Java Persistence API

- Sesión 4: Mapeo entidad-relación (relaciones)

Índice

- Conceptos previos sobre relaciones
- Definición de los distintos tipos de relaciones
- Carga perezosa

Relaciones entre entidades

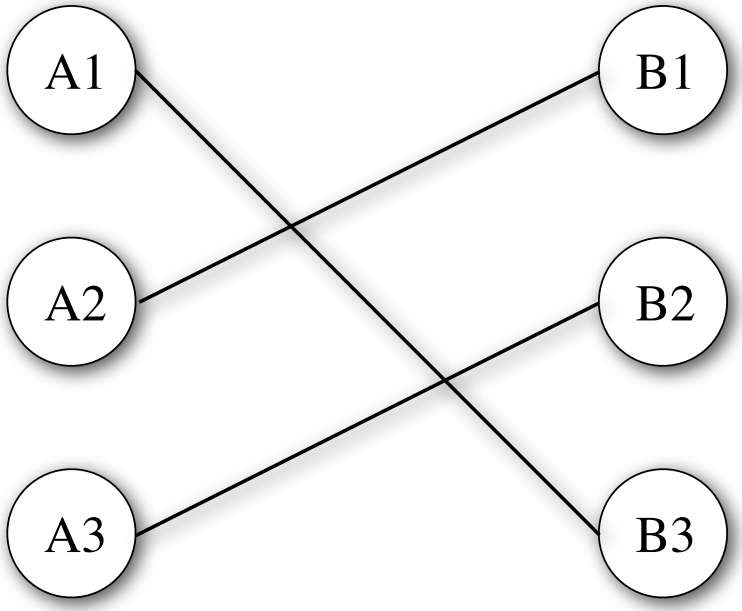
- En las relaciones se asocia una instancia de una entidad A con una o muchas instancias de otra entidad B
- En JPA se construyen las relaciones de una forma natural: definiendo en la instancia A el atributo de la relación con el tipo de la otra entidad B o como una colección de instancias de B

```
@Entity
public class Empleado {
    @OneToOne
    private Despacho despacho;
    @OneToMany
    private Collection<Proyecto> proyectos;
    //...
}
```

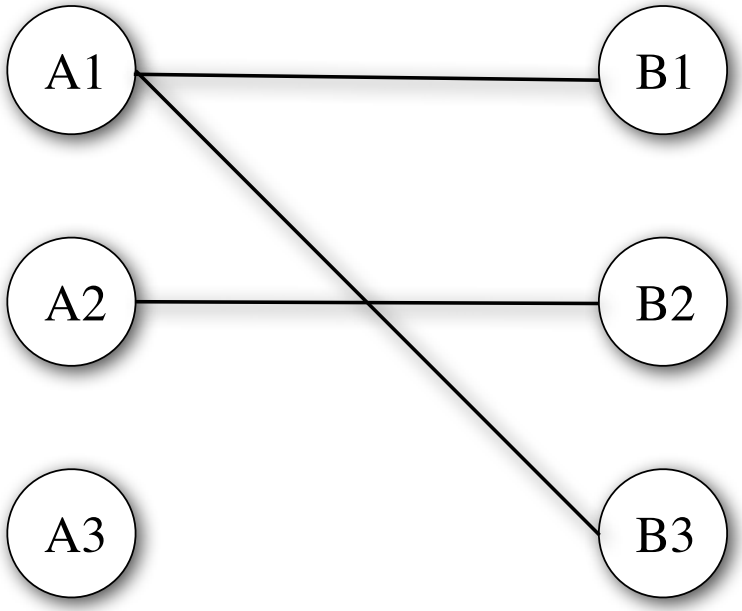
Cardinalidad

- La cardinalidad de una relación define si asociada a una instancia A hay una o muchas instancias B
- La relación inversa define también la cardinalidad: ¿la misma instancia de B puede ser el destino de más de una instancia A?
- Dependiendo de las respuestas tenemos cuatro posibles tipos de relaciones
  - one-to-one
  - one-to-many
  - many-to-one
  - many-to-many

