CHAPTER 7 (Advanced Explicit Cursor)

In this chapter we deal with the next level of explicit cursor, it's the cursor which uses parameters to pass data values from the cursor to the block itself.

The chapter also deals with a cursor that is created to prevent double updates of records, that means when this cursor is in execution it prevents other program from accessing records that are in a queue to be updated.

1. **CURSOR with parameters**.

   Cursor parameters are variables declared with datatypes but there is no need to declare the data type length unless in the case were the datatype is retrieved for the relevant tables using %TYPE. The cursor can have more than one input parameters.

   The syntax:

   CURSOR cursor_name (paremeter_name datatype, ….)
   IS
   SELECT statement.

   The parameters use prefix p_ is the first letter of the declared variable.

   Examples:

   Lets create a parameter cursor that passes the department_no and job_id to the SELECT statement so that it must retrieved records based on employee_id, first_name, last_name, job_id,salary and department name for all employees whose salary is less than R15000.

   - Pass parameter values to a cursor when a cursor is opened and the query (SELECT statement) is executed.
   - Open and explicit cursor several times with a different active set of values each time of execution.

   OPEN cursor_name (parameter_values…….) remember to enclose string values inside single quotes.

   The cursor values can be fetched into a declared record or into declared local variables that are in turn used to display those values. In the above example I will use both type of declarations.

   ```
   DECLARE
    CURSOR emp_salary_cur(p_deptno NUMBER, p_job VARCHAR2)
    IS
      SELECT employee_id,first_name||' '||last_name names,job_id,salary, department_name
      FROM employees e,departments d
      WHERE d.department_id=e.department_id
      AND salary <15000
      AND e.department_id=p_deptno
      AND e.job_id=p_job;

    v_emp_no              employees.employee_id%TYPE;
   ```

```
 v_emp_name          VARCHAR2(30);
 v_job_title         employees.job_id%TYPE;
 v_salary            employees.salary%TYPE;
 v_dept_name         departments.department_name%TYPE;

BEGIN

 OPEN emp_salary_cur(60,'IT_PROG');
 DBMS_OUTPUT.PUT_LINE('All employees employed as Programmers in department 60');
 DBMS_OUTPUT.PUT_LINE(CHR(13));
 DBMS_OUTPUT.PUT_LINE('Employee Id'||CHR(9)||'Employee
Names'||CHR(9)||CHR(9)||'Job Title'||CHR(9)||'Salary '||'Department Name');

 LOOP
   FETCH emp_salary_cur INTO v_emp_no,v_emp_name,v_job_title,v_salary,v_dept_name;
