Asignatura

Base de datos

UNIVERSAE
Instituto Superior de FP

Asignatura

Base de datos

UNIDAD 8

Las bases de datos objeto-relacionales

UNIVERSAE
Instituto Superior de FP

Base de datos de objeto-relacional

BDOR (Base de datos objeto-relacional)

- Surgen a partir de la revisión SQL:99 o SQL3
- Son base de datos relacionales
- Utilizan una capa intermedia para usar objetos
- Definición de tipos estructurados de datos

¿Base de datos relacionales o objeto-relacional?

- Por defecto es una base de datos relacional
- Cuando se utiliza las características de objetos, se pasa a objeto-relacional

Ejemplos

- Oracle
- Postgre-SQL

Características base de datos objeto-relacional (Oracle)

Tipos de objetos

- Clases

Relación entre objetos

- Herencia

Tablas de objetos

- Cada fila representa un objeto

Tablas con columnas de objetos

- La columna guarda un objeto

Tablas con columnas de arrays

- La columna guarda una colección de elementos

Tablas con columnas de tablas

- La columna guarda una tabla anidada

Tipos de objetos

Los objetos

- Es la base de la creación de objetos
- Es una estructura muy simple, definida por un nombre del tipo y luego puede contener atributos y métodos
- Se suele utilizar en PL/SQL

