



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]
```

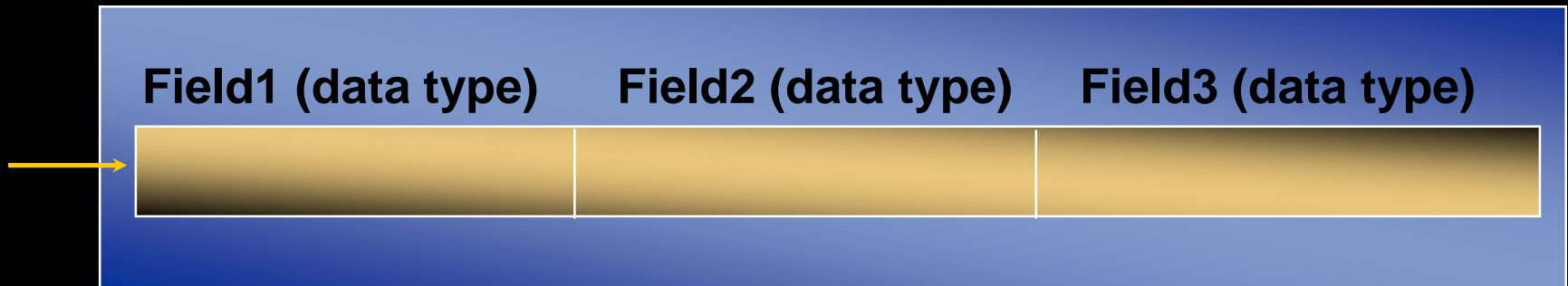
Creating a PL/SQL Record

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

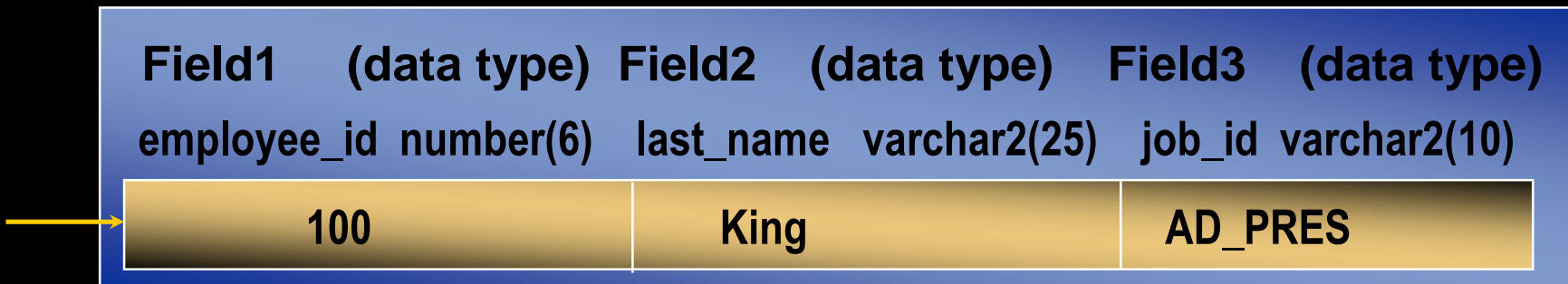
Example:

```
...  
    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



Example:



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;
```

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

Unique identifier

...
1
2
3
...

BINARY_INTEGER

Column

...
Jones
Smith
Maduro
...

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

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**

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**