

Tehnici de programare cu baze de date

#6

PL/SQL

Cursori în PL/SQL (partea II-a)

Adrian Runceanu
www.runceanu.ro/adrian

Curs 6

Cursori în PL/SQL (continuare)

Cursori în PL/SQL

- 1. LOOP-ul FOR pentru cursor**
- 2. Cursori cu parametri**
- 3. Folosirea cursorilor pentru actualizari**
- 4. Folosirea cursorilor multipli**

LOOP-ul FOR pentru cursor

- S-a studiat utilizarea cursorilor expliciti cu folosirea instructiunilor DECLARE, OPEN si FETCH.
- Putem face acelasi lucru mai usor, folosind o singura instructiune cu ajutorul **LOOP-ului FOR pentru cursor**.
- *Un loop FOR pentru cursor prelucreaza randuri intr-un cursor explicit.*
- Este o varianta mai rapida deoarece cursorul este deschis, este preluat cate un rand pentru fiecare iteratie din loop, se iese din loop dupa ce ultimul rand a fost procesat si cursorul se inchide automat.
- Si loop-ul se incheie automat la sfarsitul iteratiei, dupa prelucrarea ultimului rand.

Sintaxa

```
FOR record_name IN cursor_name  
LOOP
```

```
    Instructiune1;
```

```
    Instructiune2;
```

```
    ...
```

```
END LOOP;
```

- **record name** – numele unei inregistrari declarate implicit (**cursor_name%ROWTYPE**)
- **cursor_name** – identificator PL/SQL pentru un cursor declarat anterior

Exemple:

1)

```
◦ DECLARE
  CURSOR emp_cursor IS
  SELECT empno, ename
  FROM emp
  WHERE deptno = 30;
BEGIN
```

```
  FOR v_emp_record IN emp_cursor
  LOOP
```

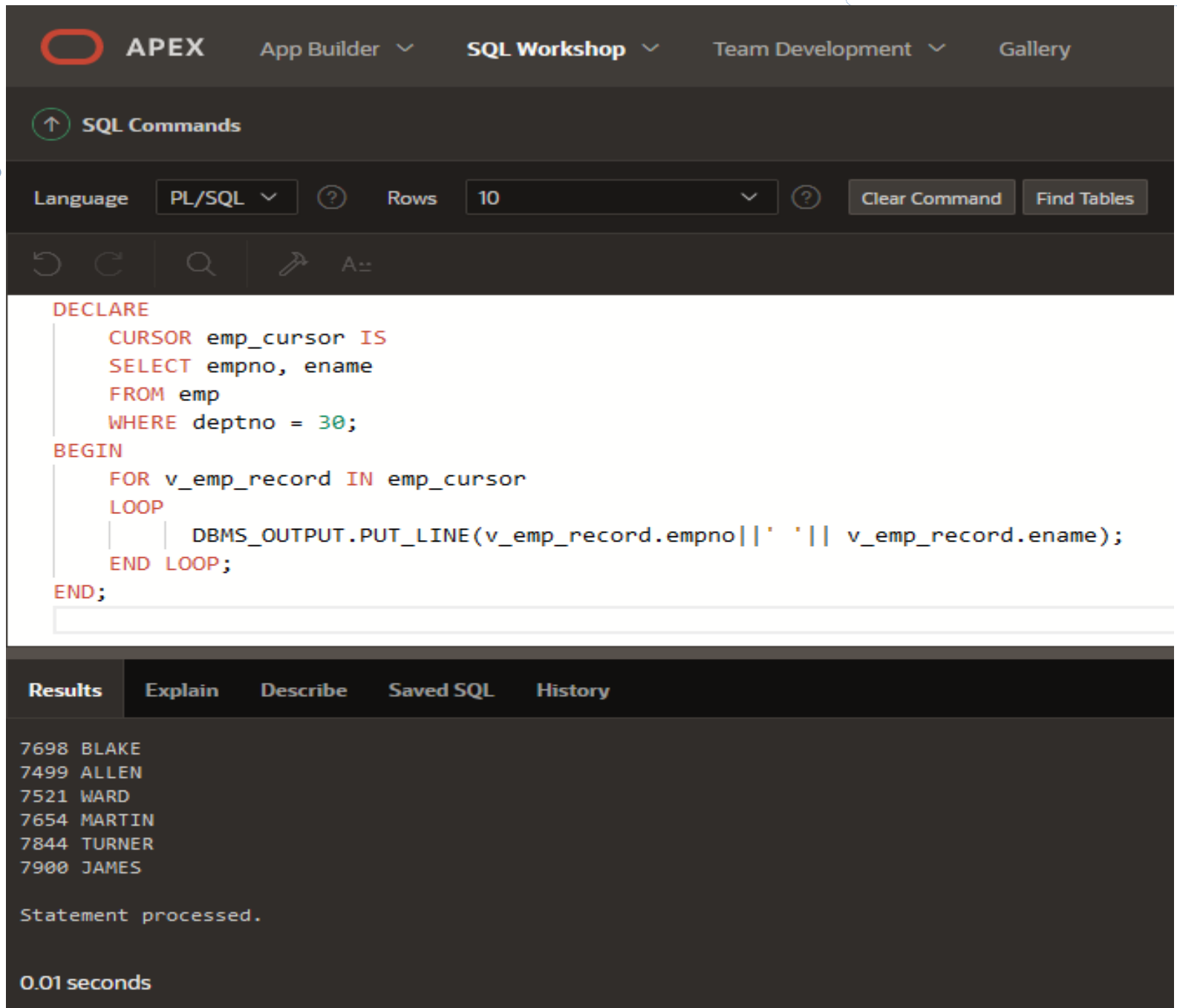
```
    DBMS_OUTPUT.PUT_LINE(v_emp_record.empno||' '||
  v_emp_record.ename);
  END LOOP;
```

```
END;
```

v_emp_record este o inregistrare declarata implicit.

- Putem accesa datele preluate cu aceasta inregistrare implicita asa cum a fost exemplificat in exemplul anterior.
- Nu sunt declarate variabile pentru a pastra informatia.
- De asemenea, codul nu contine instructiunile OPEN si CLOSE.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	4	0	1	-	-	-
	ENAME	VARCHAR2	10	-	-	-	✓	-	-
	JOB	VARCHAR2	9	-	-	-	✓	-	-
	MGR	NUMBER	-	4	0	-	✓	-	-
	HIREDATE	DATE	7	-	-	-	✓	-	-
	SAL	NUMBER	-	7	2	-	✓	-	-
	COMM	NUMBER	-	7	2	-	✓	-	-
	DEPTNO	NUMBER	-	2	0	-	✓	-	-



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing 'Language' set to 'PL/SQL' and 'Rows' set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. The main editor area contains the following PL/SQL code:

```
DECLARE
  CURSOR emp_cursor IS
  SELECT empno, ename
  FROM emp
  WHERE deptno = 30;
BEGIN
  FOR v_emp_record IN emp_cursor
  LOOP
    DBMS_OUTPUT.PUT_LINE(v_emp_record.empno || ' ' || v_emp_record.ename);
  END LOOP;
END;
```

Below the editor, the 'Results' tab is selected, showing the output of the query:

empno	ename
7698	BLAKE
7499	ALLEN
7521	WARD
7654	MARTIN
7844	TURNER
7900	JAMES

Below the results, the status 'Statement processed.' is shown, followed by the execution time '0.01 seconds'.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPTNO	NUMBER	-	2	0	1	-	-	-
	DNAME	VARCHAR2	40	-	-	-	✓	-	-
	LOC	VARCHAR2	13	-	-	-	✓	-	-

2)