   EXIT WHEN emp_salary_cur%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(v_emp_no||CHR(9)||CHR(9)||v_emp_name||CHR(9)||CHR(9)||
v_job_title||CHR(9)||v_salary||CHR(9)||v_dept_name);
  END LOOP;
  CLOSE emp_salary_cur;
  DBMS_OUTPUT.PUT_LINE(CHR(13));--Open an empty line

  DBMS_OUTPUT.PUT_LINE('All employees employed as Accountants in department 100');
  DBMS_OUTPUT.PUT_LINE(CHR(13));

  OPEN emp_salary_cur(100,'FI_ACCOUNT');

 DBMS_OUTPUT.PUT_LINE('Employee Id'||CHR(9)||'Employee Names'||CHR(9)||
CHR(9)||'Job Title'||CHR(9)||'Salary '||'Department Name');

 LOOP
   FETCH emp_salary_cur INTO v_emp_no,v_emp_name,v_job_title,v_salary,v_dept_name;
   EXIT WHEN emp_salary_cur%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(v_emp_no||CHR(9)||CHR(9)||v_emp_name||CHR(9)||CHR(9)||
v_job_title||CHR(9)||TO_CHAR(v_salary,'l999,999.99')||CHR(9)||v_dept_name);
  END LOOP;
  CLOSE emp_salary_cur;
END;
/
```

```
DECLARE
 CURSOR emp_salary_cur(p_deptno NUMBER, p_job VARCHAR2)
 IS
  SELECT employee_id,first_name||' '||last_name names,job_id,salary,department_name
  FROM employees e,departments d
  WHERE d.department_id=e.department_id
  AND salary <15000
  AND e.department_id=p_deptno
  AND e.job_id=p_job;

---Replace variables with a record
  emp_sal_rec          emp_salry_cur%ROWTYPE;
BEGIN

 OPEN emp_salary_cur(60,'IT_PROG');
 DBMS_OUTPUT.PUT_LINE('All employees employed as Programmers in department 60');
 DBMS_OUTPUT.PUT_LINE(CHR(13));
 DBMS_OUTPUT.PUT_LINE('Employee Id'||CHR(9)||'Employee Names'||CHR(9)||CHR(9)||'Job
Title'||CHR(9)||'Salary '||'Department Name');

 LOOP
  FETCH emp_salary_cur INTO emp_sal_rec;
  EXIT WHEN emp_salary_cur%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(emp_sal_rec.employee_id||CHR(9)||CHR(9)||emp_sal_rec.names
||CHR(9)||CHR(9)||emp_sal_rec.job_id||CHR(9)||TO_CHAR(emp_sal_rec.salary,'l999,999.99')||C
HR(9)||emp_sal_rec.department_name);
 END LOOP;
 CLOSE emp_salary_cur;
 DBMS_OUTPUT.PUT_LINE(CHR(13));--Open an empty line

 DBMS_OUTPUT.PUT_LINE('All employees employed as Accountants in department 100');
 DBMS_OUTPUT.PUT_LINE(CHR(13));

 OPEN emp_salary_cur(100,'FI_ACCOUNT');

 DBMS_OUTPUT.PUT_LINE('Employee Id'||CHR(9)||'Employee Names'||CHR(9)||CHR(9)||'Job
Title'||CHR(9)||'Salary '||'Department Name');

 LOOP
  FETCH emp_salary_cur INTO v_emp_no,v_emp_name,v_job_title,v_salary,v_dept_name;
  EXIT WHEN emp_salary_cur%NOTFOUND;

 DBMS_OUTPUT.PUT_LINE(emp_sal_rec.employee_id||CHR(9)||CHR(9)||emp_sal_rec.names
||CHR(9)||CHR(9)||emp_sal_rec.job_id||CHR(9)||TO_CHAR(emp_sal_rec.salary,'l999,999.99')||C
HR(9)||emp_sal_rec.department_name);
```

```
  END LOOP;
  CLOSE emp_salary_cur;
END;
/
```

