Advanced PLSQ

Lesson – REF Cursors

Lesson Objectives

On completion of this lesson on REF Cursors, you will be able to:

- State the need for REF cursors
- Understand the ways of implementing REF cursors
- Understand the use of the Strong and Weak REF cusors
- Understand the differences between Static Cursors and REF Cursors

What is a REF Cursor

- REF Cursors/Dynamic Cursors/Cursor Variables

  - REF Cursors, also known a Dynamic Cursors or Cursor variables are like C or Pascal pointers, which hold the memory location (address) of some item instead of the item itself.

  - So, declaring a cursor variable creates a pointer or reference to a work-area

  - In PL/SQL, a pointer has datatype REF X, where REF is short-form for REFERENCE and X stands for a class of objects. Therefore, a cursor variable has datatype REF CURSOR.

How are Cursors different from REF cursors

- Difference between Cursors/Static cursors and REF cursors/Dynamic Cursors

  - Like a cursor, a cursor variable points to the current row in the result set of a multi-row query. But, cursors differ from cursor variables the way constants differ from variables. Whereas a cursor is static, a cursor variable is dynamic because it is not tied to a specific query. You can open a cursor variable for any type-compatible query. This gives you more flexibility.

  - Also, you can assign new values to a cursor variable and pass it as a parameter to local and stored subprograms. This gives you an easy way to centralize data retrieval, that is, pass record or result sets as parameters.

How to define a REF cursor type and a Cursor Variable

- Steps :
  - First, you define a REF CURSOR type :

```
TYPE ref_type_name IS REF CURSOR [RETURN return_type];
```

where ref_type_name is a type specifier used in subsequent declarationsof cursor variables and return_type must represent a record or a row in a database table

  - Then, you define a Cursor Variable of the REF CURSOR type :

```
cv ref_type_name;
```

where cv is the cursor variable which would be later bound with type-specific query

1.4: Advantages of a Cursor Variable

Dynamically deciding the query

- You want more flexibility. For example, you might want to **defer your choice of schema objects until run time.** Or, you might want your program to build **different search conditions for the WHERE clause** of a SELECT statement. A more complex program might choose from various SQL operations, clauses, etc.

- You want better performance as compared to DBMS_SQL, something easier to use(Native Dynamic SQL). This is specifically an advantage of Native Dynamic SQL(using EXECUTE IMMEDIATE command)

STRONG and WEAK REF Cursor type

- REF CURSOR types can be ***strong*** **(restrictive)** or ***weak*** **(nonrestrictive)**. A strong REF CURSOR type definition specifies a return type, but a weak definition does not:

- Declaring a STRONG REF CURSOR type :

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

- Declaring a WEAK REF CURSOR type :

```
TYPE GenericCurTyp IS REF CURSOR;
```

STRONG and WEAK REF Cursor type ….contd

- Strong REF CURSOR types are less error prone because the PL/SQL compiler lets you associate a strongly typed cursor variable only with type-compatible queries.

- However, weak REF CURSOR types are more flexible because the compiler lets you associate a weakly typed cursor variable with any query.

STRONG REF Cursor type : Example 1

```
declare
        type my_emp_type is ref cursor return emp%rowtype;
        v_emp my_emp_type;
        v_edata emp%rowtype;
begin
        open v_emp for select * from emp;
        loop
                fetch v_emp into v_edata;
                exit when v_emp%notfound;
                dbms_output.put_line(v_edata.empno||' '||v_edata.ename||' '||v_edata.sal);
        end loop;
        close v_emp;
end;
/
```

…..contd

STRONG REF Cursor type : Example 2

```
declare
        type mytype is record(gnumber number, gname varchar2(50), gdesc varchar2(50));
        type my_gen_type is ref cursor return mytype;
        v_c_gen my_gen_type;
        vdata mytype;
begin
        open v_c_gen for select empno, ename, job from emp;
        loop
                fetch v_c_gen into vdata;
                exit when v_c_gen%notfound;
                dbms_output.put_line(vdata.gnumber||' '||vdata.gname||' '||vdata.gdesc);
        end loop;
        close v_c_gen;
        dbms_output.put_line('**********************************');
        open v_c_gen for select deptno, dname, loc from dept;
        loop
                fetch v_c_gen into vdata;
                exit when v_c_gen%notfound;
                dbms_output.put_line(vdata.gnumber||' '||vdata.gname||' '||vdata.gdesc);
        end loop;
        close v_c_gen;
end;
/
```

