

Desarrollo de aplicaciones web / multiplataforma



**M02B: Bases de datos**

**UF4. Bases de datos objeto-relacionales**

# Índice

1. Introducción
  - a. El paradigma de la orientación a objetos
  - b. Características de las BDOR
2. Tipos de datos objeto
  - a. Tipos Complejos
  - b. Colecciones

# 1. Introducción

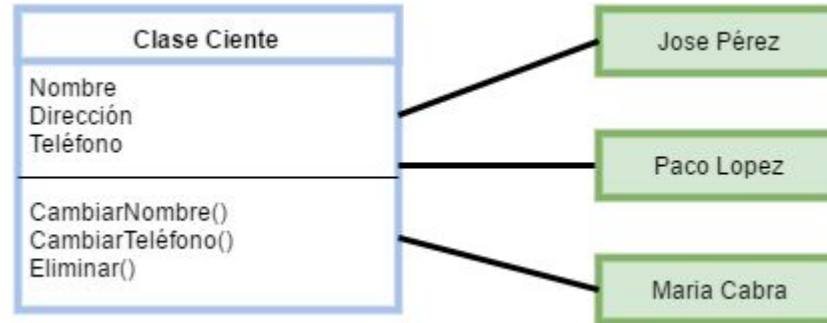
- En los años 90, de la mano de la programación Orientada a Objetos nacen las bases de datos Orientadas a Objetos.
  - Formato de información adecuado para este paradigma de programación.
  - Menos restricciones de las exigidas por el modelo relacional.
- Las BDOO puras tienen una discreta cuota de mercado, sin embargo:
  - La mayoría de los SGBD comerciales incluyen ciertos elementos de la orientación a objetos en sus bases de datos relacionales.
  - Nacimiento de las bases de datos Objeto-Relacional



# 1. Introducción

## a. El paradigma de la orientación a objetos (I)

- Proporciona una representación más directa del mundo real entendido en:
  - **Clases** → Tipos de elementos que deseamos representar.
    - Atributos → Características de la clase.
    - Métodos → Funciones o procedimientos que pueden realizar.
  - **Objetos** → Instancias de los objetos propiamente dichos



# 1. Introducción

## a. El paradigma de la orientación a objetos (II)

- Las principales características la orientación a objetos son:
  - Encapsulación → Los objetos son entidades aisladas y sólo son accesibles a través de sus métodos públicos.
  - Reutilización → Debido a la encapsulación podemos utilizar las mismas clases en diferentes contextos sin conocer su implementación.
  - Herencia → Permite crear una clase general y derivar clases con características similares.
    - E.g. Clase Persona
      - Clase Alumno
      - Clase Profesor
  - Polimorfismo → Permite redefinir el comportamiento de un método dependiendo de la clase a la que pertenezca (Dentro de un árbol de herencia).

# 1. Introducción

## b. Características de las Bases de datos Objeto-Relacionales (I)

- En el modelo objeto-relacional:
  - Cada registro de una tabla se considera un objeto
  - Y la definición de la tabla su clase.
- El modelo objeto relacional tiene capacidad para gestionar tipos de datos complejos, lo cual contradice algunas de las restricciones establecidas por el modelo relacional.
  - Permite campos multivaluados, lo cual contradice frontalmente la 1FN

Alumnos		
DNI	NOMBRE	Asignaturas
123456P	Paco	{M01,M03A}
321456O	Marta	{M05,M02B}