DECLARE

CURSOR dept_cursor IS

SELECT deptno, dname

FROM dept

ORDER BY deptno;

BEGIN

FOR v_dept_record IN dept_cursor

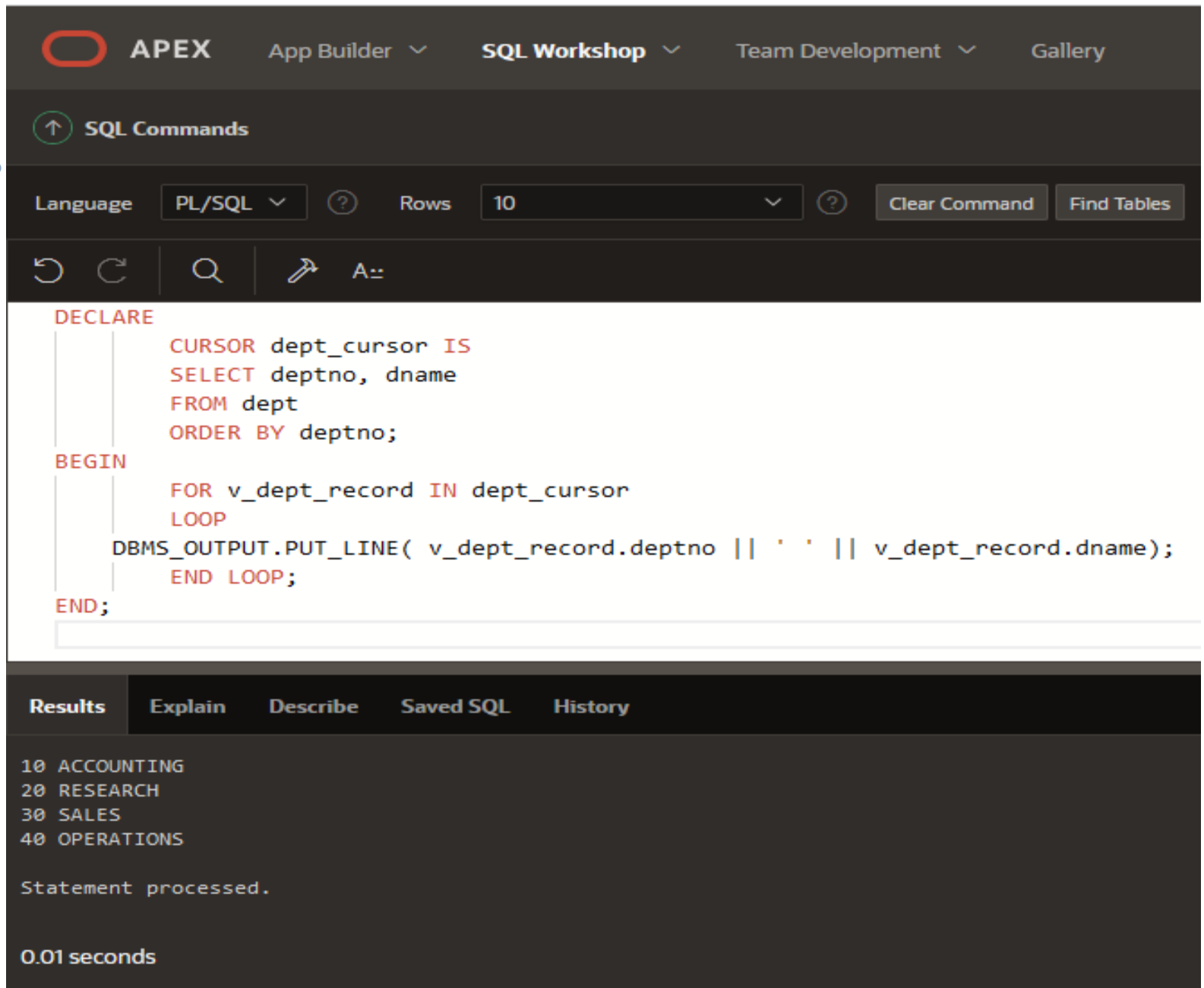
LOOP

DBMS_OUTPUT.PUT_LINE(v_dept_record.deptno

|| ' ' || v_dept_record.dname);

END LOOP;

END;



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing 'Language' set to 'PL/SQL' and 'Rows' set to '10'. There are buttons for 'Clear Command' and 'Find Tables', along with icons for undo, redo, search, and refresh.

```
DECLARE
  CURSOR dept_cursor IS
  SELECT deptno, dname
  FROM dept
  ORDER BY deptno;
BEGIN
  FOR v_dept_record IN dept_cursor
  LOOP
    DBMS_OUTPUT.PUT_LINE( v_dept_record.deptno || ' ' || v_dept_record.dname);
  END LOOP;
END;
```

The 'Results' tab is selected, showing the output of the query:

```
10 ACCOUNTING
20 RESEARCH
30 SALES
40 OPERATIONS
```

Statement processed.

0.01 seconds

Reguli de utilizare a loop-ului FOR pentru cursor

1. Nu se declara inregistrarea care controleaza loop-ul deoarece este declarata implicit
2. Domeniul de vizibilitate al inregistrarii implicite este restrictionat in interiorul loop-ului, deci nu putem referi inregistrarea in afara loop-ului
3. Putem accesa datele preluate prin:
record_name.column_name

Testarea atributelor cursorului

- Se pot testa in continuare attributele de cursor, cum ar fi ***%ROWCOUNT***.
- Urmatorul exemplu iese din loop dupa ce au fost preluate si prelucrate cinci randuri. Cursorul se inchide automat.

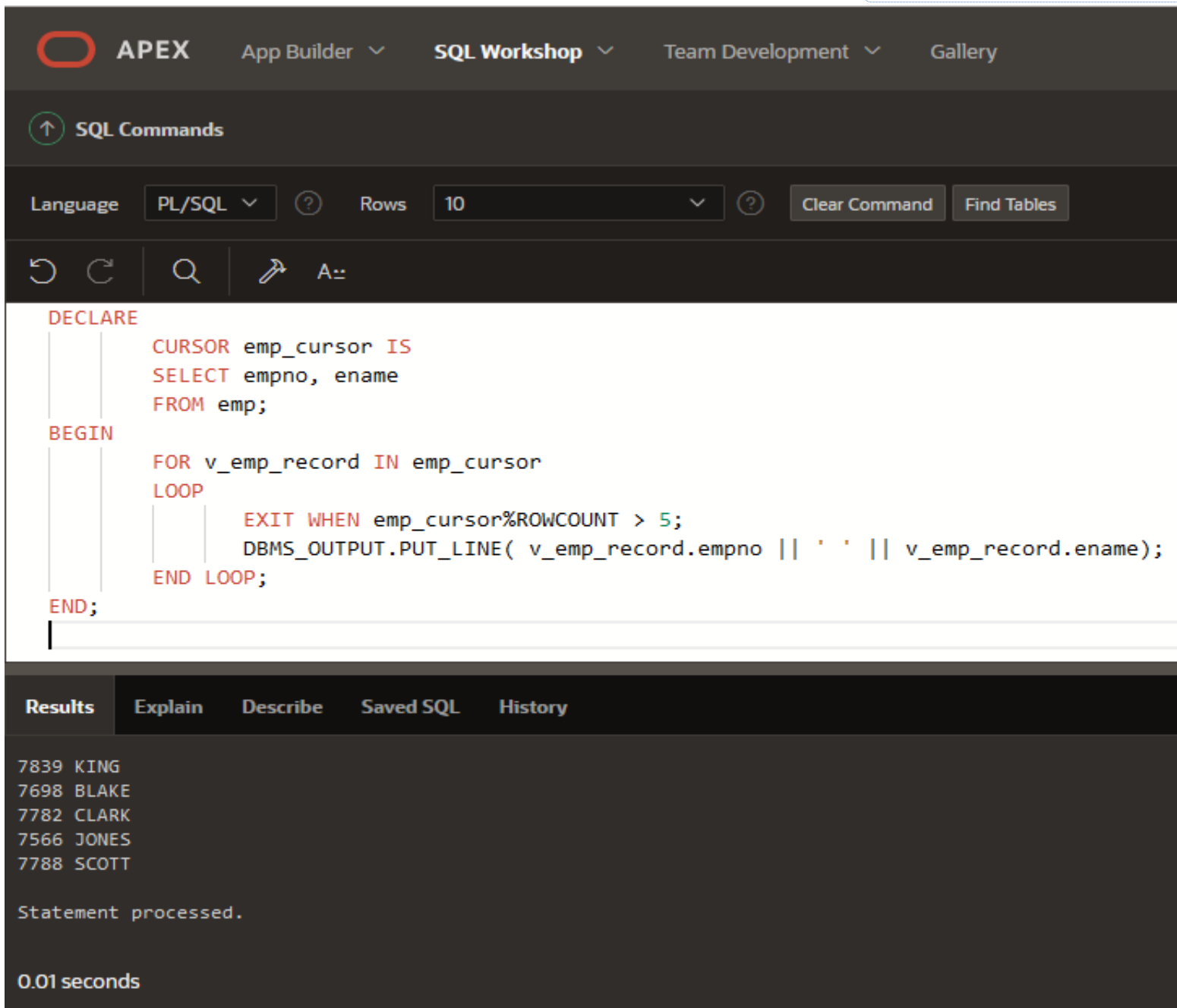
DECLARE

```
CURSOR emp_cursor IS  
SELECT empno, ename  
FROM emp;
```

