Curs 5

Cursori în PL/SQL (partea I)

Cursori în PL/SQL

1. Structura repetitivă – instrucţiuni imbricate
2. Cursori expliciţi – introducere
3. Folosirea atributelor cursorilor expliciţi

1. Structura repetitivă – instrucţiuni imbricate

Instructiunile 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;
```

SQL Workshop > SQL Commands

Rows 10 Save Run

```
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 – instrucţiuni 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 instructiunea 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;
```

```
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
```

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 – instrucţiuni imbricate
2. Cursori expliciţi – introducere
3. Folosirea atributelor cursorilor expliciţi

2. Cursori expliciţi – introducere

- Se stie ca o instrucţiune SQL într-un bloc PL/SQL rulează cu succes dacă dă un singur rezultat.
- Dar dacă avem nevoie să scriem o instrucţiune SELECT care să dea mai multe rezultate?
- Dacă de exemplu avem de facut un raport cu toţi angajaţii?
- Pentru a obţine mai multe rezultate trebuie să declarăm şi să folosim un **cursor explicit**.

2. Cursori expliciţi – introducere

Cursorii şi domeniile lor de context

- Serverul Oracle aloca zone private de memorie numite **zone (domenii) de context** pentru a stoca datele prelucrate de o instructiune SQL.
- Fiecare zona de context (si prin urmare fiecare instructiune SQL) are un cursor asociat.
- Ne putem gandi la un cursor ca la o eticheta a zonei de context sau un pointer al zonei de context.
- *De fapt un cursor este si o eticheta si un pointer.*

2. Cursori expliciţi – introducere

Cursori impliciţi şi cursori expliciţi

- **Cursorii impliciţi** – sunt definiti automat de Oracle pentru toate instructiunile DML ale SQL (INSERT, UPDATE, DELETE si MERGE) si pentru *toate instructiunile SQL care returneaza un singur rand*
- **Cursorii expliciţi** – declaraţi de programator *pentru interogările care returnează mai mult de un rand.* Cursorii expliciţi se pot folosi pentru a denumi o zonă de context şi pentru a accesa datele stocate în ea.

Limitele cursorilor impliciţi

Fie urmatorul exemplu:

