

# Tehnici de programare cu baze de date

## #5

### PL/SQL

### Cursori în PL/SQL (partea I-a)

**Adrian Runceanu**  
[www.runceanu.ro/adrian](http://www.runceanu.ro/adrian)

## Curs 5

# Cursori în PL/SQL (partea I)

# Cursori în PL/SQL

- 1. Structura repetitivă – instrucțiuni imbricate**
- 2. Cursori explițiți – introducere**
- 3. Folosirea atributelor cursorilor explițiți**

# 1. Structura repetitivă – instrucțiuni imbricate

Instrucțiunile repetitive se pot imbrica pe mai multe nivele.

***Exemple:***

1)

**BEGIN**

**FOR v\_outerloop IN 1..3 LOOP**

**FOR v\_innerloop IN REVERSE 1..5 LOOP**

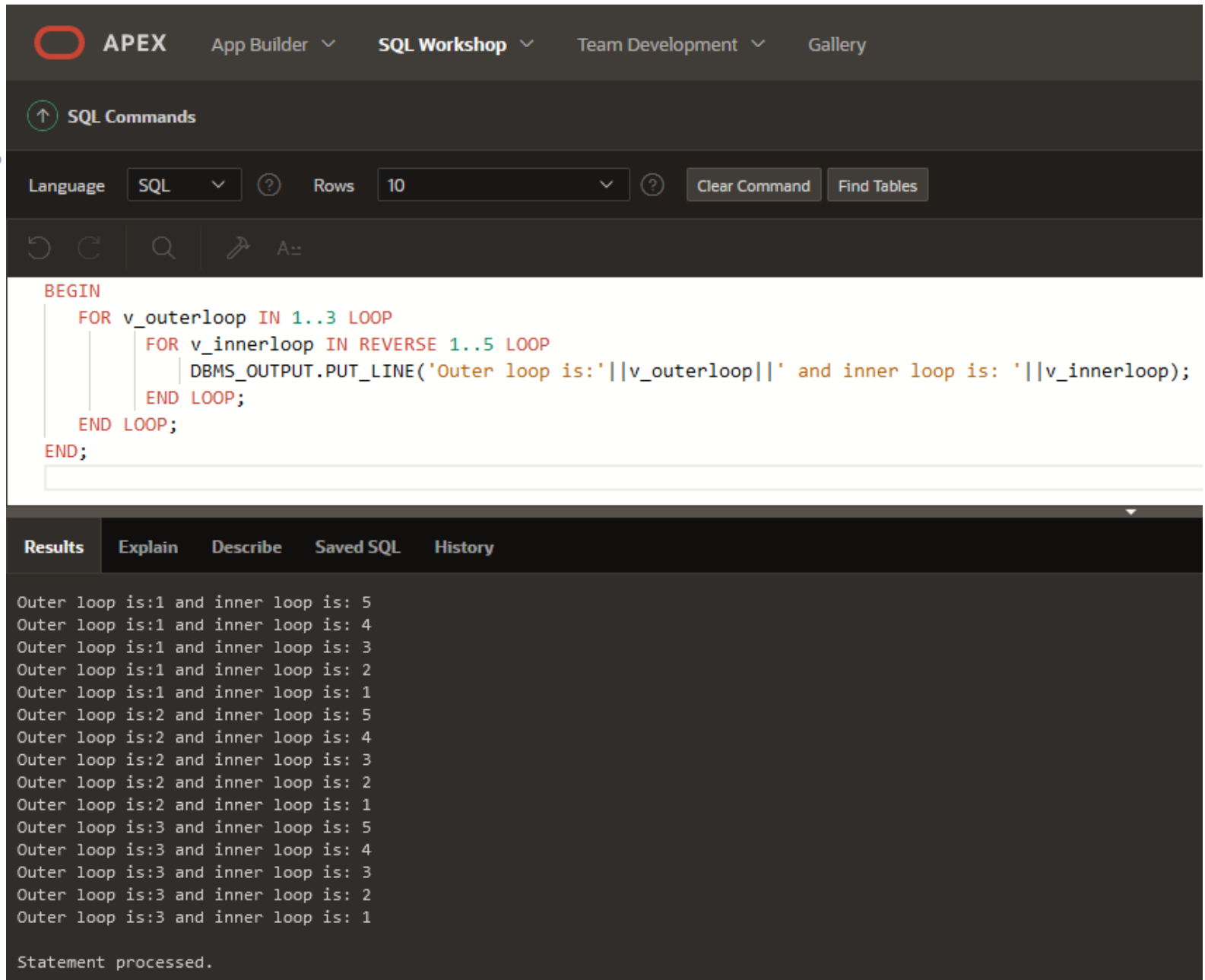
**DBMS\_OUTPUT.PUT\_LINE('Outer loop is: '||v\_outerloop||' and inner loop is:**

**'||v\_innerloop);**

**END LOOP;**

**END LOOP;**

**END;**



The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below this is the 'SQL Commands' section, which includes a 'Language' dropdown set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. The main area contains a PL/SQL block with two nested loops. The outer loop iterates from 1 to 3, and the inner loop iterates in reverse from 5 to 1. The inner loop prints the current values of both loops. Below the code, the 'Results' tab is active, displaying 15 lines of output that correspond to the 15 iterations of the nested loops. The output shows the outer loop value followed by the inner loop value, separated by 'and inner loop is:'. The final line of the results area indicates 'Statement processed.'

```
BEGIN
  FOR v_outerloop IN 1..3 LOOP
    FOR v_innerloop IN REVERSE 1..5 LOOP
      DBMS_OUTPUT.PUT_LINE('Outer loop is: '||v_outerloop||' and inner loop is: '||v_innerloop);
    END LOOP;
  END LOOP;
END;
```

**Results** Explain Describe Saved SQL History

```
Outer loop is:1 and inner loop is: 5
Outer loop is:1 and inner loop is: 4
Outer loop is:1 and inner loop is: 3
Outer loop is:1 and inner loop is: 2
Outer loop is:1 and inner loop is: 1
Outer loop is:2 and inner loop is: 5
Outer loop is:2 and inner loop is: 4
Outer loop is:2 and inner loop is: 3
Outer loop is:2 and inner loop is: 2
Outer loop is:2 and inner loop is: 1
Outer loop is:3 and inner loop is: 5
Outer loop is:3 and inner loop is: 4
Outer loop is:3 and inner loop is: 3
Outer loop is:3 and inner loop is: 2
Outer loop is:3 and inner loop is: 1
Statement processed.
```

## 2) Urmatoarea instructiune contine o conditie **EXIT** in structurile repetitive imbricate

### **DECLARE**

```
v_outer_done CHAR(3) := 'NO';  
v_inner_done CHAR(3) := 'NO';
```

### **BEGIN**

```
LOOP -- outer loop  
...  
    LOOP -- inner loop  
    ...  
    ... -- pas A  
    EXIT WHEN v_inner_done = 'YES';  
    .....  
    END LOOP;  
...  
EXIT WHEN v_outer_done = 'YES';  
...  
END LOOP;
```

### **END;**

Ce se intampla daca vrem sa iesim dintr-un loop la pasul A?

## **ETICHETE**

### **DECLARE**

...

### **BEGIN**

**<<outer\_loop>>**

**LOOP** -- outer loop

...

**<<inner\_loop>>**

**LOOP** -- inner loop

**EXIT outer\_loop WHEN ...** -- iesire din ambele loop-uri

**EXIT WHEN v\_inner\_done = 'YES';**

...

