BBDD OBJETO-RELACIONALES

Javier García-Retamero Redondo

¿QUÉ SON?

- **BDOR (Bases de datos Objeto-Relacionales):**

- Extensión de las bbdd relacionales a las que se les ha añadido conceptos del modelo orientado a objetos.
- Es un sistema relacional que permite almacenar objetos en las tablas.
- Hay que extender las características de SQL para que trabaje con objetos.

- Ejemplo: Utilizaremos Oracle

CARACTERÍSTICAS

- Ventajas:
  - El usuario puede crear sus propios tipos de datos.
  - Se pueden crear funciones para esos tipos de datos.
  - Gestión de tipos con esfuerzo mínimo
  - Herencia
  - Almacenar múltiples valores en una misma columna de una fila es posible.
  - Relaciones (tablas) anidadas
  - Es fácil el pasar aplicaciones tradicionales al nuevo modelo.

- Desventajas:
  - Aumento de la complejidad del sistema.

TIPOS DE OBJETOS

TIPOS DE OBJETOS

- Para crear tipos de objetos utilizamos: CREATE OR REPLACE TYPE

```
create or replace type direccion as object(
        calle varchar2(25),
        ciudad varchar2(20),
        codigo_post number(5));
```

```
create or replace type persona as object (
        codigo number,
        nombre varchar2(35),
        direc direccion,
        fecha_nac date);
```

El atributo direc es de tipo object

TIPOS DE OBJETOS

- Utilización de los tipos creados:

```
declare
        dir direccion := direccion(null,null,null);
        p persona := persona(null,null,null,null);
begin
        dir.calle := 'La Mina, 3';
        dir.ciudad := 'Guadalajara';
        dir.codigo_post := 19001;

        p.codigo := 1;
        p.nombre := 'Juan';
        p.direc := dir;
        p.fecha_nac := '10/11/1988';

        dbms_output.put_line('nombre: ' || p.nombre || '* calle: ' || p.direc.calle);
end;
/
```

Podemos inicializar el objeto asignando valores a los atributos

TIPOS DE OBJETOS

- Utilización de los tipos creados:

```
declare
        dir direccion;
        p persona;
begin
      dir := new direccion ('C/Madrid 10', 'Toledo', 45002);
      p := new persona (2, 'JUAN', dir, SYSDATE);
      dbms_output.put_line('nombre: ' || p.nombre || '* calle: ' || p.direc.calle);
end;
/
```

Otra forma de inicializar el objeto es utilizando new.

MÉTODOS EN LOS OBJETOS

MÉTODOS

- Al crear un objeto también solemos crear sus métodos los cuales pueden ser de varios tipos:
  - **MEMBER**: Sirven para actuar con los objetos.
    - Procedimientos
    - Funciones
  - **STATIC**: Métodos del tipo.
    - Procedimientos
    - Funciones
  - **CONSTRUCTOR**: Inicializa el objeto
    - Es una función:
      - Argumentos = valores de los atributos
      - Valor de retorno: Objeto inicializado

MÉTODOS

DECLARACIÓN DE LOS MÉTODOS:

- Fíjate como se declara la salida del constructor para indicar que devuelve el objeto (RETURN SELF AS RESULT)

```
create or replace type rectangulo as object (
                    base number,
                    altura number,
                    area number,
                    static procedure proc1 (ancho integer, alto integer),
                    member procedure proc2 (ancho integer, alto integer),
                    constructor function rectangulo (base number, altura number) return self as result
);
```

MÉTODOS

CREACIÓN DEL CUERPO DE LOS MÉTODOS:

- CREATE OR REPLACE TYPE BODY nombre_del_tipo AS
- El nombre del type body debe coincidir con el nombre del type
- Los nombres de los procedimientos o funciones deben coincidir con los especificados en el tipo
- Para que funcione el siguiente código crea la tabla

```
create table tablarec (valor integer);
```

