Tablas anidadas

- Una tabla anidada se compone de un conjunto de elementos del mismo tipo.
- Está contenida en una columna.
- La columna debe ser de tipo objeto (definido previamente).
- No es necesario indicar el tamaño máximo de la tabla anidada.

```
## Creación
CREATE TYPE nombre_tipo AS TABLE OF tipo_de_dato;

## Por ejemplo
CREATE TYPE TABLA_ANIDADA AS TABLE OF DIRECCION;

## Creación de tabla con tabla anidada
CREATE TABLE EJEMPLO_TABLA_CONTENEDORA_ANIDADA (
    ID NUMBER(2),
    APELLIDOS VARCHAR2(40),
    DIREC TABLA_ANIDADA
)
NESTED TABLE DIREC STORE AS DIREC_ANIDADA
```

- **NESTED TABLE** identifica la columna que tendrá la tabla anidada
- **STORE AS** especifica el nombre de la tabla (DIREC_ANIDADA) donde se almacenarán las direcciones representadas por el atributo DIREC de cualquier objeto de la tabla EJEMPLO_TABLA_CONTENEDORA_ANIDADA

Inserción en tablas anidadas

```
INSERT INTO EJEMPLO_TABLA_CONTENEDORA_ANIDADA VALUES( 1, 'PEREZ',
    TABLA_ANIDADA (
            DIRECCION ('Avenida Ciudad de Soria 17', 'ZARAGOZA', 50016),
            DIRECCION ('Calle Asalto, 23', 'ZARAGOZA', 50002),
            DIRECCION ('Avenida de Madrid, 220', 'ZARAGOZA', 50010)
        )
)

INSERT INTO EJEMPLO_TABLA_CONTENEDORA_ANIDADA VALUES( 2, 'MARTINEZ',
    TABLA_ANIDADA (
            DIRECCION ('Avenida Cataluña, 17', 'ZARAGOZA', 50016),
            DIRECCION ('Paseo Fernando el Católico, 22', 'CALATAYUD', 50300)
            DIRECCION ('Avenida de Navarra, 180', 'ZARAGOZA', 50010)
        )
)

## También se puede insertar registros con la tabla anidada vacía
INSERT INTO EJEMPLO_TABLA_CONTENEDORA_ANIDADA VALUES( 3, 'FERNANDEZ',
    TABLA_ANIDADA ()
)

## Recupera todos los registros de la tabla anidada, el id y apellidos de la
principal.
SELECT ID, APELLIDOS, DIRECCION.*
```

```
    FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA, TABLE(DIREC) DIRECCION;
```

Consultas con tablas anidadas

- El operador *TABLE* con la columna que es tabla anidada entre paréntesis y colocado a la derecha del FROM se emplea para acceder a todas las filas de la tabla anidada. Es necesario poner un alias a la tabla anidada.
- Con *alias*.* se obtienen todos los campos de la tabla anidada.
- Es posible establecer cursores en una consulta para acceder o establecer condiciones a las filas de una tabla anidada.

```
## columnas son las columnas del tipo de dato de la tabla anidada.

CURSOR (SELECT columnas FROM TABLE (COLUMNA_TABLA_ANIDADA));

## tabla anidada en cursor
SELECT ID, APELLIDOS, CURSOR(SELECT TD.CALLE, TD.CIUDAD FROM TABLE(DIREC) TD )
FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA;

## Tabla anidada como tabla normal

SELECT ID, APELLIDOS, COUNT(*)
FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA, TABLE(DIREC) TD
GROUP BY ID, APELLIDOS;

## EJEMPLO, aquellos registros que tienen dos direcciones en Zaragoza
SELECT ID, APELLIDOS, CURSOR (SELECT COUNT(*) FROM TABLE(DIREC) TD WHERE TD.CIUDAD
= 'ZARAGOZA')
FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA
WHERE (SELECT COUNT(*) FROM TABLE(DIREC) TD WHERE TD.CIUDAD = 'ZARAGOZA') = 2;
```

