

Week 8 Working with Composite Data Types

IM101 - Advanced Database Systems

LEARNING OUTCOMES:

At the end of the session, the students should be able to:

- 1. Create a record with the %ROWTYPE attribute.
- 2. Create and manipulate user-defined PL/SQL records.
- 3. Distinguish between an implicit and an explicit cursor
- 4. Describe why and when to use an explicit cursor in PL/SQL code.
- 5. Create PL/SQL code that successfully opens a cursor and fetches a piece of data into a variable.
- 6. Retrieve information about the state of an explicit cursor using cursor attributes.
- 7. Create PL/SQL code to declare a cursor and manipulate it in a FOR loop.

Introduction to Composite Data Types

- Composite data types are collections of multiple values.
- PL/SQL provides two main composite data types:
 - Records (used to group related variables)
 - Collections (arrays, nested tables, VARRAYs)
- Focus on PL/SQL records and cursors in this session.

Using %ROWTYPE Attribute

- %ROWTYPE allows storing an entire row from a table in a variable.
- Automatically inherits the structure of the table.

```
DECLARE
    emp_rec employees%ROWTYPE;

BEGIN

SELECT * INTO emp_rec FROM employees WHERE employee_id = 101;
    DBMS_OUTPUT_LINE('Employee Name: ' || emp_rec.first_name);
END;
```

User-Defined PL/SQL Records

 Custom records allow defining structured data types.

```
DECLARE
    TYPE emp type IS RECORD (
        emp id employee.employee id% TYPE,
        emp name employee.name% TYPE,
        emp salary employee.salary% TYPE
    );
    emp rec emp type;
BEGIN
    emp rec.emp id := 101;
    emp rec.emp name := 'John Doe';
    emp rec.emp salary := 50000;
    DBMS OUTPUT.PUT LINE('Employee: ' | emp rec.emp name);
END;
```

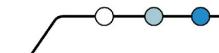
Implicit vs. Explicit Cursors

Implicit Cursor:

- Automatically created for SELECT, INSERT, UPDATE, DELETE statements.
- No need for explicit declaration.
- Used when retrieving a single row.

• Explicit Cursor:

- Manually declared and controlled.
- Used when retrieving multiple rows.
- Provides more flexibility and control.



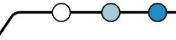
When to Use Explicit Cursors

- Required when retrieving multiple records from a table.
- Provides better performance tuning and control over row processing.
- Helps in fetching data row by row.
- Essential when looping over result sets.

Opening a Cursor and Fetching Data

Steps to use an explicit cursor:

- 1. Declare the cursor.
- 2. Open the cursor.
- 3. Fetch data from the cursor.
- 4. Close the cursor.



Opening a Cursor and Fetching Data

```
DECLARE
   CURSOR emp cursor IS SELECT name, salary FROM employee;
    emp name employee.name%TYPE;
    emp salary employee.salary%TYPE;
BEGIN
   OPEN emp cursor;
    FETCH emp cursor INTO emp name, emp salary;
   DBMS OUTPUT.PUT LINE('Employee: ' | emp name | ', Salary: ' |
emp salary);
   CLOSE emp cursor;
END;
```

Using Cursor Attributes

Cursor attributes provide information about query execution:

- %FOUND: Returns TRUE if the last FETCH affected at least one row.
- %NOTFOUND: Returns TRUE if the last FETCH did not retrieve any row.
- %ROWCOUNT: Returns the number of rows fetched.
- **%ISOPEN**: Returns TRUE if the cursor is open.

Using Cursor Attributes

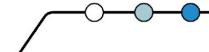
```
IF emp_cursor%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Data found!');
ELSE
    DBMS_OUTPUT.PUT_LINE('No data found.');
END IF;
```

Using Cursors in a FOR LOOP

The FOR LOOP automates opening, fetching, and closing the cursor.

Best Practices for Working with Composite Data Types

- Use %ROWTYPE to avoid declaring individual variables for each column.
- Use user-defined records for better code organization.
- Prefer implicit cursors for single-row operations.
- Use explicit cursors for multi-row processing when needed.
- Always close cursors after usage to free memory.



Summary

- PL/SQL supports composite data types like records and cursors.
- The %ROWTYPE attribute simplifies variable declarations.
- Explicit cursors offer more control over multi-row queries.
- Cursor attributes provide valuable information about query execution.
- FOR loops simplify cursor handling and iteration.

END OF PRESENTATION. THANK YOU!