Working with Composite Data Types

5

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**

ORACLE

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]
```
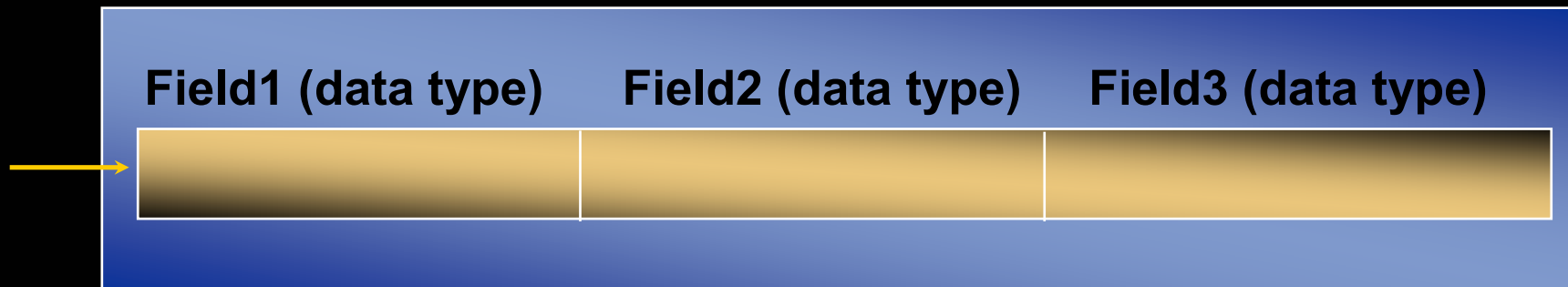
ORACLE

Creating a PL/SQL Record

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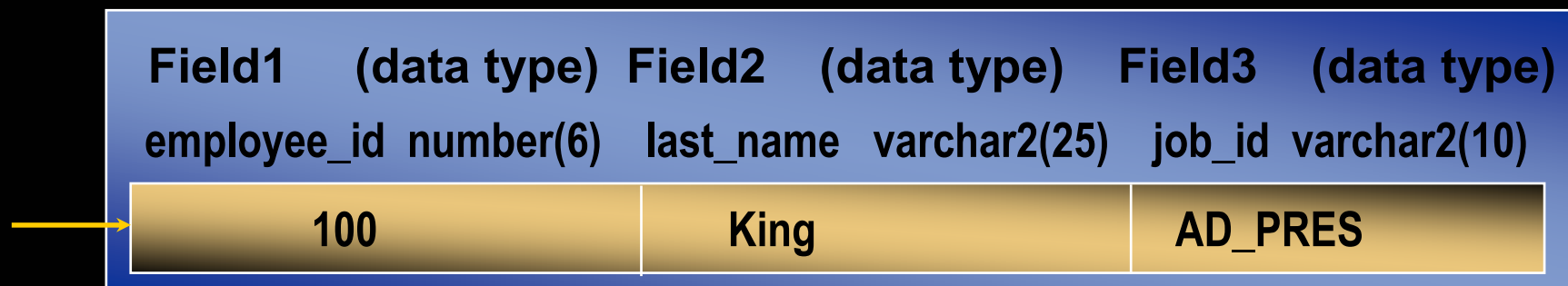
Example:

```
...
  TYPE emp_record_type IS RECORD
    (last_name    VARCHAR2(25),
     job_id       VARCHAR2(10),
     salary       NUMBER(8,2));
  emp_record       emp_record_type;
...
```

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.**

ORACLE

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.**

ORACLE

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;
```

ORACLE

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**

ORACLE

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

Unique identifier		Column	
...		...	
1		Jones	
2		Smith	
3		Maduro	
...		...	

`BINARY_INTEGER`   Scalar

Creating an `INDEX BY` Table

```
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
      INSERT INTO ...
    ...
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**

ORACLE

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):= 104;
BEGIN
  FOR i IN 100..v_count
  LOOP
        SELECT * INTO my_emp_table(i) FROM employees
        WHERE employee_id = i;
  END LOOP;
  FOR i IN my_emp_table.FIRST..my_emp_table.LAST
  LOOP
     DBMS_OUTPUT.PUT_LINE(my_emp_table(i).last_name);
  END LOOP;
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**

ORACLE

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

ORACLE

Latihan bab 5

1. Tulis blok PL/SQL untuk mencetak informasi tentang suatu Negara (diambil dari tabel COUNTRIES).

Keterangan :
* Gunakan perintah DEFINE untuk menampung country ID. Lewatkan ke blok PL/SQL melalui variable substitusi *i*SQL*Plus.
* Gunakan data bertipe record untuk menampung informasi
* Gunakan DBMS_OUTPUT.PUT_LINE untuk mencetak informasi tersebut.
* Gunakan untuk Negara dengan ID : CA, DE, UK, US.

2. Buat program PL/SQL yang menampilkan nama dari tiap departemen yang ada pada tabel DEPARTMENTS dengan menggunakan INDEX BY table.

Keterangan :
* Deklarasikan variabel MY_DEPT_TABLE bertipe INDEX BY table yang menyimpan informasi nama departemen sesuai dengan data yang ada pada tabel berikut :

Counter	DEPARTMENT_ID
1	10
2	20
3	50
4	60
5	80
6	90
7	110

jadi dalam tabel departements ada sebanyak 7 macam departemen dengan nomer departemen seperti yang tercantum pada tabel tersebut diatas
* Gunakan loop untuk menampung semua nama departemen untuk dimasukkan ke dalam variabel MY_DEPT_TABLE(*index*)
* Gunakan DBMS_OUTPUT.PUT_LINE untuk mencetak hasilnya.

3. Modifikasi program PL/SQL yang dibuat pada nomer 2 untuk menampilkan semua informasi pada tabel DEPARTMENTS meliputi nomer departemen, nama department dan lokasi departemen.

Keterangan :
* Gunakan variable MY_DEPT_TABLE yang bertipe INDEX BY table untuk menyimpan semua field dari tabel departemen
* Gunakan loop untuk menampung semua informasi
* Gunakan DBMS_OUTPUT.PUT_LINE untuk mencetak hasilnya.

4. Contoh program berikut menghapus data pegawai yang memiliki nomer pegawai dari manager (manager_id) seperti yang ada dalam daftar (kumpulan nilai manager_id).

```sql
SET SERVEROUTPUT ON
DECLARE
  TYPE dept_table_type IS TABLE OF
       DEPARTMENTS.DEPARTMENT_NAME%TYPE
       INDEX BY BINARY_INTEGER;
  my_dept_table dept_table_type;
  v_count NUMBER;
  v_deptno DEPARTMENTS.DEPARTMENT_ID%TYPE;
BEGIN
  SELECT COUNT(*) INTO v_count FROM DEPARTMENTS;
  FOR I IN 1.. v_count LOOP
     IF I=1 THEN
      v_deptno:=10;
     ELSIF I=2 THEN v_deptno:=20;
      /*   ....teruskan sendiri.....   */
     END IF;
     SELECT DEPARTMENT_NAME INTO my_dept_table(I) FROM
DEPARTMENTS WHERE DEPARTMENT_ID=v_deptno;

  END LOOP;
  FOR I IN 1..v_count LOOP
       /* TAMPILKAN MY_DEPT_TABLE DENGAN DBMS_OUTPUT.PUT_LINE
*/
      /* DBMS_OUTPUT.PUT_LINE(.......................   */

  END LOOP;
END;
/
```