Tipos de Datos en PL/SQL

Tipos de datos definidos por el usuario

- Los tipos de datos incorporados en ORACLE son escalares.
- El usuario puede definir tipos compuestos a fin de manejar datos relacionados como una sola unidad

Tipos de Datos Compuesto

1. **Registros (Records) de tipo PL/SQL**
   - **Contienen componentes internos**

2. **Las Colecciones: Permiten manejar muchas variables de un solo tipo como una sola unidad (arrays, conjuntos o listas). PL/SQL proporciona varios tipos de colecciones: Tablas y Varrays. En particular veremos la implementación de Tablas Indexadas. Una tabla indexada se caracteriza por:**
   - **Un índice de tipo BINARY_INTEGER**
   - **Columnas de tipo escalar**

1. REGISTROS PL/SQL

- Un registro es un grupo de campos, cada uno con su propio tipo de dato que puede tratarse como una sola unidad lógica.
- No es lo mismo que la estructura de un 'registro' de una tabla en la BD, pero se parece a los campos de tipo %ROWTYPE.

Campo1	Campo2	Campo3

SINTAXIS PARA CREACIÓN DE REGISTROS

```
TYPE  nombre_tipo  IS RECORD

(campo1  tipo_dato [NOT NULL {:=
| DEFAULT} expr],  campo2
tipo_dato);
```

Un registro:

- Puede tener tantos campos como sea necesario
- Se puede asignar un valor inicial y pueden ser definidos como NOT NULL
- Si no se asigna valor a un campo es inicializado a NULL

EJEMPLO DE CREACIÓN DE REGISTROS

```
DECLARE
TYPE  reg-emp  IS RECORD
(cedula  NUMBER NOT NULL,
     nombre VARCHAR2(30),
     apellido VARCHAR2(30));
      var-emp reg-emp;
BEGIN
      var-emp.cedula := 239494;
END;
```

2. TABLAS INDEXADAS

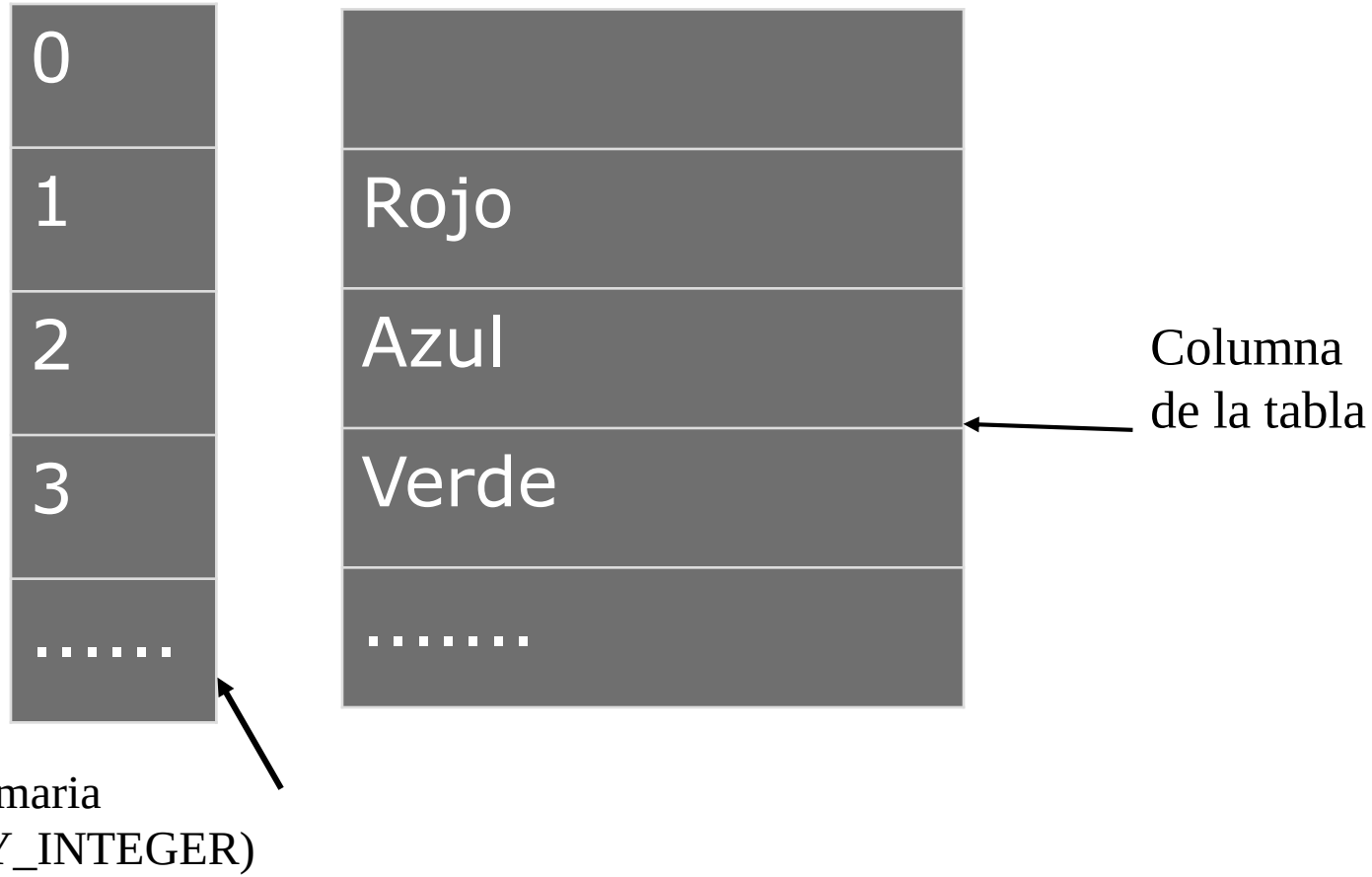
- **La tabla indexada (o tabla asociativa) es básicamente un ARRAY de una sola columna.**
- **No es igual a una TABLA de BD!!**
- **Puede crecer dinámicamente pues no se pre establece su dimensión**