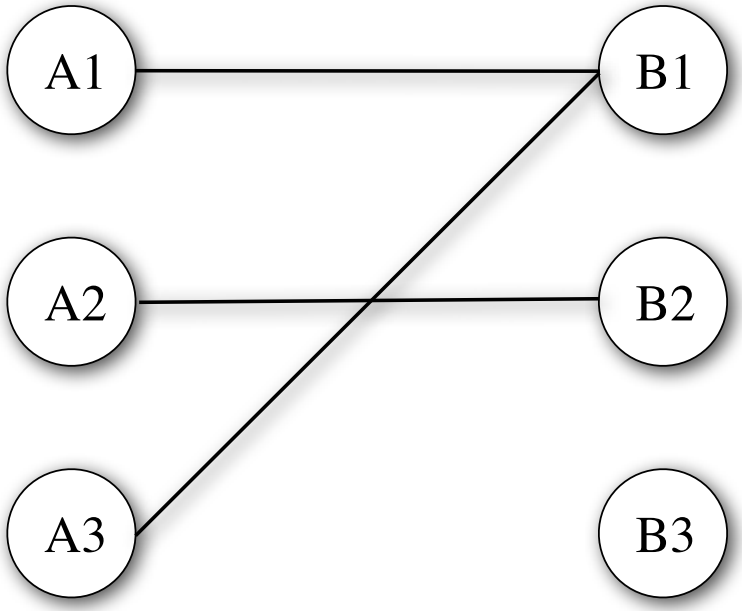
One-to-one



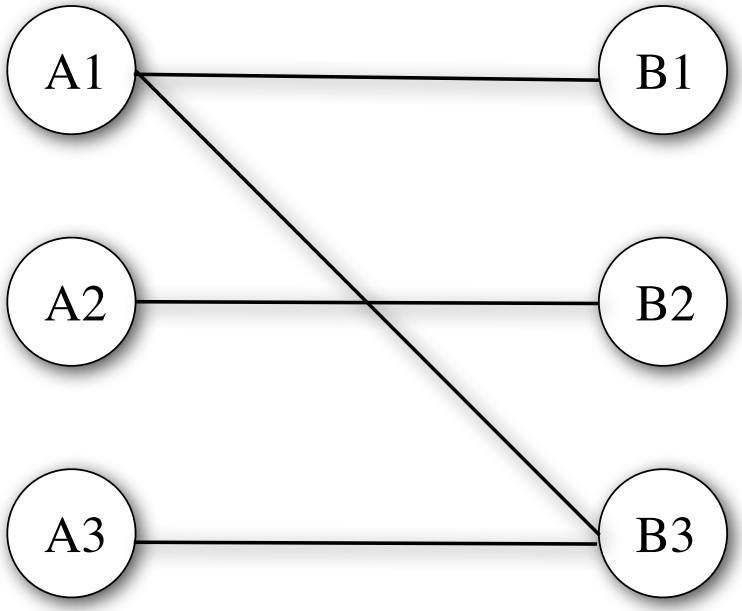
One-to-many



many-to-one



many-to-many



Cardinalidad y direccionalidad en JPA

- En JPA definimos la cardinalidad de una relación anotando los atributos que intervienen en la misma
- Posibles anotaciones
  - `@OneToOne`
  - `@OneToMany`
  - `@ManyToOne`
  - `@ManyToMany`
- La direccionalidad depede de si se define o no el atributo relacionado con A en la entidad B
- En el caso en que la relación fuera bidireccional, la relación inversa debe tener también la asocación inversa