**END LOOP;**

...

**EXIT WHEN v\_outer\_done = 'YES';**

...

**END LOOP;**

**END;**

# 1. Structura repetitivă – instrucțiuni imbricate

- Denumirile etichetelor dintr-un **loop** respecta aceleasi reguli ca orice identificator.
- O eticheta este plasata inaintea unei instructiuni fie pe aceeasi linie, fie pe linie separata.
- In instructiunile **FOR** si **WHILE** eticheta se plaseaza inainte de **FOR** sau **WHILE** cu delimitatorii de eticheta ( **<<label>>** ).
- Daca instructiunea **loop** este etichetata, denumirea etichetei poate fi inclusa optional dupa **END LOOP** pentru claritate.



**Exemplu** – La instruciunea **basic loop** se pune eticheta inainte de cuvantul LOOP intre delimitatorii de eticheta.

**DECLARE**

**v\_outerloop PLS\_INTEGER :=0;**

**v\_innerloop PLS\_INTEGER :=5;**

**BEGIN**

**<<Outer\_loop>>**

**LOOP**

**v\_outerloop := v\_outerloop + 1;**

**v\_innerloop := 5;**

**EXIT WHEN v\_outerloop > 3;**

**<<Inner\_loop>>**

**LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Outer loop  
is: '||v\_outerloop||' and inner loop is: '||v\_innerloop);**


**v\_innerloop := v\_innerloop - 1;**

**EXIT WHEN v\_innerloop =0;**

**END LOOP Inner\_loop;**

**END LOOP Outer\_loop;**

**END;**

 SQL Commands

Language SQL Rows 10 Clear Command Find Tables





 A:

```

DECLARE
    v_outerloop PLS_INTEGER :=0;
    v_innerloop PLS_INTEGER :=5;
BEGIN
    <<Outer_loop>>
    LOOP
        v_outerloop := v_outerloop + 1;
        v_innerloop := 5;
        EXIT WHEN v_outerloop > 3;
        <<Inner_loop>>
        LOOP
            DBMS_OUTPUT.PUT_LINE('Outer loop is: '||v_outerloop||' and inner loop is: '||v_innerloop);
            v_innerloop := v_innerloop - 1;
            EXIT WHEN v_innerloop =0;
        END LOOP Inner_loop;
    END LOOP Outer_loop;
END;

```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```

Outer loop is:1 and inner loop is: 5
Outer loop is:1 and inner loop is: 4
Outer loop is:1 and inner loop is: 3
Outer loop is:1 and inner loop is: 2
Outer loop is:1 and inner loop is: 1
Outer loop is:2 and inner loop is: 5
Outer loop is:2 and inner loop is: 4
Outer loop is:2 and inner loop is: 3
Outer loop is:2 and inner loop is: 2
Outer loop is:2 and inner loop is: 1
Outer loop is:3 and inner loop is: 5
Outer loop is:3 and inner loop is: 4
Outer loop is:3 and inner loop is: 3
Outer loop is:3 and inner loop is: 2
Outer loop is:3 and inner loop is: 1

```

Statement processed.

## Exemplu – loop-uri imbricate si etichete

```
...BEGIN
```

```
<<Outer_loop>>
```

```
LOOP
```

```
  v_counter := v_counter+1;
```

```
  EXIT WHEN v_counter>10;
```

```
<<Inner_loop>>
```

```
  LOOP
```

```
    ...
```

```
    EXIT Outer_loop WHEN v_total_done = 'YES';
```

```
    -- iese din ambele loop-uri
```

```
    EXIT WHEN v_inner_done = 'YES';
```

```
    -- iese numai din inner loop
```

```
    ...
```

```
  END LOOP Inner_loop;
```

```
  ...
```

```
END LOOP Outer_loop;
```

```
END;
```

# Cursori în PL/SQL

- 1. Structura repetitivă – instrucțiuni imbricate**
- 2. Cursori explițiți – introducere**
- 3. Folosirea atributelor cursorilor explițiți**

## 2. Cursori expliți – introducere

- Se știe că o instrucțiune **SQL** într-un bloc **PL/SQL** rulează cu succes dacă dă un singur rezultat.
- Dar dacă avem nevoie să scriem o instrucțiune **SELECT** care să dea mai multe rezultate?
- Dacă de exemplu avem de făcut un raport cu toți angajații?
- Pentru **a obține mai multe rezultate** trebuie să declarăm și să folosim un **cursor explicit**.

## 2. Cursori expliți – introducere

### *Cursorii și domeniile lor de context*

- Serverul **Oracle** alocă zone private de memorie numite **zone (domenii) de context** pentru a stoca datele prelucrate de o instrucțiune SQL.
- Fiecare zonă de context (și prin urmare fiecare instrucțiune SQL) are un cursor asociat.
- Ne putem gândi la un cursor ca la o etichetă a zonei de context sau un pointer al zonei de context.
- *De fapt un cursor este și o etichetă și un pointer.*

## 2. Cursori explițiți – introducere

### ***Cursori implițiți și cursori explițiți***

- **Cursorii implițiți** – sunt definiți automat de **Oracle** pentru toate instrucțiunile **DML** ale SQL (INSERT, UPDATE, DELETE și MERGE) și pentru *toate instrucțiunile SQL care returnează un singur rând*
- **Cursorii explițiți** – declarați de programator *pentru interogările care returnează mai mult de un rând*. Cursorii explițiți se pot folosi pentru a denumi o zonă de context și pentru a accesa datele stocate în ea.

## *Limitele cursorilor impliciți*

Fie urmatorul exemplu:

◦ **DECLARE**

**v\_sal emp.sal%TYPE;**

**BEGIN**

**SELECT sal**

**INTO v\_sal**

**FROM emp;**

**DBMS\_OUTPUT.PUT\_LINE(' Salary is :**

**'||v\_sal);**

**END;**

```
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 4
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
2. v_sal emp.sal%TYPE;
3. BEGIN
4. SELECT sal
5. INTO v_sal
6. FROM emp;
```

Se va afisa un mesaj de eroare, deoarece tabela **employees** are mai multe linii

```
ORA-01422: exact fetch returns more than requested number of rows
```



## 2. Cursori expliți – introducere

- Cu un **cursor explicit** putem *extrage randuri multiple din tabela*, având un pointer către fiecare rand extras și putem lucra cu un rand la un moment dat.
- Motivele pentru care folosim cursorii expliți sunt:
  1. Este singura modalitate în **PL/SQL** de a folosi mai mult de un rând dintr-o tabelă
  2. Fiecare rând este preluat de către o instrucțiune de program separată, dând programatorului mai mult control în prelucrarea rândurilor

## ***Exemplu de cursor explicit:***

Cursorul se foloseste pentru **numele tarilor** si **populatiile** pentru **continentul Africa**

### **DECLARE**

```
CURSOR eba_population_cursor IS
```

```
SELECT name, population
```

```
FROM eba_countries where region_id = 40;
```

```
v_country_name eba_countries.name%TYPE;
```

```
v_population eba_countries.population%TYPE;
```

### **BEGIN**

```
OPEN eba_population_cursor;
```

### **LOOP**

```
    FETCH eba_population_cursor INTO v_country_name,  
    v_population;
```

```
    EXIT WHEN eba_population_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE(v_country_name||'
```

