**Tutorial Week 9-10**

**PL/SQL Advance**

Today we are going to do some simple exercises involving PL/SQL. It is a programming language for Oracle Database that allows you to do much more complex things than SQL itself.

We are going to use HR database.

**Exercise 1**

Create and execute a simple anonymous block that outputs “Hello World”.

SET SERVEROUTPUT ON

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘Hello World’)

END;

**Exercise 2**

Create a PL/SQL block that selects the maximum department ID in the Department­ table and stores it in a variable. Display the maximum department ID. Steps involved in that process:

1. Declare a variable of type NUMBER in the declarative section.
2. Start the executable section with the BEGIN keyword and include a SELECT statement to retrieve the maximum department ID¨
3. Display the variable and end the executable block.

Modify the above PL/SQL block to insert a new department (Name=’EDUCATION’) into the departments table. Use SQL%ROWCOUNT to display the number of rows that are affected. Execute a SELECT statement to check whether new department is inserted or not. Include a DELETE statement to delete the department that you added.

SET SERVEROUTPUT ON

DECLARE

v\_dept\_name departments.department\_name%TYPE:= 'Education';

v\_dept\_id NUMBER;

v\_max\_deptno NUMBER;

BEGIN

SELECT MAX(department\_id) INTO v\_max\_deptno FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is : ' ||

v\_max\_deptno);

v\_dept\_id := 10 + v\_max\_deptno;

INSERT INTO departments (department\_id, department\_name, location\_id)

VALUES (v\_dept\_id,v\_dept\_name, NULL);

DBMS\_OUTPUT.PUT\_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);

END;

/

SELECT \* FROM departments WHERE department\_id=280;

**Exercise 3 (Loops and IF/ELSE statements)**

Create a message table using following command:

DROP TABLE messages;

CREATE TABLE messages (results VARCHAR2(80));

Write a PL/SQL block to insert numbers in messages table from 1 to 10 excluding 6. Commit before end of the block

BEGIN

FOR i in 1..10 LOOP

IF i = 6 THEN

null;

ELSE

INSERT INTO messages(results)

VALUES (i);

END IF;

END LOOP;

COMMIT;

END;

/

SELECT \* FROM messages;

**Exercise 4 (Loops and IF/ELSE statements)**

Execute following commands to create an emp table.

DROP TABLE emp;

CREATE TABLE emp AS SELECT \* FROM employees;

ALTER TABLE emp ADD stars VARCHAR2(50);

Create a PL/SQL block that inserts an asterisk in the stars column for every column for every $1000 of an employee’s salary. You can use default value for employee number as 176.

SET VERIFY OFF

DECLARE

v\_empno emp.employee\_id%TYPE := 176;

v\_asterisk emp.stars%TYPE := NULL;

v\_sal emp.salary%TYPE;

BEGIN

SELECT NVL(ROUND(salary/1000), 0) INTO v\_sal

FROM emp WHERE employee\_id = v\_empno;

FOR i IN 1..v\_sal

LOOP

v\_asterisk := v\_asterisk ||'\*';

END LOOP;

UPDATE emp SET stars = v\_asterisk

WHERE employee\_id = v\_empno;

COMMIT;

END;

/

SELECT employee\_id,salary, stars

FROM emp WHERE employee\_id =176;

**Exercise 5 (Cursors)**

In this exercise,

* First, you use an explicit cursor to process a number of rows from a table and

populate another table with the results using a cursor FOR loop.

* Second, you write a PL/SQL block that processes information with two cursors,

including one that uses a parameter.

Create a PL/SQL block to perform following:

1. Declare a cursor which retrieves the last\_name, salary, and manager\_id of employees working in the department specified (deptno is a varialble)
2. In the executable section, use the cursor for loop to operate on the data retrieved. If the salary of the employee is less than 5,000 and if the manager ID is either 101 or 124, display the message “<<last\_name>> Due for a raise”. Otherwise, display the message “<<last\_name>> Not Due for a raise.”
3. Test your PL/SQL block for the following Department IDs: 10,20,50,80

Now modify above PL/SQL block, to write two cursors-one without a parameter and one with a parameter. The first cursor retrieves the department number and the department name from the Department table for all departments whose ID number is less than 100. The second cursor receives the department number as a parameter, and retrieves employee details for those who work in that department and whose employee\_id is less than 120. You need to declare variables to hold the values retrieved from each cursor. Use the %Type attribute while declaring variables.

SET SERVEROUTPUT ON

SET VERIFY OFF

SET ECHO OFF

DECLARE

v\_deptno NUMBER := 10;

CURSOR c\_emp\_cursor IS

SELECT last\_name, salary,manager\_id

FROM employees

WHERE department\_id = v\_deptno;

BEGIN

FOR emp\_record IN c\_emp\_cursor

LOOP

IF emp\_record.salary< 5000 AND (emp\_record.manager\_id=101 OR emp\_record.manager\_id=124) THEN

DBMS\_OUTPUT.PUT\_LINE (emp\_record.last\_name || ' Due for a raise');

ELSE

DBMS\_OUTPUT.PUT\_LINE (emp\_record.last\_name || ' Not Due for a raise');

END IF;

END LOOP;

END;

SET SERVEROUTPUT ON

DECLARE

CURSOR c\_dept\_cursor IS

SELECT department\_id,department\_name

FROM departments

WHERE department\_id< 100

ORDER BY department\_id;