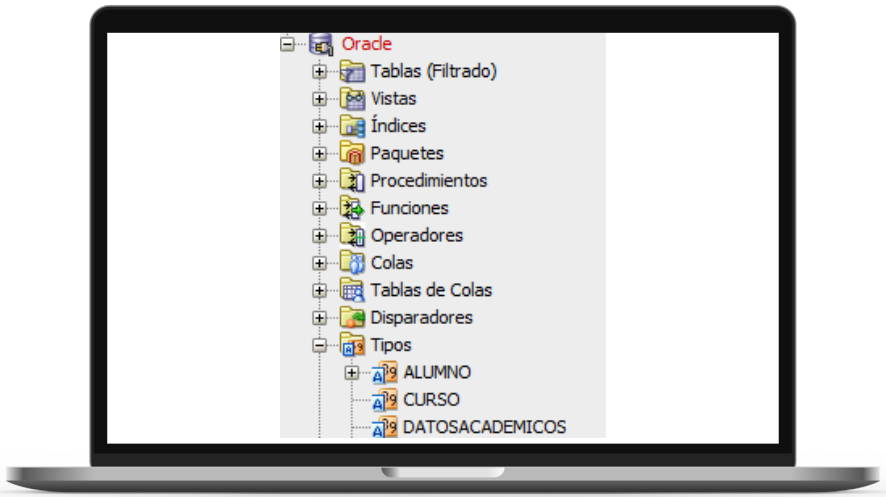
Sintaxis

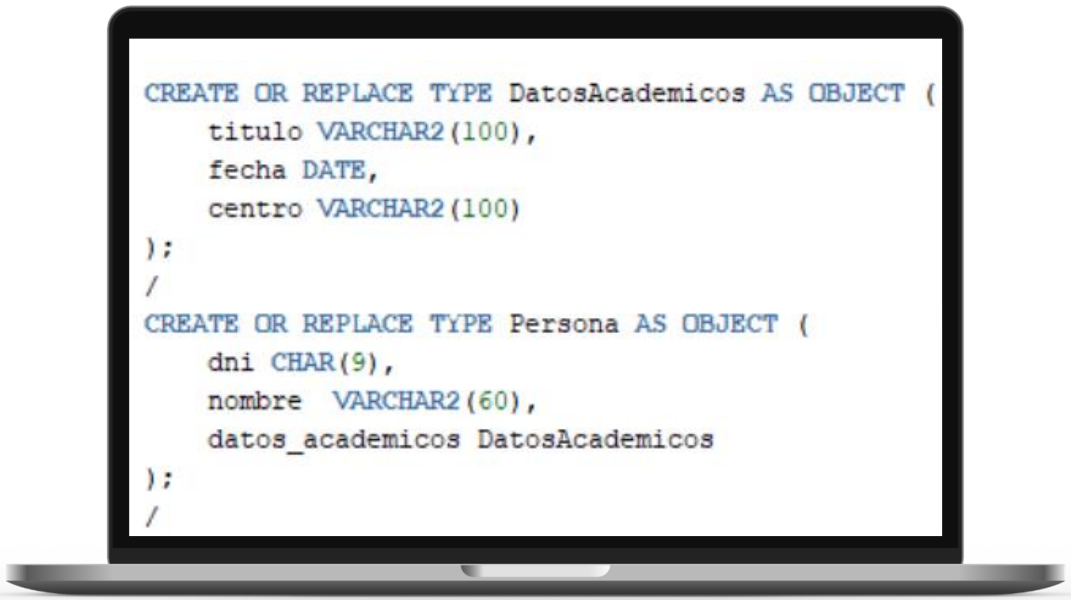
CREATE TYPE nombre AS OBJET {

Atributo Tipo,

....

}





```
CREATE OR REPLACE TYPE DatosAcademicos AS OBJECT (
    titulo VARCHAR2(100),
    fecha DATE,
    centro VARCHAR2(100)
);
/
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre  VARCHAR2(60),
    datos_academicos DatosAcademicos
);
/
```

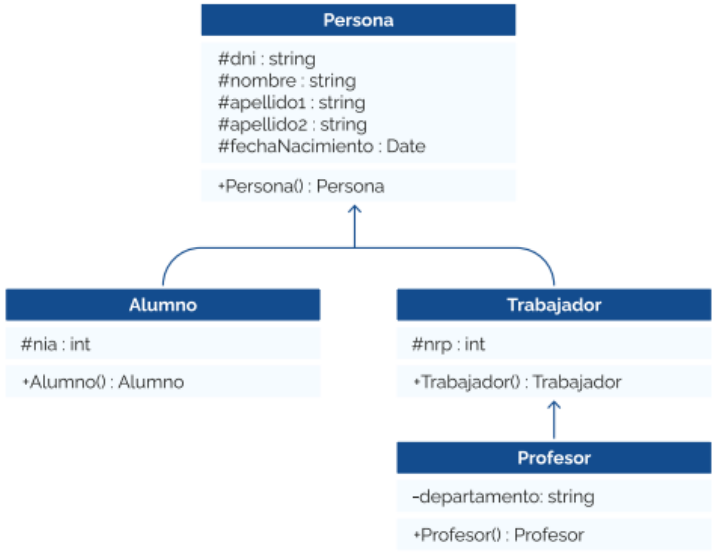
Herencia

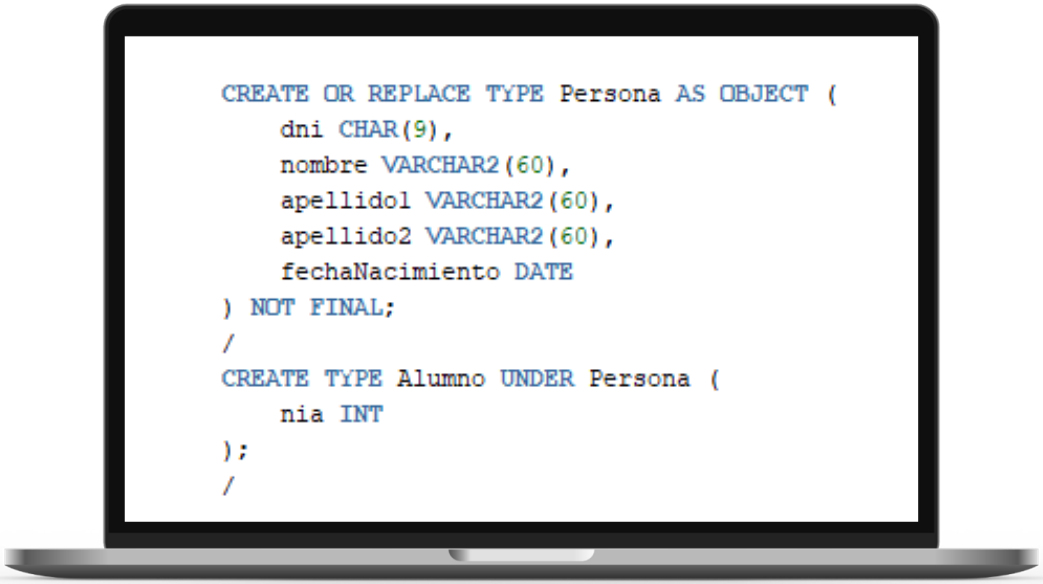
La herencia

- Existe un objeto padre o base
- Existe objetos hijos o subclase que extienden del objeto padre
- Herencia simple de tipos