All employees employed as Programmers in department 60

Employee Id	Employee Names	Job Title	Salary	Department Name
103	Alexander Hunold	IT_PROG	R9,000.00	IT
104	Bruce Ernst	IT_PROG	R6,000.00	IT
105	David Austin	IT_PROG	R4,800.00	IT
106	Valli Pataballa	IT_PROG	R4,800.00	IT
107	Diana Lorentz	IT_PROG	R4,200.00	IT

All employees employed as Accountants in department 100

Employee Id	Employee Names	Job Title	Salary	Department Name
109	Daniel Faviet	FI_ACCOUNT	R9,000.00	Finance
110	John Chen	FI_ACCOUNT	R8,200.00	Finance
111	Ismael Sciarra	FI_ACCOUNT	R7,700.00	Finance
112	Jose Manuel Urman	FI_ACCOUNT	R7,800.00	Finance
113	Luis Popp	FI_ACCOUNT	R6,900.00	Finance

PL/SQL procedure successfully completed.

The other way is the parameters to accept values that are prompted by the cursor then display the output of the block. In this scenario the cursor is opened ones and closed ones compare to the above example were it is opened more than ones with different values supplied.

```
DECLARE
 CURSOR emp_salary_cur(p_deptno NUMBER, p_job VARCHAR2)
 IS
  SELECT employee_id,first_name||' '||last_name names,job_id,salary,department_name
  FROM employees e,departments d
  WHERE d.department_id=e.department_id
  AND salary <15000
  AND e.department_id=p_deptno
  AND e.job_id=p_job;

 emp_sal_rec  emp_salary_cur%ROWTYPE;
BEGIN

 OPEN emp_salary_cur(&_dept_no,UPPER('&job_id'));
 DBMS_OUTPUT.PUT_LINE('All employees employed as per user entry);
 DBMS_OUTPUT.PUT_LINE(CHR(13));--make a space
```

```
 DBMS_OUTPUT.PUT_LINE('Employee Id'||CHR(9)||'Employee Names'||CHR(9)||CHR(9)||'Job
Title'||CHR(9)||'Salary '||'Department Name');

 LOOP
  FETCH emp_salary_cur INTO emp_sal_rec;
  EXIT WHEN emp_salary_cur%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(emp_sal_rec.employee_id||CHR(9)||CHR(9)||emp_sal_rec.names||CH
R(9)||CHR(9)||emp_sal_rec.job_id||CHR(9)||TO_CHAR(emp_sal_rec.salary,'l999,999.99')||CHR(9)
||emp_sal_rec.department_name);
 END LOOP;
 CLOSE emp_salary_cur;

END;
/
```

Enter value for _dept_no: 50
Enter value for job_id: st_man
All employees employed as per user value entry

Employee Id	Employee Names	Job Title	Salary	Department Name
120	Matthew Weiss	ST_MAN	R8,000.00	Shipping
121	Adam Fripp	ST_MAN	R8,200.00	Shipping
122	Payam Kaufling	ST_MAN	R7,900.00	Shipping
123	Shanta Vollman	ST_MAN	R6,500.00	Shipping
124	Kevin Mourgos	ST_MAN	R5,800.00	Shipping

PL/SQL procedure successfully completed.

2. **FOR UPDATE clause**

This clause explicitly locks a row that is being updated so that it denies access for the duration of a transaction. It locks the rows before the UPDATE or DELETE.
Include the **FOR UPDATE** clause in the cursor statement to lock the rows first

It uses the cursor to update or delete the current row. Then use the **WHERE CURRENT OF** clause to reference the current row from an explicit cursor

Syntax

```
DECLARE
   CURSOR cursor_name IS
    SELECT statement
   FOR UPDATE OF salary NOWAIT; --the field that is updated
BEGIN
```

```
LOOP
  UPDATE table_name
  WHERE CURRENT OF cursor_name;
END;
```

An example is when only records of all employees earning less than R5000 must be update by 5.5% and this must be done immediately. The transaction must deny access to any of these records while the process is still on.

All employees earning less than R5000

Employee Id	Employee Names	Job Title	Salary
105	David Austin	IT_PROG	R4,800.00
106	Valli Pataballa	IT_PROG	R4,800.00
116	Shelli Baida	PU_CLERK	R2,900.00
117	Sigal Tobias	PU_CLERK	R2,800.00
137	Renske Ladwig	ST_CLERK	R3,600.00
.			
.			
.			
197	Kevin Feeney	SH_CLERK	R3,000.00
198	Donald OConnell	SH_CLERK	R2,600.00
199	Douglas Grant	SH_CLERK	R2,600.00
200	Jennifer Whalen	AD_ASST	R4,400.00

PL/SQL procedure successfully completed.

The below code shows how updating should be done.

```
DECLARE
 CURSOR salary_cur IS
  SELECT employee_id,first_name||' '||last_name names,job_id,salary
  FROM employees e
  WHERE salary <5000
  FOR UPDATE OF salary NOWAIT;
BEGIN

 FOR salary_rec IN salary_cur LOOP
  IF salary_rec.salary <5000 THEN
    UPDATE employees
    SET salary = salary_rec.salary * 1.05
    WHERE CURRENT OF salary_cur;
  END IF;
 END LOOP;
END;
/
```

Study the Cursor subquery

CHAPTER 8 (Handling Exception)

- An **Exception** is an identifier in **PL/SQL** that is raised during execution.
-
- It is when an Oracle error occurs or can be raised by the programmer explicitly.
- Exception can be handled by:
    - Trapping it with the handler.
    - Propagating it to the calling environment.

