT',20)||lpad('VENIT',10));
  dbms_output.put_line(rpad('=', 10, '=')||rpad('=', 20, '=')||
    rpad('=',20, '=')||lpad('=',10, '='));
  loop
    fetch c_angajati into angajat;
    exit when c_angajati%NOTFOUND;
    venit := round(angajat.sal+nvl(angajat.comm, 0));
    dbms_output.put_line(rpad(angajat.empno, 10)||rpad(angajat.ename, 20)||
      rpad(angajat.dname,20)||lpad(venit,10));
    venit := 0;
  end loop;
  close c_angajati;
end;
```

Cursoare explicite

- Setul activ de date definite de cursorul *c_angajati* este procesat linie cu linie;
- Ciclul iterativ de citire și procesare a liniilor individuale continuă până la procesarea tuturor liniilor din setul activ;
- Ieșirea din buclă se face testând condiția %NOTFOUND care se setează pe true după ce cursorul procesează toate liniile;
- După închiderea cursorului datele din setul activ sunt resetate.

Cursoare explicite

- Ciclul **FOR** este folosit pentru simplificarea sintaxei de prelucrare a liniilor dintr-un cursor;
- Cursorul însuși este cel care determină momentul în care se părăsește ciclul;
- Sintaxa este:

```
DECLARE
        CURSOR cursor_name IS select ...;
        variable cursor_name%rowtype;
BEGIN
        FOR variacle IN cursor_name
        LOOP
                ...
        END LOOP;
END;
```

Cursoare explicite

- Ciclul **FOR** simplifică codul programului și efectuează următoarele operații:
  - La inițializarea ciclului se execută în mod implicit o instrucțiune **OPEN** ;
  - La fiecare parcurgere se execută în mod implicit o instrucțiune **FETCH** ;
  - La părăsirea ciclului se execută în mod implicit o instrucțiune **CLOSE**.

Cursoare explicite

- Ex. 6. Rescriere exemplu 5 folosind **FOR.**

```
set serveroutput on;
declare
  cursor c_angajati is
    select d.dname, e.empno, e.ename, e.sal, e.comm
      from emp e inner join dept d on d.deptno = e.deptno
      where lower(e.job) like lower('Manager');
  angajat c_angajati%rowtype;
  venit number := 0;
begin
  dbms_output.put_line(rpad('ECUSON', 10)||rpad('NUME', 20)||
    rpad('DEPARTAMENT',20)||lpad('VENIT',10));
  dbms_output.put_line(rpad('=', 10, '=')||rpad('=', 20, '=')||
    rpad('=',20, '=')||lpad('=',10, '='));
  for angajat in c_angajati
  loop
    venit := round(angajat.sal+nvl(angajat.comm, 0));
    dbms_output.put_line(rpad(angajat.empno, 10)||rpad(angajat.ename, 20)||
      rpad(angajat.dname,20)||lpad(venit,10));
    venit := 0;
  end loop;
end;
```

Cursoare explicite

- Într-un ciclu **FOR** se poate folosi un SELECT care este tot un cursor, dar care nu trebuie să mai fie declarat în secțiunea DECLARE;
- Sintaxa este:

```
BEGIN
        FOR variable IN (SELECT column_manes FROM table_names
                                        WHERE conditions)
        LOOP
                ...
        END LOOP;
END;
```

Cursoare explicite

- Ex. 7. Să se rescrie exercițiul 6 folosind un SELECT în ciclul FOR.

```
set serveroutput on;
declare
  venit number := 0;
begin
  dbms_output.put_line(rpad('ECUSON', 10)||rpad('NUME', 20)||
    rpad('DEPARTAMENT',20)||lpad('VENIT',10));
  dbms_output.put_line(rpad('=', 10, '=')||rpad('=', 20, '=')||
    rpad('=',20, '=')||lpad('=',10, '='));
  for angajat in (select d.dname, e.empno, e.ename, e.sal, e.comm
      from emp e inner join dept d on d.deptno = e.deptno
      where lower(e.job) like lower('Manager'))
  loop
    venit := round(angajat.sal+nvl(angajat.comm, 0));
    dbms_output.put_line(rpad(angajat.empno, 10)||rpad(angajat.ename, 20)||
      rpad(angajat.dname,20)||lpad(venit,10));
    venit := 0;
  end loop;
end;
```

Cursoare explicite

- Când ne referim la rândul curent dintr-un cursor explicit comenzile SQL pot folosi clauza **WHERE CURRENT OF cursor_name**;
- Această clauză permite actualizarea sau ștergerea în punctul în care se află cursorul, fără a fi necesară folosirea condițiilor în clauza **WHERE** pentru identificarea unică a liniei;
- Trebuie să se folosească clauza **FOR UPDATE** în definirea cursorului pentru a se bloca rândurile la deschidere (se face un *lock*);
- Clauza **WHERE CURRENT OF cursor_name** se poate folosi si in comenzile UPDATE și DELETE.

Cursoare explicite

- Ex. 8. Să se modifice comisionul cu 10% din salariu pentru angajații care au peste 20 ani vechime în companie.

