Database Programming with PL/SQL

Indexing Tables of Records

Objectives

This lesson covers the following objectives:

- Create an `INDEX BY` table
- Create an `INDEX BY` table of records
- Describe the difference between records, tables, and tables of records

Purpose

You have learned that you can store a whole record in a single variable, either by using `%ROWTYPE` or by creating your own record structure as a type and then declaring a variable of that type.

Wouldn't it be even better to be able to store many whole records in a single variable?

In this lesson, you learn how to define and use collections. A PL/SQL collection is a named set of many occurrences of the same thing.

What is a Collection?

A collection is a set of occurrences of the same kind of data. For example, the set of all employees' last names. Or, the set of all department rows.

In PL/SQL, a collection is a type of composite variable, just like user-defined records and `%ROWTYPE`.

What is a Collection? (cont.)

You see two kinds of collections in this lesson:

- An INDEX BY TABLE, which is based on a single field or column; for example, on the `last_name` column of `EMPLOYEES`.
- An INDEX BY TABLE OF RECORDS, which is based on a composite record type; for example, on the whole `DEPARTMENTS` row.

Because collections are PL/SQL variables, their data is stored in a private memory area like any other PL/SQL variable.

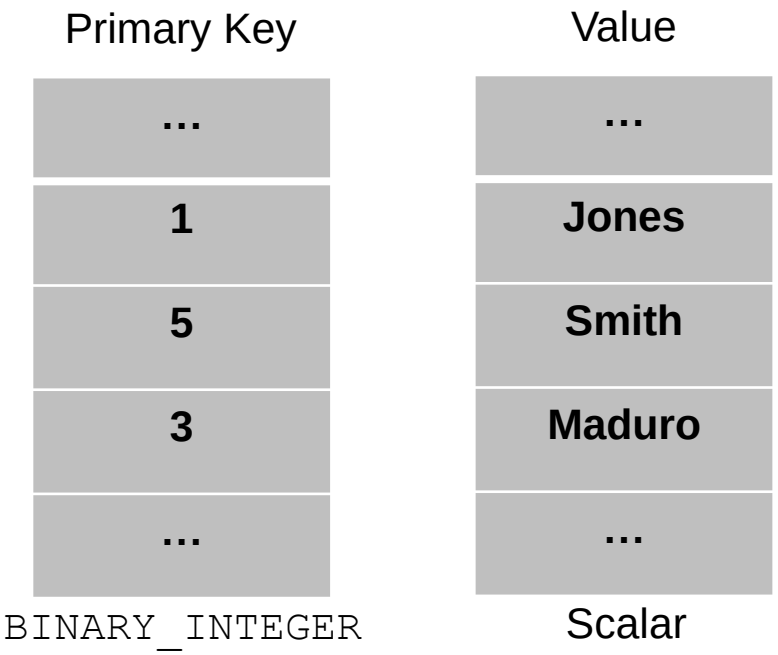
An INDEX BY Table Has a Primary Key

You need to able to distinguish between individual values in the table, so that you can reference them individually. Therefore, every `INDEX BY` table must have a primary key, of any valid Oracle data type, which serves as an index into the table.

The primary key is most often of datatype `BINARY_INTEGER` (the default) or `PLS_INTEGER`. The primary key can be negative as well as positive.

INDEX BY Table Structure

The primary key could be meaningful business data such as an employee id.



Declaring an INDEX BY Table

Like user-defined records, you must first declare a type and then declare “real” variables of that type.

This example declares two INDEX BY tables of the same type.

```
DECLARE TYPE t_names IS TABLE OF VARCHAR2(50)
                        INDEX BY BINARY_INTEGER;
  last_names_tab   t_names;
  first_names_tab  t_names;
```

Populating an INDEX BY Table

This example populates the INDEX BY table with employees' last names, using `employee_id` as the primary key.

```
DECLARE
 TYPE t_names IS  TABLE OF VARCHAR2(50)
                  INDEX BY BINARY_INTEGER;
 last_names_tab   t_names;
BEGIN
 FOR emp_rec IN (SELECT employee_id, last_name
                   FROM employees) LOOP
   last_names_tab(emp_rec.employee_id) := emp_rec.last_name;
  END LOOP;
END;
```

Using `INDEX BY` Table Methods

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

EXISTS	PRIOR
COUNT	NEXT
FIRST	DELETE
LAST	TRIM

You use these methods by dot-prefixing the method-name with the table-name.

Using INDEX BY Table Methods (cont.)

```
DECLARE
 TYPE t_names IS  TABLE OF VARCHAR2(50)
                  INDEX BY BINARY_INTEGER;
 last_names_tab   t_names;
 v_count          INTEGER;
BEGIN
 -- populate the INDEX BY table with employee data as before
  v_count := last_names_tab.COUNT;                      --1
  FOR i IN last_names_tab.FIRST .. last_names_tab.LAST --2
   LOOP
    IF last_names_tab.EXISTS(i) THEN                    --3
      DBMS_OUTPUT.PUT_LINE(last_names_tab(i));
    END IF;
   END LOOP;
END;
```

INDEX BY Table of Records

Even though an index by table can have only one data field, that field can be a composite data type, such as a `RECORD`.

The record can be `%ROWTYPE` or a user-defined record. This example declares an INDEX BY table to store complete employee rows:

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

Using an INDEX BY Table of Records

```
DECLARE
  TYPE t_emprec IS  TABLE OF employees%ROWTYPE
                    INDEX BY BINARY_INTEGER;
 employees_tab   t_emprec;
BEGIN
  FOR emp_rec IN (SELECT * FROM employees) LOOP
   employees_tab(emp_rec.employee_id) := emp_rec;          --1
  END LOOP;
 FOR i IN employees_tab.FIRST .. employees_tab.LAST
   LOOP
    IF employees_tab.EXISTS(i) THEN
       DBMS_OUTPUT.PUT_LINE(employees_tab(i).first_name); --2
    END IF;
   END LOOP;
END;
```

Using an INDEX BY Table of Records: A Second Example

We've copied the whole `DEPARTMENTS` table to another variable with a single statement.

```
DECLARE
  TYPE t_deptrec IS   TABLE OF departments%ROWTYPE
                      INDEX BY BINARY_INTEGER;
 departments_tab      t_deptrec;
 departments_tab_new  t_deptrec.
BEGIN
  FOR dept_rec IN
   (SELECT * FROM departments WHERE department_id = 20) LOOP
   departments_tab(dept_rec.department_id) := dept_rec;
  END LOOP;
  departments_tab_new := departments_tab;
  ...
END;
```

Terminology

Key terms used in this lesson included:

- Collection
- `INDEX BY TABLE`
- `INDEX BY TABLE OF RECORDS`

Summary

In this lesson, you should have learned how to:

- Create an INDEX BY table
- Create an INDEX BY table of records
- Describe the difference between records, tables, and tables of records