Methods for raising an Exception

- An Oracle error occurs, and the associated exception is raised automatically. For example, if the error ORA-01403 occurs when no rows are retrieved from the database in a SELECT statement, then PL/SQL raise the exception NO_DATA_FOUND.
- You raise an exception explicitly by issuing the RAISE statement within the block.

Propagating an Exception

The moment the exception is raised in the executable section of the block and there is no corresponding exception handler, the PL/SQL block terminates with the failure and the exception is propagated to the calling environment.

Example:

```
DECLARE
 v_emp_id      employees.employee_id%TYPE:=&employee_no;
 v_fname       employees.first_name%TYPE;
 v_lname       employees.last_name%TYPe;
 v_job         employees.job_id%TYPE;
 v_salary      employees.salary%TYPE;
BEGIN
 SELECT first_name,last_name,job_id,salary
 INTO v_fname,v_lname,v_job,v_salary
 FROM employees
 WHERE employee_id=v_empid;

 DBMS_OUTPUT.PUT_LINE(UPPER(v_fname||' '||v_lname)||' is employed as '||v_job||'
and earns '||TO_CHAR(v_salary,'l999,999.99'));
END;
/
```

The moment it's executed the following happens;

```
Enter value for employee_no: 300
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 8
```

Due to the fact that exception handler is not used the PL/SQL display this message to display the error the block has encountered when the it was executed.

Trapping an Exception

The moment the exception is raised in the executable section of the block, processing branches to the to the corresponding exception handler in the exception section of the block. If PL/SQL handles the exception, then the exception does not propagate to the enclosed block.

PL/SQL has three types of Exceptions , of which two are implicitly raised and the other one is explicitly raised.

- Predefined Oracle server
- Nonpredefined Oracle server

} **Implicitly raised**

- User-defined → **Explicitly raised**

Predefined error is one of approximately 20 errors that occur most often in PL/SQL code like example I showed above in the propagation of exception. Do not declare and allow the Oracle Server to raise them implicitly. Predefined errors are listed in page 9 to 10 of this chapter.

Nonpredefined error is any other standard Oracle Server error. Declare within the declarative section and allow the Oracle server to raise them implicitly.

User-defined error is a condition that the developer determine as abnormal. Declare within the declarative section and raise explicitly.

Let us start with example of a predefine exception. Exception trapping is the last section of your anonymous block. The Block consists of the declaration section, the executable section (an environment were we write the code to be executed) and lastly the exception section that is used to trap the errors.

a) **Predefined Exception**
It has predefined names that can be used to specific error trapping, no declaration is needed.

Enter value for job_title: ad_assit
AD_ASSIT does not exists.
PL/SQL procedure successfully completed.

SQL> /
Enter value for job_title: It_prog
IT_PROG has more than one employee.
PL/SQL procedure successfully completed.

SQL> /
Enter value for job_title: ad_pres
STEVEN KING(100) is employed as AD_PRES and earns R24,000.

The code for the above output is posted below.

```
DECLARE
 v_emp_id      employees.employee_id%TYPE;
 v_fname       employees.first_name%TYPE;
 v_lname       employees.last_name%TYPe;
 v_job         employees.job_id%TYPE:=UPPER('&job_title');
 v_salary      employees.salary%TYPE;
BEGIN
 SELECT employee_id,first_name,last_name,job_id,salary
 INTO v_emp_id,v_fname,v_lname,v_job,v_salary
 FROM employees
 WHERE job_id=v_job;

 DBMS_OUTPUT.PUT_LINE(UPPER(v_fname||' '||v_lname)||'('||v_emp_id||') is employed
as '||v_job||' and earns '||TO_CHAR(v_salary,'fml999,999.99'));

EXCEPTION
 WHEN NO_DATA_FOUND THEN --if no record is found by the query then this error will be
raised
   DBMS_OUTPUT.PUT_LINE(v_job||' does not exists.');
 WHEN TOO_MANY_ROWS THEN --if more than one record is found by the query then this error
will be raised
   DBMS_OUTPUT.PUT_LINE(v_job||' has more than one employee.');
 WHEN OTHERS THEN --this only become raised if the other errors are met
   DBMS_OUTPUT.PUT_LINE('No other errors');
END;
/
```