```
    '||v_population);
```

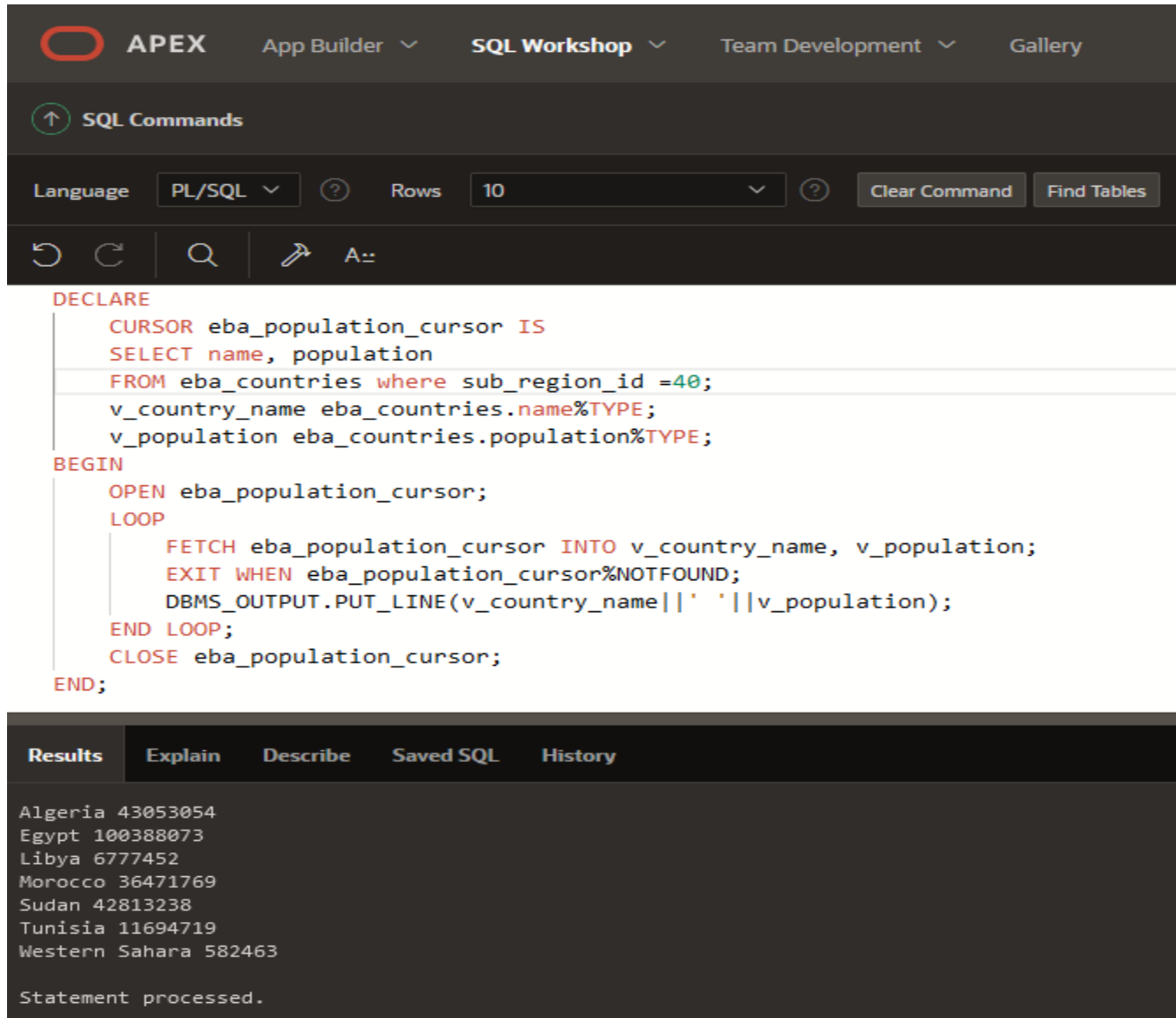
```
END LOOP;
```

```
CLOSE eba_population_cursor;
```

```
END;
```

## Exemplu de cursor explicit:

Cursorul se foloseste pentru numele tarilor si populatiile pentru continentul Africa



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a search icon and a 'Language' dropdown set to 'PL/SQL'. There are also 'Rows' and 'Clear Command' buttons. The main area contains the following SQL code:

```
DECLARE
  CURSOR eba_population_cursor IS
  SELECT name, population
  FROM eba_countries where sub_region_id =40;
  v_country_name eba_countries.name%TYPE;
  v_population eba_countries.population%TYPE;
BEGIN
  OPEN eba_population_cursor;
  LOOP
    FETCH eba_population_cursor INTO v_country_name, v_population;
    EXIT WHEN eba_population_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_country_name||' '||v_population);
  END LOOP;
  CLOSE eba_population_cursor;
END;
```

Below the code, there is a 'Results' section with a table of data. The table has two columns: country name and population. The results are as follows:

Country	Population
Algeria	43053054
Egypt	100388073
Libya	6777452
Morocco	36471769
Sudan	42813238
Tunisia	11694719
Western Sahara	582463

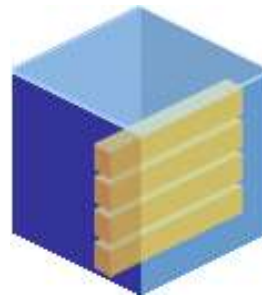
At the bottom of the results section, it says 'Statement processed.'

## 2. Cursori explițiți – introducere

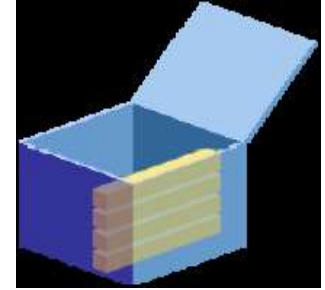
### *Operații – cursori explițiți*

- O mulțime de rânduri furnizate de o *interogare multiple-row* este denumită **mulțime activă (active set)** și este stocată în zona de context.
- Dimensiunea sa este numărul de rânduri care îndeplinesc criteriul de căutare.
- Ne putem imagina zona de context ca fiind o cutie, conținutul cutiei fiind mulțimea activă.
- Pentru a prelua date trebuie deschisă (**OPEN**) cutia și sunt preluate (**FETCH**) rândurile din cutie, pe rând, câte unul.
- Când am terminat trebuie să închidem (**CLOSE**) cutia.

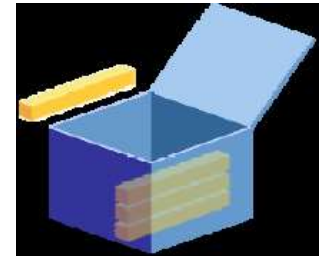
# Cursor explicit:



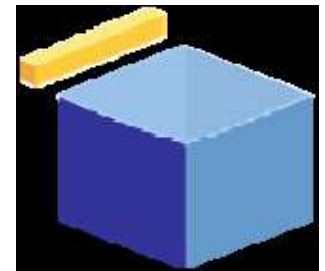
1. *Deschiderea cursorului*



2. *Preluarea a câte unui rând*



3. *Închiderea cursorului*



## ***Declararea și controlul cursorilor expliciți***

- Pașii pentru utilizarea cursorilor expliciți sunt:
1. **Declarare (DECLARE)** – se face în secțiunea declarativă și se denumește setul activ
  2. **Deschidere (OPEN)** – Instrucțiunea **OPEN** execută interogarea asociată cursorului, identifică setul de rezultate și poziționează cursorul înaintea primului rând.
  3. **Preluare (FETCH)** – Instrucțiunea **FETCH** extrage rândul curent și avansează cursorul la rândul următor.
  4. **Închiderea (CLOSE)** cursorului se face după ce a fost prelucrat ultimul rând. Instrucțiunea **CLOSE** dezactivează cursorul.

# 1. Declararea unui cursor

- Setul activ al unui cursor este determinat de instructiunea SELECT din declararea cursorului.

## Sintaxa