Sintaxis

- El objeto padre debe de incluir NOT FINAL
- Los objetos hijos deben de utilizar UNDER





```
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre VARCHAR2(60),
    apellido1 VARCHAR2(60),
    apellido2 VARCHAR2(60),
    fechaNacimiento DATE
) NOT FINAL;
/
CREATE TYPE Alumno UNDER Persona (
    nia INT
);
/
```

Objetos por filas y por columnas

Por filas (Row object)

- Cada fila representa un objeto
- Sintaxis: CREATE TABLE nombre OF nombre objeto

Por columnas (Object column)

- Cada columna representa un objeto
- Sintaxis: en cada atributo especificar como tipo el nombre del objeto

```
CREATE TABLE personas OF Persona (
    dni PRIMARY KEY,
    nombre NOT NULL
);

INSERT INTO personas VALUES (Persona('11223344X','Pepe'));
```

```
CREATE TABLE personas (
    referencia INT PRIMARY KEY,
    persona Persona not null
);

INSERT INTO personas VALUES (1, Persona('11223344X','Pepe'));
```

Referencias de objetos

Las referencias

- Cada objeto tiene una referencia (OID)
- Permite que ese objeto pueda ser usado en diferentes tablas.
- Si se modifica en una tabla, se modifica en todas.

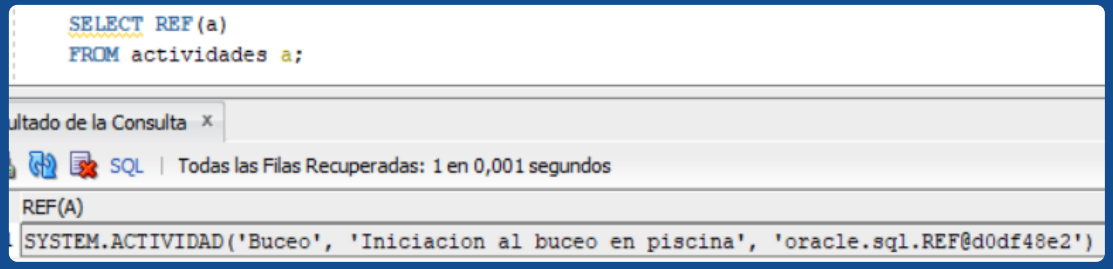
Procedimiento

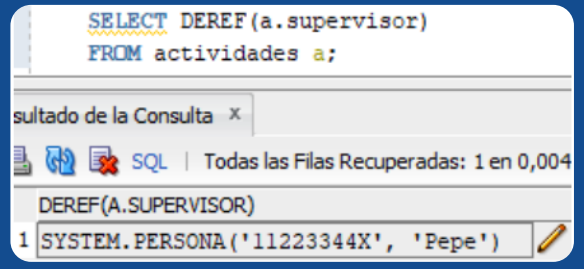
- Usar **REF** nombreObjeto para especificar que se guardara como referencia como campo de un tipo
- Usar **SCOPE IS** para especificar el ámbito de donde obtenerlo

```
CREATE OR REPLACE TYPE Actividad AS OBJECT (
    nombre VARCHAR2(100),
    descripcion VARCHAR2(100),
    supervisor REF Persona
);
/
CREATE TABLE actividades OF Actividad (
    nombre NOT NULL,
    descripcion NOT NULL,
    supervisor SCOPE IS personas
);
```

Consulta de un campo con referencias y sus valores

- Con **REF** podemos ver el OID del objeto
- Con **DEREF** podemos ver los valores del objeto





Arrays y tablas anidadas

Arrays

- Uso igual que en los lenguajes de programación.
- Sintaxis: VARRAY(tamaño) OF tipo

```
CREATE OR REPLACE TYPE Telefono AS VARRAY(2) OF CHAR(9);
/
CREATE OR REPLACE TYPE Persona AS OBJECT (
    dni CHAR(9),
    nombre  VARCHAR2(60),
    telefonos Telefono
) NOT FINAL;
/
CREATE TABLE personas OF Persona (
    dni PRIMARY KEY,
    nombre NOT NULL
);
```

```
INSERT INTO personas
VALUES (Persona('22334455Y',
                'Maria',
                Telefono('600111222','700111222')));
```