MÉTODOS

```
create or replace type body rectangulo as
    static procedure proc1 (ancho integer, alto integer) is
        begin
                insert into tablarec values (ancho*alto);
                commit;
                dbms_output.put_line('Fila insertada');
        end;
    member procedure proc2 (ancho integer, alto integer) is
        begin
                self.altura :=alto;
                self.base:= ancho;
                area:=altura*base;
                insert into tablarec values(area);
                commit;
                dbms_output.put_line('Fila insertada');
        end;
```

MÉTODOS

```
    constructor function rectangulo (base number, altura number)
    return self as result is
        begin
            self.base := base;
            self.altura := altura;
            self.area := base*altura;
            return;
        end;
end;
```

Llamada al método static

Llamada al método member

```
declare
  r1 rectangulo;
  r2 rectangulo;
  r3 rectangulo := rectangulo(null,null,null);
begin
  r1 := new rectangulo(10,20,500);
  dbms_output.put_line ('Area R1: '||r1.area);
  r2 := new rectangulo(10,20);
  dbms_output.put_line ('Area R2: '||r2.area);
  r3.base := 5;
  r3.altura := 15;
  r3.area := r3.base*r3.altura;
  dbms_output.put_line ('Area R3: '||r3.area);
  RECTANGULO.PROC2(10,20);
  R1.PROC2(5,6);
end;
/
```

MÉTODOS

- EJEMPLO: Tipo Direccion para almacenar los datos de la dirección postal

```
create or replace type direccion as object(
        calle varchar2(25),
        ciudad varchar2(20),
        codigo_post number(5),
        member procedure set_calle(c varchar2),
        member function get_calle return varchar2,
);
```

MÉTODOS

```
create or replace type body direccion as
        member procedure set_calle(c varchar2) is
            begin
                    calle:=c;
            end;
        member function get_calle return varchar2 is
            begin
                    return calle;
            end;
end;
/
```

MÉTODOS

```
DECLARE
        DIR DIRECCION:= DIRECCION(NULL, NULL, NULL);
BEGIN
        DIR.SET_CALLE('La Mina, 3');
        DBMS_OUTPUT.PUT_LINE(DIR.GET_CALLE);
        DIR:=NEW DIRECCION('C/Madrid 10', 'Toledo', 45002);
        DBMS_OUTPUT.PUT_LINE(DIR.GET_CALLE);
END;
/
```

COMPARACIÓN DE OBJETOS

UTILIZANDO MAP

UTILIZANDO MAP

- En muchas ocasiones tenemos que comparar u ordenar datos de tipos definidos como OBJECT. Para ello tenemos que crear un método MAP u ORDER teniendo que definir al menos uno por cada objeto que se quiere comparar.

- Los métodos MAP son una función que devuelve un valor de tipo escalar (CHAR, VARCHAR2, NUMBER, DATE, etc.) que será el que se utilice en las comparaciones y ordenaciones aplicando los criterios establecidos para este tipo de datos.

UTILIZANDO MAP

```
create or replace type persona as object (
        codigo number,
        nombre varchar2(35),
        direc direccion,
        fecha_nac date,
        MAP MEMBER FUNCTION POR_CODIGO RETURN NUMBER
);
/
create or replace type body persona as
        MAP MEMBER FUNCTION POR_CODIGO RETURN NUMBER IS
        begin
                return codigo;
        end;
end;
/
```

MÉTODOS

COMPARACIÓN DE OBJETOS:

```
declare
        p1 persona := persona(null, null, null, null);
        p2 persona := persona(null, null, null, null);
begin
        p1.codigo :=1;
        p1.nombre := 'JUAN';
        p2.codigo :=1;
        p2.nombre := 'MANUEL';
        IF p1 = p2 then
                dbms_output.put_line('objetos iguales');
        else
                dbms_output.put_line('objetos distintos');
        end if;
end;
/
```

Al ejecutar este código nos dirá que son iguales ya que los Dos objetos tienen como código el 1.
Modifica el código de p2 para que sea 2 y vuelve a ejecutar, Ahora debería aparecer el mensaje de que son distintos.