BEGIN

```
FOR v_emp_record IN emp_cursor  
LOOP  
    EXIT WHEN emp_cursor%ROWCOUNT > 5;  
    DBMS_OUTPUT.PUT_LINE(  
v_emp_record.empno || ' ' || v_emp_record.ename);  
    END LOOP;
```

END;



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is visible, with 'Language' set to 'PL/SQL' and 'Rows' set to '10'. The main area contains a PL/SQL script that declares a cursor, selects employee data, and loops through the first five rows to output their employee IDs and names. The 'Results' tab is active, showing the output of the query: five rows of employee data (KING, BLAKE, CLARK, JONES, SCOTT) and a confirmation message 'Statement processed.' with an execution time of '0.01 seconds'.

```
DECLARE
  CURSOR emp_cursor IS
  SELECT empno, ename
  FROM emp;
BEGIN
  FOR v_emp_record IN emp_cursor
  LOOP
    EXIT WHEN emp_cursor%ROWCOUNT > 5;
    DBMS_OUTPUT.PUT_LINE( v_emp_record.empno || ' ' || v_emp_record.ename);
  END LOOP;
END;
```

Results Explain Describe Saved SQL History

```
7839 KING
7698 BLAKE
7782 CLARK
7566 JONES
7788 SCOTT
```

Statement processed.

0.01 seconds

Folosirea subinterogărilor în loop-ul FOR pentru cursor

- **Putem să nu declaram cursorul deloc!**
- În schimb, putem specifica direct în loop-ul FOR instrucțiunea SELECT care stă la baza cursorului.
- Avantajul constă în faptul că toată definiția cursorului este cuprinsă într-o singură instrucțiune FOR.
- Astfel codul poate fi ulterior modificat mai ușor și mai rapid.

Exemplu

```
BEGIN
```

```
FOR v_emp_record IN (SELECT empno, ename  
FROM emp WHERE deptno = 30)
```

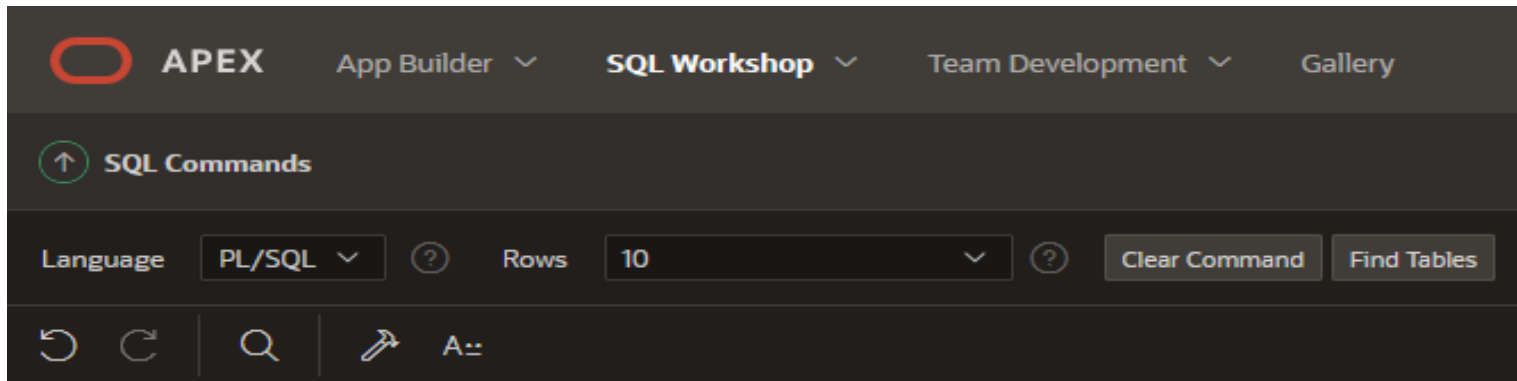
```
LOOP
```

```
  DBMS_OUTPUT.PUT_LINE(v_emp_record.empno  
  || ' ' || v_emp_record.ename);
```

```
END LOOP;
```

```
END;
```

- Clauza **SELECT** in instructiunea **FOR** este practic o subinterogare, deci trebuie inclusa intre paranteze.



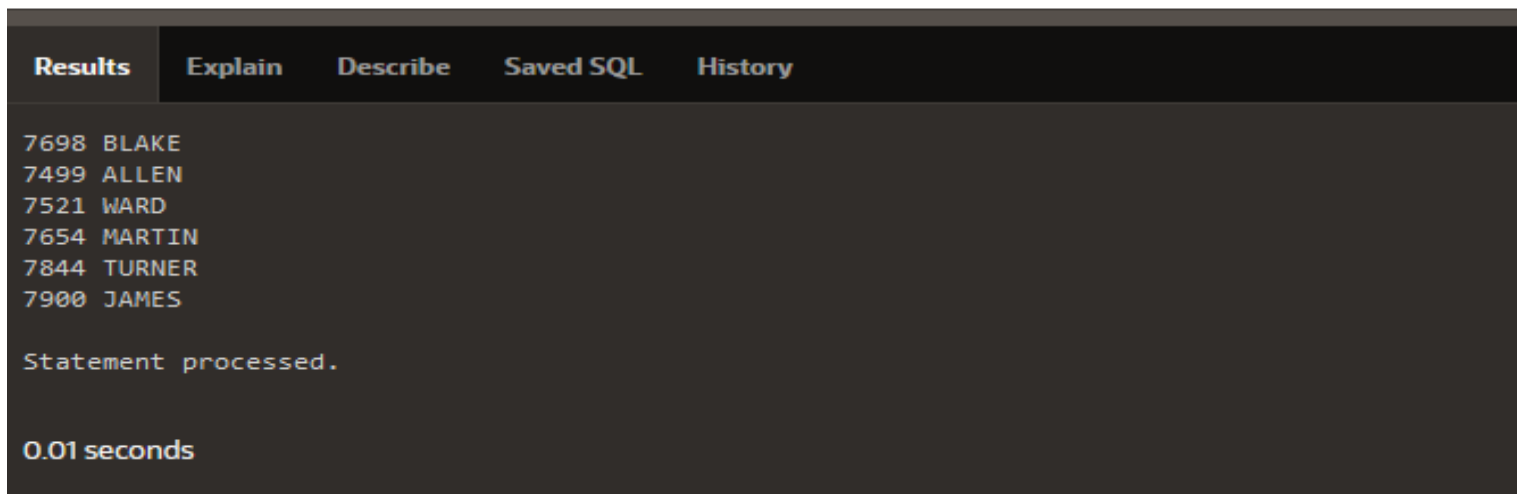
APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