Tablas anidadas

- Una columna con una tabla dentro
- Procedimiento:
  1. Crear un tipo con referencia a una tabla
  2. Crear una tabla con una columna con el tipo objeto tabla

```
CREATE OR REPLACE TYPE Alumno AS TABLE OF REF Persona;
/
CREATE OR REPLACE TYPE Curso AS OBJECT (
    nombre VARCHAR2(50),
    fecha  DATE,
    alumnos Alumno
);
/
CREATE TABLE cursos OF Curso (
    nombre PRIMARY KEY
) NESTED TABLE alumnos STORE AS alumnos_tab;
```

```
INSERT INTO cursos
VALUES (Curso('Informatica',
              null,
              Alumno((SELECT REF(e)
                      FROM personas e))
              )
       );
```

Métodos

- Los objetos están compuestos por atributos y acciones.
- Las acciones pueden ser:
    - Constructores
    - Procedimientos
    - Funciones

Sintaxis del objeto

- Ahora los objetos tendrán dos partes. La **interfaz** y el **cuerpo**.
- La interfaz establece la cabecera de los métodos.

    CREATE OR REPLACE TYPE nombre AS OBJECT
- El cuerpo desarrolla las acciones.

    CREATE OR REPLACE TYPE BODY nombre AS

Sintaxis de los métodos

- CONSTRUCTOR FUNCTION nombre (parámetros) RETURN tipo
- MEMBER FUNCTION nombre (parámetros) RETURN tipo
- MEMBER PROCEDURE nombre (parámetros)

Métodos. Ejemplos

```
CREATE OR REPLACE TYPE rectangulo AS OBJECT (
    id       NUMBER,
    longitud NUMBER,
    ancho    NUMBER,
    area     NUMBER,
    CONSTRUCTOR FUNCTION rectangulo (id NUMBER, longitud NUMBER, ancho NUMBER )
        RETURN SELF AS RESULT,
    MAP MEMBER FUNCTION getid
        RETURN NUMBER,
    MEMBER PROCEDURE mostrar_datos ( SELF IN OUT NOCOPY rectangulo )
);
/
CREATE OR REPLACE TYPE BODY rectangulo AS
    CONSTRUCTOR FUNCTION rectangulo (id NUMBER, longitud NUMBER, ancho NUMBER )
        RETURN SELF AS RESULT AS
    BEGIN
        self.id := id;
        self.longitud := longitud;
        self.ancho := ancho;
        self.area := longitud * ancho;
        RETURN;
    END;
    MAP MEMBER FUNCTION getid RETURN NUMBER IS
    BEGIN
        RETURN id;
    END;
    MEMBER PROCEDURE mostrar_datos ( SELF IN OUT NOCOPY rectangulo )
    IS
    BEGIN
        dbms_output.put_line('longitud:' || to_char(longitud)
                             || ' ancho:' || to_char(ancho));
        dbms_output.put_line(' area:' || to_char(area));
    END;
END;
```

```
CREATE TABLE rectangulos (
   rectangulo rectangulo
);

INSERT INTO rectangulos VALUES (rectangulo(1, 5, 10));
INSERT INTO rectangulos VALUES (rectangulo(1, 5, 15));
INSERT INTO rectangulos VALUES (rectangulo(2, 5, 7));
```

```
SELECT * FROM rectangulos;
```