```
CURSOR cursor_name IS  
select_statement;
```

**unde:**

**cursor\_name** - reprezinta un identificator  
PL/SQL

**select\_statement;** - reprezinta o instructiune  
SELECT fara o clauza INTO

# 1. Declararea unui cursor

## Exemplu 1

Cursorul *emp\_cursor* este declarat pentru a extrage coloanele *empno* si *ename* pentru angajatii care lucreaza in departamentul pentru care *deptno* este 30.

**DECLARE**

**CURSOR emp\_cursor IS**

**SELECT empno, ename**

**FROM    WHERE deptno =30;**

...



# 1. Declararea unui cursor

## Exemplu 2

Cursorul ***dept\_cursor*** este declarat pentru a extrage toate informatiile pentru departamentele care au ***loc*** 1700. Preluam si prelucram randurile alfabetic dupa ***dname***.

**DECLARE**

**CURSOR dept\_cursor IS**

**SELECT \***

**FROM dept**

**WHERE loc = 1700**

**ORDER BY dname;**

**...**

# 1. Declararea unui cursor

## Exemplu 3

O instructiune **SELECT** in declararea unui cursor poate include join-uri, functii de grup si subinterogari. Acest exemplu extrage fiecare departament care are cel putin doi angajati furnizand numele departamentului si numarul de angajati.

**DECLARE**

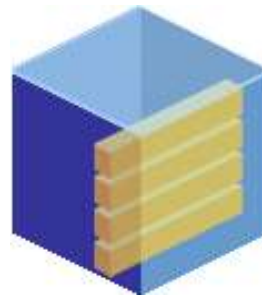
```
CURSOR dept_emp_cursor IS  
SELECT dname, COUNT(*) AS how_many  
FROM dept d, emp e  
WHERE d.deptno = e.deptno  
GROUP BY d.dname  
HAVING COUNT(*) > 1;
```

...

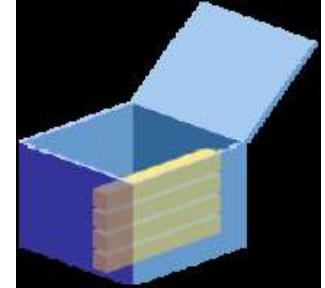
## Reguli de declarare a unui cursor

1. Nu se include clauza **INTO** in declararea cursorului deoarece va aparea mai tarziu in instructiunea **FETCH**
2. Daca randurile se prelucreaza intr-o ordine specificata atunci se foloseste clauza **ORDER BY**
3. Cursorul poate fi orice instructiune **SELECT** valida ce poate include join-uri, subinterogari etc.
4. Daca declararea unui cursor se refera la variabile **PL/SQL**, atunci aceste variabile trebuie declarate inainte de cursor.

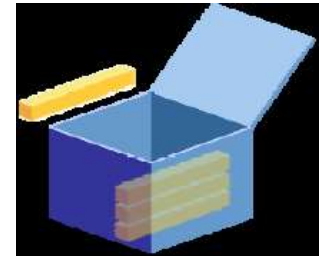
# Cursor explicit:



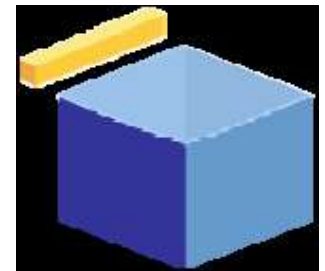
1. ***Deschiderea cursorului***



2. ***Preluarea a câte unui rând***



3. ***Închiderea cursorului***



## 2. Deschiderea unui cursor

- Instructiunea **OPEN** executa interogarea asociata cursorului, identifica multimea activa si pozitioneaza pointerul cursorului catre primul rand.
- Instructiunea **OPEN** este inclusa in partea executabila a unui bloc **PL/SQL**.

**DECLARE**

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

...

**BEGIN**

```
OPEN emp_cursor;
```

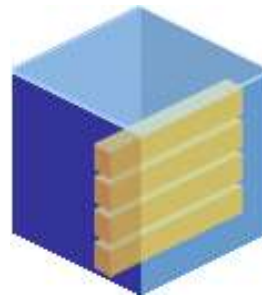
...

## 2. Deschiderea unui cursor

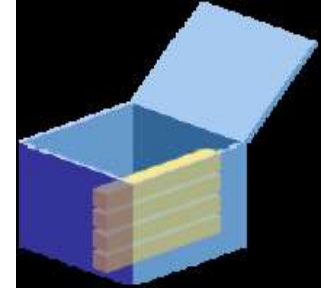
Instructiunea **OPEN** efectueaza urmatoarele 3 operatii:

1. Aloca memorie pentru zona de context (*creeza "cutia"*)
2. Executa instructiunea **SELECT** din declararea cursorului, returnand rezultatele in multimea activa (*umple „cutia” cu date*)
3. Pozitioneaza pointerul la primul rand din multimea activa (*deschide "cutia"*)

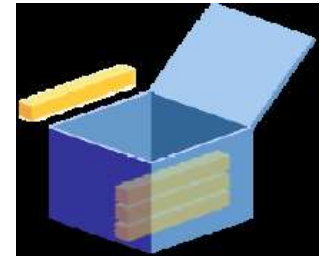
# Cursor explicit:



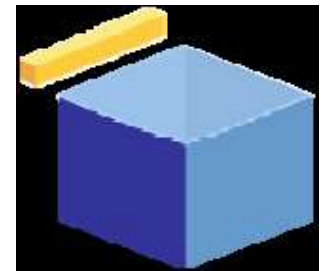
1. ***Deschiderea cursorului***



2. ***Preluarea a câte unui rând***



3. ***Închiderea cursorului***



### 3. Preluarea datelor de la cursor

- Instructiunea **FETCH** extrage randuri de la cursor, unul singur la un moment dat.
- Dupa fiecare preluare, cursorul avanseaza la urmatorul rand in multimea activa.



### 3. Preluarea datelor de la cursor

Doua variabile ***v\_empno*** si ***v\_lname*** sunt declarate pentru a retine valorile preluate de cursor.

**DECLARE**

**CURSOR emp\_cursor IS**

**SELECT empno, ename**

**FROM emp**

**WHERE depno =10;**

**v\_empno emp.empno%TYPE;**

**v\_ename emp.ename%TYPE;**

**BEGIN**

**OPEN emp\_cursor;**

**FETCH emp\_cursor INTO v\_empno, v\_ename;**

**DBMS\_OUTPUT.PUT\_LINE(v\_empno || ' '||v\_ename);**

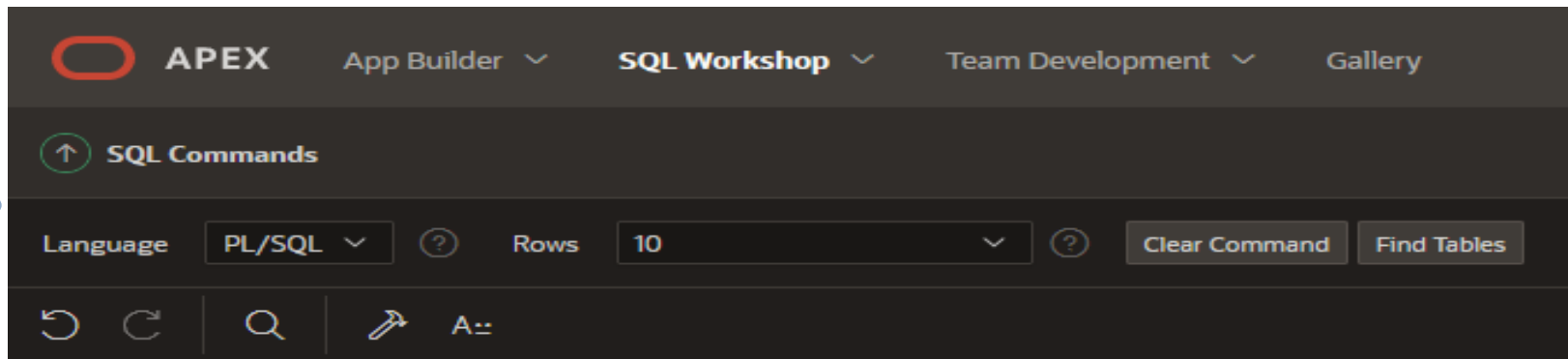
**...**

**END;**

Pentru a prelua toate randurile avem nevoie de **loop**-uri.

## DECLARE

