East West University
Department of Computer Science and Engineering
Course: CSE464
Instructor: Antu Chowdhury

Lab 03: Working with Cursors and Exception Handling in PL/SQL

Lab Objectives:

- Understand the concept of cursors in PL/SQL and their role in processing query results row-by-row.
- Differentiate between implicit and explicit cursors.
- Use cursor attributes such as %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN.
- Implement PL/SQL blocks using explicit cursors with loops and conditional statements.
- Handle exceptions such as NO_DATA_FOUND during cursor operations.

Topics Covered:

- Cursor basics: implicit vs explicit cursors
- Cursor declaration, opening, fetching, and closing
- Cursor attributes:
  - %FOUND
  - %NOTFOUND
  - %ROWCOUNT
  - %ISOPEN
- Exception handling with cursors (e.g., NO_DATA_FOUND)
- Combining cursors with loops and conditional statements
- Using %TYPE to declare variables based on table columns

Lab Outcomes:

By the end of this lab session, students were able to:

1. Understand and explain the difference between implicit and explicit cursors.
2. Write PL/SQL blocks using explicit cursors to fetch and process multiple rows.
3. Use cursor attributes to control flow and output information about query processing.
4. Handle exceptions such as NO_DATA_FOUND in cursor operations gracefully.
5. Declare variables dynamically using %TYPE to ensure compatibility with table columns.
6. Combine loops and conditionals in PL/SQL blocks involving cursors for effective row-by-row processing.

Lab Activities

Cursor:
A cursor in PL/SQL is a pointer to the memory area (called the context area) where the result set of a SQL query is stored. It allows you to retrieve and process query results one row at a time, making it especially useful when dealing with multiple rows returned by a query.

Key Points:

- A cursor holds the active data set returned by a SQL statement.
- It provides controlled, row-by-row access to query results.

Types:

- **Implicit cursors** are automatically created and managed by Oracle whenever an SQL statement is executed. In PL/SQL, an implicit cursor is automatically generated to process SQL statements such as INSERT, UPDATE, DELETE, or single-row SELECT INTO queries, requiring no explicit declaration or control by the programmer.
- **Explicit cursors** are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Cursor Attributes:

Cursor Attribute	Implicit Cursor Attribute	Explicit Cursor Attribute
%ISOPEN	__	CursorName%ISOPEN
%FOUND	SQL%FOUND	CursorName%FOUND
%NOTFOUND	SQL%NOTFOUND	CursorName%NOTFOUND
%ROWCOUNT	SQL%ROWCOUNT	CursorName%ROWCOUNT

(prerequisite) Create Table:

```
CREATE TABLE Employee (
    E_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Salary NUMBER
);
```

E_ID	Name	Salary
1	Alim	10000
2	Bob	4500
3	Carol	5500
4	David	6000
5	Eva	4800

Practice: Implicit Cursor

- **Write a PL/SQL block that increases the salary by 500 for all employees whose salary is less than 5000. Print the number of rows updated using SQL%ROWCOUNT. (using implicit cursor)**

```
BEGIN
   UPDATE employee
   SET salary = salary + 500
   WHERE salary < 5000;

   DBMS_OUTPUT.PUT_LINE('Rows updated: ' || SQL%ROWCOUNT);
END;
```

- **Using %TYPE, declare variables to hold Name and Salary. Write a PL/SQL block to select the Name and Salary of an employee with E_ID = 2 into those variables and print the values. (using implicit cursor)**

```
DECLARE
   v_name employee.name%TYPE;
   v_salary employee.salary%TYPE;
BEGIN
   SELECT name, salary INTO v_name, v_salary
   FROM employee
   WHERE E_ID = 2;

   DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ' Salary: ' || v_salary);
END;
```

- **Write a PL/SQL block to increase the salary by 300 for employee ID 3. Use SQL%FOUND to print whether the update succeeded or no record was found. (using implicit cursor)**

```
BEGIN
   UPDATE employee
   SET salary = salary + 300
   WHERE e_id = 3;

   IF SQL%FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Update succeeded.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Record not found.');
   END IF;
END;
```

- **Declare variables for Name and Salary using %TYPE. Write a PL/SQL block to select employee details for E_ID = 10. Handle the NO_DATA_FOUND exception and print "Employee not found". (using implicit cursor)**

```
DECLARE
   v_name   employee.name%TYPE;
   v_salary employee.salary%TYPE;
BEGIN
   SELECT name, salary INTO v_name, v_salary
   FROM employee
   WHERE e_id = 10;
   DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Salary: ' || v_salary);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee not found.');
END;
```

- **Write a PL/SQL block to delete an employee with E_ID = 100 from the employee table. Use the SQL%NOTFOUND attribute to check whether any row was actually deleted. (using implicit cursor)**

