

Limbajul PL/SQL- Interactiunea cu Oracle

DECLARE

TYPE TypEmp IS REF CURSOR RETURN emp%ROWTYPE;

emp_cv1 TypEmp;

emp_cv2 TypEmp;

emp_rec emp%ROWTYPE;

BEGIN

emp_cv2:=emp_cv1; -- inutil

OPEN emp_cv1 FOR SELECT * FROM emp...;

/* emp_cv1 pointeaza catre o zona de lucru */

FETCH emp_cv1 INTO emp_rec;

FETCH emp_cv2 INTO emp_rec; --exceptia INVALID_CURSOR

EXCEPTION

WHEN INVALID_CURSOR THEN

/* facem ca emp_cv2 si emp_cv1 sa pointeze aceeasi zona de lucru*/121

Limbajul PL/SQL- Interactiunea cu Oracle

```
emp_cv2:=emp_cv1;
  FETCH emp_cv2 INTO emp_rec; -- returneaza a doua linie.
  OPEN emp_cv2 FOR SELECT * FROM emp1...;
/* reutilizarea zonei de lucru pentru alta interogare */
  FETCH emp_cv1 INTO rep_rec;
/* furnizeaza prima linie din tabela emp1 */
END;
```

Daca parametrii actuali si cei formali nu au acelasi tip- exceptia

ROWTYPE_MISMATCH

```
emp_cv2:=emp_cv1; -- primul este un alias pentru al doilea
PROCEDURE proc1 (emp_cv1 IN OUT TypEmp,
                 emp_cv2 IN OUT TypEmp) IS
emp_rec emp%ROWTYPE;
```

Limbajul PL/SQL- Interactiunea cu Oracle

```
BEGIN
```

```
OPEN emp_cv1 FOR SELECT * FROM emp;
```

```
emp_cv2:=emp_cv1;
```

```
FETCH emp_cv1 INTO emp_rec; -- prima linie
```

```
FETCH emp_cv2 INTO emp_rec; -- a doua linie
```

```
FETCH emp_cv1 INTO emp_rec; -- a treia linie
```

```
CLOSE emp_cv1;
```

```
FETCH emp_cv2 INTO emp_rec; -- exceptia INVALID_CURSOR;
```

```
END proc1;
```

Aceeasi eroare apare cind un parametru apare de 2 ori intr-un apel.

```
Procedure proc2 (emp_cv1 IN OUT TypEmp,emp_cv2 IN OUT  
TypEmp) IS ...
```

```
emp_cv TypEmp;      proc2(emp_cv,emp_cv);
```

Limbajul PL/SQL- Interactiunea cu Oracle

- Restrictii asupra variabilelor cursor:
 - Nu se pot declara in pachete deoarece nu au o stare persistenta.
 - O interogare asociata cu o variabila cursor intr-o instructiune OPEN-FOR nu poate fi FOR UPDATE.
 - Nu putem folosi variabile cursor in operatori de comparare:=,!=, NULL
 - Nu putem asigna NULL la variabile cursor
 - Nu putem folosi tipul REF CURSOR pentru o coloana in comenzile CREATE TABLE si CREATE VIEW.
 - Nu putem folosi REF CURSOR sa specificam tipul elementelor unei colectii.
 - Cursorii si variabilele cursor nu sunt interoperabile.

```
CURSOR emp_1 IS SELECT * FROM emp;
```

```
TYPE TypEmp IS REF CURSOR RETURN emp%ROWTYPE;
```

```
emp_cv TypEmp;
```

```
... FOR emp_rec IN emp_cv LOOP ... --illegal
```

Limbajul PL/SQL- Interactiunea cu Oracle

```
DECLARE TYPE EmpTyp IS REF CURSOR RETURN
emp%ROWTYPE;
empv EmpTyp;
PROCEDURE process_emp_cv (emp_cv IN EmpTyp) IS
person emp%ROWTYPE;
BEGIN
dbms_output.put_line('*****');
LOOP
FETCH emp_cv INTO person;
EXIT WHEN emp_cv%NOTFOUND;
dbms_output.put_line('Nume = ' || person.first_name || ' ' ||
person.last_name);
END LOOP;
END;
```

Limbajul PL/SQL- Interactiunea cu Oracle

```
BEGIN
```

```
OPEN empv FOR SELECT * FROM emp WHERE ROWNUM <  
10;
```

```
process_emp_cv(empv);
```

```
CLOSE emp;
```

```
END;
```

- Expresii cursor:

Returneaza un cursor imbricat. O linie returnata de un cursor obisnuit poate contine valori sau o submultime de alte linii produse de subinterogari.

Sintaxa expresiei cursor: `CURSOR(interogare SELECT)`

Expresiile cursor pot apare in declaratii cursor, declaratii REF CURSOR, interogari SQL.