- Para seleccionar filas de una tabla anidada se puede usar la cláusula *THE* con *SELECT*.

```
SELECT ... FROM THE (subconsulta tabla anidada) WHERE ...

SELECT CALLE
    FROM THE (SELECT DIREC FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA WHERE ID = 2)
WHERE CIUDAD = 'CALATAYUD'
```

- La cláusula *TABLE* a la derecha de INTO se usa para acceder a la fila que interesa.

```
INSERT INTO TABLE
    (SELECT DIREC FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA WHERE ID = 1)
    VALUES(DIRECCION('Calle Bilbao 32', 'Madrid', 28021));
```

- Modificación de registros en tablas anidadas.
    - El alias recoge los datos devueltos por el SELECT (una fila en este caso).
    - Mediante SET VALUE(ALIAS) se asigna el valor nuevo al objeto cuyo valor coincida con el del filtro.

```
UPDATE TABLE
    (SELECT DIREC FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA WHERE ID = 1) PRIMERA
SET VALUE(PRIMERA) = DIRECCION('Calle Bailén, 23', 'Madrid', 28014)
WHERE VALUE(PRIMERA) = DIRECCION ('Avenida Ciudad de Soria 17', 'ZARAGOZA', 50016);
```

- Eliminación en tablas anidadas

```
DELETE FROM TABLE
(SELECT DIREC FROM EJEMPLO_TABLA_CONTENEDORA_ANIDADA WHERE ID = 1) PRIMERA
WHERE
VALUE(PRIMERA)=DIRECCION('Avenida de Madrid, 220', 'ZARAGOZA', 50010);

DELETE FROM TABLE
(SELECT DIREC FROM EJEMPLO_TABLA_ANIDADA WHERE ID = 1) PRIMERA
WHERE
 PRIMERA.CIUDAD ='CALATAYUD';
```

- La vista **USER_NESTED_TABLES** devuelve información sobre las tablas anidadas.

```
## Se define una función que comprueba si existe una dirección en un identificador
concreto. La función recibe el identificador y una dirección e indica si existe o
no.

CREATE OR REPLACE FUNCTION EXISTE_DIREC
   (IDEN NUMBER, PCALLE VARCHAR2,  PCIU VARCHAR2, CP NUMBER)
RETURN VARCHAR2 AS
  IDT NUMBER;
  TABLAANID TABLA_ANIDADA;
  CUENTA NUMBER;
BEGIN
 --COMPROBAR SI EXISTE ID,
   SELECT COUNT(ID) INTO CUENTA
       FROM EJEMPLO_TABLA_ANIDADA WHERE ID = IDEN;
   IF CUENTA = 0 THEN
       RETURN 'NO EXISTE EL ID: '||IDEN||', EN LA TABLA';
   END IF ;
   IF CUENTA > 1 THEN
      RETURN 'EXISTEN VARIOS REGISTROS CON EL MISMO  ID: '||IDEN ;
   END IF;

 --EL ID EXISTE, COMPROBAR SI LA CALLE EXISTE:
  SELECT ID INTO IDT
            FROM EJEMPLO_TABLA_ANIDADA, TABLE(DIREC)
            WHERE ID= IDEN
            AND UPPER(CALLE)=UPPER(PCALLE)
            AND UPPER(CIUDAD) = UPPER(PCIU)
            AND CODIGO_POST= CP;
```

```
  RETURN ('LA DIRECCIÓN : '||PCALLE || '*' ||PCIU
              || '*' || CP  || 'YA EXISTE PARA ESE ID: '||IDEN);
EXCEPTION
WHEN NO_DATA_FOUND THEN
  RETURN 'NO EXISTE LA DIRECCION : '||PCALLE || '*' ||PCIU
              || '*' || CP  || ' PARA EL ID: '||IDEN;
END EXISTE_DIREC;
```