INTRODUCCIÓN A LOS OBJETOS

BD Relacionales	BD Orientados a Objetos	BD Relacionales OO
• Tipos de datos sencillos • Lenguajes potentes • Protección elevada	• Tipos de datos complejos • Integrados con lenguajes de programación • Elevado Rendimiento	• Tipos de datos complejos • Lenguajes potentes • Protección elevada

MODELOS RELACIONALES EXTENDIDOS

- El modelo relacional extendido, permite relaciones que no cumplen con las formas normales. Ejemplo: Campos con valores múltiples
- Esta característica se da mediante la agregación de "tipos enriquecidos" en el DDL del SQL, así como extensiones al mecanismo de consultas para contemplar los tipos complejos de datos y relaciones anidadas.

ORACLE: EJEMPLO DE BD RELACIONAL CON EXTENSIONES DE ORIENTACIÓN A OBJETO

- Creación de tipos de datos definidos por el usuario
- Permite crear objetos almacenados y referenciarlos por otros objetos
- Permite el almacenamiento de "estructuras desnormalizadas"
- No soporta la definición de clases, herencia y encapsulamiento con el mismo concepto que el de los LOO

Estructura de un tipo OBJETO

Interfaz Pública

Especificación (TYPE)

- Declaración de atributos
- Declaración formal de los métodos

Implementación Privada

Cuerpo (TYPE BODY)

- Desarrollo (cuerpo) de los métodos

SINTAXIS

```
CREATE [OR REPLACE] TYPE nombre_tipo
   IS| AS OBJECT
(atributo1              tipodato,
 atributo2              tipodato,
[MEMBER <declaración formal de función
   o procedimiento>,
.... ]
);
/
CREATE [OR REPLACE] TYPE_BODY
   nombre_tipo IS | AS
[MEMBER <cuerpo de la función o
   procedimiento]);
```

Restricciones

- En la especificación del objeto, primeramente deben declararse los atributos, luego los métodos.
- Si el objeto tiene solo atributos, no requiere un body.
- No se pueden declarar atributos en el body
- Las declaraciones de la especificación del objeto son públicas

CONSIDERACIONES ACERCA DE LOS OBJETOS

- Cuando se crea un objeto, sus atributos miembros deben considerar lo siguiente:
  - Un atributo es declarado con un nombre y tipo
  - Los tipos de datos no pueden ser LONG , NCHAR, NCLOB, ROWID o tipos de datos PL/SQL
  - El tipo de datos no puede tener valores por default
  - No se puede asignar la cláusula NOT NULL

EJEMPLO

Definición de un nuevo tipo simple:

```
CREATE TYPE tipo_persona AS OBJECT
(Nombre VARCHAR2(30),
 id              NUMBER(8),
 Fecha_nac DATE);
```

Definición de un tipo compuesto:

```
CREATE TYPE profesión_ejercida AS OBJECT
(profesional    tipo_persona ,
 profesion      VARCHAR2(30));
```

Los métodos

- Son los subprogramas declarados en la especificación del tipo utilizando la palabra clave MEMBER. Los métodos tienen la especificación y el cuerpo.
- Los métodos, al igual que los subprogramas de los paquetes, pueden ser sobrecargados.

Los métodos caen en las siguientes categorías:

- **MEMBER**: Es una función o procedimiento que siempre tiene implícitamente un parámetro SELF que contiene el tipo objeto que contiene al método.
- **STATIC**: Es una función o procedimiento que no tiene el parámetro SELF. Dichos métodos pueden ser invocados calificando el método con el nombre del tipo.
- **Métodos de comparación** para comparar instancias de objetos.

El método CONSTRUCTOR

Cada tipo de objeto tiene también definido un método implícito: el **método constructor**, que tiene el mismo nombre del tipo de objeto.

Como argumentos, toma los valores de los atributos del objeto.

El PL/SQL no invoca al constructor explícitamente, de manera que debe llamarse explícitamente.

El método constructor tiene:

- El mismo nombre del tipo objeto
- Los parámetros formales y el retorno son los mismos del tipo objeto

El constructor debe ser utilizado para instanciar el tipo de objeto

Ordenamiento de Objetos -

- **Métodos MAP**: El método MAP utiliza la propia habilidad de ORACLE para realizar comparaciones de los tipos de datos incorporados en el lenguaje. Se define como una función que devuelve un valor escalar. En el método MAP se define el atributo, u operación de atributos que establecen el criterio de comparación entre dos objetos.

Ordenamiento de Objetos -

- **Métodos ORDER**: Usa su propio método interno para poder determinar el resultado. Recibe como parámetro un objeto del mismo tipo y debe devolver:
  - -1 si el parámetro es mayor que SELF
  - 1 si el parámetro es menor que SELF
  - 0 si el parámetro es igual a SELF

EJEMPLO : OBJETO CON MIEMBROS USANDO ORDER

```
CREATE  or REPLACE TYPE tipo_clientes AS
OBJECT

(cli_id NUMBER,

nombre VARCHAR2(30),

MEMBER FUNCTION ver_nombre  RETURN
varchar2,

ORDER MEMBER FUNCTION orden_cli (cli1
tipo_clientes)

RETURN NUMBER);

/
```