```
CURSOR emp_cursor IS  
SELECT empno, ename  
FROM emp  
WHERE deptno=30;  
v_empno emp.empno%TYPE;  
v_ename emp.ename%TYPE;  
BEGIN  
    OPEN emp_cursor;  
    LOOP  
        FETCH emp_cursor INTO v_empno,  
v_ename;  
        EXIT WHEN emp_cursor%NOTFOUND;  
        DBMS_OUTPUT.PUT_LINE( v_empno ||  
'||v_ename);  
    END LOOP;  
END;
```



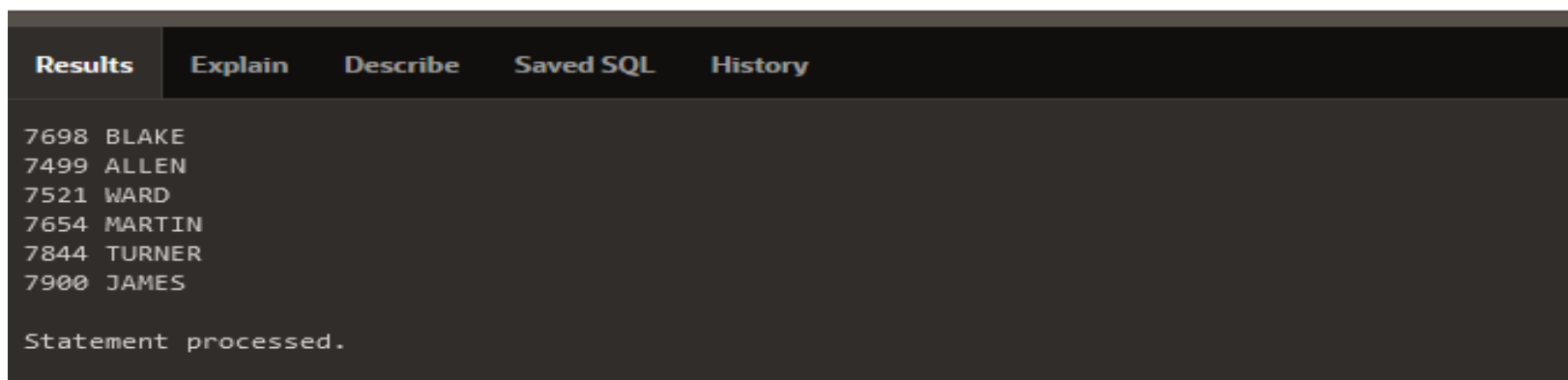
APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

Undo Redo Search A::

```
DECLARE
  CURSOR emp_cursor IS
    SELECT empno, ename
    FROM emp
    WHERE deptno=30;
  v_empno emp.empno%TYPE;
  v_ename emp.ename%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_empno, v_ename;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empno || ' ' ||v_ename);
  END LOOP;
END;
```



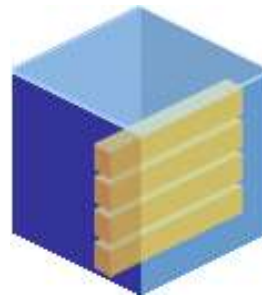
Results	Explain	Describe	Saved SQL	History
7698	BLAKE			
7499	ALLEN			
7521	WARD			
7654	MARTIN			
7844	TURNER			
7900	JAMES			

Statement processed.

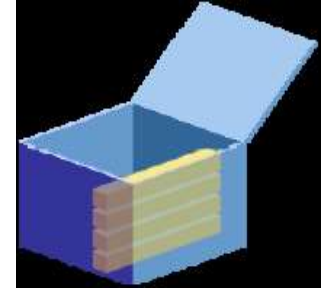
# Reguli pentru preluarea datelor de la un cursor

1. Trebuie sa includem **acelasi numar de variabile** in clauza **INTO** a instructiunii **FETCH** ca si **numarul de coloane** din instructiunea **SELECT**, iar tipurile de date trebuie sa fie compatibile.
2. Corespondenta dintre variabile si coloane trebuie sa fie de unu la unu.
3. Trebuie sa verificam daca cursorul contine randuri. Daca o preluare nu capata valori, atunci nu mai sunt randuri de procesat in multimea activa si nu se inregistreaza erori. Ultimul rand este prelucrat din nou.
4. Putem folosi atributul de cursor **%NOTFOUND** pentru a testa o conditie de iesire.

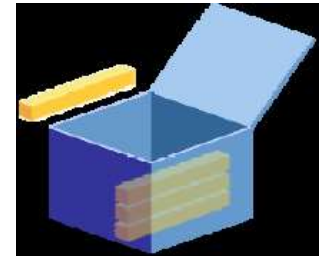
# Cursor explicit:



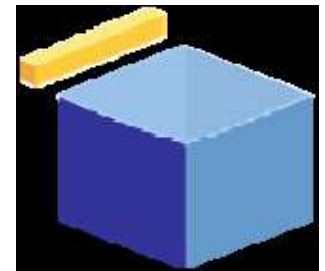
1. *Deschiderea cursorului*



2. *Preluarea a câte unui rând*



3. *Închiderea cursorului*



## 4. Inchiderea cursorilor

- Instructiunea **CLOSE** dezactiveaza cursorul, elibereaza zona de context si zona ramane nedefinita.
- Cursorul se inchide dupa prelucrarea completa a instructiunii **FETCH**.
- Cursorul se poate deschide mai tarziu daca este nevoie.

## 4. Inchiderea cursorilor

Ne putem gandi la inchiderea cursorului ca la golirea si inchiderea „cutiei”, deci nu mai putem lua nimic din continutul ei.

...

**LOOP**

**FETCH emp\_cursor INTO v\_empno,  
v\_ename;**

**EXIT WHEN emp\_cursor%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE( v\_empno ||'**

**'||v\_ename);**

**END LOOP;**

**CLOSE emp\_cursor;**

**END;**

# Reguli pentru închiderea unui cursor

1. Un cursor poate fi redeschis numai dacă este închis.
2. Dacă încercați să preluați date dintr-un cursor după ce a fost închis atunci apare o excepție **INVALID\_CURSOR**
3. Dacă mai târziu redeschidem cursorul atunci instrucțiunea **SELECT** asociată este executată din nou pentru a reumple zona de context cu cele mai recente date din baza de date.



## ***Exemplu de cursor explicit:***

Cursorul se foloseste pentru **numele tarilor** si **populatiile** pentru **continentul Africa**

### **DECLARE**