⏪ ⏩ 🔍 A::

```
BEGIN
  FOR v_emp_record IN (SELECT empno, ename FROM emp WHERE deptno = 30)
  LOOP
    DBMS_OUTPUT.PUT_LINE(v_emp_record.empno || ' ' || v_emp_record.ename);
  END LOOP;
END;
```



Results	Explain	Describe	Saved SQL	History
7698	BLAKE			
7499	ALLEN			
7521	WARD			
7654	MARTIN			
7844	TURNER			
7900	JAMES			

Statement processed.

0.01 seconds

- Comparati urmatoarele doua exemple. Logic sunt identice.
- Diferenta este la modalitatea de scriere.

1)

```
BEGIN
```

```
FOR v_dept_rec IN (SELECT * FROM dept)
```

```
LOOP
```

```
DBMS_OUTPUT.PUT_LINE(v_dept_rec.dname);
```

```
END LOOP;
```

```
END;
```


2)

DECLARE**CURSOR dept_cursor IS****SELECT * FROM dept;****v_dept_rec dept_cursor%ROWTYPE;****BEGIN****OPEN dept_cursor;****LOOP****FETCH dept_cursor INTO v_dept_rec;****EXIT WHEN dept_cursor%NOTFOUND;****DBMS_OUTPUT.PUT_LINE(v_dept_rec.dname);****END LOOP;****CLOSE dept_cursor;****END;**

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

SQL Editor toolbar with icons for undo, redo, search, and edit.

```

BEGIN
  FOR v_emp_record IN (SELECT empno, ename FROM emp WHERE deptno = 30)
  LOOP
    DBMS_OUTPUT.PUT_LINE(v_emp_record.empno || ' ' || v_emp_record.ename);
  END LOOP;
END;

```

Results Explain Describe Saved SQL History

```

7698 BLAKE
7499 ALLEN
7521 WARD
7654 MARTIN
7844 TURNER
7900 JAMES

```

Statement processed.

0.01 seconds

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

SQL Editor toolbar with icons for undo, redo, search, and edit.

```

DECLARE
  CURSOR dept_cursor IS
  SELECT * FROM dept;
  v_dept_rec dept_cursor%ROWTYPE;
BEGIN
  OPEN dept_cursor;
  LOOP
    FETCH dept_cursor INTO v_dept_rec;
    EXIT WHEN dept_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_dept_rec.dname);
  END LOOP;
  CLOSE dept_cursor;
END;

```

Results Explain Describe Saved SQL History

```

ACCOUNTING
RESEARCH
SALES
OPERATIONS

```

Statement processed.

0.04 seconds

Cursori în PL/SQL

1. LOOP-ul FOR pentru cursor
2. **Cursori cu parametri**
3. Folosirea cursorilor pentru actualizari
4. Folosirea cursorilor multipli

2. Cursorsi cu parametri

- *Un parametru este o variabila al carei nume este folosit in declararea cursorului.*
- Cand se deschide cursorul, valoarea parametrului este transmisa serverului **Oracle** care o foloseste pentru a decide ce randuri sa extraga in multimea activa a cursorului.
- Aceasta inseamna ca putem inchide si deschide un cursor explicit de cateva ori intr-un bloc sau in diferite executii ale aceluiasi bloc, returnand de fiecare data alta multime activa.
- Consideram un exemplu in care transmitem cursorului orice valoare pentru **region_id** si acesta returneaza numele tarilor din acea regiune.

DECLARE

```
CURSOR c_country (p_region_id NUMBER) IS  
SELECT name, CAPITAL  
FROM eba_countries  
WHERE region_id = p_region_id;  
v_country_record c_country%ROWTYPE;
```

BEGIN

```
OPEN c_country(50);  
LOOP  
    FETCH c_country INTO v_country_record;  
    EXIT WHEN c_country%NOTFOUND;  
    DBMS_OUTPUT.PUT_LINE(v_country_record.name ||  
' ' || v_country_record.CAPITAL);  
END LOOP;  
CLOSE c_country;
```

END;

APEX App Builder SQL Workshop Team Development Gallery

↑ SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

↶ ↷ 🔍 ↵ A::

```

DECLARE
    CURSOR c_country (p_region_id NUMBER) IS
    SELECT name, CAPITAL
    FROM eba_countries
    WHERE region_id = p_region_id;
    v_country_record c_country%ROWTYPE;
BEGIN
    OPEN c_country(50);
    LOOP
        FETCH c_country INTO v_country_record;
        EXIT WHEN c_country%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_country_record.name || ' ' || v_country_record.CAPITAL);
    END LOOP;
    CLOSE c_country;
END;

```

Results Explain Describe Saved SQL History

Burkina Faso Ouagadougou
 Burundi Bujumbura
 Cabo Verde Praia
 Cameroon Yaoundé
 Central African Republic Bangui
 Chad N'Djamena
 Comoros Moroni
 Congo Brazzaville
 Congo, Democratic Republic of the Kinshasa
 Côte d'Ivoire Yamoussoukro
 Djibouti Djibouti

Definirea cursorilor cu parametri

- Fiecare parametru din declararea cursorului trebuie sa aiba o valoare corespunzatoare in instructiunea **OPEN**.
- Tipurile de date ale parametrilor sunt aceleasi ca cele pentru variabilele scalare, dar nu le precizam dimensiunile.
- Denumirile parametrilor sunt folosite in clauza **WHERE** ale instructiunii **SELECT** corespunzatoare cursorului.

Sintaxa

```
CURSOR cursor_name  
[( parameter_name datatype, ...)]  
IS  
select_statement;
```

In sintaxa:

- **cursor_name** este un identificator PL/SQL pentru cursorul declarat
- **parameter_name** este numele parametrului
- **select_statement** este o instructiune SELECT fara clauza INTO

Deschiderea cursorilor cu parametri

Sintaxa

```
OPEN cursor_name(parameter_value,...);
```

- Atunci cand se deschide un cursor transmitem valori parametrilor.
- De aceea, putem deschide un singur cursor explicit de mai multe ori si putem prelua mai multe multimi active diferite.