```
DECLARE
   v_salary employees.salary%TYPE;
BEGIN
   SELECT salary
   INTO v_salary
   FROM employees;
   DBMS_OUTPUT.PUT_LINE(' Salary is :
   '||v_salary);
END;
```

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 expliciţi – introducere

- Cu un **cursor explicit** putem *extrage randuri multiple din tabela*, avand un pointer catre fiecare rand extras si putem lucra cu un rand la un moment dat.

- Motivele pentru care folosim cursorii expliciti 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 instrucţiune de program separata, dând programatorului mai mult control în prelucrarea rândurilor

Exemplu de cursor explicit:

Cursorul se foloseste pentru numele tarilor si sarbatorile nationale pentru continentul Asia

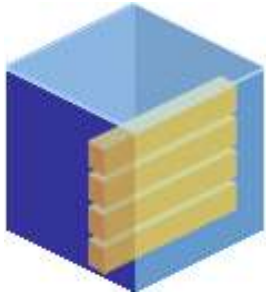
```
DECLARE
   CURSOR wf_holiday_cursor IS
   SELECT country_name, national_holiday_date
   FROM wf_countries where region_id IN(30,34,35);
   v_country_name wf_countries.country_name%TYPE;
   v_holiday wf_countries.national_holiday_date%TYPE;
BEGIN
   OPEN wf_holiday_cursor;
   LOOP
         FETCH wf_holiday_cursor INTO v_country_name,
   v_holiday;
         EXIT WHEN wf_holiday_cursor%NOTFOUND;
         DBMS_OUTPUT.PUT_LINE(v_country_name||'
   '||v_holiday);
   END LOOP;
   CLOSE wf_holiday_cursor;
END;
```

2. Cursori expliciţi – introducere

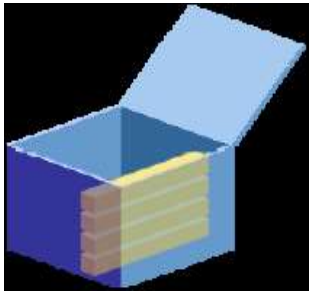
Operaţii – cursori expliciţi

- O mulţime de rânduri furnizate de o *interogare multiple-row* este denumită **mulţime activă** (**active set**) şi 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, conţinutul cutiei fiind mulţimea activă.
- Pentru a prelua date trebuie deschisă (OPEN) cutia şi sunt preluate (FETCH) rândurile din cutie, pe rând, câte unul.
- Când am terminat trebuie să închidem (CLOSE) cutia.

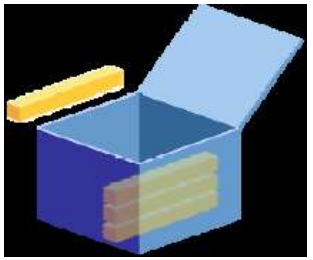
Cursor explicit:



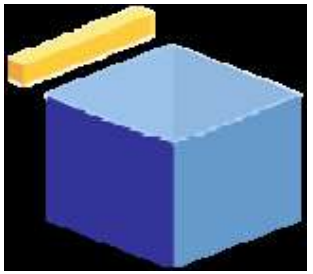
1. ***Deschiderea cursorului***



2. ***Preluarea a câte unui rând***



3. ***Închiderea cursorului***



Declararea şi controlul cursorilor expliciţi

- Paşii pentru utilizarea cursorilor expliciţi sunt:
  1. ***Declarare*** (**DECLARE**) – se face in sectiunea declarativa si se denumeste setul activ
  2. ***Deschidere*** (**OPEN**) – Instructiunea OPEN executa interogarea asociata cursorului, identifica setul de rezultate si pozitioneaza cursorul inainte de primul rand.
  3. ***Preluare*** (**FETCH**) – Instructiunea FETCH extrage randul curent si avanseaza cursorul la randul urmator.
  4. ***Închiderea*** (**CLOSE**) cursorului se face dupa ce a fost prelucrat ultimul rand. Instructiunea CLOSE dezactiveaza 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 ***employee_id*** si ***last_name*** pentru angajatii care lucreaza in departamentul pentru care ***department_id*** este 30.

```
DECLARE
  CURSOR emp_cursor IS
  SELECT employee_id, last_name
  FROM employees
  WHERE department_id =30;
...
```

1. Declararea unui cursor

Exemplu 2

Cursorul ***dept_cursor*** este declarat pentru a extrage toate informatiile pentru departamentele care au ***location_id*** 1700. Preluam si prelucram randurile alfabetic dupa ***department_name***.

```
DECLARE
    CURSOR dept_cursor IS
    SELECT *
    FROM departments
    WHERE location_id = 1700
    ORDER BY department_name;
    …
```

1. Declararea unui cursor

Exemplu 3

O instructiune SELECT in declararea unui cursor poate include join-uri, functii de grup si subinterogari.

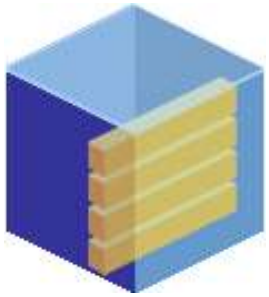
Acest exemplu extrage fiecare departament care are cel putin doi angajati furnizand numele departamentului si numarul de angajati.