Limbajul PL/SQL- Interactiunea cu Oracle

```
DECLARE TYPE emp_typ IS REF CURSOR;
    emp_c emp_typ;
    dept_name departments.department_name%TYPE;
    emp_name emp.last_name%TYPE;
    CURSOR c1 IS SELECT department_name,
        CURSOR ( SELECT e.last_name FROM emp e
            WHERE e.department_id = d.department_id ) employees
    FROM departments d WHERE department_name like
    'INFO%';
BEGIN
    OPEN c1;
    LOOP FETCH c1 INTO dept_name, emp_c;
    EXIT WHEN c1%NOTFOUND;
```

Limbajul PL/SQL- Interactiunea cu Oracle

```
dbms_output.put_line('Department: ' || dept_name);  
-- Pentru fiecare linie a rezultatului prelucram submultimea  
definita de subinterogare.  
    LOOP FETCH emp_c INTO emp_name;  
    EXIT WHEN emp_c%NOTFOUND;  
    dbms_output.put_line(' Employee: ' || emp_name);  
    END LOOP;  
END LOOP;  
CLOSE c1;  
END;
```

Restrictii: -Nu putem folosi o expresie cursor cu un cursor implicit
-Nu pot apare in declaratii view
- Nu putem realiza operatii EXECUTE pe expresii cursor

Limbajul PL/SQL- Interactiunea cu Oracle

- Attribute cursor : %FOUND, %ISOPEN, %NOTFOUND, %ROWCOUNT

```
CURSOR C1...
```

```
IF C1%ISOPEN THEN – cursor deschis
```

```
...
```

```
ELSE -- il deschidem
```

```
OPEN C1;
```

```
...
```

```
END IF;
```

Inainte de FETCH, %NOTFOUND este NULL.

```
EXIT WHEN C1%NOTFOUND OR C1%NOTFOUND IS NULL
```

Inainte de deschiderea unui cursor sau variabila cursor

%ROWCOUNT este 0.

Limbajul PL/SQL- Interactiunea cu Oracle

LOOP

```
FETCH C1 INTO inreg;
```

```
IF C1%ROWCOUNT >11 THEN
```

```
...
```

```
END IF;
```

```
...
```

```
END LOOP;
```

In cazul cind cursorul e nedeschis referirea la %ROWCOUNT produce exceptia INVALID_CURSOR.

Dupa prima executie FETCH daca nu exista linii in multimea respectiva, atunci %FOUND este FALSE, %NOTFOUND este TRUE si %ROWCOUNT este 0.

- Atributele pentru cursori impliciti sunt aceleasi, dar returneaza informatii despre executia comenzilor INSERT, UPDATE, DELETE, SELECT.

Limbajul PL/SQL- Interactiunea cu Oracle

- %FOUND

Inainte ca instructiunea SQL sa fie executata, %FOUND este NULL. Apoi %FOUND este TRUE daca INSERT, UPDATE sau DELETE a afectat cel putin o linie, sau SELECT INTO returneaza cel putin o linie. Altfel %FOUND este FALSE.

```
DELETE FROM emp WHERE empno=emp1;  
IF SQL%FOUND THEN – stergere cu succes  
    INSERT INTO emp_new VALUES(emp1,..);
```

- %ISOPEN

Oracle inchide automat cursorul SQL dupa executia instructiunii SQL. Deci%ISOPEN este mereu FALSE.

- %NOTFOUND

Produce TRUE daca INSERT, UPDATE sau DELETE nu afecteaza nicio linie, sau SELECT INTO nu returneaza nicio linie. Altfel, ea produce FALSE.

Limbajul PL/SQL- Interactiunea cu Oracle

- `%ROWCOUNT`

Produce numarul de linii afectate de `INSERT`, `UPDATE`, `DELETE` sau returnate de `SELECT INTO`.

Daca `SELECT INTO` returneaza cel putin 2 linii, atunci apare exceptia `TOO_MANY_ROWS` si `%ROWCOUNT` este 1.

Valorile atributelor cursor se refera la cea mai recenta instructiune SQL executata.

Daca `SELECT INTO` nu returneaza nicio linie , atunci PL/SQL produce exceptia `NO_DATA_FOUND`.

`SELECT INTO` cu o functie de grupare nu produce exceptia de sus, deoarece functia de grupare produce o valoare sau `NULL`. Atunci `%NOTFOUND` este `FALSE`.

Limbajul PL/SQL- Interactiunea cu Oracle

- Prelucrarea tranzactiilor

-COMMIT

Face permanente schimbarile din timpul tranzactiei.

```
BEGIN
```

```
SELECT suma INTO suma1 FROM cont WHERE nrcont=1500;
```

```
UPDATE cont SET suma= suma1-debit
```

```
WHERE nrcont=1500;
```

```
...
```

```
UPDATE cont SET suma=suma1+debit
```

```
WHERE nrcont=1600;
```

```
COMMIT WORK;
```

```
END;
```

```
COMMIT COMMENT 'mesaj';
```

Limbajul PL/SQL- Interactiunea cu Oracle

- ROLLBACK

```
DECLARE
```

```
emp_id INTEGER;
```

```
...
```

```
BEGIN
```

```
  SELECT empno,... INTO emp_id,... FROM ..
```

```
  ...
```

```
  INSERT INTO emp VALUES (emp_id,...);
```

```
  INSERT INTO emp1 VALUES (emp_id,...);
```

```
  INSERT INTO emp2 VALUES (emp_id,...);
```

```
  ...
```

```
EXCEPTION
```

```
  WHEN DUP_VAL_ON_INDEX THEN
```

```
    ROLLBACK;
```

```
END;
```

