5 Working with Composite Data Types

Objectives

After completing this lesson, you should be able to do the following:

- Create user-defined PL/SQL records
- Create a record with the %ROWTYPE attribute
- Create an INDEX BY table
- Create an INDEX BY table of records
- Describe the difference between records, tables, and tables of records.

Composite Data Types

- Are of two types:
  - ➢ PL/SQL RECORDs
  - ➢ PL/SQL Collections
    - – INDEX BY Table
    - – Nested Table
    - – VARRAY
- Contain internal components
- Are reusable

PL/SQL Records

- Must contain one or more components of any scalar, RECORD, or INDEX BY table data type, called fields
- Are similar in structure to records in a third generation language (3GL)
- Are not the same as rows in a database table
- Treat a collection of fields as a logical unit
- Are convenient for fetching a row of data from a table for processing

Creating a PL/SQL Record

Syntax:

```
TYPE type_name IS RECORD
          (field_declaration[, field_declaration]…);
identifier type_name;
```

Where *field_declaration* is:

```
field_name {field_type | variable%TYPE
                 | table.column%TYPE | table%ROWTYPE }
                 [[NOT NULL] {:= | DEFAULT} expr]
```

ORACLE

Creating a PL/SQL Record

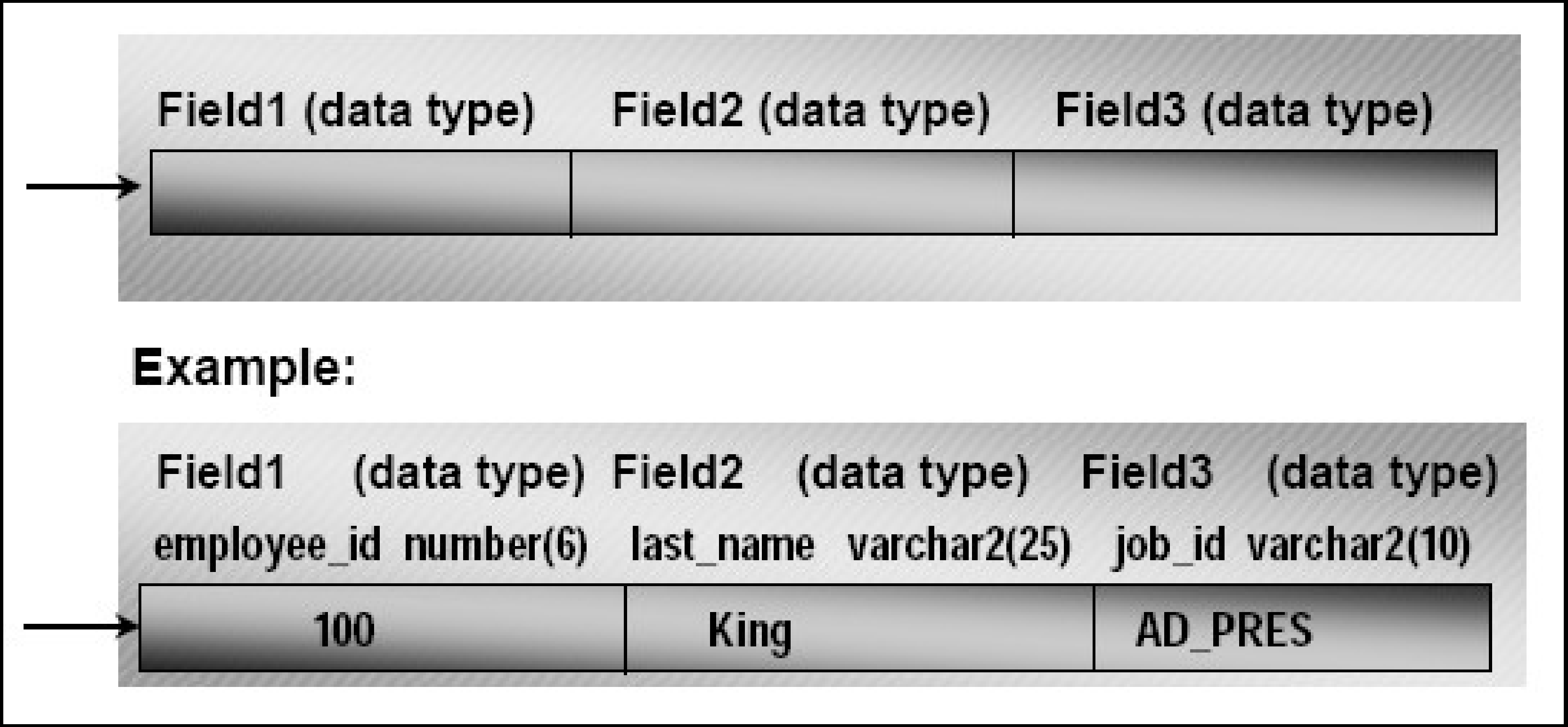
Declare variables to store the name, job, and salary of a new employee.

```
SET SERVEROUTPUT ON
DECLARE
  TYPE EmpRec IS RECORD (
    emp_name    VARCHAR2(50),
    job_title   VARCHAR2(9),
    salary      NUMBER(7,2));
  emp_info EmpRec;
BEGIN
  SELECT first_name||’ ‘||last_name, job_id, salary
  INTO emp_info.emp_name, emp_info.job_title, emp_info.salary
  FROM employees
  WHERE employee_id = 105;
  DBMS_OUTPUT.PUT_LINE ('Nhan vien : '||emp_info.emp_name||' lam cong viec '||
                                   emp_info.job_title ||' co muc luong '||emp_info.salary );
END;
/
```