```
CURSOR eba_population_cursor IS
```

```
SELECT name, population
```

```
FROM eba_countries where region_id IN(30,34,35);
```

```
v_country_name eba_countries.name%TYPE;
```

```
v_population eba_countries.population%TYPE;
```

### **BEGIN**

```
OPEN eba_population_cursor;
```

### **LOOP**

```
    FETCH eba_population_cursor INTO v_country_name,  
    v_population;
```

```
    EXIT WHEN eba_population_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE(v_country_name||'
```

```
    ||v_population);
```

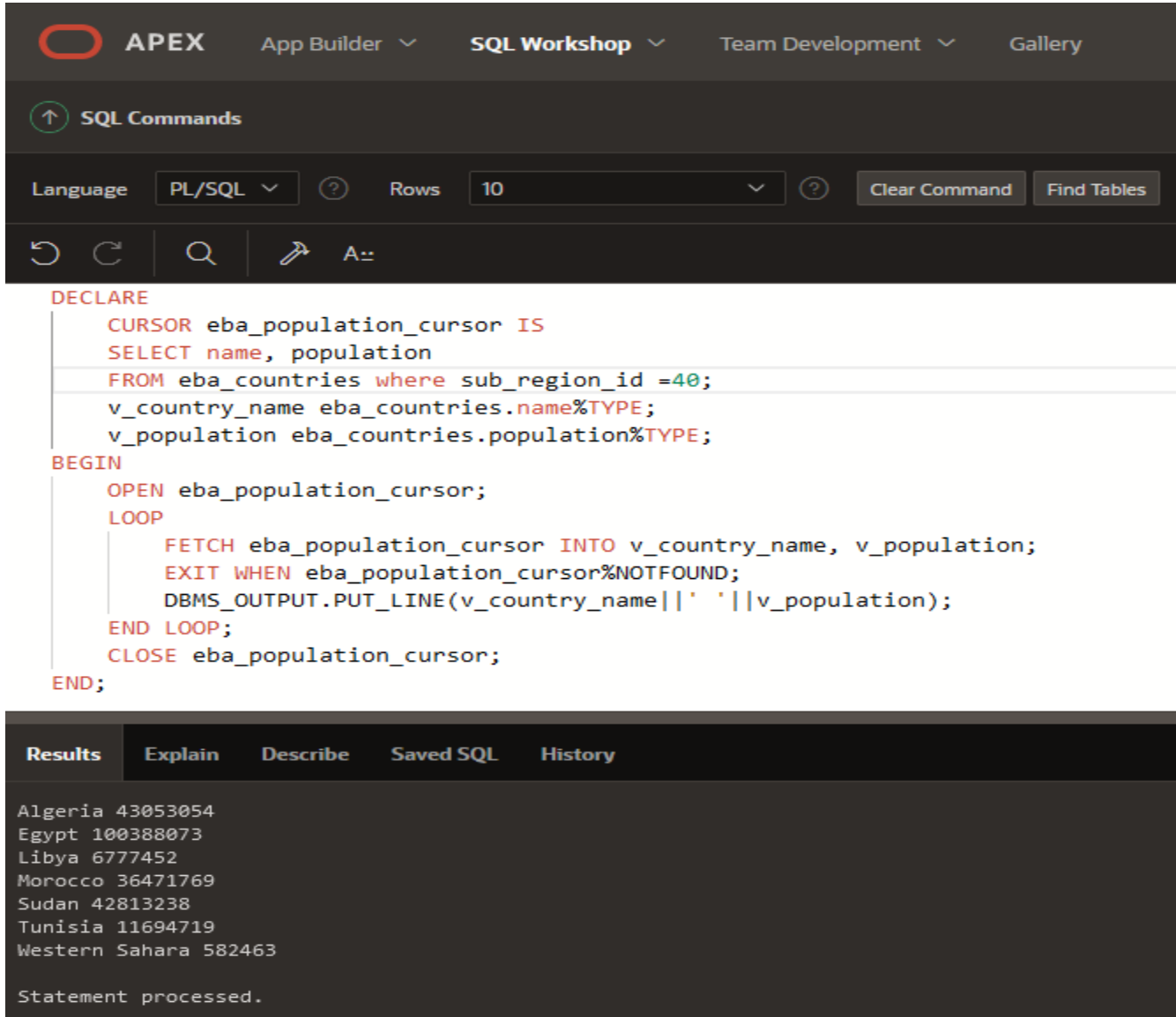
```
END LOOP;
```

```
CLOSE eba_population_cursor;
```

```
END;
```

## Exemplu de cursor explicit:

Cursorul se foloseste pentru numele tarilor si populatiile pentru continentul Africa



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a 'Language' dropdown set to 'PL/SQL', a 'Rows' limit of '10', and buttons for 'Clear Command' and 'Find Tables'. The main area contains the following PL/SQL code:

```
DECLARE
  CURSOR eba_population_cursor IS
  SELECT name, population
  FROM eba_countries where sub_region_id =40;
  v_country_name eba_countries.name%TYPE;
  v_population eba_countries.population%TYPE;
BEGIN
  OPEN eba_population_cursor;
  LOOP
    FETCH eba_population_cursor INTO v_country_name, v_population;
    EXIT WHEN eba_population_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_country_name||' '||v_population);
  END LOOP;
  CLOSE eba_population_cursor;
END;
```

Below the code, the 'Results' tab is active, showing the output of the query:

Results	Explain	Describe	Saved SQL	History
Algeria 43053054				
Egypt 100388073				
Libya 6777452				
Morocco 36471769				
Sudan 42813238				
Tunisia 11694719				
Western Sahara 582463				

At the bottom of the results area, it states 'Statement processed.'

# Cursori în PL/SQL

- 1. Structura repetitivă – instrucțiuni imbricate**
- 2. Cursori explițiți – introducere**
- 3. Folosirea atributelor cursorilor explițiți**

### 3. FOLOSIREA ATRIBUTELOR CURSORILOR EXPLICITI

- **Înregistrările** cursorului ne permit să declarăm o singură variabilă pentru toate coloanele selectate în cursor.
- **Atributele** cursorului ne permit să extragem informații cu privire la starea cursorului explicit.

## 3.1. *Cursori și înregistrări*

- Cursorul din urmatorul exemplu este bazat pe o instrucțiune SELECT care extrage *doar doua coloane* din fiecare rând al tablei.

## DECLARE

```
v_emp_id emp.empno%TYPE;  
v_ename emp.ename%TYPE;  
CURSOR emp_cursor IS  
SELECT empno, ename  
FROM emp  
WHERE deptno =30;
```

## BEGIN