alida de Script | Resultado de la Cons

SQL | Todas las Filas Recupe

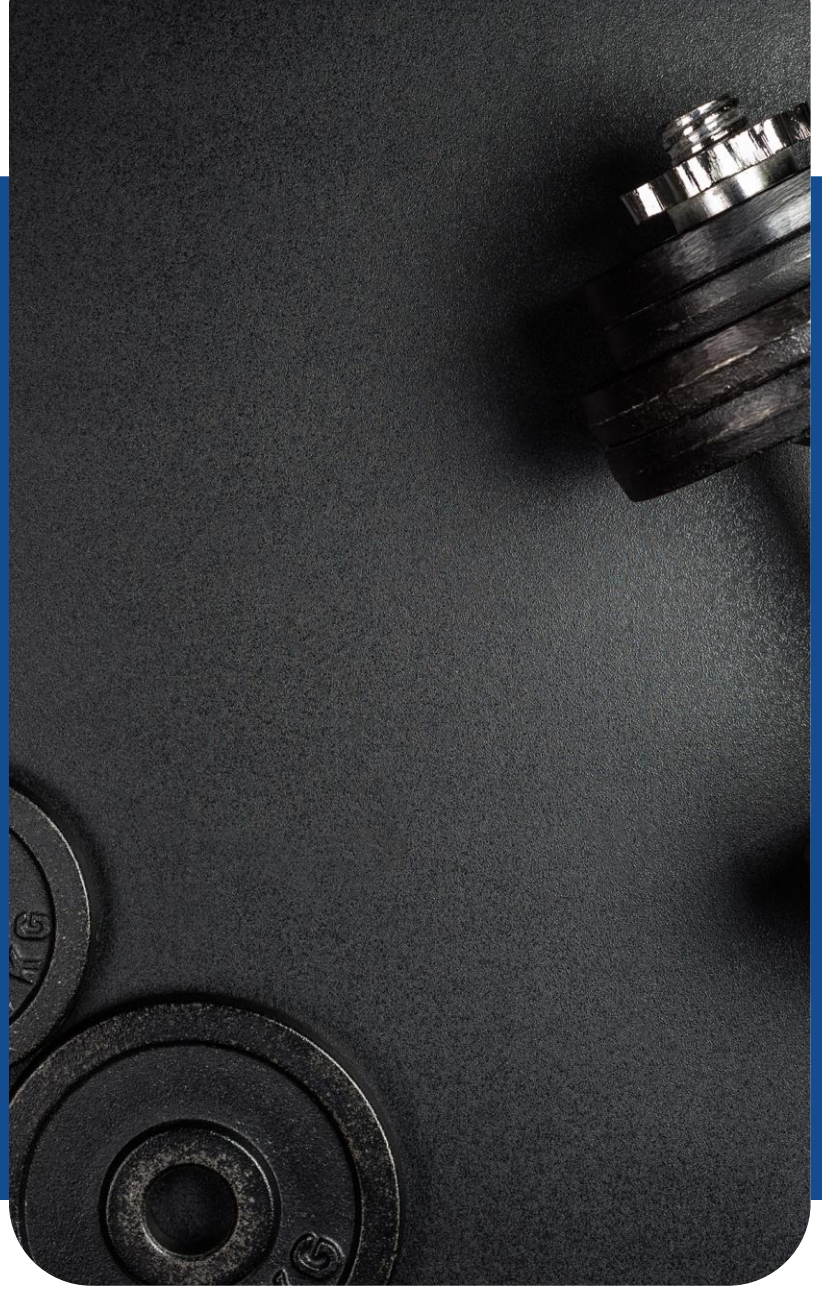
	RECTANGULO
1	[SYSTEM.RECTANGULO]
2	[SYSTEM.RECTANGULO]
3	[SYSTEM.RECTANGULO]

```
SELECT r.rectangulo.getId() FROM rectangulos r;
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 3 en 0,004 segundos

	R.RECTANGULO.GETID()
1	1
2	1
3	2



Planteamiento de ejercicios

1. Crear un objeto que represente un animal con los atributos que consideréis oportunos
2. Crear diferentes tipos de animales (Marino, Terrestre, etc.) añadir los atributos que consideréis. Tenéis que tener en cuenta el ejercicio1.
3. Crear una tabla en la que se pueda tener un conjunto de animales de un tipo por columna. Por ejemplo, columna 1 = Marino, Columna 2 = Terrestre, etc.
4. Modifica el objeto animal para que realice un procedimiento llamado ataque y muestre por mensaje si el animal puede atacar o no.

Resumen

1. Base de datos de objeto-relacional
2. Características base de datos objeto-relacional (Oracle)
3. Tipos de objetos
4. Herencia
5. Objetos por filas y por columnas
6. Referencias de objetos
7. Arrays y tablas anidadas
8. Métodos
9. Métodos. Ejemplos
10. Ejercicios propuestos

UNIVERSAE

CHANGE YOUR WAY