Exemplu 1 – cursor deschis de mai multe ori

```
DECLARE
    CURSOR c_country (p_region_id NUMBER)
IS
    SELECT name, CAPITAL
    FROM eba_countries
    WHERE region_id = p_region_id;
    v_country_record c_country%ROWTYPE;
BEGIN
    OPEN c_country(50);
    ...
    CLOSE c_country;
    OPEN c_country (30);
    ...
```

Exemplu 2

DECLARE

```

v_deptno emp.deptno%TYPE;
CURSOR empcursor (p_deptno NUMBER) IS
SELECT empno, sal
FROM emp
WHERE deptno = p_deptno;
v_emp_rec empcursor%ROWTYPE;

```

BEGIN

```

SELECT MAX(deptno) INTO v_deptno
FROM emp;
OPEN empcursor(v_deptno);

```

LOOP

```

    FETCH empcursor INTO v_emp_rec;
    EXIT WHEN empcursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_rec.empno || ' ' ||
v_emp_rec.sal);
    END LOOP;
    CLOSE empcursor;

```

END;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	4	0	1	-	-	-
	ENAME	VARCHAR2	10	-	-	-	✓	-	-
	JOB	VARCHAR2	9	-	-	-	✓	-	-
	MGR	NUMBER	-	4	0	-	✓	-	-
	HIREDATE	DATE	7	-	-	-	✓	-	-
	SAL	NUMBER	-	7	2	-	✓	-	-
	COMM	NUMBER	-	7	2	-	✓	-	-
	DEPTNO	NUMBER	-	2	0	-	✓	-	-

APEX App Builder **SQL Workshop** Team Development Gallery

SQL Commands

Language **PL/SQL** Rows **10** **Clear Command** **Find Tables**

⏪ ⏩ 🔍 ↗ A=

```

DECLARE
    v_deptno emp.deptno%TYPE;
    CURSOR empcursor (p_deptno NUMBER) IS
    SELECT empno, sal
    FROM emp
    WHERE deptno = p_deptno;
    v_emp_rec empcursor%ROWTYPE;
BEGIN
    SELECT MAX(deptno) INTO v_deptno
    FROM emp;
    OPEN empcursor(v_deptno);
    LOOP
        FETCH empcursor INTO v_emp_rec;
        EXIT WHEN empcursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_emp_rec.empno || ' ' || v_emp_rec.sal);
    END LOOP;
    CLOSE empcursor;
END;

```

Results

Explain

Describe

Saved SQL

History

```

7698 2850
7499 1600
7521 1250
7654 1250
7844 1500
7900 950

```

Statement processed.

Cursori cu parametri multipli

Exemplu 1- cursor cu doi parametri

DECLARE

**CURSOR countrycursor2 (p_region_id NUMBER,
p_population NUMBER) IS**

SELECT country_id, name, population

FROM eba_countries

WHERE region_id = p_region_id OR population >

p_population;

BEGIN

FOR v_country_record IN countrycursor2(145,1000000)

LOOP

**DBMS_OUTPUT.PUT_LINE(v_country_record.country_id
|| ' ' || v_country_record.name || ' '
v_country_record.population);**

END LOOP;

END;

Cursori cu parametri multipli

Exemplu 1- cursor cu doi parametri

Results	Explain	Describe	Saved SQL	History
1	Afghanistan	38041754		
3	Algeria	43053054		
6	Angola	31825295		
10	Argentina	44780677		
13	Australia	25203198		
15	Azerbaijan	10047718		
18	Bangladesh	163046161		
21	Belgium	11539328		
23	Benin	11801151		
26	Bolivia (Plurinational State of)	11513100		
31	Brazil	211049527		
35	Burkina Faso	20321378		
36	Burundi	11530580		
38	Cambodia	16486542		
39	Cameroon	25876380		

ucb.adrian.runceanu@gmail.com oracle_studenti_ain en

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

↶ ↷ 🔍 📌 A::

```

DECLARE
CURSOR countrycursor2 (p_region_id NUMBER, p_population NUMBER) IS
    SELECT country_id, name, population
    FROM eba_countries
    WHERE region_id = p_region_id OR population > p_population;
BEGIN
    FOR v_country_record IN countrycursor2(145,10000000)
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_country_record.country_id || ' ' || v_country_record.name || ' ' || v_country_record.population);
    END LOOP;
END;
```

Exemplu 2 – codul preia toti programatorii IT care castiga mai mult de 1500\$

DECLARE

**CURSOR emp_cursor3 (p_job VARCHAR2, p_sal
NUMBER) IS**

SELECT empno, ename

FROM emp

WHERE job = p_job AND sal > p_sal;

BEGIN

**FOR v_emp_record IN emp_cursor3('ANALYST',
1500)**

LOOP

**DBMS_OUTPUT.PUT_LINE(v_emp_record.empno
|| ' ' || v_emp_record.ename);**

END LOOP;

END;

APEX App Builder **SQL Workshop** Team Development Gallery

↑ SQL Commands

Language **SQL** Rows **10** Clear Command Find Tables

↶ ↷ 🔍 📌 A::

```
DECLARE
CURSOR emp_cursor3 (p_job VARCHAR2, p_sal NUMBER) IS
    SELECT empno, ename
    FROM emp
    WHERE job = p_job AND sal > p_sal;
BEGIN
    FOR v_emp_record IN emp_cursor3('ANALYST', 1500)
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_emp_record.empno || ' ' || v_emp_record.ename);
    END LOOP;
END;
```

Results Explain Describe Saved SQL History

```
7788 SCOTT
7902 SMITH

Statement processed.

0.01 seconds
```


Cursori în PL/SQL

1. LOOP-ul FOR pentru cursor
2. Cursori cu parametri
3. Folosirea cursorilor pentru actualizari
4. Folosirea cursorilor multipli

3. Folosirea cursorilor pentru actualizari

- Atunci cand sunt mai multi utilizatori conectati in acelasi timp la baza de date, exista posibilitatea ca un alt utilizator sa actualizeze randurile dintr-o anumita tabela dupa ce v-ati deschis cursorul si ati preluat randurile.
- Putem bloca randurile la deschiderea cursorului pentru a preveni modificari facute asupra lor de catre alti utilizatori.
- Este important sa facem acest lucru daca vrem sa modificam aceleasi randuri noi insine

Declararea unui cursor folosind clauza **FOR UPDATE**

- Atunci cand declaram un cursor **FOR UPDATE**, fiecare rand este blocat cum deschidem cursorul.
- Acest lucru previne modificarea randurilor de catre alti utilizatori cat timp cursorul este deschis.
- De asemenea, ni se permite noua sa modificam randurile folosind o clauza **...WHERE CURRENT OF...**

Sintaxa

```
CURSOR cursor_name IS  
SELECT ... FROM ...  
FOR UPDATE [OF column_reference  
n][NOWAIT | WAIT ];
```

Acest lucru nu impiedica vizualizarea randurilor de catre alti utilizatori.

- **column_reference** – este o coloana din tabela ale carei randuri este necesar sa le blocam

Daca randurile au fos deja blocate de alta sesiune:

- **NOWAIT** furnizeaza o eroare imediata serverului ORACLE
- **WAIT n** asteapta **n** secunde si returneaza o eroare daca o alta sesiune inca blocheaza randurile dupa cele **n** secunde

*Cuvantul cheie **NOWAIT** in clauza **FOR UPDATE***

- Cuvantul cheie optional **NOWAIT** spune serverului ORACLE sa nu astepte daca oricare dintre randurile solicitate sunt deja blocate de catre alt utilizator.
- Controlul este imediat dat programului nostru deci putem face altceva inainte de a incerca din nou sa realizam blocarea.
- Daca omitem cuvantul cheie **NOWAIT** atunci serverul ORACLE asteapta nedefinit pana cand randurile sunt disponibile.