```
set serveroutput on;
declare
  cursor c_angajati is
    select d.dname, e.empno, e.ename, e.sal, e.comm, e.hiredate
      from emp e inner join dept d on d.deptno = e.deptno for update of comm;
  angajat c_angajati%rowtype;
  comisionNou number default 0;
begin
  open c_angajati;
  loop
    fetch c_angajati into angajat;
    exit when c_angajati%notfound;
    if add_months(angajat.hiredate, 240)< sysdate then
      comisionNou := nvl(angajat.comm, 0) + round(0.1*angajat.sal, 0);
      update emp set comm = comisionNou where current of c_angajati;
    end if;
  end loop;
  close c_angajati;
end;
```

Cursoare explicite

- Ex. 9. Să se șteargă din tabela EMP toți angajații care au comision.

```
set serveroutput on;
declare
  cursor c_angajati is
    select d.dname, e.empno, e.ename, e.sal, e.comm, e.hiredate
      from emp e inner join dept d on d.deptno = e.deptno for update;
  angajat c_angajati%rowtype;
  comisionNou number default 0;
begin
  for angajat in c_angajati
  loop
    if not (nvl(angajat.comm, 0) = 0) then
      delete from emp where current of c_angajati;
    end if;
  end loop;
end;
/
```

Cursoare explicite

- Parametrii permit transmiterea unor valori efective unui cursor când acesta este deschis;
- Parametrii formali se definesc în momentul declarării cursorului;
- Tipurile parametrilor sunt aceleași cu ale variabilelor scalare, dar nu primesc dimensiuni;
- Parametrii sunt folosiți pentru referirea în cadrul expresiei de definire din cadrul cursorului și pot fi tratați și ca variabile PL/SQL.

Cursoare explicite

- Ex. 10. Să se facă o listă cu angajații care fac parte dintr-un departament specificat, au o anumită funcție și au venit în companie la o anumită dată specificată. Aceste condiții să fie transmise ca parametri unui cursor.

```
set serveroutput on;
declare
  cursor c_angajati(idDept number, functie char, dataAng date) is
    select deptno, ename, job, hiredate from emp
      where deptno=idDept and lower(job)=lower(functie) and hiredate>dataAng;
  angajat c_angajati%rowtype;
begin

  open c_angajati(10, 'clerk', '1-JUL-81');
  loop
    fetch c_angajati into angajat;
    exit when c_angajati%notfound;
    dbms_output.put_line(rpad(angajat.deptno, 10)||rpad(angajat.ename, 20)||
      rpad(angajat.job,20)||lpad(angajat.hiredate,10));
  end loop;
  close c_angajati;
```

Cursoare explicite

```
open c_angajati(idDept => 20, functie =>'clerk', dataAng => '1-JUL-81');
loop
  fetch c_angajati into angajat;
  exit when c_angajati%notfound;
  dbms_output.put_line(rpad(angajat.deptno, 10)||rpad(angajat.ename, 20)||
    rpad(angajat.job,20)||lpad(angajat.hiredate,10));
end loop;
close c_angajati;

open c_angajati(functie =>'clerk', dataAng => '1-JUL-81', idDept => 10);
loop
  fetch c_angajati into angajat;
  exit when c_angajati%notfound;
  dbms_output.put_line(rpad(angajat.deptno, 10)||rpad(angajat.ename, 20)||
    rpad(angajat.job,20)||lpad(angajat.hiredate,10));
end loop;
close c_angajati;

open c_angajati(20, functie=>'clerk',  dataAng =>'1-JUL-81');
loop
  fetch c_angajati into angajat;
  exit when c_angajati%notfound;
  dbms_output.put_line(rpad(angajat.deptno, 10)||rpad(angajat.ename, 20)||
    rpad(angajat.job,20)||lpad(angajat.hiredate,10));
end loop;
close c_angajati;
```

Cursoare explicite