```
DECLARE
        CURSOR dept_emp_cursor IS
        SELECT department_name, COUNT(*) AS
    how_many
        FROM departments d, employees e
        WHERE d.department_id = e.department_id
        GROUP BY d.department_name
        HAVING COUNT(*) > 1;
...
```

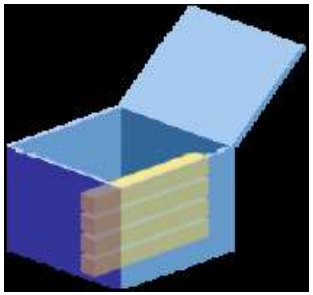
Reguli de declarare a unui cursor

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.

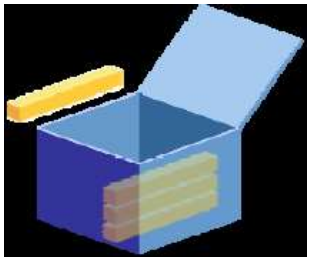
Cursor explicit:



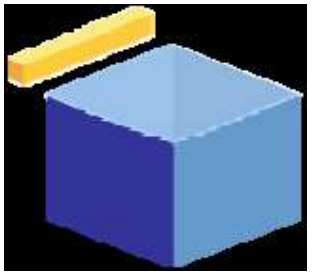
1. ***Deschiderea cursorului***



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.

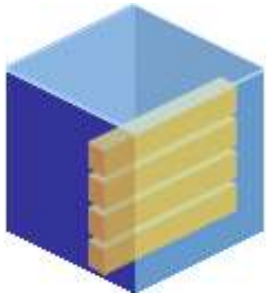
```
DECLARE
        CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM employees
        WHERE department_id =30;
        …
BEGIN
        OPEN emp_cursor;
…
```

2. Deschiderea unui cursor

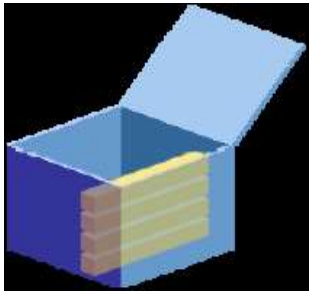
Instructiunea **OPEN** efectueaza urmatoarele 3 operatii:

1. Aloca memorie pentru zona de context (*creeaza "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"*)

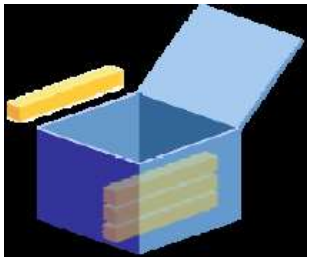
Cursor explicit:



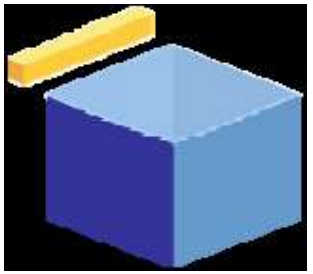
1. ***Deschiderea cursorului***



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 employee_id, last_name
        FROM employees
        WHERE department_id =10;
   v_empno employees.employee_id%TYPE;
   v_lname employees.last_name%TYPE;
BEGIN
        OPEN emp_cursor;
        FETCH emp_cursor INTO v_empno, v_lname;
        DBMS_OUTPUT.PUT_LINE(v_empno ||' '||v_lname);
        …
END;
```

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

```
DECLARE
        CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM employees
        WHERE department_id =50;
   v_empno employees.employee_id%TYPE;
   v_lname employees.last_name%TYPE;
BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor INTO v_empno,
   v_lname;
                EXIT WHEN emp_cursor%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE( v_empno ||'
   '||v_lname);
        END LOOP;
END;
```

```
DECLARE
                CURSOR emp_cursor IS
                SELECT employee_id, last_name FROM employees
                WHERE department_id =50;
        v_empno employees.employee_id%TYPE;
        v_lname employees.last_name%TYPE;
BEGIN
                OPEN emp_cursor;
                LOOP
                        FETCH emp_cursor INTO v_empno, v_lname;
                        EXIT WHEN emp_cursor%NOTFOUND;
                        DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_lname);
                END LOOP;
END;
```

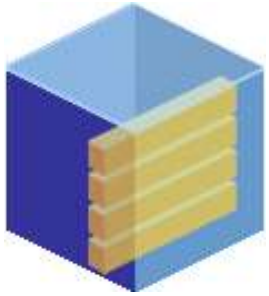
Results Explain Describe Saved SQL History

```
124 Mourgos
141 Rajs
142 Davies
143 Matos
144 Vargas

Statement processed.


0.55 seconds
```

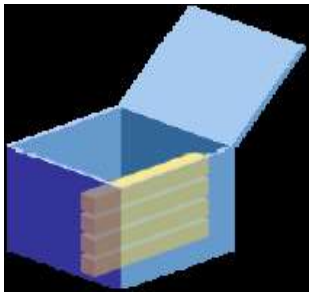
Reguli pentru preluarea datelor de la un cursor

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.

