PL/SQL

Lecture 4 :
Cursors and parametrized cursors

Cursors (01/13)

- Why ?
  - Reminder : you can initialize PL/SQL variables by fetching database values (SELECT ... INTO)

```
SELECT field_1, field_2, ... , field_n
INTO variable_1, variable_2, ... , variable_n
FROM ...
```

  - Problem : you cannot use SELECT ... INTO if it returns several lines (TOO_MANY_ROWS)
  - Actually, SELECT...INTO is too rigid
  - What if we could manipulate a whole SQL request into PL/SQL ?

Cursors (02/13)

- What ?
  - A memory zone
  - Fixed size
  - Used by Oracle in order to analyze and interpret any SQL order
- What for ?
  - It is used to treat several lines of a SQL request into PL/SQL
  - Therefore, it provides a better interaction between PL/SQL and SQL

Cursors (03/13)

- Two types of cursor : explicit and implicit
- Explicit cursors
  - Created and managed by the programmer
  - Associated with SELECT statements
  - Defined in the DECLARE section
- Working with explicit cursors involves 4 steps
  - **Declaring** the cursor for initializing the memory
  - **Opening** the cursor for allocating memory
  - **Fetching** the cursor for retrieving data
  - **Closing** the cursor to release allocated memory

Cursors (04/13)

- Declaring the cursor
  - Defines the cursor with a name and the associated SELECT statement.
  - Syntax

```
CURSOR cursor_name IS select_statement;
```

  - Example

```
DECLARE
    CURSOR dpt_10 IS
        SELECT ename, sal FROM emp WHERE deptno=10 order by sal ;
    ...
BEGIN
    ...
END;
```

Cursors (05/13)

- Opening the cursor
  - Allocates memory for the cursor
  - Makes it ready for fetching the rows returned by the SQL statement into it.
  - Syntax

```
OPEN cursor_name;
```

  - Example

```
BEGIN
    OPEN dpt_10;
    ...
END;
```

Cursors (06/13)

- Fetching the cursor
  - Once opened, the cursor contains all the lines of the requested result
  - They are collected, or "fetched", one by one
  - Syntax : FETCH cursor_name INTO variables
  - Example

```
DECLARE ...
    nom emp.ename%TYPE ;
    salaire emp.sal%TYPE ;
BEGIN ...
    LOOP
        FETCH dept_10 INTO nom, salaire ; ...
    END LOOP ; ...
END;
```

Cursors (07/13)

- Closing the cursor
  - Releases the allocated memory (don't forget it !)
  - Usually done before the END of the (sub-)program
  - Syntax

```
CLOSE cursor_name;
```

  - Example

```
BEGIN
    LOOP
        ...
    END LOOP ;
    CLOSE dpt_10 ;
END;
```

Cursors (08/13)

- Explicit cursor : full example

```
DECLARE
    CURSOR dpt_10 IS
        SELECT ename, sal FROM emp WHERE deptno=10 order by sal ;
    name emp.ename%TYPE ;
    salary emp.sal%TYPE ;
BEGIN
    OPEN dpt_10 ;
    LOOP
        FETCH dept_10 INTO name, salary ;
        IF salary > 2500 THEN
            insert into result values (name, salary) ;
        END IF;
        EXIT WHEN salary = 5000 ;
    END LOOP ;
CLOSE dpt_10 ;
END;
/
```

Cursors (09/13)

- Implicit cursors
  - Created and managed by Oracle
  - Used whenever a SQL statement is executed, when there no explicit cursor for the statement
  - Associated with INSERT, UPDATE and DELETE
- Cursor attributes
  - Used to check the state of a cursor
  - Syntax
    - Explicit attributes : cursor_name%attribute_name
    - Implicit attributes : SQL%attribute_name

Cursors (10/13)

- Most used pre-defined attributes
  - %FOUND : the statement produced some lines
  - %NOTFOUND : the statement produced no line
  - %ISOPEN : the cursor is opened
    - Always false in case of an implicit cursor
  - %ROWCOUNT : the number of lines produced
- Explicit cursors and FOR loop
  - There is a FOR loop that does it all : opening, fetching and closing the cursor
  - Syntax