Al comparar los dos objetos llamará a cada una de las funciones MAP de los objetos y obtendrá el valor devuelto por la función. Esos valores devueltos será los que se compararán

TABLAS DE OBJETOS

CREACIÓN DE TABLAS

¿QUÉ ES UNA TABLA DE OBJETOS?

- Una vez creados los tipos de objetos podemos utilizarlos para:
  - Definir nuevos tipos
  - Definir columnas de tablas
  - Definir tablas que almacenan objetos

- TABLA DE OBJETOS: Tabla que almacena un objeto en cada fila.

CREACIÓN

- Creamos una tabla de alumnos donde cada fila es un objeto del tipo PERSONA:

```
create table alumnos OF PERSONA(
    codigo primary key
);
```

El campo codigo será la clave primaria

- Realizamos un DESC de alumnos para visualizar los campos.

TABLAS DE OBJETOS

INSERCIÓN DE DATOS

INSERCIÓN DE DATOS

CON UN INSERT NORMAL

- El insert es similar a un insert sobre una tabla normal.
- Si uno de los campos es un objeto, tendremos que utilizar el constructor para almacenar los datos dentro de ese campo.

insert into alumnos values(1, 'Juan Pérez', DIRECCION('C/Los Espartales 25', 'GUADALAJARA', 19005), '18/12/1991');

Como dirección es un objeto entonces lo insertamos así

INSERCIÓN DE DATOS

INSERTANDO UN OBJETO

- Como las filas de la tabla son objetos, podemos insertar directamente un objeto al cual le hemos asignado los valores.

```
declare
        dir direccion := direccion('C/Sevilla 20', 'GUADALAJARA', 19004);
        per persona := persona(5, 'MANUEL', dir, '20/10/1987');
begin
        insert into alumnos values(per);
        commit;
end;
/
```

TABLAS DE OBJETOS

CONSULTA DE DATOS

CONSULTA DE DATOS

SELECT PARA MOSTRAR LOS CAMPOS NORMALES DEL OBJETO

- Si queremos mostrar los datos de los campos que no son objetos.

```
select codigo, nombre, fecha_nac from alumnos;
```

CONSULTA DE DATOS

SELECT PARA MOSTRAR UN CAMPO QUE ES UN OBJETO

- Si queremos mostrar uno de los campos que contiene un objeto.

```
select codigo, nombre,
        A.direc.calle, A.direc.ciudad, A.direc.codigo_post,
        fecha_nac
from alumnos A;
```

TABLAS DE OBJETOS

MODIFICACIÓN DE DATOS

MODIFICACIÓN DE DATOS

MODIFICACIÓN DE CAMPOS INDIVIDUALES

- Podemos modificar los valores contenidos en campos que no son objetos de la forma tradicional.

```
update alumnos set fecha_nac = '21/11/2001' where codigo = 1;
```

- Si el campo está dentro de un objeto, tendremos que anteponer el nombre del objeto.

```
update alumnos A set A.direc.ciudad = lower(A.direc.ciudad)
where A.direc.ciudad = 'GUADALAJARA';
```

MODIFICACIÓN DE DATOS

MODIFICACIÓN ASIGNANDO OBJETOS

- Si el campo a modificar es un objeto, podemos asignarle otro objeto.

```
declare
        d direccion := direccion('C/Galiano 5', 'Guadalajara', 19004);
begin
        update alumnos set direc = d where nombre = 'Juan Pérez';
        commit;
end;
/
```

TABLAS DE OBJETOS

BORRADO DE DATOS

BORRADO DE DATOS

BORRADO DE FILAS

```
delete alumnos A where A.direc.ciudad='guadalajara';
```