# 1. Introducción

## b. Características de las Bases de datos Objeto-Relacionales (II)

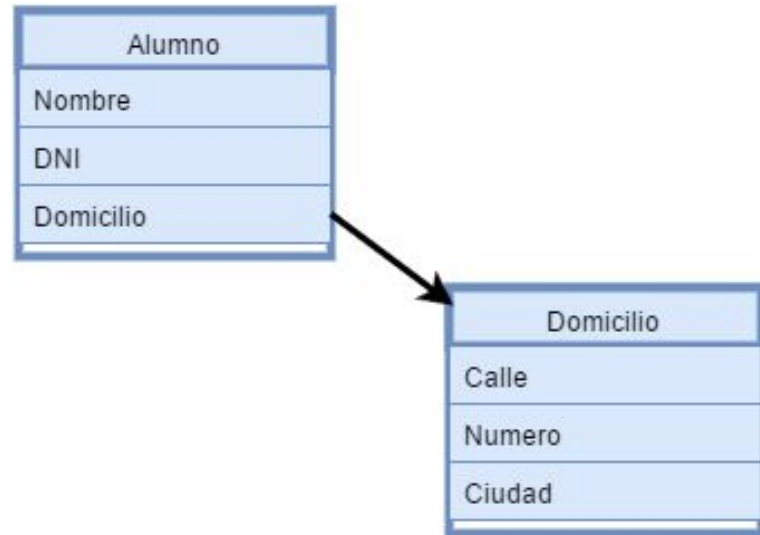
- El modelo objeto-relacional contradice ... (Continuación):
  - Las tablas dejan de ser elementos bidimensionales para convertirse en estructuras de datos más complejas.
    - E.g. Una columna puede ser de tipo “Asignatura” conteniendo toda la información de esta y no únicamente su clave ajena.

Alumnos		
DNI	NOMBRE	Asignaturas
123456P	Paco	{aNombre: Bases de datos aCodigo: M02A}

# 1. Introducción

## b. Características de las Bases de datos Objeto-Relacionales (III)

- El modelo objeto-relacional contradice ... (Continuación) bis:
  - Del mismo modo, es posible que los valores que adopte un campo sean registros de otra tabla (Es decir, colecciones de objetos). Esto se conoce como “Modelo relacional anidado”





## 2. Tipos de datos Objeto

### a. Tipos Complejos (Object Types) I

- Para construir los tipos de usuario se utilizan los tipos básicos provistos por el sistema y otros tipos de usuario previamente definidos. Un tipo define una estructura y un comportamiento común para un conjunto de datos de las aplicaciones.
- Un tipo objeto representa a una entidad del mundo real.
- Se compone de los siguientes elementos:
  - Nombre→ Sirve para identificar el tipo de los objetos.
  - Atributos→ Modelan la estructura y los valores de los datos de ese tipo.
    - Cada atributo puede ser de un tipo de datos básico o de un tipo de usuario.
  - Métodos→ Procedimientos o funciones escritos en el lenguaje PL/SQL (almacenados en la base de datos).

# 2. Tipos de datos Objeto

## a. Tipos Complejos (Object Types) II

- Definición de un tipo objeto.

```
CREATE or REPLACE TYPE nombreObjeto AS OBJECT (  
    atributo TIPO,  
    atributo2 TIPO,  
    ...,  
    MEMBER FUNCTION nombreFuncion RETURN Tipo,  
    MEMBER FUNCTION nombreFuncion2(nomVar TIPO) RETURN Tipo,  
    MEMBER PROCEDURE nombreProcedimiento  
    PRAGMA RESTRICT_REFERENCES (nombreFuncion, restricciones),  
    PRAGMA RESTRICT_REFERENCES (nombreFuncion2, restricciones),  
    PRAGMA RESTRICT_REFERENCES (nombreProcedimiento, restricciones)  
);
```

Dónde “restricciones” puede ser cualquiera de las siguientes o una combinación de ellas:

- **WNDS**: Evita que el método pueda modificar las tablas de la BD.
- **RNDS**: Evita que el método pueda leer las tablas de la base de datos.
- **WNPS**: Evita que el método modifique variables del paquete PL/SQL.
- **RNPS**: Evita que el método pueda leer las variables del paquete PL/SQL.