CURSOR c\_emp\_cursor(v\_deptno NUMBER) IS

SELECT last\_name,job\_id,hire\_date,salary

FROM employees

WHERE department\_id = v\_deptno

AND employee\_id< 120;

v\_current\_deptno departments.department\_id%TYPE;

v\_current\_dname departments.department\_name%TYPE;

v\_ename employees.last\_name%TYPE;

v\_job employees.job\_id%TYPE;

v\_hiredate employees.hire\_date%TYPE;

v\_sal employees.salary%TYPE;

BEGIN

OPEN c\_dept\_cursor;

LOOP

FETCH c\_dept\_cursor INTO v\_current\_deptno, v\_current\_dname;

EXIT WHEN c\_dept\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE ('Department Number : ' || v\_current\_deptno || ' Department Name : ' || v\_current\_dname);

IF c\_emp\_cursor%ISOPEN THEN

CLOSE c\_emp\_cursor;

END IF;

OPEN c\_emp\_cursor (v\_current\_deptno);

LOOP

FETCH c\_emp\_cursor INTO v\_ename,v\_job,v\_hiredate,v\_sal;

EXIT WHEN c\_emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE (v\_ename || ' ' || v\_job || ' ' || v\_hiredate || ' ' || v\_sal);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------------------------------------');

CLOSE c\_emp\_cursor;

END LOOP;

CLOSE c\_dept\_cursor;

END;

**Exercise 6 (Exception)**

In this practice, you write a PL/SQL block that applies a predefined exception in order to process only one record at a time. The PL/SQL block selects the name of the employee with a given salary value.

1. Execute following command to re-create the message table

DROP TABLE messages;

CREATE TABLE messages (results VARCHAR2(80));

1. In the declarative section, declare two variables: v\_ename of type employees.last\_name and emp\_sal of type employees.salary. Initialize the latter to 6000.
2. In the executable section, retrieve the last names of employees whose salaries are equal to the value emp\_sal. If the salary entered returns only one row, insert into the message table the employee’s name and the salary amount.

Note: Do not use explicit cursors.

1. If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert into the messages table the message “No employee with a salary of <salary>”.
2. f the salary entered returns multiple rows, handle the exception with an appropriate exception handler and insert into the messages table the message More than one employee with a salary of <salary>.
3. Handle any other exception with an appropriate exception handler and insert into the messages table the message Some other error occurred.
4. Display the rows from the messages table to check whether the PL/SQL block has executed successfully.
5. Change the initialized value of emp\_salto 2000 and re-execute.

SET VERIFY OFF

DECLARE

v\_ename employees.last\_name%TYPE;

v\_emp\_sal employees.salary%TYPE := 6000;

BEGIN

SELECT last\_name

INTO v\_ename

FROM employees

WHERE salary = v\_emp\_sal;

INSERT INTO messages (results)

VALUES (v\_ename || ' - ' || v\_emp\_sal);

EXCEPTION

WHEN no\_data\_found THEN

INSERT INTO messages (results)

VALUES ('No employee with a salary of '|| TO\_CHAR(v\_emp\_sal));

WHEN too\_many\_rows THEN

INSERT INTO messages (results)

VALUES ('More than one employee with a salary of '||

TO\_CHAR(v\_emp\_sal));

WHEN others THEN

INSERT INTO messages (results)

VALUES ('Some other error occurred.');

END;

/

SELECT \* FROM messages;

**Exercise 7 (Trigger)**

We would like to create a trigger on insert and update operation on employee table such that the employees’ salary should remain within a range according to their job type. Please follow these steps for that:

1. Create a trigger on employees table that fires before insert or update operation on each row. Trigger must should check the salary whether it is between the minimum and maximum range for salary for a specified job. If job salary does not fall into the range, then raise an error. To raise an application error, you can use following statement:

RAISE\_APPLICATION\_ERROR(-20100, “error in the salary range);

CREATE OR REPLACE TRIGGER check\_salary\_trg

BEFORE INSERT OR UPDATE OF job\_id, salary

ON employees

FOR EACH ROW

Declare

v\_minsal jobs.min\_salary%type;

v\_maxsal jobs.max\_salary%type;

BEGIN

SELECT min\_salary, max\_salary INTO v\_minsal, v\_maxsal

FROM jobs

WHERE job\_id = UPPER(:new.job\_id);

IF :new.salary NOT BETWEEN v\_minsal AND v\_maxsal THEN

RAISE\_APPLICATION\_ERROR(-20100,

'Invalid salary $' ||:new.salary ||'. '||

'Salaries for job '|| :new.job\_id ||

' must be between $'|| v\_minsal ||' and $' || v\_maxsal);

END IF;

END;

Testing the trigger:

* 1. Now try by updating the salary of employee 115 to 2000. What is the result
  2. Now update the salary of employee 115 to 2800. What is the result?

Now we want to write a trigger that prevent rows from being deleted during business hours i.e. Monday to Friday (9AM to 6PM). To get the hours from current date you use following function: TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24'));

CREATE OR REPLACE TRIGGER delete\_emp\_trg

BEFORE DELETE ON employees

DECLARE

the\_day VARCHAR2(3) := TO\_CHAR(SYSDATE, 'DY');

the\_hour PLS\_INTEGER := TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24'));

BEGIN

IF (the\_hour BETWEEN 9 AND 18) AND (the\_day NOT IN ('SAT','SUN')) THEN

RAISE\_APPLICATION\_ERROR(-20150,

'Employee records cannot be deleted during the business hours of 9AM and 6PM');

END IF;

END;

/

SHOW ERRORS

DELETE FROM employees

WHERE job\_id = 'SA\_REP'

AND department\_id IS NULL;