FOR variable IN cursor_name LOOP ... END LOOP;

Cursors (11/13)

- Explicit cursors : Full Example 2

```
DECLARE
    CURSOR emp_cur IS
        SELECT * FROM EMP;
    line emp_cur%rowtype;
BEGIN
    OPEN emp_cur;
    LOOP
        FETCH emp_cur INTO line;
        EXIT WHEN emp_cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(line.ename);
    END LOOP;
    CLOSE emp_cur;
END;
/
```

Cursors (12/13)

- Explicit cursors : Full example 2 (version 2)

```
DECLARE
    CURSOR emp_cur IS
        SELECT * FROM EMP;
    line emp_cur%rowtype;
BEGIN
    OPEN emp_cur;
    FETCH emp_cur INTO line;
    WHILE emp_cur%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(line.ename);
        FETCH emp_cur INTO line;
    END LOOP;
    CLOSE emp_cur;
END;
/
```

Cursors (13/13)

- Explicit cursors : Full example 2 (version 3)

```
DECLARE
    CURSOR emp_cur IS
        SELECT * FROM EMP;
BEGIN
    FOR line IN emp_cur LOOP
        DBMS_OUTPUT.PUT_LINE(line.ename);
    END LOOP;
END;
/
```

Parametrized cursors (01/06)

- Question : is this code valid ?

```
DECLARE
    n NUMBER := 14;
BEGIN
    DECLARE
        CURSOR C IS
            SELECT * FROM PERSON WHERE numpers >= n;
        ROW C%rowType;
    BEGIN
        FOR ROW IN C LOOP
            DBMS_OUTPUT.PUT_LINE(ROW.numpers);
        END LOOP;
    END;
END;
/
```

Parametrized cursors (02/06)

- Answer : no !
  - A cursor request cannot contain variables whose values are't fixed

    n NUMBER

    ...

    ... WHERE numpers >= n !! Prohibited !!
  - Why ? Because these values can change from the moment the cursor is declared to the moment the cursor is opened
- Solution : add parameters to cursors

Parametrized cursors (03/06)

- What are they ?
  - Explicit cursors whose request contain variables whos values will be fixed only in the opening
- Declaration
  - After the cursor's name, give the list of parameter names and types (just like a function)
  - Syntax

```
CURSOR cursor_name(param1 : type 1, param2 : type2...) IS
```

  - Example

```
CURSOR children (numparent NUMBER) IS
    SELECT * FROM PERSON WHERE father = numparent OR mother = numparent;
```

Parametrized cursors (04/06)

- Opening
  - Give the name of the cursor, then pass in parameter the values of the variables (just like a function or a procedure)
  - Syntax

    ```
    OPEN cursor_name(param_value1, param_value2,...);
    ```

  - Example

    ```
    OPEN children(1);
    ```

- Fetching and closing
  - Same rules as "normal" explicit cursors

Parametrized cursors (05/06)

- FOR loop
  - Pass the parameter values after the cursor name
  - Syntax

```
FOR variable_name IN cursor_name (param_value1,
param_value2, ...) LOOP
    ... instructions ...
END LOOP;
```

  - Example

```
FOR e IN children(1) LOOP
    DBMS_OUTPUT.PUT_LINE(e.namepers || ' ' || e.firstnamepers);
END LOOP;
```

Parametrized cursors (06/06)

- Full example

```
DECLARE
    CURSOR parent IS
        SELECT * FROM PERSON;
    p parent%rowtype;
    CURSOR children (numparent NUMBER) IS
        SELECT * FROM PERSON WHERE father = numparent OR mother =
        numparent; e children%rowtype;
BEGIN
    FOR p IN parent LOOP
        DBMS_OUTPUT.PUT_LINE('Children of '|| p.firstname || ' ' || p.lastname
        || ' are : ');
        FOR e IN children(p.numpers) LOOP
            DBMS_OUTPUT.PUT_LINE(' * '|| e.firstname || ' ' || e.lastname );
        END LOOP;
    END LOOP;
END;
/
```