Working with Composite Datatypes

Composite Datatypes

- Types:
  - PL/SQL RECORDS
  - Index By TABLES
- Contain internal components
- Are reusable

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
    (ename    VARCHAR2(10),
     job          VARCHAR2(9),
     sal          NUMBER(7,2));
  emp_record     emp_record_type;
...
```

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 datatypes from the columns of the table or view.

The %ROWTYPE Attribute

Examples

Declare a variable to store the same information about a department as it is stored in the DEPT table.

```
dept_record    dept%ROWTYPE;
```

Declare a variable to store the same information about an employee as it is stored in the EMP table.

```
emp_record     emp%ROWTYPE;
```

PL/SQL Tables

- Are composed of two components:
  - Primary key of datatype BINARY_INTEGER
  - Column of scalar or record datatype
- Increase dynamically because they are unconstrained

Creating a Index By Table

Syntax

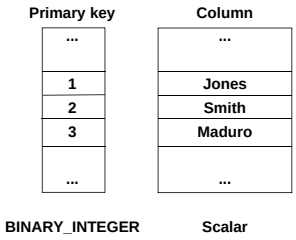
```
TYPE type_name IS TABLE OF
     {column_type | variable%TYPE
     | table.column%TYPE} [NOT NULL]
     [INDEX BY BINARY_INTEGER];
Variable_name  type_name;
```

Declare a (Index By table to store names.

Example

```
...
TYPE ename_table_type IS TABLE OF emp.ename%TYPE
  INDEX BY BINARY_INTEGER;
ename_table ename_table_type;
...
```

PL/SQL Table Structure



Creating a Index By Table

```
DECLARE
  TYPE ename_table_type IS TABLE OF emp.ename%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 PL/SQL tables easier to use:

- EXISTS
- COUNT
- FIRST and LAST
- PRIOR
- NEXT
- EXTEND
- TRIM
- DELETE

Index By Table of Records

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

Example

```
DECLARE
  TYPE dept_table_type IS TABLE OF dept%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

```
DECLARE
  TYPE e_table_type IS TABLE OF emp.Ename%Type
  INDEX BY BINARY_INTEGER;
  e_tab e_table_type;
BEGIN
  e_tab(1) := 'SMITH';
  UPDATE emp
  SET sal = 1.1 * sal
  WHERE Ename = e_tab(1);
  COMMIT;
END;
/
```

Summary

- Define and reference PL/SQL variables of composite datatypes:
  - PL/SQL records
  - PL/SQL tables
  - PL/SQL table of records
- Define a PL/SQL record by using the %ROWTYPE attribute.