Limbajul PL/SQL- Interactiunea cu Oracle

- SAVEPOINT

Numeste si marcheaza un punct curent in prelucrarea unei tranzactii.

```
DECLARE
```

```
emp_id emp.empno%TYPE:
```

```
BEGIN
```

```
    UPDATE emp SET ... WHERE empno=emp_id;
```

```
    DELETE FROM emp WHERE ...
```

```
    ...
```

```
SAVEPOINT insert;
```

```
    INSERT INTO emp VALUES (emp_id,...);
```

```
EXCEPTION
```

```
    WHEN DUP_VAL_ON_INDEX THEN
```

```
        ROLLBACK TO insert;
```

```
END;
```

Limbajul PL/SQL- Interactiunea cu Oracle

- SET TRANSACTION

DECLARE

vinzari_zi REAL;

vinzari_saptamina REAL;

vinzari_luna REAL;

BEGIN

COMMIT;

SET TRANSACTION READ ONLY;

SELECT SUM(cant) INTO vinzari_zi FROM sales
WHERE data=SYSDATE;

SELECT SUM(cant) INTO vinzari_saptamina FROM sales
WHERE data >SYSDATE-7;

SELECT SUM(cant) INTO vinzari_luna FROM sales
WHERE data>SYSDATE-30;

COMMIT; END;

Limbajul PL/SQL- Interactiunea cu Oracle

- Restrictii
 - Numai SELECT INTO, OPEN, FETCH, CLOSE, LOCK TABLE, COMMIT si ROLLBACK sunt permise in tranzactii read-only.
 - Interogările nu pot contine FOR UPDATE
- Blocari
 - Oracle blocheaza linia ce apare pentru UPDATE sau DELETE.
 - Folosim FOR UPDATE sa blocam toate liniile inainte de comanda UPDATE sau DELETE.
 - Blocarea intregii tabele cu LOCK TABLE

DECLARE

```
CURSOR C1 IS SELECT empno, sal FROM emp
```

```
WHERE job='saleman' AND comm >sal FOR UPDATE NOWAIT;
```

BEGIN

```
OPEN C1; -- toate liniile lui C1 sunt blocate, deblocarea:COMMIT
```

```
-- sau ROLLBACK
```

Limbajul PL/SQL- Interactiunea cu Oracle

In interogari cu mai multe tabele putem bloca numai linii din anumite tabele.

```
DECLARE
```

```
CURSOR C1 IS SELECT ename, dname FROM emp, dept  
WHERE emp.deptno=dept.deptno AND job='MANAGER'  
FOR UPDATE OF sal;
```

In instructiunea UPDATE sau DELETE folosim CURRENT OF:

```
DECLARE
```

```
CURSOR C1 IS SELECT empno, job, sal FROM emp FOR UPDATE;  
BEGIN  
OPEN C1;  
LOOP  
FETCH c1 INTO ...  
UPDATE emp SET sal =new_sal WHERE CURRENT OF C1;  
END LOOP;
```

Limbajul PL/SQL- Interactiunea cu Oracle

- FETCH si COMMIT. FETCH nu poate urma lui COMMIT.

```
DECLARE
```

```
CURSOR C1 IS SELECT ename FROM emp FOR UPDATE OF sal;
```

```
ctr NUMBER:=0;
```

```
BEGIN
```

```
FOR emp_rec IN C1 LOOP
```

```
...
```

```
ctr:=ctr+1;
```

```
INSERT INTO temp VALUES (ctr,'info');
```

```
IF ctr>=10 THEN
```

```
    COMMIT;
```

```
END IF;
```

```
END LOOP;
```

```
END;
```

Limbajul PL/SQL- Interactiunea cu Oracle

In loc de FOR UPDATE si CURRENT OF- ROWID

DECLARE

CURSOR C1 IS SELECT ename, job, rowid FROM emp;

ename1 emp.ename%TYPE;

job1 emp.ename%TYPE;

rowid1 ROWID;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO ename1, job1, rowid1;

EXIT WHEN C1%NOTFOUND;

UPDATE emp SET sal=sal *1.1 WHERE ROWID= rowid1;

COMMIT;

END LOOP;

CLOSE C1; END;

Limbajul PL/SQL- Interactiunea cu Oracle

- Utilizarea atributului %ROWTYPE cu cursori ce refera ROWID

```
DECLARE
```

```
    CURSOR C1 IS SELECT ename, sal, rowid FROM emp;
```

```
    emp_rec C1%ROWTYPE;
```

```
BEGIN
```

```
    OPEN C1;
```

```
    LOOP
```

```
        FETCH C1 INTO emp_rec;
```

```
        EXIT WHEN C1%NOTFOUND;
```

```
        ...
```

```
        IF ... THEN
```

```
            DELETE FROM emp WHERE ROWID=emp_rec.ROWID;
```

```
        END IF;
```

```
    END LOOP;
```

```
    CLOSE C1; END;
```