

# Index by Tables of Records

## What Will I Learn?

In this lesson, you will learn to:

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

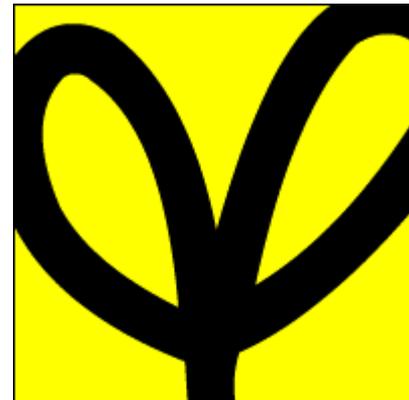


## Why Learn It?

You have learned that you can pass a whole record between your PL/SQL subprogram and the calling environment, using a single parameter.

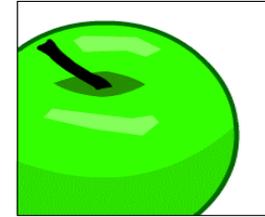
Wouldn't it be more efficient to pass many records at the same time?

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.



# Tell Me/Show Me

## 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`. 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.



## Tell Me/Show Me

### An INDEX BY Table Has a Primary Key

You need to be able to distinguish between individual values in the table, so that you can reference them individually. Therefore, every `INDEX BY` table automatically has a numeric primary key, which serves as an index into the table.

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

 **Tell Me/Show Me****INDEX BY Table Structure****Primary Key**

...
1
5
3
...

**BINARY\_INTEGER****Value**

...
Jones
Smith
Maduro
...

**Scalar**

The primary key must be numeric, but could be meaningful business data, for example an employee id.



# Tell Me/Show Me

## Declaring an INDEX BY Table

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

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.



# Tell Me/Show Me

## Populating an INDEX BY Table

```
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;
```

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



# Tell Me/Show Me

## 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
- COUNT
- FIRST and LAST
- PRIOR
- NEXT
- DELETE
- TRIM

You use these methods by dot-prefixing the method-name with the table-name. The next slide shows some examples.



# Tell Me/Show Me

## Using INDEX BY Table Methods (continued)

```
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;
```



# Tell Me/Show Me

## 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;
```



# Tell Me/Show Me

## 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;
```

# Tell Me/Show Me

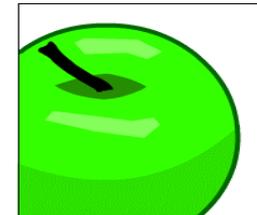
## Terminology

Key terms used in this lesson include:

Collection

INDEX BY TABLE

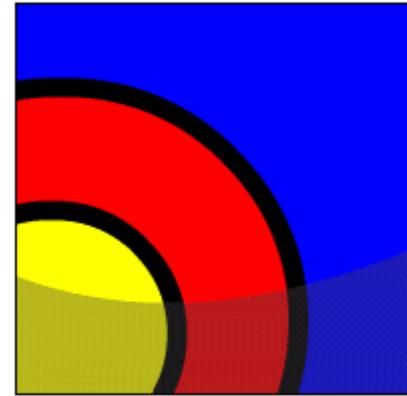
INDEX BY TABLE OF RECORDS



## Summary

In this lesson, you learned to:

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



## Try It/Solve It

This practice covers the following topics:

- Creating an INDEX BY table
- Creating an INDEX BY table of records
- Describing the difference between records, tables, and tables of records