Exemplu

```
DECLARE  
CURSOR emp_cursor IS  
SELECT empno, ename  
FROM emp  
WHERE deptno = 80 FOR UPDATE  
NOWAIT;  
...
```

- Daca randurile sunt deja blocate de alta sesiune si am specificat **NOWAIT**, atunci la deschiderea cursorului va rezulta eroare.
- Putem incerca sa deschidem cursorul mai tarziu.
- Se poate folosi **WAIT n** in loc de **NOWAIT** si sa specificam numarul de secunde de asteptare

Clauza FOR UPDATE OF

- Dacă cursorul are la baza un *join dintre doua tabele* poate dorim sa blocam randurile dintr-o tabela, dar nu si din cealalta.
- Dacă dorim acest lucru, specificam *orice coloana a tablei pe care vrem sa o blocam.*

DECLARE

```
CURSOR emp_cursor IS  
SELECT e.empno, d.dname  
FROM emp e, dept d  
WHERE e.deptno = d. deptno AND  
deptno = 80  
FOR UPDATE OF salary;
```

...

Clauza WHERE CURRENT OF

- Clauza **WHERE CURRENT OF** este folosita impreuna cu clauza **FOR UPDATE** pentru a referi randul curent (randul preluat cel mai recent) intr-un cursor explicit.
- Clauza **WHERE CURRENT OF** este folosita in instructiunile **UPDATE** sau **DELETE**, in timp ce clauza **FOR UPDATE** este specificata in declararea cursorului.

Sintaxa

WHERE CURRENT OF cursor-name;

- **cursor_name** – este numele unui cursor declarat (cursorul trebuie sa fi fost declarat cu clauza **FOR UPDATE**)
- Putem folosi **WHERE CURRENT OF** pentru actualizarea sau stergerea randului curent din tabela.
- Aceasta ne permite sa aplicam actualizari si stergeri ale randului curent fara a fi necesara folosirea clauzei **WHERE**.
- Putem include clauza **FOR UPDATE** in interogarea cursorului astfel incat randurile sa fie blocate la deschidere (**OPEN**).

3. Folosirea cursorilor pentru actualizari

- Cursorii se pot folosi pentru a actualiza si a sterge randul curent.
- Se include clauza **FOR UPDATE** in interogarea cursorului pentru a bloca randul mai intai
- Se foloseste clauza **WHERE CURRENT OF** pentru a referi randul curent dintr-un cursor explicit.

Exemple

1)

UPDATE EMP

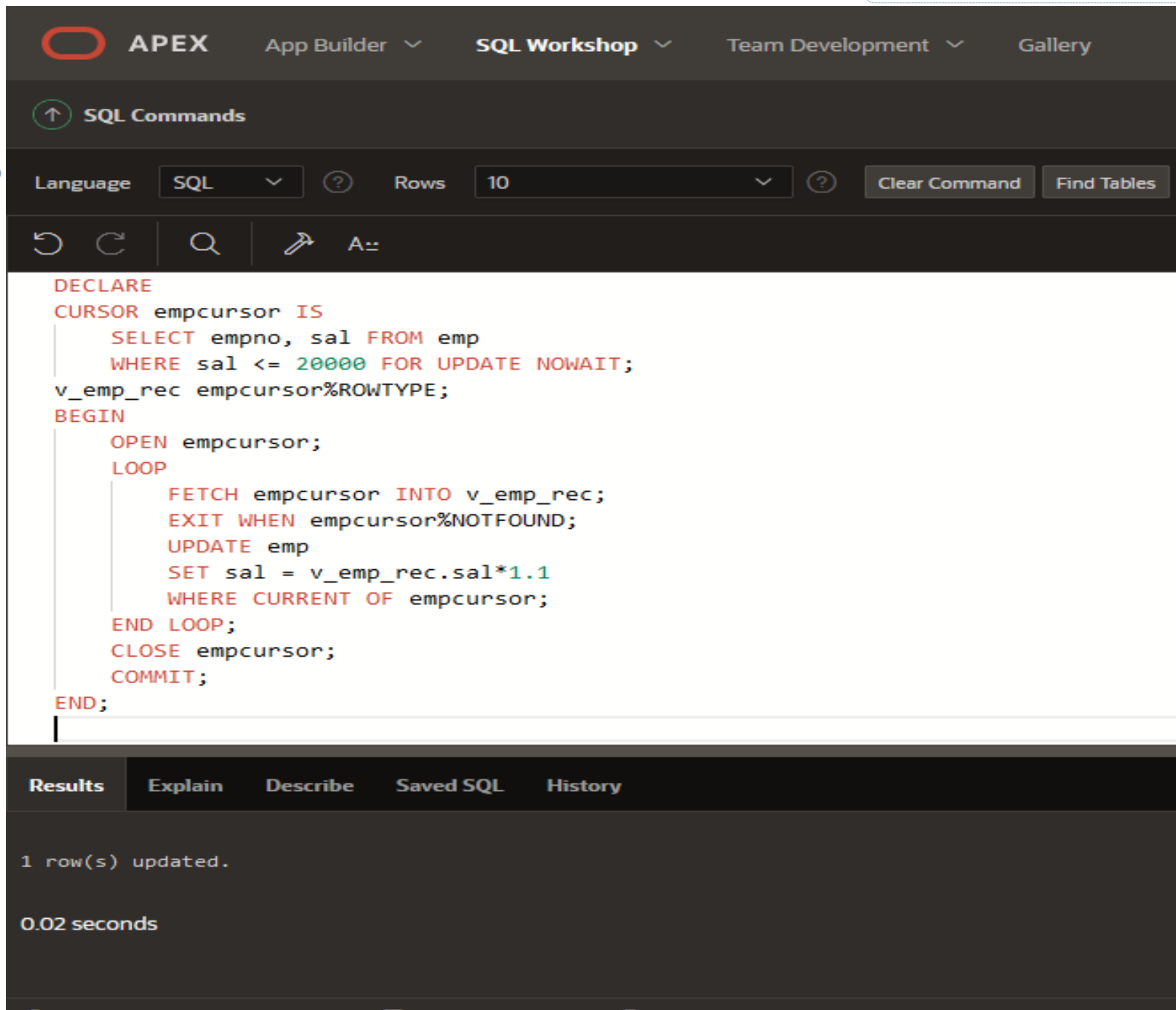
SET salary = ...

WHERE CURRENT OF emp_cursor;

Exemple

- 2) In acest exemplu nu avem nevoie de o coloana referinta in clauza FOR UPDATE deoarece cursorul nu se bazeaza pe un join.

```
DECLARE  
CURSOR empcursor IS  
    SELECT empno, sal FROM emp  
    WHERE sal <= 20000 FOR UPDATE NOWAIT;  
v_emp_rec empcursor%ROWTYPE;  
BEGIN  
    OPEN empcursor;  
    LOOP  
        FETCH empcursor INTO v_emp_rec;  
        EXIT WHEN empcursor%NOTFOUND;  
        UPDATE emp  
        SET sal = v_emp_rec.sal*1.1  
        WHERE CURRENT OF empcursor;  
    END LOOP;  
    CLOSE empcursor;  
    COMMIT;  
END;
```



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing a 'Language' dropdown set to 'SQL' and 'Rows' set to '10'. There are buttons for 'Clear Command' and 'Find Tables', along with standard SQL editor icons like undo, redo, search, and save.

```
DECLARE
CURSOR empcursor IS
    SELECT empno, sal FROM emp
    WHERE sal <= 20000 FOR UPDATE NOWAIT;
v_emp_rec empcursor%ROWTYPE;
BEGIN
    OPEN empcursor;
    LOOP
        FETCH empcursor INTO v_emp_rec;
        EXIT WHEN empcursor%NOTFOUND;
        UPDATE emp
        SET sal = v_emp_rec.sal*1.1
        WHERE CURRENT OF empcursor;
    END LOOP;
    CLOSE empcursor;
    COMMIT;
END;
```

Below the code editor, the 'Results' tab is selected, showing the execution output: '1 row(s) updated.' and '0.02 seconds'.