```
  open c_angajati(10, 'clerk',  dataAng =>'1-JUL-81');
  loop
    fetch c_angajati into angajat;
    exit when c_angajati%notfound;
    dbms_output.put_line(rpad(angajat.deptno, 10)||rpad(angajat.ename, 20)||
      rpad(angajat.job,20)||lpad(angajat.hiredate,10));
  end loop;
  close c_angajati;

end;
```

Tipuri de variabile
REF CURSOR și RECORD

- Se poate asocia o interogare unui cursor astfel:
  - Se declară un tip de date **REF CURSOR** și se declară o variabilă cu acest tip;
  - Se declară o variabilă cu tipul de date **SYS_REFCURSOR.**
- Acest mod de a declara un cursor este util în momentul în care se dorește să se transmită seturi de date la subprograme(proceduri, funcții, pachete)
- Utilizatorul poate să-și definească propriul tip de date pentru a păstra o linie dintr-un cursor folosind o variabilă de tip RECORD .

Tipuri de variabile
REF CURSOR și RECORD

- Sintaxa **REF CURSOR**:

```
DECLARE
        TYPE ref_cursor IS REF CURSOR;
        c_variable ref_cursor;
        variable variable_type;
BEGIN
        OPEN c_variable FOR SELECT column_manes FROM table_names WHERE
                conditions  [FOR UPDATE [OF col1_name[, col2_name, …]]];
        LOOP
                FETCH c_variable INTO variable;
                EXIT WHEN condition
                …
        END LOOP;
        CLOSE c_variable;
END;
```

Tipuri de variabile
REF CURSOR și RECORD

unde:

- **ref_cursor** – numele variabilei de tip **REF CURSOR**
- **c_variable** – numele variabilei cursor
- **variable** – numele unei variabile normale
- **conditions** – condițiile de ieșire din buclă puse pentru **c_variable**

Tipuri de variabile
REF CURSOR și RECORD

- Sintaxa **SYS_REFCURSOR**:

```
DECLARE
        c_variable SYS_REFCURSOR;
        variable variable_type;
BEGIN
        OPEN c_variable FOR SELECT column_manes FROM table_names WHERE
                conditions  [FOR UPDATE [OF col1_name[, col2_name, …]]];
        LOOP
                FETCH c_variable INTO variable;
                EXIT WHEN condition
                …
        END LOOP;
        CLOSE c_variable;
END;
```

Tipuri de variabile
REF CURSOR și RECORD

- Ex. 11. Să se facă o listă cu toate departamentele.

```
set serveroutput on;
declare
  type r_cursor is REF CURSOR;
  c_dept r_cursor;
  depart dept.dname%type;
begin
  open c_dept for select dname from dept;
  loop
    fetch c_dept into depart;
    exit when c_dept%notfound;
    dbms_output.put_line(depart);
  end loop;
  close c_dept;
end;
```

Tipuri de variabile
REF CURSOR și RECORD

- Ex. 12. Să se listeze toți angajatii din tabela EMP.

```
set serveroutput on;
declare
  c_emp SYS_REFCURSOR;
  angajat emp%rowtype;
begin
  open c_emp for select * from emp;
  loop
    fetch c_emp into angajat;
    exit when c_emp%notfound;
    dbms_output.put_line(angajat.ename||' '||angajat.job||' '||angajat.sal);
  end loop;
  close c_emp;
end;
/
```

Tipuri de variabile
REF CURSOR și RECORD

- Ex. 13. Să se folosească un record pentru a păstra numele, funcția și salariul pentru toți angajații.

