TIPOS DE DATOS AVANZADOS

Existen en PL/SQL otros tipos de datos que el usuario puede crear.

Estos son:

- Registros
- Tablas

1. REGISTROS

DECLARACIÓN DE UN REGISTRO.

Los registros son una estructura estática de datos presente en casi todos los lenguajes clásicos. Se trata de un tipo de datos que se compone de datos más simple.
Por ejemplo el registro ***persona*** se compondría de los datos simples ***nombre, apellidos, dirección, fecha de nacimiento,*** etc.
En PL/SQL su interés radica en que cada fila de una tabla o vista se puede interpretar como un registro, ya que cada fila se compone de datos simples.
Los campos de un registro pueden ser inicializados y pueden ser definidos como NOT NULL. Aquellos campos que no sean inicializados explícitamente, se inicializarán a NULL.

SINTAXIS

La sintaxis general para definir un tipo de registro es:

```
TYPE tipo_registro IS RECORD (
Campo1 Tipo1 [NOT NULL] [:= expr1],
Campo2 Tipo2 [NOT NULL] [:= expr2],
Campo3 Tipo3 [NOT NULL] [:= expr3],
…… … .. …… ……
Campo(n) Tipo(n) [NOT NULL] [:= expr(n)]);
```

Es típico en el trabajo con BD el declarar registros con el mismo formato que las filas de las tablas (%TYPE)

El siguiente ejemplo crea un tipo PAIS, que tiene como campos el código, el nombre y el continente.

```
TYPE PAIS IS RECORD
(
CO_PAIS NUMBER ,
NOMBRE VARCHAR2(50),
CONTINENTE VARCHAR2(20)

);
```

Los registros son un tipo de datos, por lo que se puede declarar variables de dicho tipo de datos.

Para hacer referencia a los campos de un registro se utiliza la notación punto (.)

```
Nombre_registro.nombrecampo
```

EJEMPLO:

```
DECLARE

TYPE PAIS IS RECORD
(
CO_PAIS NUMBER ,
NOMBRE VARCHAR2(50),
CONTINENTE VARCHAR2(20)
);
miPAIS PAIS;
BEGIN
/* Asignamos valores a los campos de la variable.

*/
miPAIS.CO_PAIS := 27;
miPAIS.NOMBRE := 'ITALIA';
miPAIS.CONTINENTE := 'EUROPA';

END;
```

EJEMPLO DE REGISTRO CON BASE DE DATOS

LIBRO

COD_LIBRO	TITULO	ID_AUTOR	AÑO	NOM_EDITORIAL	PRECIO
102	Los tres mosqueteros	10	2003	Altazor	15700
101	El conde de Montecristo	10	2004	Antares	12500
103	Las Aventuras de Tom Sawyer	11	2002	Contrapunto	17800
104	cantos de Vida y Esperanza	12	2002	Antares	19800
105	Los Miserables	13	2003	Contrapunto	16700

```
DECLARE

TYPE LIBRO_CAMPOS IS RECORD (
CODIGO_LIBRO libro.cod_libro%type ,
NOMBRE_LIBRO libro.titulo%type,
NOMBRE_EDITORIAL libro.nom_editorial%type);

LIB LIBRO_CAMPOS;
BEGIN/* Asignamos valores a los campos de la variable.*/

SELECT cod_libro,titulo,nom_editorial into LIB
from libro
where  precio= 15700

dbms_output.put_line(lib.nombre_libro);
END;
```

2. TABLAS

Las tablas de PL/SQL son tipos de datos que permiten almacenar varios valores del mismo tipo de datos.

Una tabla PL/SQL:

1. Es similar a un array
2. Tiene dos componentes: Un índice de tipo BINARY_INTEGER que permite acceder a los elementos en la tabla PL/SQL y una columna de escalares o registros que contiene los valores de la tabla PL/SQL
3. Puede incrementar su tamaño dinámicamente.

La sintaxis general para declarar una tabla de PL/SQL es la siguiente:

```
TYPE <nombre_tipo_tabla> IS TABLE OF
<tipo_datos> [NOT NULL]
INDEX BY BINARY_INTEGER ;
```

El rango de binary integer es –2147483647.. 2147483647, por lo tanto el índice puede ser negativo, lo cual indica que el índice del primer valor no tiene que ser necesariamente el uno.

Una vez que se ha definido el tipo, se pueden declarar variables y asignar valores.

```
DECLARE
/* Definimos el tipo PAISES como tabla PL/SQL */

TYPE PAISES IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;

/* Declaramos una variable del tipo PAISES */
tPAISES PAISES;
BEGIN
tPAISES(1) := 10;
tPAISES(2) := 20;
tPAISES(3) := 30;
END;
```

FUNCIONES PARA EL MANEJO DE TABLAS PL/SQL

Cuando trabajamos con tablas de PL podemos utilizar las siguientes funciones:

- **FIRST**: Devuelve el menor índice de la tabla. NULL si está vacía.
- **LAST**: Devuelve el mayor índice de la tabla. NULL si está vacía.
- **EXISTS**(i): Utilizada para saber si en un cierto índice hay almacenado un valor. Devolverá TRUE si en el índice i hay un valor.
- **COUNT**: Devuelve el número de elementos de la tabla PL/SQL.
- **PRIOR** (n): Devuelve el número del índice anterior a n en la tabla.
- **NEXT** (n): Devuelve el número del índice posterior a n en la tabla.
- **TRIM:** Borra un elemento del final de la tabla PL/SQL.
- **TRIM(n):** borra n elementos del final de la tabla PL/SQL.
- **DELETE:** borra todos los elementos de la tabla PL/SQL.
- **DELETE(n):** borra el correspondiente al índice n.
- **DELETE(m,n) :** borra los elementos entre m y n.