



Relaciones entre entidades

Concepto

- Si las tablas tienen campos comunes, las entidades correspondiente se pueden relacionar
 - Cuando se relacionan entidades, un objeto de una entidad contiene una referencia al objeto u objetos de la entidad relacionada
 - Las relaciones simplifican el acceso a la capa de persistencia, permitiendo establecer joins entre entidades
- 

Tipos de relaciones

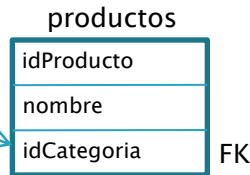
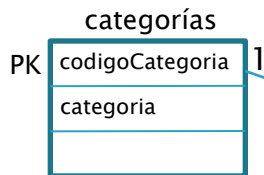
- Uno a muchos / muchos a uno. Son las más frecuentes. La entidad del lado uno está asociada a un conjunto de objetos del lado muchos. A su vez, cada entidad del lado muchos está asociada a un objeto del lado uno
 - Muchos a muchos. Cada entidad de ambos tiene asociada un conjunto de objetos de la otra entidad. Requieren una tabla de unión
- 

Uno a muchos/muchos a uno

- Las entidades contienen atributos con los objetos relacionados

```
public class Categoria{  
    private int codigoCategoria;  
    private String categoria;  
    :  
    private List<Producto> productos;  
    :  
}
```

```
public class Producto{  
    private int idProducto;  
    private String nombre;  
    :  
    private Categoria categoria;  
    :  
}
```



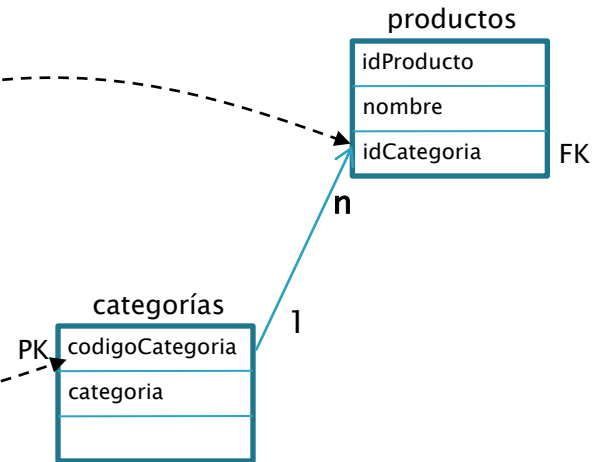
Las relaciones no tienen porque ser bidireccionales. Pueden ser unidireccionales, es decir, solo una entidad contiene referencia a la otra

Configuración

- Se utilizan unas anotaciones en los atributos de relación, a través de los que se indica la relación entre las tablas:

```
public class Categoria{  
    @OneToMany(mappedBy="categoria")  
    private List<Producto> productos;  
    :  
}
```

```
public class Producto{  
    @ManyToOne()  
    @JoinColumn(name="idCategoria",  
        referencedColumnName = "codigoCategoria")  
    private Categoria categoria;  
}
```



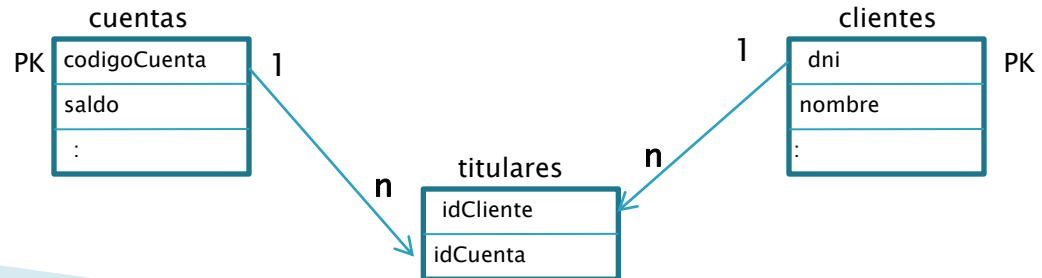
Muchos a muchos

- Las entidades contienen una colección con los objetos relacionados de la otra entidad

```
public class Cuenta{  
    private int codigoCuenta;  
    private double saldo;  
    private List<Cliente> clientes;  
    :  
}
```

```
public class Cliente{  
    private int dni;  
    private String nombre;  
    private List<Cuenta> cuentas;  
    :  
}
```

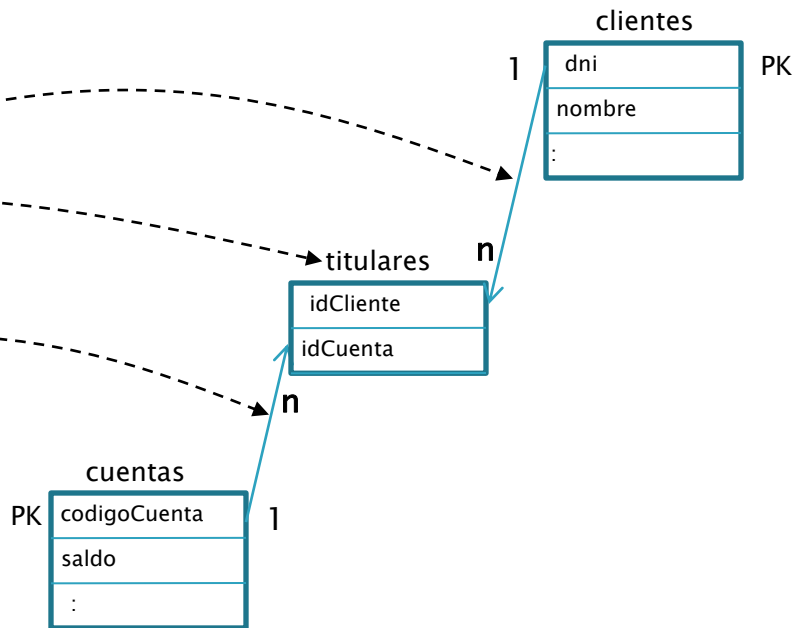
- Requiere tabla de unión:



Configuración

```
public class Cuenta{  
    @ManyToMany(mappedBy="cuentas")  
    private List<Cliente> clientes;  
    :  
}
```

```
public class Cliente{  
    @ManyToMany()  
    @JoinTable(name = "titulares",  
        joinColumns =  
            @JoinColumn(name="idCliente",  
                referencedColumnName = "dni"),  
        inverseJoinColumns =  
            @JoinColumn(name="idCuenta",  
                referencedColumnName = "codigoCuenta"))  
    private List<Cuenta> cuentas;  
    :  
}
```



Joins

- Un join permite definir uniones entre entidades relacionadas en una consulta JPQL
- Gracias a los join, podemos operar sobre una entidad en base a condiciones que afectan a la entidad relacionada.
- Dos tipos de join:
 - Join implícitos.
 - Join explícitos.

Join implícito

- No necesita la utilización de la palabra join
- Se emplea en relaciones muchos a uno cuando se opera sobre la entidad del lado muchos y la condición afecta a la entidad del lado uno
- Ejemplos:
 - `Select p From Producto p where p.categoria.nombre='Alimentación';`
 - `Select e From Empleado e where e.departamento.nombre='Informatica'`

Join explícito

- Se emplea la cláusula join dentro de from.
- Se utiliza cuando la condición afecta al lado muchos. Puede darse en relaciones muchos a muchos y uno a muchos cuando se opera en el lado uno.
- Ejemplos:
 - `Select distinct(c) From Categoria c join c.productos p where p.nombre like "%cable%"`
 - `Select c From Cliente c join c.cuentas b where b.saldo > 1000`