WEAK REF Cursor type : Example 1

```
DECLARE
    TYPE gen_cur IS REF CURSOR;
    v_weak_cursor gen_cur;
    v_emp_rec EMP%ROWTYPE;
    v_dept_rec DEPT%ROWTYPE;
    v_operation NUMBER(1) := &op;
BEGIN
    IF v_operation = 1 THEN
                OPEN v_weak_cursor FOR SELECT * FROM emp;
                LOOP
                            FETCH v_weak_cursor INTO v_emp_rec;
                            EXIT WHEN v_weak_cursor%NOTFOUND;
                            DBMS_OUTPUT.PUT_LINE('Ename :'||v_emp_rec.ename);
                END LOOP;
                CLOSE v_weak_cursor;
    ELSE
                OPEN v_weak_cursor FOR SELECT * FROM dept;
                LOOP
                            FETCH v_weak_cursor INTO v_dept_rec;
                            EXIT WHEN v_weak_cursor%NOTFOUND;
                            DBMS_OUTPUT.PUT_LINE('Deptname :'||v_dept_rec.dname);
                END LOOP;
                CLOSE v_weak_cursor;
    END IF;
END;
/
```

1.8: Cursor Variables as Parameters

Cursor variables can be passed a parameters

- Procedure which accepts a cursor variable as a parameter :

```
create or replace procedure p1(x in sys_refcursor) is
        erec emp%rowtype;
begin
        loop
                fetch x into erec;
                exit when x%notfound;
                dbms_output.put_line(erec.empno||erec.ename);
        end loop;
end;
/
```

Cursor variables can be passed a parameters ……contd

Anonymous block calling the Procedure which accepts a cursor variable as a parameter :

```
declare
		xx sys_refcursor;
begin
		open xx for select * from emp;
		p1(xx);
end;
/
```

Cursor variables cannot be compared like other scalar variables

Anonymous blocks to compare cursor variables :

```
declare
		v_weak_cursor sys_refcursor;
		v_weak_cursor1 sys_refcursor;
begin
		OPEN v_weak_cursor FOR SELECT * FROM emp;
		OPEN v_weak_cursor1 FOR SELECT * FROM emp;
		if v_weak_cursor=v_weak_cursor1 then
				dbms_output.put_line('are same');
		end if;
end;
/
ERROR at line 7:
ORA-06550: line 7, column 21:
PLS-00306: wrong number or types of arguments in call to '='
ORA-06550: line 7, column 5:
PL/SQL: Statement ignored
```

Cursor variables cannot be compared like other scalar variables

Cursor variables can be compared for NULL values :

```
declare
            v_weak_cursor sys_refcursor;
            v_weak_cursor1 sys_refcursor;
begin
            OPEN v_weak_cursor FOR SELECT * FROM emp;
            if v_weak_cursor is null then
                        dbms_output.put_line('is null');
            end if;
            if v_weak_cursor is not null then
                        dbms_output.put_line('is not null');
            end if;

            if v_weak_cursor1 is null then
                        dbms_output.put_line('is null');
            end if;
            if v_weak_cursor1 is not null then
                        dbms_output.put_line('is not null');
            end if;
end;
/
```

CURSOR FOR LOOP cannot be used for cursor variables

Unlike Static Cursors, Cursor variables are not compatible with CURSOR FOR LOOP :

```
declare
     v_weak_cursor sys_refcursor;
     emprec emp%rowtype;
BEGIN
     OPEN v_weak_cursor FOR SELECT * FROM emp;
     if v_weak_cursor is null then
                        dbms_output.put_line('is null');
     end if;
     if v_weak_cursor is not null then
                        dbms_output.put_line('is not null');
     end if;
     --for erec in v_weak_cusror loop
                        -- dbms_output.put_line(erec.ename);
     --end loop;
     LOOP
                        FETCH v_weak_cursor INTO emprec;
                        EXIT WHEN v_weak_cursor%NOTFOUND;
                        DBMS_OUTPUT.PUT_LINE('Ename :'||emprec.ename);
     END LOOP;
     v_weak_cursor:=null;
     if v_weak_cursor is null then
                        dbms_output.put_line('is null');
     end if;
     if v_weak_cursor is not null then
                        dbms_output.put_line('is not null');
     end if;
END;
/
```

Cannot fetch from a non-open cursor variable

We cannot fetch from a cursor-variable which is not associated with a query or which is null :

```
declare
		v_weak_cursor sys_refcursor;
		emprec emp%rowtype;
begin
  LOOP
  FETCH v_weak_cursor INTO emprec;
				EXIT WHEN v_weak_cursor%NOTFOUND;
				DBMS_OUTPUT.PUT_LINE('Ename :'||emprec.ename);
  END LOOP;
END;
/
ERROR at line 1:
ORA-01001: invalid cursor
ORA-06512: at line 6
```

You cannot declare cursor variables in a package

Unlike packaged variables, cursor variables do not have persistent state. Remember, declaring a cursor variable creates a pointer, not an item :

```
Create or replace package mypack is
        TYPE DeptCurTyp IS REF CURSOR RETURN dept%ROWTYPE;
        dept_cv DeptCurTyp; -- declare cursor variable
end;
/

Show errors
Errors for PACKAGE MYPACK:

LINE/COL        ERROR
--------        -----------------------------------------------------------------
3/10            PL/SQL: Declaration ignored
3/10            PLS-00994: Cursor Variables cannot be declared as part of a package
```