Exemple

3)

FOR UPDATE OF sal blocheaza numai randurile din **emp** nu si din **dept**. Si sa nu uitam ca noi actualizam tabela, nu cursorul.

DECLARE

CURSOR ed_cursor IS

SELECT empno, sal, dname

FROM emp e, dept d

WHERE e.deptno = d.deptno

FOR UPDATE OF sal NOWAIT;

BEGIN

FOR v_ed_rec IN ed_cursor LOOP

UPDATE emp

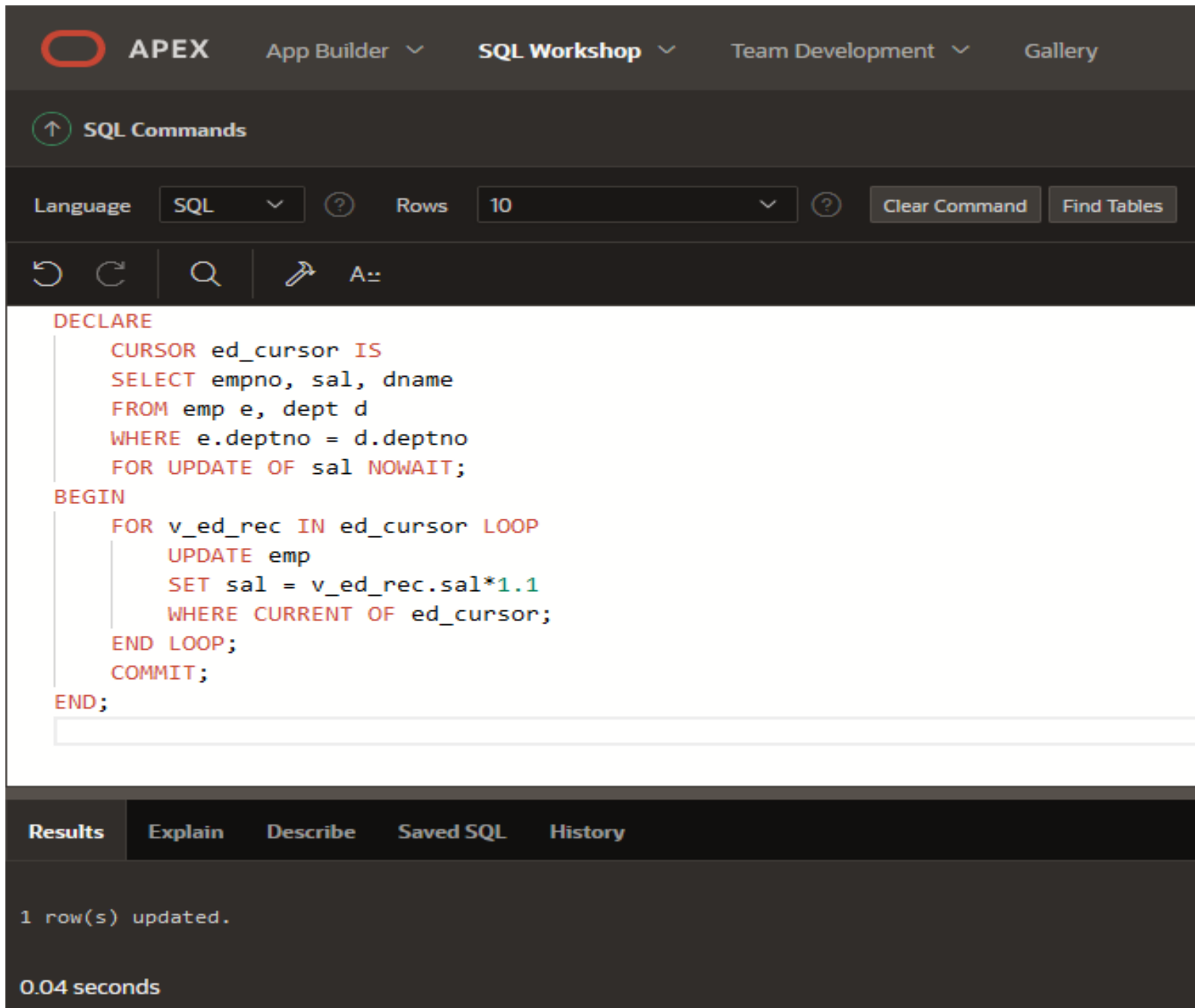
SET sal = v_ed_rec.sal*1.1

WHERE CURRENT OF ed_cursor;

END LOOP;

COMMIT;

END;



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing a dropdown menu set to 'SQL' and a 'Rows' limit of '10'. There are buttons for 'Clear Command' and 'Find Tables'. A toolbar with icons for undo, redo, search, and save is visible. The main area contains the following SQL code:

```
DECLARE
  CURSOR ed_cursor IS
  SELECT empno, sal, dname
  FROM emp e, dept d
  WHERE e.deptno = d.deptno
  FOR UPDATE OF sal NOWAIT;
BEGIN
  FOR v_ed_rec IN ed_cursor LOOP
    UPDATE emp
    SET sal = v_ed_rec.sal*1.1
    WHERE CURRENT OF ed_cursor;
  END LOOP;
  COMMIT;
END;
```

At the bottom, the 'Results' tab is selected, showing the output: '1 row(s) updated.' and '0.04 seconds'.

Cursori în PL/SQL

- 1. LOOP-ul FOR pentru cursor**
- 2. Cursori cu parametri**
- 3. Folosirea cursorilor pentru actualizari**
- 4. Folosirea cursorilor multipli**

4. Folosirea cursorilor multipli

- In programe avem adesea nevoie sa declaram si sa folosim doi sau mai multi cursori in acelasi bloc **PL/SQL**.
- Adesea acesti cursori sunt legati unii de altii prin parametri.

1. Un exemplu de problema

Avem nevoie sa realizam un raport care listeaza fiecare departament ca un subtitlu urmat imediat de o lista a angajatilor din acel departament, apoi urmatorul departament, etc.

Avem nevoie de doi cursori, cate unul pentru fiecare din cele doua tabele.

Cursorul care are la baza tabela EMP este deschis de cateva ori, o data pentru fiecare departament.

Solutia problemei

Pas 1

Declaram doi cursori, cate unul pentru fiecare tabela, plus structurile de tip inregistrare asociate.

DECLARE

CURSOR *c_dept* IS

```
SELECT deptno, dname  
FROM dept ORDER BY dname;
```

CURSOR *c_emp* (p_deptid NUMBER) IS

```
SELECT first_name, last_name  
FROM EMP  
WHERE deptno = p_deptid  
ORDER BY last_name;  
v_deptrec c_dept%ROWTYPE;  
v_emprec c_emp%ROWTYPE;
```

De ce cursorul *c_emp* este declarat cu un parametru?

Pas 2

Deschidem cursorul **c_dept**, preluam si afisam randurile din dept ca de obicei.

```
DECLARE
CURSOR c_dept IS .....;
CURSOR c_emp (p_deptid NUMBER) IS .....;
v_deptrec c_dept%ROWTYPE;
v_emprec c_emp%ROWTYPE;

BEGIN
  OPEN c_dept;
  LOOP
    FETCH c_dept INTO v_deptrec;
    EXIT WHEN c_dept%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_deptrec.dname);
  END LOOP;
CLOSE c_dept;
END;
```

Pas 3

- După preluarea și afișarea fiecărui rând din tabelul DEPT, avem nevoie să preluăm și să afișăm angajații din acel departament.
- Pentru aceasta, deschidem cursorul EMP, îi preluăm și îi afișăm rândurile într-un loop imbricat și închidem cursorul.
- Apoi facem același lucru îl facem pentru următorul rând din dept etc.

