

User-defined records

Indexing tables of records



Oracle Academy Study Materials

User-defined records

- PL/SQL record structures that correspond to the data fetched by a cursor use the %ROWTYPE attribute.

```
record_name      table_name%ROWTYPE;
```

```
DECLARE
  v_emp_record  employees%ROWTYPE;
BEGIN
  SELECT * INTO v_emp_record
    FROM employees
   WHERE employee_id = 100;
  DBMS_OUTPUT.PUT_LINE('Email for ' || v_emp_record.first_name ||
    ' ' || v_emp_record.last_name || ' is ' || v_emp_record.email ||
    '@oracle.com.');
```

```
END;
```

User-defined records

- You can use %ROWTYPE to declare a record based on another record:

```
DECLARE
  v_emp_record  employees%ROWTYPE;
  v_emp_copy_record  v_emp_record%ROWTYPE;
BEGIN
  SELECT * INTO v_emp_record
    FROM employees
   WHERE employee_id = 100;
  v_emp_copy_record := v_emp_record;

  v_emp_copy_record.salary := v_emp_record.salary * 1.2;

  DBMS_OUTPUT.PUT_LINE(v_emp_record.first_name ||
    ' ' || v_emp_record.last_name || ': Old Salary - ' ||
    v_emp_record.salary || ', Proposed New Salary - ' ||
    v_emp_copy_record.salary || '.');
END;
```

User-defined records

PL/SQL records:

- Must contain one or more components/fields of any scalar or composite type (at least one field and the fields may be defined using scalar data types such as DATE, VARCHAR2, or NUMBER, or using attributes such as %TYPE and %ROWTYPE)
- Are not the same as rows in a database table
- Can be assigned initial values and can be defined as NOTNULL
- Can be components of other records(nested records).

Syntax for User-Defined Records:

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

User-defined records

Example 1:

```
DECLARE
  TYPE person_dept IS RECORD
    (first_name      employees.first_name%TYPE,
     last_name       employees.last_name%TYPE,
     department_name departments.department_name%TYPE);
  v_person_dept_rec person_dept;
BEGIN
  SELECT e.first_name, e.last_name, d.department_name
  INTO v_person_dept_rec
    FROM employees e JOIN departments d
   ON e.department_id = d.department_id
   WHERE employee_id = 200;
  DBMS_OUTPUT.PUT_LINE(v_person_dept_rec.first_name ||
    ' ' || v_person_dept_rec.last_name || ' is in the ' ||
    v_person_dept_rec.department_name || ' department. ');
END;
```

User-defined records

Example 2:

```
DECLARE
  TYPE dept_info_type IS RECORD
    (department_id      departments.department_id%TYPE,
     department_name     departments.department_name%TYPE);
  TYPE emp_dept_type IS RECORD
    (first_name         employees.first_name%TYPE,
     last_name          employees.last_name%TYPE,
     dept_info          dept_info_type);

  v_emp_dept_rec       emp_dept_type;
BEGIN
  ...
END;
```

Indexing tables of records

When you need to temporarily store multiple rows of data, you can use collections.

A Collection is:

- a named set of many occurrences of the same kind of data stored as a variable.
- a type of composite variable, similar to user-defined records.
- Two main collections can be distinguished:
 - INDEX BY tables, and
 - INDEX BY tables of records.
- There are other types of collection variables, for instance, Nested Tables and Varrays.
- Because collections are PL/SQL variables, they are stored in memory like other PL/SQL variables.
- They are not stored on the disk like data in a database table.

Indexing tables of records

- An INDEX BY Table Has a Primary Key – we need to be able to reference each row in an INDEX BY table.
- The primary key is typically a `BINARY_INTEGER`, but it may be a `VARCHAR2`.