```
OPEN emp_cursor;  
LOOP  
    FETCH emp_cursor INTO v_emp_id,  
v_ename;  
    ...
```

Dar daca extrage 6 coloane sau 7, 8, 9,...coloane?

Urmatorul cursor extrage in intregime randurile din tabela angajati:

**DECLARE**

**v\_empno emp.empno%TYPE;**

**v\_ename emp.ename%TYPE;**

...

**v\_deptno emp.deptno%TYPE;**

**CURSOR emp\_cursor IS**

**SELECT \* FROM emp**

**WHERE deptno =30;**

**BEGIN**

**OPEN emp\_cursor;**

**LOOP**

**FETCH emp\_cursor**

**INTO v\_empno, v\_ename, v\_sal ...v\_deptno;**

...

Am declarat si  
utilizat cate o  
variabila  
pentru fiecare  
coloana a  
tabelei emp

Este mult de scris si este incomod, nu?

Un cod mai simplu pentru a extrage aceleasi informatii este urmatorul:

◦ **DECLARE**

**CURSOR** emp\_cursor IS

**SELECT \***

**FROM emp**

**WHERE deptno =30;**

**v\_emp\_record emp\_cursor%ROWTYPE;**

**BEGIN**

**OPEN emp\_cursor;**

**LOOP**

**FETCH emp\_cursor INTO**

**v\_emp\_record;**

**...**



- Acest cod folosește **%ROWTYPE** pentru a declara o *structura de tip înregistrare bazată pe cursor*.
- *O înregistrare este un tip de date compus în PL/SQL.*
- O înregistrare este formată din mai multe câmpuri, fiecare cu propriul nume și tip de date.
- Putem referi câmpurile prefixându-le denumirile de caracterul punct și denumirea înregistrării.
- **%ROWTYPE** declară o înregistrare cu aceleași câmpuri ca și cursorul pe care se bazează.

Exemplu:

```
DECLARE  
CURSOR emp_cursor IS  
SELECT empno, ename, sal  
FROM emp  
WHERE deptno =30;  
v_emp_record emp_cursor%ROWTYPE;  
  
...
```

<code>v_emp_record.empno</code>	<code>v_emp_record.ename</code>	<code>v_emp_record.sal</code>
100	King	24000

**%ROWTYPE** este convenabil pentru prelucrarea randurilor din multimea activa deoarece putem prelua datele intr-o modalitate mai simpla, folosind inregistrarile.

Exemplu:

**DECLARE**

**CURSOR** emp\_cursor IS

**SELECT \* FROM emp**

**WHERE deptno =30;**

**v\_emp\_record emp\_cursor%ROWTYPE;**

**BEGIN**

**OPEN** emp\_cursor;

**LOOP**

**FETCH** emp\_cursor INTO v\_emp\_record;

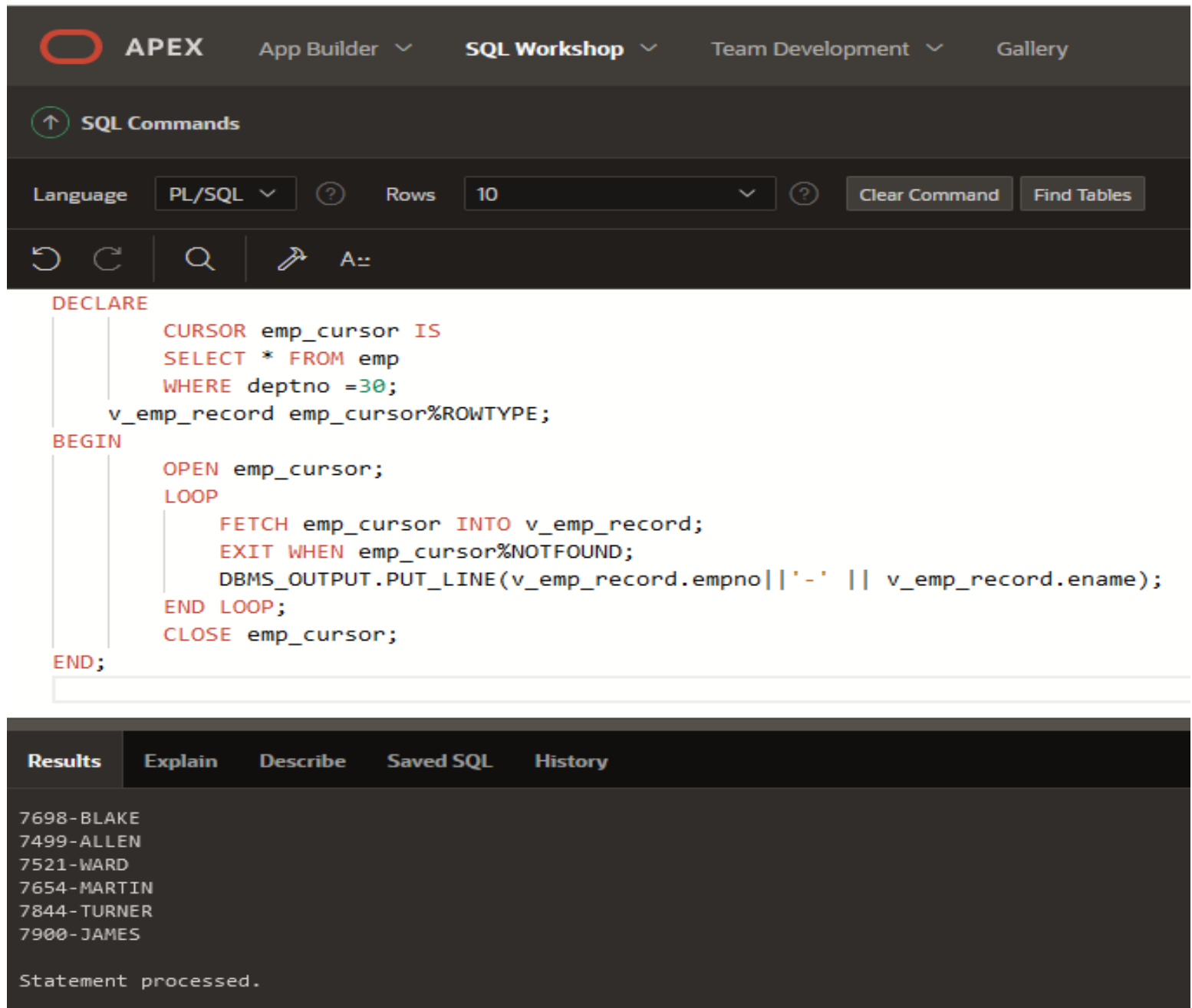
**EXIT WHEN** emp\_cursor%NOTFOUND;

**DBMS\_OUTPUT.PUT\_LINE(v\_emp\_record.empno||'-' ||  
v\_emp\_record.ename);**

**END LOOP;**

**CLOSE** emp\_cursor;

**END;**



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a 'Language' dropdown set to 'PL/SQL' and 'Rows' set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. The main area contains a PL/SQL script:

```
DECLARE
    CURSOR emp_cursor IS
    SELECT * FROM emp
    WHERE deptno =30;
    v_emp_record emp_cursor%ROWTYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_emp_record;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_emp_record.empno || '-' || v_emp_record.ename);
    END LOOP;
    CLOSE emp_cursor;
END;
```

Below the script is a 'Results' tab with sub-tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the following output:


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

Statement processed.
```

Apelul fiecarui camp al inregistrarii prin specificarea **denumirii campului** precedat de constructia **.nume\_inregistrare**

