26-06-2024 Working with Composite Data Types

Creating a PL/SQL Record

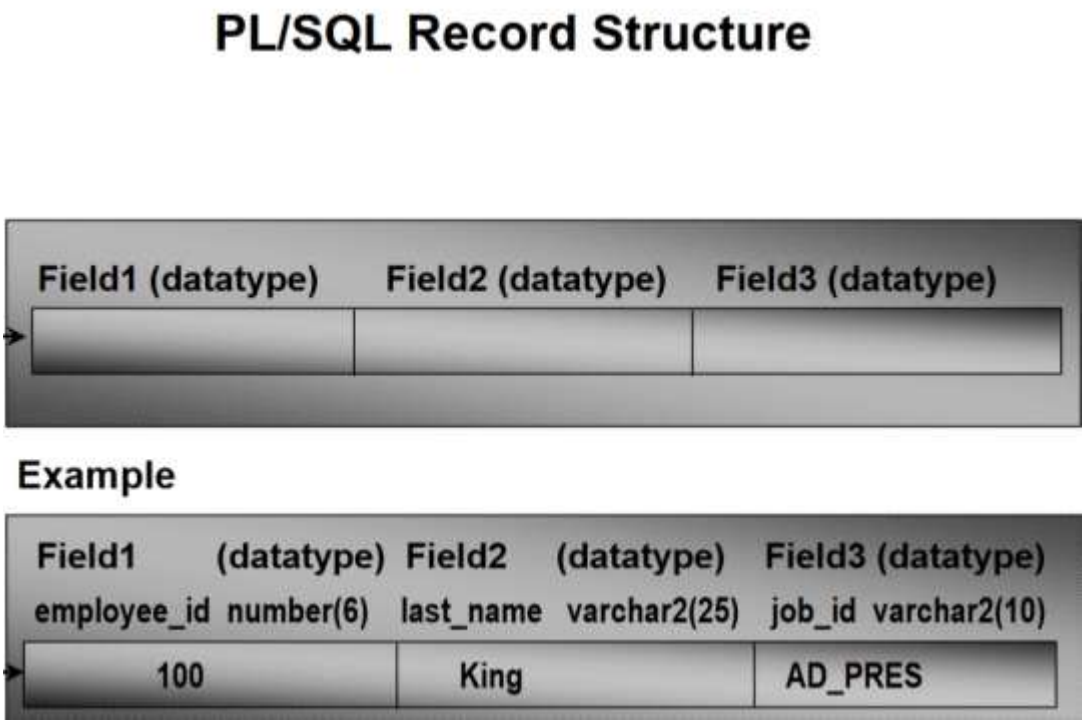
1.TYPE emp_record_type IS RECORD

(last_name VARCHAR2(25),

job_id VARCHAR2(10),

salary NUMBER(8,2));

emp_record emp_record_type;



The %ROWTYPE Attribute

2.DEFINE employee_number = 124

DECLARE

emp_rec employees%ROWTYPE;

BEGIN

SELECT * INTO emp_rec FROM employees

WHERE employee_id = &employee_number;

INSERT INTO retired_emps(empno, ename, job, mgr, hiredate,

leavedate, sal, comm, deptno)

VALUES (emp_rec.employee_id, emp_rec.last_name, emp_rec.job_id,

emp_rec.manager_id, emp_rec.hire_date, SYSDATE, emp_rec.salary,

emp_rec.commission_pct, emp_rec.department_id);

Writing Explicit Cursors

Cursor Type	Description
Implicit	Implicit cursors are declared by PL/SQL implicitly for all DML and PL/SQL SELECT statements, including queries that return only one row.
Explicit	For queries that return more than one row, explicit cursors are declared and named by the programmer and manipulated through specific statements in the block's executable actions.

Declaring the Cursor

3.DECLARE

CURSOR emp_cursor IS

SELECT employee_id, last_name

FROM employees;

CURSOR dept_cursor IS

SELECT *

FROM departments

WHERE location_id = 170;

4.SET SERVEROUTPUT ON

DECLARE

v_empno employees.employee_id%TYPE;

v_ename employees.last_name%TYPE;

CURSOR emp_cursor IS

SELECT employee_id, last_name

FROM employees;

BEGIN

OPEN emp_cursor;

```
FOR i IN 1..10 LOOP
FETCH emp_cursor INTO v_empno, v_ename;
DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
||' '|| v_ename);
END LOOP;
END ;
```

5.DECLARE

```
v_empno employees.employee_id%TYPE;
v_ename employees.last_name%TYPE;
CURSOR emp_cursor IS
SELECT employee_id, last_name
FROM employees;
BEGIN
OPEN emp_cursor;
LOOP
FETCH emp_cursor INTO v_empno, v_ename;
EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
emp_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)
||' '|| v_ename);
END LOOP;
CLOSE emp_cursor;
END ;
```

Cursors and Records

6.DECLARE

```
CURSOR emp_cursor IS
SELECT employee_id, last_name
FROM employees;
emp_record emp_cursor%ROWTYPE;
```

BEGIN

OPEN emp_cursor;

LOOP

FETCH emp_cursor INTO emp_record;

EXIT WHEN emp_cursor%NOTFOUND;

INSERT INTO temp_list (empid, empname)

VALUES (emp_record.employee_id, emp_record.last_name);

END LOOP;

CLOSE emp_cursor;

END;

Cursor FOR Loops

7.DECLARE

CURSOR emp_cursor IS

SELECT last_name, department_id

FROM employees;

BEGIN

FOR emp_record IN emp_cursor LOOP

IF emp_record.department_id = 80 THEN

END LOOP;

END;

Cursor FOR Loops Using Subqueries

8.SET SERVEROUTPUT ON

BEGIN

FOR emp_record IN (SELECT last_name, department_id

FROM employees) LOOP

IF emp_record.department_id = 80 THEN

DBMS_OUTPUT.PUT_LINE (’Employee ’ || emp_record.last_name

|| ’ works in the Sales Dept. ’);

END IF;

END LOOP; END ;

9.SET SERVEROUTPUT ON

DECLARE

v_employee_id employees.employee_id%TYPE;

v_job_id employees.job_id%TYPE;

v_start_date DATE;

v_end_date DATE;

CURSOR emp_cursor IS

SELECT employee_id, job_id, start_date, end_date

FROM job_history

ORDER BY employee_id;

BEGIN

OPEN emp_cursor;

LOOP

FETCH emp_cursor

INTO v_employee_id, v_job_id, v_start_date, v_end_date;

DBMS_OUTPUT.PUT_LINE (’Employee #: ’ || v_employee_id ||

’ held the job of ’ || v_job_id || ’ FROM ’ ||

v_start_date || ’ TO ’ || v_end_date);

EXIT WHEN emp_cursor%ROWCOUNT > 4 OR

emp_cursor%NOTFOUND;

END LOOP;

CLOSE emp_cursor;

END;