b) **Nonpredefined Exception**

It is similar to predefined exception but do not have predefined names but uses Oracle error number (ORA,'####') and error message. It uses EXCEPTION_INIT function. You need to declare.

```
Enter value for job_title: sh_clerk
SH_CLERK has more than one employee.
PL/SQL procedure successfully completed.

SQL> /
Enter value for job_title: ac_pres
AC_PRES does not exists.
PL/SQL procedure successfully completed.
```

```
DECLARE
 v_emp_id      employees.employee_id%TYPE;
 v_fname       employees.first_name%TYPE;
 v_lname       employees.last_name%TYPE;
 v_job         employees.job_id%TYPE:=UPPER('&job_title');
 v_salary      employees.salary%TYPE;

--Declare the non-predefine exception her
 e_no_records          EXCEPTION;
 PRAGMA EXCEPTION_INIT(e_no_records,+100);

 e_more_records EXCEPTION;
 PRAGMA EXCEPTION_INIT(e_more_records,-01422);
BEGIN
 SELECT employee_id,first_name,last_name,job_id,salary
 INTO v_emp_id,v_fname,v_lname,v_job,v_salary
 FROM employees
 WHERE job_id=v_job;

 DBMS_OUTPUT.PUT_LINE(UPPER(v_fname||' '||v_lname)||'('||v_emp_id||') is
employed as '||v_job||' and earns '||TO_CHAR(v_salary,'fml999,999.99'));

EXCEPTION
 WHEN e_no_records THEN --if no record is found by the query then this error will be
raised
   DBMS_OUTPUT.PUT_LINE(v_job||' does not exists.');
 WHEN e_more_records THEN --if more than one record is found by the query then this
error will be raised
   DBMS_OUTPUT.PUT_LINE(v_job||' has more than one employee.');
END;
/
```

c) **User-defined Exception**

```
Enter value for job_title: ac_pres
AC_PRES does not exists.
PL/SQL procedure successfully completed.

SQL> /
Enter value for job_title: it_prog
IT_PROG has more than one employee.
PL/SQL procedure successfully completed.

SQL> /
Enter value for job_title: ad_pres
STEVEN KING(100) is employed as AD_PRES and earns R24,000.
```

```
DECLARE
 v_emp_id      employees.employee_id%TYPE;
 v_fname       employees.first_name%TYPE;
 v_lname       employees.last_name%TYPE;
 v_job         employees.job_id%TYPE:=UPPER('&job_title');
 v_salary      employees.salary%TYPE;
--Declare the non-predefine exception her

 e_no_records          EXCEPTION;
 e_more_records EXCEPTION;
 v_count       NUMBER(2):=0;
BEGIN
 SELECT COUNT(*) count
 INTO v_count
 FROM employees
 WHERE job_id=v_job;

 IF v_count = 0 THEN
   RAISE e_no_records;
 ELSIF v_count > 1 THEN
      RAISE e_more_records;
 ELSE
    SELECT employee_id,first_name,last_name,job_id,salary
    INTO v_emp_id,v_fname,v_lname,v_job,v_salary
    FROM employees
    WHERE job_id=v_job;
    DBMS_OUTPUT.PUT_LINE(UPPER(v_fname||' '||v_lname)||'('||v_emp_id||') is
          employed as '||v_job||' and earns '||TO_CHAR(v_salary,'fml999,999.99'));
 END IF;
```

```
EXCEPTION
 WHEN e_no_records THEN --if no record is found by the query then this error will be
        raised
   DBMS_OUTPUT.PUT_LINE(v_job||' does not exists.');
 WHEN e_more_records THEN --if more than one record is found by the query then this
        error will be raised
   DBMS_OUTPUT.PUT_LINE(v_job||' has more than one employee.');
END;
/
```