## DECLARE

```
CURSOR emp_dept_cursor IS  
SELECT ename, sal, dname  
FROM emp e, dept d  
WHERE e.deptno = d.deptno;  
v_emp_dept_record emp_dept_cursor%ROWTYPE;  
BEGIN  
OPEN emp_dept_cursor;  
LOOP  
    FETCH emp_dept_cursor INTO v_emp_dept_record;  
    EXIT WHEN emp_dept_cursor%NOTFOUND;  
    DBMS_OUTPUT.PUT_LINE( v_emp_dept_record.ename ||  
– '|| v_emp_dept_record.sal ||' – '|| v_emp_dept_record.dname);  
END LOOP;  
CLOSE emp_dept_cursor;  
END;
```

 SQL Commands

 Language PL/SQL ▾ ? Rows 10 ▾ ? Clear Command Find Tables
    A::

```

DECLARE
    CURSOR emp_dept_cursor IS
    SELECT ename, sal, dname
    FROM emp e, dept d
    WHERE e.deptno = d.deptno;
    v_emp_dept_record emp_dept_cursor%ROWTYPE;
BEGIN
    OPEN emp_dept_cursor;
    LOOP
        FETCH emp_dept_cursor INTO v_emp_dept_record;
        EXIT WHEN emp_dept_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( v_emp_dept_record.ename || ' - ' || v_emp_dept_record.sal || ' - ' || v_emp_dept_record.dname);
    END LOOP;
    CLOSE emp_dept_cursor;
END;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```

CLARK - 2450 - ACCOUNTING
MILLER - 1300 - ACCOUNTING
KING - 5000 - ACCOUNTING
FORD - 3900 - RESEARCH
SCOTT - 3300 - RESEARCH
JONES - 3272.5 - RESEARCH
SMITH - 800 - RESEARCH
ADAMS - 1100 - RESEARCH
WARD - 1250 - SALES
MARTIN - 1250 - SALES
TURNER - 1500 - SALES
JAMES - 950 - SALES
ALLEN - 1600 - SALES
BLAKE - 2850 - SALES
```

Statement processed.

## 3.2. Atributele cursorilor expliți

- Ca și pentru cursorii implicați, sunt câteva atribute utile pentru a obține informații cu privire la starea cursorilor expliți.
- Aceste atribute ne dau informații utile cu privire la execuția unei instrucțiuni de manipulare a cursorului.

## 3.2. Atributele cursorilor expliți

Attribute	Type	Description
<b>%ISOPEN</b>	Boolean	Evaluates to TRUE if the cursor is open
<b>%NOTFOUND</b>	Boolean	Evaluates to TRUE if the most recent fetch did not return a row
<b>%FOUND</b>	Boolean	Evaluates to TRUE if the most recent fetch returned a row; opposite of %NOTFOUND
<b>%ROWCOUNT</b>	Number	Evaluates to the total number of rows FETCHed so far



## a) Atributul **%ISOPEN**

- Se stie ca putem extrage randuri doar atunci cand cursorul este deschis.
- Pentru a verifica daca este deschis cursorul se foloseste atributul **%ISOPEN**.
- **%ISOPEN** ne da starea cursorului:
  - **TRUE** daca acesta este deschis
  - si **FALSE** in caz contrar.

### *Exemplu:*

```
IF NOT emp_cursor%ISOPEN THEN  
    OPEN emp_cursor;  
END IF;  
LOOP  
FETCH emp_cursor...
```

De obicei, aceste 2 atribute se folosesc intr-un **loop** pentru a determina iesirea din loop.

## **b) Atributul %ROWCOUNT**

- Atributul **%ROWCOUNT** se foloseste pentru:
  1. Pentru prelucrarea unui anumit numar de randuri
  2. Pentru a numara randurile preluate intr-un loop si/sau pentru a determina cand se iese din loop

## **c) Atributul %NOTFOUND**

- Atributul **%NOTFOUND** se foloseste pentru:
  1. Pentru a determina daca interogarea a gasit randuri care se potrivesc criteriului
  2. Pentru a determina cand se face iesirea din loop

## Exemplu pentru **%ROWCOUNT** si **%NOTFOUND**

### DECLARE

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

```
v_emp_record emp_cursor%ROWTYPE;
```

### BEGIN

```
OPEN emp_cursor;
```

### LOOP

```
FETCH emp_cursor INTO v_emp_record;
```

```
EXIT WHEN emp_cursor%ROWCOUNT>10
```

```
OR emp_cursor%NOTFOUND;
```

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

```
END LOOP;
```

```
CLOSE emp_cursor;
```

```
END;
```

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables



```

DECLARE
    CURSOR emp_cursor IS
    SELECT empno, ename
    FROM emp;
    v_emp_record emp_cursor%ROWTYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_emp_record;
        EXIT WHEN emp_cursor%ROWCOUNT>10 OR emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_emp_record.empno||' '|| v_emp_record.ename );
    END LOOP;
    CLOSE emp_cursor;
END;

```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```

7839 KING
7698 BLAKE
7782 CLARK
7566 JONES
7788 SCOTT
7902 FORD
7369 SMITH
7499 ALLEN
7521 WARD
7654 MARTIN

```

Statement processed.

0.01 seconds

Atributele cursorilor expliciti nu pot fi folosite direct in instructiunile

**SQL.**

Urmatorul cod da eroare:

**DECLARE**

```
CURSOR emp_cursor IS  
SELECT emp, sal  
FROM em  
ORDER BY SAL DESC;  
v_emp_record emp_cursor%ROWTYPE;  
v_count NUMBER;  
BEGIN  
OPEN emp_cursor;  
LOOP  
    FETCH emp_cursor INTO v_emp_record;  
    EXIT WHEN emp_cursor%NOTFOUND;  
    INSERT INTO top_paid_emps (empno, rank, sal)  
    VALUES (v_emp_record.empno,  
emp_cursor%ROWCOUNT, v_emp_record.sal);  
END LOOP;  
END;    ...
```

Atributele cursorilor expliciti nu pot fi folosite direct in instructiunile SQL.  
 Urmatorul cod da eroare:

```

APEX App Builder SQL Workshop Team Development Gallery
SQL Commands
Language PL/SQL Rows 10 Clear Command Find Tables
DECLARE
    CURSOR emp_cursor IS
    SELECT emp, sal
    FROM em
    ORDER BY SAL DESC;
    v_emp_record emp_cursor%ROWTYPE;
    v_count NUMBER;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_emp_record;
        EXIT WHEN emp_cursor%NOTFOUND;
        INSERT INTO top_paid_emps (empno, rank, sal)
        VALUES (v_emp_record.empno, emp_cursor%ROWCOUNT, v_emp_record.sal);
    END LOOP;
END;
    
```

Results Explain Describe Saved SQL History

```

ORA-06550: line 14, column 42:
PL/SQL: ORA-00911: invalid character
ORA-06512: at "SYS.WWV_DBMS_SQL_APEX_210200", line 673
ORA-06550: line 13, column 4:
PL/SQL: SQL Statement ignored
ORA-06512: at "SYS.DBMS_SYS_SQL", line 1658
ORA-06512: at "SYS.WWV_DBMS_SQL_APEX_210200", line 659
ORA-06512: at "APEX_210200.WWV_FLOW_DYNAMIC_EXEC", line 1829

1. DECLARE
2.     CURSOR emp_cursor IS
3.     SELECT emp, sal
    
```

0.00 seconds



**Întrebări?**