TABLAS INDEXADAS

Una tabla indexada se caracteriza por:

- **Un índice de tipo BINARY_INTEGER, PLS_INTEGER, o VARCHAR2**
- **Columnas de tipo escalar**

Representación gráfica de una tabla PL/SQL



SINTAXIS PARA CREACIÓN DE TABLAS PL/SQL

```
TYPE nombre_tipo IS TABLE OF
tipo_de_dato

[NOT NULL] INDEX BY BINARY_INTEGER
```

Algo más sobre las tablas PL/SQL

- Los elementos de una tabla pl/sql existen sólo cuando se les asigna un valor. Así mismo, los elementos no tienen que crearse necesariamente de manera consecutiva.

- Como las tablas son exclusivas de PL/SQL no pueden utilizarse en operaciones de SELECT u operaciones de tipo DML.

EJEMPLO DE APLICACIÓN

```
DECLARE
TYPE tabla_color  IS  TABLE OF
VARCHAR2(15)INDEX  BY BINARY_INTEGER;
varcolor     tabla_color;
BEGIN
     varcolor(0) := 'ROJO';
     varcolor(1) := 'AMARILLO';
     varcolor(2) := 'AZUL';
     FOR I IN 0..2 LOOP
        DBMS_OUTPUT.PUT_LINE(varcolor(I));
     END LOOP;
END;
```

Algunas operaciones sobre tablas

- Asignar un valor:

```
tab (1) := 23;
```

- Borrar todas las filas de la tabla:

```
tab.DELETE;
```

- Determinar el último valor del índice en el cual se asignó un elemento en la tabla:

```
i = tab.LAST;
```

(LAST devuelve NULL si la tabla no tiene elementos)

Usando Métodos de Tablas

- Los siguientes métodos hacen a las tablas mas fáciles de usar
  - EXISTS
  - DELETE
  - FIRST AND LAST
  - PRIOR
  - NEXT
  - COUNT

Syntaxis

```
Nombre_tabla.nombre_método[(parámetros)]
```

Método	Descripción
EXISTS (n)	Retorna VERDADERO si existe un elemento en la posición **n** de la tabla PL/SQL .
COUNT	Retorna el numero de elementos que contiene la tabla PL/SQL
FIRST / LAST	Retorna el índice del primer y el último elemento tabla PL/SQL. Retorna NULO si la tabla PL/SQL esta vacía.
PRIOR (n)	Retorna el numero de índice anterior al índice **n** en la tabla PL/SQL
NEXT (n)	Retorna el numero de índice que sigue al índice **n** en la tabla PL/SQL
TRIM	TRIM elimina un elemento del final de la tabla PL/SQL. TRIM (n): elimina **n** elementos al final de la tabla PL/SQL.
DELETE	DELETE elimina todos los elementos de la tabla PL/SQL. DELETE (n) elimina el **n-avo** elemento de la tabla PL/SQL. DELETE (m, n) elimina todos los elementos en el rango m..n de la tabla PL/SQL.

Ejemplo: Definición de una tabla de registros

```
DECLARE
        TYPE  reg-emp  IS RECORD
        (id  NUMBER NOT NULL, nombre VARCHAR2(30),
        apellido VARCHAR2(30));

        TYPE tab-emp is table of reg-emp index by
         binary_integer ;
        var_tab_emp tab-emp;
BEGIN
        FOR i in 1..100 LOOP
                var_tab_emp(i).id := i
                .
                .
        END LOOP;
END;
```