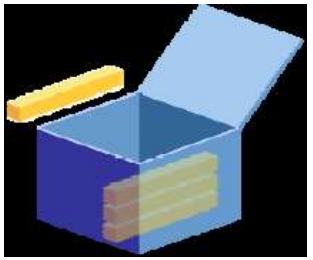
Cursor explicit:



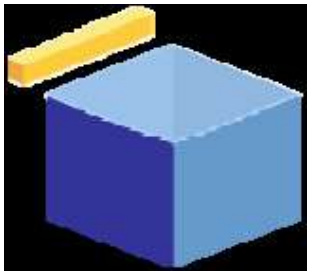
1. ***Deschiderea cursorului***



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_lname;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( v_empno ||'
  '||v_lname);
  END LOOP;
  CLOSE emp_cursor;
END;
```

Reguli pentru închiderea unui cursor

1. Un cursor poate fi redeschis numai daca este inchis.

2. Dacă încercaţi să preluati date dintr-un cursor dupa ce a fost inchis atunci apare o exceptie **INVALID_CURSOR**

3. Daca mai tarziu redeschidem cursorul atunci instructiunea SELECT asociata este executata din nou pentru a reumple zona de context cu cele mai recente date din baza de date.

Exemplu:

Urmatorul exemplu declara si prelucreaza un cursor pentru a obtine pentru toate tarile din Asia denumirea tarii si sarbatoarea nationala.

```
DECLARE
        CURSOR wf_holiday_cursor IS
        SELECT country_name, national_holiday_date
        FROM wf_countries where region_id IN(30,34,35);
   v_country_name wf_countries.country_name%TYPE;
   v_holiday wf_countries.national_holiday_date%TYPE;
BEGIN
        OPEN wf_holiday_cursor;
        LOOP
                FETCH wf_holiday_cursor INTO
   v_country_name, v_holiday;
                EXIT WHEN wf_holiday_cursor%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(v_country_name||'
   '||v_holiday);
        END LOOP;
        CLOSE wf_holiday_cursor;
END;
```

Cursori în PL/SQL

1. Structura repetitivă – instrucţiuni imbricate
2. Cursori expliciţi – introducere
3. Folosirea atributelor cursorilor expliciţi

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 informaţii cu privire la starea cursorului explicit.

3.1. Cursori şi înregistrări

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

```
DECLARE
        v_emp_id   employees.employee_id%TYPE;
        v_last_name   employees.last_name%TYPE;
        CURSOR emp_cursor IS
        SELECT employee_id, last_name
        FROM employees
        WHERE department_id =30;
BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor INTO v_emp_id,
  v_last_name;
        …
```

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

Urmatorul cursor extrage in intregime randurile din tabela angajati:

```
DECLARE
        v_emp_id employees.employee_id%TYPE;
        v_first_name employees.first_name%TYPE;
        v_last_name employees.last_name%TYPE;
        …
        v_department_id employees.department_id%TYPE;
        CURSOR emp_cursor IS
        SELECT * FROM employees
        WHERE department_id =30;
BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor
                INTO v_emp_id, v_first_name, v_last_name
...v_department_id;
…
```

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

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 employees
        WHERE department_id =30;
        v_emp_record  emp_cursor%ROWTYPE;
BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor  INTO
   v_emp_record;
        …
```

- Acest cod foloseste **%ROWTYPE** pentru a declara o *structura de tip inregistrare bazata pe cursor*.
- *O înregistrare este un tip de date compus in PL/SQL.*
- O inregistrare este formata din mai multe campuri, fiecare cu propriul nume si tip de date.
- Putem referi câmpurile prefixându-le denumirile de caracterul punct şi denumirea înregistrării.
- **%ROWTYPE** declară o înregistrare cu aceleaşi câmpuri ca şi cursorul pe care se bazează.