Creating a PL/SQL Record

```
SET SERVEROUTPUT ON
DECLARE
  TYPE EmpRec IS RECORD (
    emp_id    employees.employee_id%TYPE,
    job_title   VARCHAR2(9),
    salary       NUMBER(7,2));
  emp_info    EmpRec;
  emp_null    EmpRec;
  emp           EmpRec;
BEGIN
  emp_info.emp_id := 7788;
  emp_info.job_title := 'ANALYST';
  emp_info.salary := 3500;
  DBMS_OUTPUT.PUT_LINE (‘Ma : ‘||emp_info.emp_id||’-’||emp_info.job_title||
                                        ' - '||emp_info.salary );
  emp := emp_info;
  DBMS_OUTPUT.PUT_LINE (‘Ma : ‘||emp.emp_id||’ – ‘||emp.job_title ||' - '||emp.salary );
  emp := emp_null;  -- Tat ca cac field trong emp deu co gia tri null
  DBMS_OUTPUT.PUT_LINE (‘Ma : ‘||emp.emp_id||’ - '||emp.job_title ||' - '||emp.salary );
END;
/
```

PL/SQL Record Structure


Field1 (data type)
Field2 (data type)
Field3 (data type)
Example:
Field1 (data type) Field2 (data type) Field3 (data type)
employee_id number(6) last_name varchar2(25) job_id varchar2(10)
100
King
AD_PRES

The %ROWTYPE Attribute

- Declare a variable according to a collection of columns in a database table or view.
- Prefix %ROWTYPE with the database table.
- Fields in the record take their names and data types from the columns of the table or view.

Advantages of Using %ROWTYPE

- The number and data types of the underlying database columns need not be known.
- The number and data types of the underlying database column may change at run time.
- The attribute is useful when retrieving a row with the SELECT * statement.

The %ROWTYPE Attribute

Examples:

Declare a variable to store the information about a department from the DEPARTMENTS table.

```
dept_record departments%ROWTYPE;
```

Declare a variable to store the information about an employee from the EMPLOYEES table.

```
emp_record employees%ROWTYPE;
```

Creating a PL/SQL Record

```
CREATE TABLE e_temp
  AS SELECT * FROM employees WHERE employee_id = 50;

SET SERVEROUTPUT ON
DECLARE
    emp_rec employees%ROWTYPE;
BEGIN
   SELECT * INTO emp_rec FROM employees
   WHERE employee_id = 105;
   INSERT INTO e_temp
   VALUES (emp_rec.employee_id, emp_rec.first_name, emp_rec.last_name,
     emp_rec.email, emp_rec.phone_number, emp_rec.hire_date, emp_rec.job_id,
     emp_rec.salary, emp_rec.commission_pct, emp_rec.manager_id,
     emp_rec.department_id);
   COMMIT;
END;
/
SELECT * FROM e_temp;
```

INDEX BY Tables

- Are composed of two components:
  - Primary key of data type BINARY_INTEGER
  - Column of scalar or record data type
- Can increase in size dynamically because they are unconstrained

Creating an INDEX BY Table

Syntax:

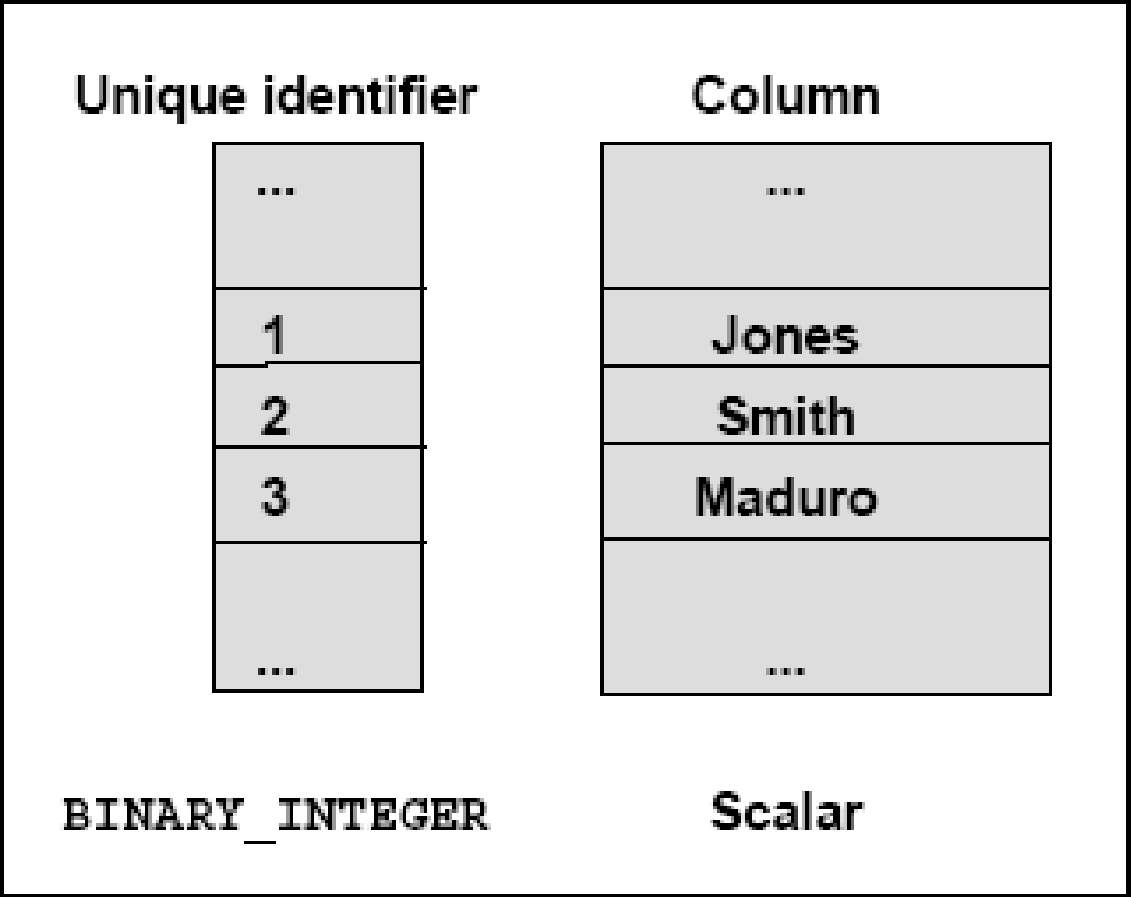
```
TYPE type_name  IS  TABLE OF
                 { column_type | variable%TYPE
                 | table.column%TYPE } [NOT NULL]
                 | table.%ROWTYPE
                 [INDEX BY BINARY_INTEGER];
identifier        type_name;
```

Declare an INDEX BY table to store names.

Example:

```
...
TYPE ename_table_type IS TABLE OF  employees.last_name%TYPE
                 INDEX BY BINARY_INTEGER;
ename_table ename_table_type;
...
```

INDEX BY Table Structure



Creating an INDEX BY Table

```
SET SERVEROUTPUT ON
DECLARE
   TYPE ename_table_type IS TABLE OF
                  employees.last_name%TYPE
                  INDEX BY BINARY_INTEGER;
    TYPE hiredate_table_type IS TABLE OF DATE
                  INDEX BY BINARY_INTEGER;
    ename_table ename_table_type;
    hiredate_table hiredate_table_type;
BEGIN
    ename_table(1) := 'CAMERON';
    hiredate_table(8) := SYSDATE + 7;
        IF ename_table.EXISTS(1) THEN
             DBMS_OUTPUT.PUT_LINE (ename_table(1) || ‘ – ‘||hiredate_table(2));
        END IF;
    EXCEPTION
       WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE (‘Xu ly loi : ‘|| ename_table(1) || ‘ – ‘ ||hiredate_table(8));
END;
/
```

Using INDEX BY Table Methods

The following methods make INDEX BY tables easier to use:

– EXISTS
– COUNT
– FIRST and LAST
– PRIOR
– NEXT
– TRIM
– DELETE

INDEX BY Table of Records

- Define a TABLE variable with a permitted PL/SQL data type.
- Declare a PL/SQL variable to hold department information.