```
set serveroutput on;
declare
  type r_cursor is ref cursor;
  type rec_ang is record(
    nume emp.ename%type,
    functie emp.job%type,
    salariu emp.sal%type
  );
  c_ang r_cursor;
  angajat rec_ang;
begin
  open c_ang for select ename, job, sal from emp;
  loop
    fetch c_ang into angajat;
    exit when c_ang%notfound;
    dbms_output.put_line(angajat.nume||' '||angajat.functie
      ||' '||angajat.salariu);
  end loop;
  close c_ang;
end;
```

Eficiența cursoarelor

- Pentru a mări eficiența cursoarelor trebuie să se țină cont de următoarele facilități:
  - Crearea de indecși pe coloanele folosite în clauzele WHERE și JOIN;
  - Prelucrarea procedurală permite un control puternic și flexibil al liniilor din baza de date, dar fiecare INSERT sau UPDATE provoacă o reorganizare a indecșilor;
  - Efectuarea, pe cât posibil, a calculelor în cererea SELECT;
  - Prefixarea coloanelor cu numele userului și a tabelei;
  - În cazul unui JOIN indexat, să se facă referire la tabela cea mai mică la sfârșit.

Eficiența cursoarelor

- Pentru a mări eficiența cursoarelor trebuie să se țină cont de următoarele facilități:
  - Să se folosească IF pentru comparație, dacă toate datele de prelucrat sunt stocate în variabile;
  - Să se evite SELECT ... FROM SYS.DUAL deoarece produce deschiderea unui cursor și transmiterea unor cereri către RDBMS;
  - Cursoarele explicite evită un FETCH ulterior față de cele implicite care execută două cereri(testează și linia următoare);
  - Dacă se prelucrează mai multe linii dintr-o tabelă, este bine să se facă într-un singur pas, evitând trecerile repetate prin tabelă.

Eficiența cursoarelor

- Performanțele cursoarelor pot fi îmbunătățite prin ajustarea parametrului de sistem CURSOR_SPACE_FOR_TIME care precizează momentul în care zona partajată (*shared pool*) poate fi dealocată pentru a face loc unei noi comenzi SQL și are valoarea implicită FALSE;
- Aceasta înseamnă că SGBD-ul poate dealoca zona partajată chiar dacă este deschis un cursor al unei aplicații;
- Valoarea TRUE înseamnă că o zonă partajată poate fi dealocată numai atunci când toate cursoarele aplicației sunt închise, cu condiția ca zona să fie suficient de mare pentru ca toate cursoarele să fie deschise simultan;
- Setarea parametrului pe TRUE face ca SGBD-ul să realizeze o economie de timp și resurse și poate îmbunătăți performanțele apelurilor de execuție;
- Nu se recomandă totdeauna setarea pe TRUE deoarece parametrul trebuie corelat cu alți parametri de sistem pentru dimesionarea zonei partajate și a numărului maxim de cursoare deschise simultan.

Cursoare explicite

- Ex. 14. Să se scrie un cursor explicit PL/SQL care face o listă pentru acordarea de comisioane tuturor șefilor de departament(care se identifică în coloana mgr din tabela EMP), după următorul algoritm:
  - Dacă salariu_sef < salariu_mediu_departament atunci comision_sef = 10% * salariu_mediu_departament
  - Dacă salariu_sef >= salariu_mediu_departament atunci comision_sef = 20% * salariu_mediu_departament

Cursoare explicite

```
set serveroutput on;
declare
  cursor c_sef is select distinct mgr from emp where mgr is not null;
  numeAng emp.ename%type;
  idDept emp.deptno%type;
  numeDept dept.dname%type;
  salariu emp.sal%type;
  idAng emp.empno%type;
  salMediu number;
  comision number;
begin
  dbms_output.put_line(rpad('Departament', 20)||rpad('Salariu mediu', 15)
    ||rpad('Nume', 20)||lpad('Salariu',10)||lpad('Comision', 10));
  dbms_output.put_line(rpad('-', 20, '-')||rpad('-', 15, '-')
    ||rpad('-', 20, '-')||lpad('-',10, '-')||lpad('-', 10, '-'));
```

Cursoare explicite

```
  for rec in c_sef
  loop
    select ename, deptno, sal, empno into numeAng, idDept, salariu, idAng
      from emp where empno = rec.mgr;
    select round(avg(sal)) into salMediu from emp
      where deptno=idDept;
    select dname into numeDept from dept where deptno=idDept;

    if salariu<salMediu then
      comision := round(0.1*salMediu);
    else
      comision := round(0.2*salMediu);
    end if;
    dbms_output.put_line(rpad(numeDept, 20)||rpad(salMediu, 15)
    ||rpad(numeAng, 20)||lpad(salariu,10)||lpad(comision, 10));
  end loop;
end;
```