EJEMPLO : OBJETO CON MIEMBROS

```
CREATE  OR REPLACE TYPE BODY tipo_clientes IS
 MEMBER FUNCTION ver_nombre RETURN varchar2 IS
 BEGIN
   RETURN nombre;
 END;

ORDER MEMBER FUNCTION orden_cli (cli1 tipo_clientes)
RETURN NUMBER IS
BEGIN
       IF cli_id < cli1.cli_id THEN RETURN (-1);
        ELSIF cli_id = cli1.cli_id THEN RETURN (0);
        ELSE RETURN (1);
       END IF;
END;
END;
```

Ejemplo de aplicación

```
DECLARE
  A tipo_clientes := tipo_clientes(1, 'MARIO BENEDETI');
  B tipo_clientes := tipo_clientes(2, 'JOSELUCHO');
BEGIN
  IF A.orden_cli(B) = 1 then
       dbms_output.put_line('A > B');
  ELSIF A.orden_cli(B) = 0 then
       dbms_output.put_line('A = B');
  ELSE
       dbms_output.put_line('A < B');
  END IF;
  IF A > B then
     DBMS_OUTPUT.PUT_LINE(' A MAYOR B');
  ELSE
     DBMS_OUTPUT.PUT_LINE('MENOR O IGUAL');
 END IF;
END;
```

Uso de los objetos en tablas de la BD

- **Tablas de Objeto: Object Tables**

Una tabla objeto es una tabla de BD en la cual cada fila representa un objeto. A continuación un ejemplo:

```
CREATE TABLE clientes OF tipo_clientes;
```

```
SQL> desc clientes
Name                      Null?    Type
 ------------------------------ -------- ----------------
 CLI_ID                                   NUMBER
 NOMBRE                                   VARCHAR2(30)
```

Cuando se trata de objetos de fila se pueden usar directamente las columnas o una referencia al objeto:

- INSERT INTO clientes VALUES (1, 'Juan Carlos');
- INSERT INTO clientes VALUES (tipo_clientes(2, 'Maria'));
- SELECT cli_id, nombre from clientes;
- SELECT VALUE(p) from clientes p;
- SELECT c.ver_nombre() from clientes c;
- DELETE clientes where cli_id= 1;
- DELETE from clientes p WHERE p.cli_id = 1;

Uso de los objetos en tablas de la BD

- **Objetos de Columna:**

Un objeto de columna se almacena como una columna de una tabla de Base de Datos. A continuación el ejemplo:

```
CREATE TABLE clientes2
(cedula              number(8),
 descripcion   tipo_clientes);
```

```
SQL> desc clientes
Name                      Null?    Type
------------------------- -------- -------------------
CEDULA                             NUMBER(8)
DESCRIPCION                        TIPO_CLIENTES
```

Cuando se trata de objetos de columna, deben utilizarse el método constructor:

- INSERT INTO clientes2 VALUES (2345, tipo_clientes(1, 'Juan Carlos'));
- SELECT * FROM clientes2 c WHERE c.descripcion.cli_id = 1;
- SELECT c.descripcion.ver_nombre() FROM clientes2 c;
- DELETE clientes2 c WHERE c.descripcion.cli_id = 1;
- UPDATE clientes2 c SET c.descripcion = tipo_clientes (2, 'Juan Manuel');

OTROS DETALLES ACERCA DE LOS OBJETOS

Métodos sin parámetros

- Si en un programa PL/SQL se llama a un método que no tiene ningún parámetro, el uso del paréntesis en la llamada es opcional
- Sin embargo, cuando se llama un método desde SQL, es necesario usar el paréntesis

Métodos sin parámetros. Ej:

```
CREATE TYPE T_EMPLEADO AS OBJECT
(ID NUMBER,
 NOMBRE VARCHAR2(60),
 MEMBER FUNCTION OBTENER_NOMBRE RETURN VARCHAR2);
/
CREATE TYPE BODY T_EMPLEADO AS
MEMBER FUNCTION OBTENER_NOMBRE RETURN VARCHAR2
 IS
  BEGIN
     RETURN NOMBRE;
  END;
END;
/

CREATE TABLE EMP OF T_EMPLEADO;
```

Métodos sin parámetros. Ej:

INCORRECTO

```
SQL> SELECT E.OBTENER_NOMBRE FROM EMP E;
SELECT E.OBTENER_NOMBRE FROM EMP E
       *
ERROR at line 1:
ORA-00904: "E"."OBTENER_NOMBRE": invalid identifier
```

CORRECTO

```
SELECT E.OBTENER_NOMBRE() FROM EMP E
```

FUNCIÓN VALUE:

- **Se utiliza en un SELECT en tablas de objetos.**
- **VALUE devuelve el objeto de fila para dicha tabla**
- **Ej:**

SQL> SELECT VALUE(E) FROM EMP E;

VALUE(E)(ID, NOMBRE)

T_EMPLEADO(1, 'JUANCHO')

FUNCIÓN REF:

- **REF devuelve la referencia al objeto de fila (recordar que VALUE devuelve el objeto de fila en sí)**

```
SQL> SELECT REF(E) FROM EMP E;

REF(E)
--------------------------------------------------------------------------------
--------------------------------------------

00002802095490E6E72324106ACBC24914103780A40AFF
7EC210D4465B49002AAC2FC810C018004A20000
```

EJERCICIOS