Example:

```
DECLARE
   TYPE dept_table_type IS TABLE OF
           departments%ROWTYPE
           INDEX BY BINARY_INTEGER;
   dept_table dept_table_type;
   -- Each element of dept_table is a record
```

Example of INDEX BY Table of Records

```
SET SERVEROUTPUT ON
DECLARE
    TYPE emp_table_type is table of
                employees%ROWTYPE INDEX BY BINARY_INTEGER;
    my_emp_table emp_table_type;
    v_count NUMBER(3):= 10;
BEGIN
    FOR i IN 1..v_count
    LOOP
                SELECT * INTO my_emp_table(i) FROM employees
                WHERE employee_id = 100 + i;
    END LOOP;
    FOR i IN my_emp_table.FIRST..my_emp_table.LAST
    LOOP
        DBMS_OUTPUT.PUT_LINE(‘Row : ‘||i ||’ – ‘||my_emp_table(i).last_name ||
              ‘ co muc luong ‘ || my_emp_table(i).salary);
    END LOOP;
END;
/
```

Example of INDEX BY Table of Records

```
SET SERVEROUTPUT ON
DECLARE
   TYPE emp_table_type is table of
              employees%ROWTYPE INDEX BY BINARY_INTEGER;
   my_table emp_table_type;
   v_count NUMBER(3):= 130;
BEGIN
   FOR i IN 100..v_count
   LOOP
              SELECT * INTO my_table(i) FROM employees
              WHERE employee_id =  i;
   END LOOP;
   IF my_table.exists(99) THEN
      DBMS_OUTPUT.PUT_LINE(‘Khong ton tai trong bang’);
   ELSE
      DBMS_OUTPUT.PUT_LINE(my_table(106).last_name);
   END IF;
END;
/
```

```
SET SERVEROUTPUT ON
DECLARE
    TYPE emp_table_type is table of
                employees%ROWTYPE INDEX BY BINARY_INTEGER;
    my_table emp_table_type;
    v_count NUMBER(3):= 130;
    truoc   NUMBER(3):= 1;
    sau   NUMBER(3):= 1;
BEGIN
    FOR i IN 100..v_count     LOOP
                SELECT * INTO my_table(i) FROM employees  WHERE employee_id =  i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('TEST 1: '||my_table.count);
    DBMS_OUTPUT.PUT_LINE('Row first: '||my_table.first||' Row last: '||my_table.last);
    sau := my_table.NEXT(125);
    truoc := my_table.PRIOR(113);
    DBMS_OUTPUT.PUT_LINE('Row prior: '||truoc|| ‘ - Row next : ‘||sau);
END;
```

```
SET SERVEROUTPUT ON
DECLARE
   TYPE emp_table_type is table of
              employees%ROWTYPE INDEX BY BINARY_INTEGER;
   my_table emp_table_type;
   v_count NUMBER(3):= 130;
   truoc   NUMBER(3):= 1;
   sau   NUMBER(3):= 1;
BEGIN
   FOR i IN 100..v_count     LOOP
              SELECT * INTO my_table(i) FROM employees  WHERE employee_id =  i;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('TEST 1: '||my_table.count);
   DBMS_OUTPUT.PUT_LINE('Row first: '||my_table.first||' Row last: '||my_table.last);
--    my_table.DELETE; -- Xoa tat ca vung nho cap phat cho my_table
--    my_table.DELETE(102); -- Xoa dong 102
   my_table.DELETE(100,120); -- Xoa dong 100 --> 120
   DBMS_OUTPUT.PUT_LINE('TEST 2: '||my_table.count);
   DBMS_OUTPUT.PUT_LINE ('Row first: '||my_table.first||' Row last: '||my_table.last);
END;
```

ORACLE

```
SET SERVEROUTPUT ON
DECLARE
  TYPE CourseList IS TABLE OF VARCHAR2(20);
  courses CourseList;
BEGIN
  courses := CourseList ( 'Oracle9i SQL', 'Oracle FUNI', 'Oracle FUN II',
                          'Tuning', 'Oracle PL/SQL', 'Oracle Form',
                          'Oracle Report','Oracle 10g');
  dbms_output.put_line ('1. '||courses.count||' - '||courses(courses.last));
  courses.TRIM(2);
  dbms_output.put_line ('2. '||courses.count||' - '||courses(courses.last));
  courses.DELETE(courses.LAST);
  dbms_output.put_line ('3. '||courses.count||' - '||courses(courses.last));
  courses.TRIM(4);
  dbms_output.put_line('4. '||courses.count||' - '||courses(courses.last));
END;
```

Summary

In this lesson, you should have learned to:

- Define and reference PL/SQL variables of composite data types:
  - PL/SQL records
  - INDEX BY tables
  - INDEX BY table of records
- Define a PL/SQL record by using the %ROWTYPE attribute

Practice 5 Overview

This practice covers the following topics:

- Declaring INDEX BY tables
- Processing data by using INDEX BY tables
- Declaring a PL/SQL record
- Processing data by using a PL/SQL record