Exemplu:

```
DECLARE
   CURSOR emp_cursor IS
   SELECT employee_id, last_name, salary
   FROM employees
   WHERE department_id =30;
   v_emp_record emp_cursor%ROWTYPE;
...
```

v_emp_record.employee_id	v_emp_record.last_name	v_emp_record.salary
100	King	24000

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

- Exemple

```
DECLARE
        CURSOR emp_cursor IS
        SELECT * FROM employees
        WHERE department_id =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.employee_id||'-'
   || v_emp_record.last_name);
        END LOOP;
        CLOSE emp_cursor;
END;
```

Apelul fiecarui camp al inregistrarii prin specificarea **denumirii campului** precedat de constructia **.nume_inregistrare**

```
DECLARE
        CURSOR emp_dept_cursor IS
        SELECT first_name, last_name, department_name
        FROM employees e, departments d
        WHERE e.department_id = d.department_id;
   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.first_name
||' – '|| v_emp_dept_record.last_name ||' – '||
   v_emp_dept_record.department_name);
        END LOOP;
        CLOSE emp_dept_cursor;
END;
```

```
DECLARE
                CURSOR emp_dept_cursor IS
                SELECT first_name, last_name, department_name
                FROM employees e, departments d
                WHERE e.department_id = d.department_id;
        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.first_name
||' - '||v_emp_dept_record.last_name ||' - '||v_emp_dept_record.department_name);
                END LOOP;
                CLOSE emp_dept_cursor;
END;
```

Results Explain Describe Saved SQL History

```
Jennifer - Whalen - Administration
Michael - Hartstein - Marketing
Pat - Fay - Marketing
Curtis - Davies - Shipping
Peter - Vargas - Shipping
Trenna - Rajs - Shipping
Kevin - Mourgos - Shipping
Randall - Matos - Shipping
Alexander - Hunold - IT
Bruce - Ernst - IT
Diana - Lorentz - IT
Jonathon - Taylor - Sales
Eleni - Zlotkey - Sales
Ellen - Abel - Sales
Lex - De Haan - Executive
Steven - King - Executive
Neena - Kochhar - Executive
Shelley - Higgins - Accounting
William - Gietz - Accounting
```

3.2. Atributele cursorilor expliciţi

- Ca şi pentru cursorii impliciţi, sunt câteva atribute utile pentru a obţine informaţii cu privire la starea cursorilor expliciţi.

- Aceste atribute ne dau informaţii utile cu privire la execuţia unei instrucţiuni de manipulare a cursorului.

3.2. Atributele cursorilor expliciţi

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch did not return a row
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returned a row; opposite of %NOTFOUND
%ROWCOUNT	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 employee_id, last_name
        FROM employees;
   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.employ
   ee_id||' '|| v_emp_record.last_name );
        END LOOP;
   CLOSE emp_cursor;
END;
```

```
DECLARE
                CURSOR emp_cursor IS
                SELECT employee_id, last_name FROM employees;
        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.employee_id||' '|| v_emp_record.last_name );
                END LOOP;
        CLOSE emp_cursor;
END;
```

Results Explain Describe Saved SQL History

```
100 King
101 Kochhar
102 De Haan
200 Whalen
205 Higgins
206 Gietz
149 Zlotkey
174 Abel
176 Taylor
178 Grant

Statement processed.


0.02 seconds
```

Atributele cursorilor expliciti nu pot fi folosite direct in instructiunile SQL.

Urmatorul cod da eroare:

```
DECLARE
        CURSOR emp_cursor IS
        SELECT employee_id, salary
        FROM employees
        ORDER BY SALARY 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 (employee_id,
   rank, salary)
                VALUES (v_emp_record.employee_id,
   emp_cursor%ROWCOUNT, v_emp_record.salary);
```

Întrebări?