Explicit Cursor Syntax:

1. **Declaring the cursor for initializing the memory**
    - **Syntax:**
    - CURSOR cursor_name IS select_statement;
    - **Meaning:**
      This defines the cursor with a name and the SQL SELECT statement it will execute. This step only initializes memory for the cursor but does not execute the query yet.
    - **Example:**
    - CURSOR c_employee IS
    - SELECT e_id, name, salary FROM employee;
2. **Opening the cursor for allocating the memory**
    - **Syntax:**
    - OPEN cursor_name;
    - **Meaning:**
      Opens the cursor and executes the associated SQL query. This allocates memory for the result set, making the cursor ready for fetching rows.
    - **Example:**
    - OPEN c_employee;
3. **Fetching the cursor for retrieving the data**
    - **Syntax:**
    - FETCH cursor_name INTO variable1, variable2, ...;

- **Meaning:**
  Retrieves the next row from the cursor's result set into the specified variables. Each FETCH moves the cursor forward by one row.
- **Example:**
- FETCH c_employee INTO v_eid, v_name, v_salary;

4. **Closing the cursor to release the allocated memory**
   - **Syntax:**
   - CLOSE cursor_name;
   - **Meaning:**
     Releases the memory and resources allocated to the cursor. Always close the cursor after processing to free resources.
   - **Example:**
   - CLOSE c_employee;

Complete Example Combining All Steps:

- **Write a PL/SQL block using a user-defined cursor to fetch the name and salary from the employee table and print the total number of rows fetched.**

```
DECLARE
   v_eid    employee.e_id%TYPE;
   v_name   employee.name%TYPE;
   v_salary employee.salary%TYPE;

   CURSOR c_employee IS
      SELECT e_id, name, salary FROM employee;
BEGIN
   OPEN c_employee;

   LOOP
      FETCH c_employee INTO v_eid, v_name, v_salary;

      EXIT WHEN c_employee%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('ID: ' || v_eid || ', Name: ' || v_name || ', Salary: ' || v_salary);
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Total rows fetched: ' || c_employee%ROWCOUNT);

   CLOSE c_employee;
END;
```

- **Write a PL/SQL program using an explicit cursor that retrieves all rows from the EMPLOYEE table and checks each employee's salary. If an employee's salary is less than 7000, increase it by 500 using an UPDATE statement with a proper WHERE clause to ensure only the matching row is updated. For each row processed, display a message using DBMS_OUTPUT.PUT_LINE indicating whether the salary was updated or not, based on the condition. The program should use cursor control statements (OPEN, FETCH, EXIT WHEN, CLOSE) and conditional logic inside a loop to handle per-row processing.**

Problem Practice:

Lab Task # 01 (Create table):

1.(a). Write SQL statement to create a table 'instructor_your_student_id' which has 4 attributes –
i) id (number) [Primary Key]
ii) name (varchar)
iii) dept_name (varchar)
iv) salary (number)

Example:

```
create table instructor_2020360001 ( …… );
```

Lab Task # 02 (Inserting data into a table):

2.(a). Write SQL statements to insert the following records into 'instructor_your_student_id' table:

ID	Name	Dept_Name	Salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wasif	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Goblin	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000

Lab Task # 03: PL/SQL Cursor and %TYPE Practice (Based on Today's Session)

1. Write a PL/SQL block using **implicit cursor** to update the salary of instructor with id = 15151 by increasing it by 500. Use SQL%ROWCOUNT to print how many rows were updated.
2. Declare variables for name and salary using %TYPE. Write a PL/SQL block to select the details of the instructor with id = 22222 into these variables and print them.
3. Write a PL/SQL block using **exception handling** to select instructor details for id = 99999. If no data is found, print "Instructor not found".
4. Write a PL/SQL block using an **explicit cursor** to fetch and print the details of all instructors from the table. Include steps for declaring, opening, fetching in a loop, and closing the cursor.
5. Write a PL/SQL block to delete an instructor with id = 12345. Use SQL%NOTFOUND to print "Instructor not found" if no record was deleted, otherwise print "Instructor deleted successfully".
6. Write a PL/SQL block using an explicit cursor to fetch all instructors whose salary is less than 60000, update their salary by adding 1000, and count how many instructors were updated. Print the total count after processing.

Submission Instructions:

1. For each question:
    - Run the PL/SQL code in your database environment.
    - Take screenshots of your SQL code and its output.
    - Paste the code and screenshots into a Word document (.doc or .docx).
2. Save the document as a PDF named:
   `<YourStudentID>_LAB03.pdf`
   (Replace `<YourStudentID>` with your actual student ID.)
3. Submit the PDF file using the assignment section in your classroom.