## 2. Tipos de datos Objeto

### a. Tipos Complejos (Object Types) III

- Definición del cuerpo de un objeto.

```
CREATE OR REPLACE TYPE BODY nombreObjeto AS
  MEMBER FUNCTION nombreFuncion
  RETURN Tipo
  IS
    BEGIN
      ...
    END nombreFuncion;
  MEMBER FUNCTION nombreFuncion2(nomVar TIPO)
  RETURN Tipo
  IS
    BEGIN
      ...
    END nombreFuncion2;
  MEMBER PROCEDURE nombreProcedimiento
  IS BEGIN
    ...
  END nombreProcedimiento;
END; /
```

# 2. Tipos de datos Objeto

## a. Tipos Complejos (Object Types) IV

- Ejemplo:

```
CREATE or REPLACE TYPE persona AS OBJECT (  
    idpersona number,  
    dni varchar2(9),  
    nombre varchar2(15),  
    apellidos varchar2(30),  
    fecha_nac date,  
    MEMBER FUNCTION muestraEdad RETURN NUMBER,  
    PRAGMA RESTRICT_REFERENCES (muestraEdad,WNDS)  
);
```

### Definición clase

### Cuerpo de la clase

```
CREATE OR REPLACE TYPE BODY persona AS  
    MEMBER FUNCTION muestraEdad  
    RETURN NUMBER  
    IS  
        BEGIN  
            return to_char(sysdate, 'YYYY')-to_char(fecha_nac, 'YYYY');  
        END muestraEdad;  
END;
```

## 2. Tipos de datos Objeto

### a. Tipos Complejos (Object Types) V

- Una vez hemos definido la clase y su cuerpo, podemos usar este objeto como si de un tipo cualquiera se tratase.

```
DECLARE
    Trabajador1 Persona;
Begin
    Trabajador1:= NEW Persona(1, '123456', 'Alberto', 'Oliva', '22/12/1989');

    DBMS_OUTPUT.put_line('El empleado '||Trabajador1.nombre|| ' ha sido creado');
END;
```

## 2. Tipos de datos Objeto

### a. Colecciones

- Para poder contar con atributos multivaluados es necesario que los organicemos en una estructura de datos, esta estructura es conocida como Array o Colección. Para crear un atributo de tipo array debemos usar la siguiente sintaxis.

```
-- Lista de un tipo primitivo varchar2  
create type colec_nombres as varray(10) of varchar2(20);  
  
-- Lista de un tipo de usuario: Persona  
create type colec_personas as varray(10) of Persona;
```

## 2. Tipos de datos Objeto

### a. Colecciones II

- Una vez hemos creado un tipo colección podemos usarla dentro de la definición de una clase:

```
Create or Replace Type Departamento as Object(  
    NombreDept Varchar2(100),  
    Empleados colec_personas  
);
```

## 2. Tipos de datos Objeto

### a. Colecciones II

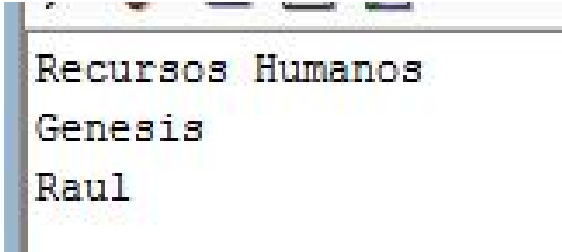
- Ejemplo de uso de objeto con clases:

```
Declare
    RRHH Departamento;
Begin
    RRHH:= new Departamento('Recursos Humanos', colec_personas(
        Persona(2, '1111', 'Genesis', 'Martinez', '12/12/1212'),
        Persona (3, '2222', 'Raul', 'Garcia', '23/11/1765'))
    );

    DBMS_OUTPUT.put_line (RRHH.nombredept);

    DBMS_OUTPUT.PUT_LINE (RRHH.EMPLEADOS(1).nombre);

    DBMS_OUTPUT.PUT_LINE (RRHH.EMPLEADOS(2).nombre);
End;
```



```
Recursos Humanos
Genesis
Raul
```



## 2. Tipos de datos Objeto

### a. Tablas derivadas de objetos

- Del mismo modo que creamos objetos haciendo uso de los tipos definidos, podemos hacerlo con las tablas:

```
create table Empleados(  
    NombreDept Varchar2(50),  
    Datos_Empleado Persona  
);
```

- Para hacer un insert debemos indicar el tipo de objeto que estamos insertando

```
insert into Empleados  
values ('Informática', Persona(1, '1111', 'Paco', 'Gonzalez', '23/12/1988'));
```

**¡Ánimo Chicos!**

