

# Bases de datos 2

Persistencia de objetos

## Problemas de los sistemas tradicionales

- La performance de los sistemas convencionales es inaceptable en algunos tipos de aplicaciones.
- Hay conceptos muy simples que son difíciles de implementar con los sistemas relacionales (árbol recursivo).
- Diferencias de “impedancia” entre los lenguajes de programación y los lenguajes de bases de datos.

## ¿Cómo solucionar las deficiencias?

- Extender el modelo relacional.
- Bases de datos orientadas a objetos.
- Mapeo de Objetos a Bases de Datos Relacionales.

## Puntos a favor de las alternativas 1 y 3

- Existe una base matemática para el lenguaje de consulta (álgebra relacional y cálculo de tuplas).
- Mucha experiencia adquirida.
- Gran base de clientes/usuarios instalada.
- Un estándar en la industria: SQL.

## Puntos a favor de la alternativa 2

- El modelo de datos orientado a objetos ya incluye conceptos como herencia, encapsulamiento, polimorfismo, agregación, generalización.
- Los lenguajes de “programación” O.O. pueden ser extendidos para obtener un lenguaje de “programación” y “bd” unificado, como consecuencia se obtiene homogeneidad.
- Las bdoos soportan naturalmente el modelo de “navegación” utilizado por la web.

# Persistencia

- Ortogonalidad: un elemento A es ortogonal a un elemento B cuando no se debe prestar atención a B cuando se está trabajando con A.

- Persistencia: es el almacenamiento de información desde la memoria de manera que pueda ser recuperada cuando la aplicación se ejecute nuevamente.

- Persistencia ortogonal: es una forma de persistencia de objetos que adhiere a los siguientes principios (Atkinson & Morrison, 1995):

- Principio de independencia de la persistencia:

- Los programas se ven igual aunque manipulen información volátil o duradera.

- Principio de ortogonalidad de tipos:

- Todos los objetos son persistentes, más allá de su respectivo tipo.

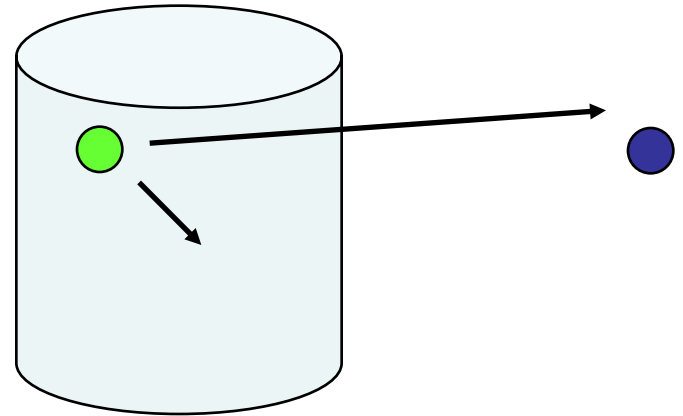
# Persistencia

- Persistencia por alcance:

- Toda instancia a la cual se puede llegar a partir de una instancia persistente, es a su vez persistente.


● Instancia no persistente

● Instancia persistente




# Persistencia

- Persistencia por alcance:
  - A nivel de código fuente, ¿qué significa la animación de la transparencia anterior?

 Instancia persistente

```
public class Person {  
  
    protected Car car;  
  
    public Person() {  
  
    }  
  
    public void setCar(Car aCar) {  
        this.car=aCar;  
    }  
  
    public Car getCar() {  
        return this.car;  
    }  
}
```

 Instancia no persistente

```
public class Car {  
  
    protected String brand;  
  
    public Car() {  
  
    }  
  
    //getters y setters  
}
```