DECLARE

CURSOR c_dept IS;

CURSOR c_emp (p_deptid NUMBER) IS;

v_deptrec c_dept%ROWTYPE;

v_emprec c_emp%ROWTYPE;

BEGIN

```
OPEN c_dept;
LOOP
FETCH c_dept INTO v_deptrec;
EXIT WHEN c_dept%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_deptrec.dname);
  OPEN c_emp (v_deptrec.deptno);
  LOOP
    FETCH c_emp INTO v_emprec;
    EXIT WHEN c_emp%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emprec.last_name
  || ' ' || v_emprec.first_name);
  END LOOP;
  CLOSE c_emp;
END LOOP;
CLOSE c_dept;
END;
```

2. Exemplu de problema

- Avem nevoie sa realizam un raport care afiseaza fiecare locatie in care sunt situate departamentele urmate de departamentele din locatiile respective.
- Din nou, avem nevoie de doi cursori, cate unul pentru fiecare din cele doua tabele.
- Cursorul care are la baza tabela DEPT va deschis de cateva ori, de fiecare data pentru fiecare locatie.

DECLARE

```

CURSOR c_loc IS SELECT * FROM locations;
CURSOR c_dept (p_locid NUMBER) IS
SELECT * FROM dept WHERE location_id = p_locid;
v_locrec c_loc%ROWTYPE;
v_deptrec c_dept%ROWTYPE;

```

BEGIN

```

OPEN c_loc;

```

LOOP

```

FETCH c_loc INTO v_locrec;
EXIT WHEN c_loc%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_locrec.city);
OPEN c_dept (v_locrec.location_id);
LOOP
FETCH c_dept INTO v_deptrec;
EXIT WHEN c_dept%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_deptrec.dname);
END LOOP;
CLOSE c_dept;

```

```

END LOOP;

```

```

CLOSE c_loc;

```

```

END;

```

Object Type: TABLE Object: LOCATIONS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
LOCATIONS	CITY	VARCHAR2	20	-	-	-	✓	-	-
	LOCATION_ID	VARCHAR2	20	-	-	-	✓	-	-
	COUNTRY_ID	VARCHAR2	20	-	-	-	✓	-	-
	POSTAL_CODE	NUMBER	-	10	0	-	✓	-	-
	STREET_ADDRESS	NUMBER	-	10	0	-	✓	-	-

Results Explain Describe Saved SQL History

Toronto
Marketing
Oxford
Sales
Southlake
IT
South San Francisco
Shipping
Seattle
Administration
Executive
Accounting
Contracting

Statement processed.

0.12 seconds

Folosirea instructiunii FOR cu cursori multipli

- Putem folosi loop-ul FOR (si alte tehnici pentru cursori cum ar fi FOR UPDATE) cu cursori multipli ca si atunci cand folosim un singur cursor.

```
DECLARE
```

```
CURSOR c_loc IS SELECT * FROM locations;
```

```
CURSOR c_dept (p_locid NUMBER) IS
```

```
SELECT * FROM dept WHERE location_id = p_locid;
```

```
BEGIN
```

```
FOR v_locrec IN c_loc
```

```
LOOP
```

```
    DBMS_OUTPUT.PUT_LINE(v_locrec.city);
```

```
    FOR v_deptrec IN c_dept (v_locrec.location_id)
```

```
        LOOP
```

```
            DBMS_OUTPUT.PUT_LINE(v_deptrec.dname);
```

```
        END LOOP;
```

```
END LOOP;
```

```
END;
```

```
DECLARE
  CURSOR c_loc IS SELECT * FROM locations;
  CURSOR c_dept (p_locid NUMBER) IS
    SELECT * FROM departments WHERE location_id = p_locid;
BEGIN
  FOR v_locrec IN c_loc
    LOOP
      DBMS_OUTPUT.PUT_LINE(v_locrec.city);
      FOR v_deptrec IN c_dept (v_locrec.location_id)
        LOOP
          DBMS_OUTPUT.PUT_LINE(v_deptrec.department_name);
        END LOOP;
      END LOOP;
    END LOOP;
END;
```

Results Explain Describe Saved SQL History

Toronto
Marketing
Oxford
Sales
Southlake
IT
South San Francisco
Shipping
Seattle
Administration
Executive
Accounting
Contracting

Statement processed.

0.01 seconds

Exemplu

Afisarea tuturor angajatilor din toate departamentele si marirea salariilor unora dintre ei

DECLARE

CURSOR c_dept IS SELECT * FROM my_dept;

CURSOR c_emp (p_DEPT_id NUMBER) IS

**SELECT * FROM my_EMP WHERE deptno = p_DEPT_id
FOR UPDATE NOWAIT;**

BEGIN

FOR v_DEPTrec IN c_dept

LOOP

DBMS_OUTPUT.PUT_LINE(v_deptrec.dname);

FOR v_emprec IN c_emp (v_deptrec.deptno)

LOOP

DBMS_OUTPUT.PUT_LINE(v_emprec.last_name);

IF v_deptrec.location_id = 1700 AND v_emprec.salary < 10000

THEN UPDATE my_EMP SET salary = salary * 1.1

WHERE CURRENT OF c_emp;

END IF;

END LOOP;

END LOOP;

END;

Results Explain Describe Saved SQL History

```
Administration
Whalen
Marketing
Hartstein
Fay
Shipping
Mourgos
Rajs
Davies
Matos
Vargas
IT
Hunold
Ernst
Lorentz
Sales
Zlotkey
Abel
Taylor
Executive
King
Kochhar
De Haan
Accounting
Higgins
Gietz
Contracting

1 row(s) updated.
```

Alte probleme

1. Sa se afiseze salariatii care au salariul mai mic de 7000\$, in urmatoarea forma:

Salariatul <nume> are salariul <salariu>

Solutie:

```
BEGIN
  FOR v_rec IN
    (SELECT ename, sal
     FROM emp
     WHERE sal >= 7000)
  LOOP
    DBMS_OUTPUT.PUT_LINE ( ' Salariatul ' ||
      v_rec.ename || ' are salariul: ' || v_rec.sal);
  END LOOP;
END;
```

Alte probleme

2. Să se declare un cursor cu un parametru de tipul codului departamentului, care regăsește numele și salariul angajaților din departamentul respectiv, pentru care nu s-a specificat comisionul.

Să se declare o variabilă `v_nume` de tipul unei linii a cursorului.

DECLARE

CURSOR c_nume (p_idDep
emp.deptno%TYPE) IS

SELECT ename, sal*12 salariu_anual

FROM emp

WHERE comm IS NULL

AND deptno = p_idDep;

BEGIN

FOR v_rec IN c_nume (20) LOOP

DBMS_OUTPUT.PUT_LINE (' Nume:' ||
v_rec.ename || ' salariu : ' || v_rec.sal_an);

END LOOP;

END;

3. Să se dubleze valoarea salariilor celor angajați înainte de 1 ianuarie 1995, care nu câștigă comision.

DECLARE

CURSOR before95 IS

SELECT *

FROM emp

WHERE comm IS NULL

AND hire_date <= TO_DATE('01-JAN-1995','DD-MON-YYYY')

FOR UPDATE OF sal NOWAIT;

BEGIN

FOR x IN before95 LOOP

UPDATE emp

SET sal = sal*2

WHERE CURRENT OF before95;

END LOOP;

-- ce efect ar avea urmatoarea comanda? Explicati.

-- DBMS_OUTPUT.PUT_LINE('Au fost actualizate '||
before95%ROWCOUNT || ' linii');

COMMIT; -- se permanentizeaza actiunea si se elibereaza
blocarea

END;



Întrebări?