Primary Key	Value
...	...
100	Jones
157	Smith
135	Maduro
...	...
<code>BINARY_INTEGER</code>	Scalar

Indexing tables of records

- First a type must be declared and then a variable of that type.
- The syntax is:

```
TYPE type_name IS TABLE OF DATA_TYPE  
    INDEX BY PRIMARY_KEY_DATA_TYPE;  
identifier    type_name;
```

- Example:

```
TYPE t_hire_date IS TABLE OF DATE  
    INDEX BY BINARY_INTEGER;  
v_hire_date_tab    t_hire_date;
```

Indexing tables of records

- The syntax to populate the INDEX BY table:

```
DECLARE
  TYPE type_name IS TABLE OF DATA_TYPE
  INDEX BY PRIMARY_KEY_DATA_TYPE;
  identifier    type_name;
BEGIN
  FOR record IN (SELECT column FROM table)
  LOOP
    identifier(primary_key) := record.column;
  END LOOP;
END;
```

- The primary key can be initialized using a unique column from the selected table or an incrementing integer.

Indexing tables of records

Example 1:

```
DECLARE
  TYPE t_hire_date IS TABLE OF employees.hire_date%TYPE
  INDEX BY BINARY_INTEGER;
  v_hire_date_tab    t_hire_date;
BEGIN
  FOR emp_rec IN
    (SELECT employee_id, hire_date FROM employees)
  LOOP
    v_hire_date_tab(emp_rec.employee_id)
      := emp_rec.hire_date;
  END LOOP;
END;
```

Indexing tables of records

Example 2:

```
DECLARE
  TYPE t_hire_date IS TABLE OF employees.hire_date%TYPE
  INDEX BY BINARY_INTEGER;
  v_hire_date_tab    t_hire_date;
  v_count BINARY_INTEGER := 0;
BEGIN
  FOR emp_rec IN
    (SELECT hire_date FROM employees)
  LOOP
    v_count := v_count + 1;
    v_hire_date_tab(v_count) := emp_rec.hire_date;
  END LOOP;
END;
```

Using INDEX BY Table Methods

- There are built-in procedures and functions (called methods) to reference single elements of the INDEX BY table, or to read successive elements.
- The available methods are:

EXISTS	PRIOR
COUNT	NEXT
FIRST	DELETE
LAST	TRIM

- They can be used by dot-prefixing the method-name with the table-name.

Using INDEX BY Table Methods

- Example of the method COUNT:

```
DECLARE
    TYPE t_hire_date IS TABLE OF employees.hire_date%TYPE
    INDEX BY BINARY_INTEGER;
    v_hire_date_tab    t_hire_date;
    v_hire_date_count  NUMBER(4);
BEGIN
    FOR emp_rec IN
        (SELECT employee_id, hire_date FROM employees)
    LOOP
        v_hire_date_tab(emp_rec.employee_id)
            := emp_rec.hire_date;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(v_hire_date_tab.COUNT);
END;
```

INDEX BY table of records

- INDEX BY table can have only one data field, which can be a composite data type such as a RECORD – It is called an INDEX BY table of records.
- The record can be %ROWTYPE or a user-defined record.
- Example:

```
DECLARE
    TYPE t_emp_rec IS TABLE OF employees%ROWTYPE
    INDEX BY BINARY_INTEGER;
    v_employees_tab    t_emprec;
```

- Individual fields within a table of records can be referenced by adding an index value in parentheses after the table of records name.
- Syntax: table(index).field
- Example: v_employees_tab(index).hire_date
- The index value in the example could be an actual value (ex. 1, 5, 12, etc.) or a reference to a value (v_emp_rec_tab.LAST).

INDEX BY table of records

– Example:

```
DECLARE
  TYPE t_emp_rec IS TABLE OF employees%ROWTYPE
  INDEX BY BINARY_INTEGER;
  v_emp_rec_tab t_emp_rec;
BEGIN
  FOR emp_rec IN (SELECT * FROM employees) LOOP
    v_emp_rec_tab(emp_rec.employee_id) := emp_rec;
    DBMS_OUTPUT.PUT_LINE(
      v_emp_rec_tab(emp_rec.employee_id).salary);
  END LOOP;
END;
```