4. REGISTROS Y TABLAS:

4.1.- INTRODUCCIÓN

Los tipos de datos compuestos son:

- **Registro**: grupo de elementos almacenados en campos, cada uno con su propio nombre y tipo de datos. Se utiliza para tratar datos relacionados pero distintos, como una unidad lógica.
- **Tablas**: contienen una columna y una clave primaria (índice), para dar acceso de tipo vector a determinadas filas. Se utiliza para hacer referencia y manipular las colecciones de datos como un objeto completo.

4.2.- REGISTROS

Los registros presentan las siguientes características:

- Los registros deben contener una o más componentes llamados *campos*.
- No es lo mismo que la fila de una tabla.
- Tratan una colección de campos como unidad lógica.
- Son adecuados para recuperar una fila de datos de una tabla.
- Se declaran en la zona *declare*.

Sintaxis:

```
TYPE nb_tipo_reg IS RECORD
(field_declaration [, field_declaration] …);

nb_reg nb_tipo_reg;
```

Donde *field_declaration* tienen la siguiente estructura:

```
Nb_campo {tip_dato_campo | variable%TYPE | tabla.column%TYPE | tabla%ROWTYPE}
[[NOT NULL]{:= | DEFAULT} expr ]
```

Donde:

NB_TIPO_REG: es el nombre del tipo de registro.
NB_REG: es el nombre del registro.
NB_CAMPO: nombre del campo que está dentro del registro.
TIPO_DATO_CAMPO: tipo de dato de un campo.
EXPR: valor inicial.

Ejemplo 1:

Inicializar un registro ALUMNO con los siguientes valores nombre, número de matricula y código en la zona declarativa.

```
TYPE REG_ALUMNO IS RECORD
(NOM_ ALUMNO VARCHAR2(25)
NUM_MATRI NUMBER(7,2),
CODIGO NUMBER(3));

ALUMNO REG_ALUMNO;
```

Ejemplo 2:

Inicializar un registro DEPARTAMENTO con el código de departamento en la zona declarativa.

```
TYPE REG_DEPT IS RECORD
(DEPTNO DEPT.DEPTNO%TYPE);

DEPARTAMENTO REG_DEPT;
```

Ejemplo 3:

Crear un bloque PL/SQL que mediante un cursor seleccione de la tabla DEPT el campo DEPTNO y DNAME, lo cargue en un registro y visualice los campos del registro posteriormente. Se introducirá un valor por teclado que indicará el número de filas del cursor que se deben ir cargando en el registro y visualizando. El orden de salida será de menor a mayor.

```
DECLARE
        CURSOR C1 IS
        SELECT DEPTNO, DNAME
        FROM DEPT
        ORDER BY DEPTNO;

        TYPE REGISTRO IS RECORD
                (DEPTNO DEPT.DEPTNO%TYPE,
                DNAME DEPT.DNAME%TYPE);

        REG REGISTRO;

        V_NUM NUMBER:=&NUMERO_FILA;

BEGIN
        OPEN C1;
        FETCH C1 INTO REG;
        WHILE (C1%ROWCOUNT<=V_NUM) AND (C1%FOUND)LOOP
                DBMS_OUTPUT.PUT_LINE(REG.DEPTNO||''||REG.DNAME);
                FETCH C1 INTO REG;
```

```
        END LOOP;
        CLOSE C1;
END;
/
```

4.3.- ATRIBUTO %ROWTYPE

Para declarar un registro basándose en una colección de columnas de una tabla o vista o cursor se utiliza al atributo %ROWTYPE

Los campos del registro tomarán automáticamente sus nombres y tipos de datos de la columna de la tabla, vista o cursor. El registro almacena por tanto, una fila entera de los elementos.

Sintaxis:

```
Nb_registro objeto_referenciado%ROWTYPE;
```

Ventajas del atributo %ROWTYPE

- El número y tipo de datos de las columnas de la base de datos pueden no ser conocidos.
- El número y tipos de datos de las columnas de la base de datos pueden cambiar en el momento de la ejecución.
- Es útil para recuperar una fila con la sentencia SELECT.

Ejemplo:

Realizar el bloque anterior utilizando todos los campos de la tabla y el atributo %ROWTYPE:

```
DECLARE
        CURSOR C1 IS
        SELECT * FROM DEPT
        ORDER BY DEPTNO;

        REG C1%ROWTYPE;

        V_NUM NUMBER:= &NÚMERO_FILAS;

BEGIN
        OPEN C1;
        FETCH C1 INTO REG;
        WHILE (C1%ROWCOUNT <= V_NUM) AND (C1%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE     (REG.DEPTNO||' '||REG.DNAME||'
                '||REG.LOC);
                FETCH C1 INTO REG;
        END LOOP;
        CLOSE C1;
END;
```