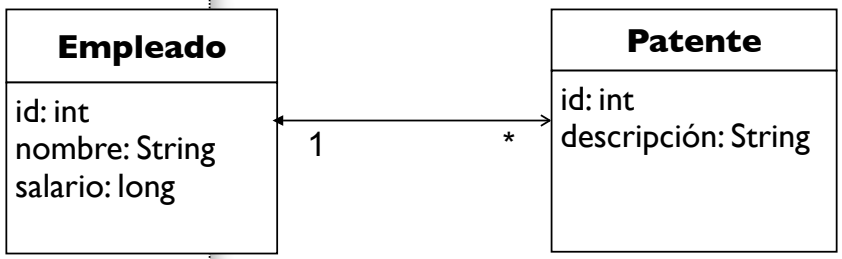
Mapeo de una relación

- Existen básicamente dos formas de mapear una relación en SQL
  - utilizando claves ajenas en una de las tablas
  - utilizando una tabla adicional (join) que implementa la asociación
- Las tablas con claves ajenas pueden implementar las relaciones one-to-one y one-to-many (y la many-to-one bidireccional)
- La tabla join se usa para implementar las asociaciones many-to-many

Un ejemplo

- Una relación bidireccional one-to-many

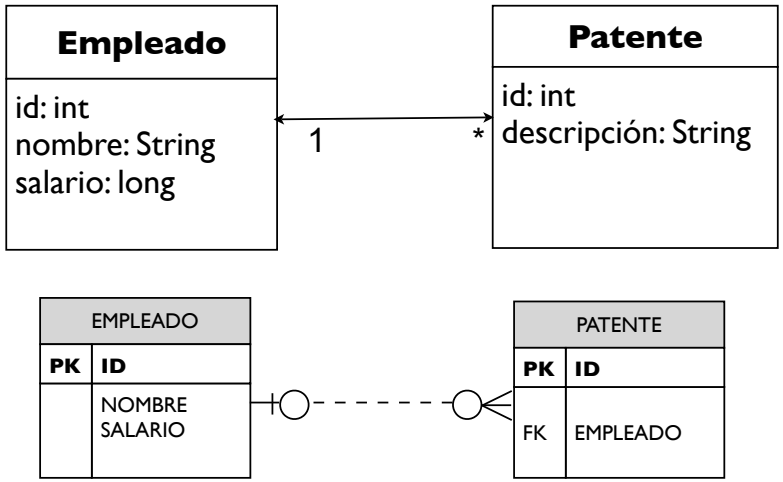
```
@Entity
public class Empleado {
    @OneToMany(mappedBy="empleado")
    private Collection<Patente> patentes;
    // ...
    }
}

@Entity
public class Patente {
    @ManyToOne
    private Empleado empleado;
    //...
    }
}
```



Mapeo con clave ajena

- En JPA se indica la tabla que contiene la clave ajena (la entidad propietaria de la relación) con el elemento `mappedBy` en el otro lado de la relación

```
@Entity
public class Empleado {
   @OneToMany(mappedBy="empleado")
   private Collection<Patente> patentes;
   // ...
   }
}

@Entity
public class Patente {
   @ManyToOne
   private Empleado empleado;
   //...
   }
}
```



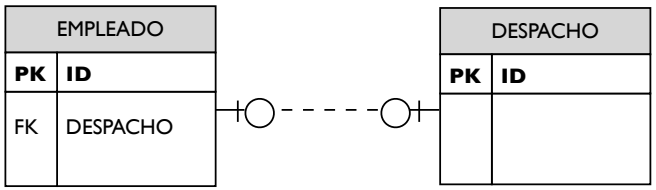
One-to-one unidireccional

```
@Entity
public class Empleado {
   // ...
   @OneToOne
   private despacho Despacho;
   // ...

   public void setDespacho(Despacho despacho) {
      this.despacho = despacho;
   }
}

@Entity
public class Despacho {
   // ...
}
```

One-to-one bidireccional

Entidad propietaria de la relación

Clave ajena

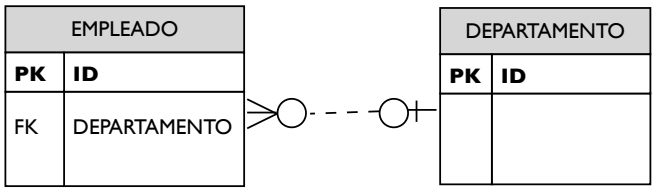



```
@Entity
public class Empleado {
   @OneToOne
   private Despacho despacho;
   // ...

   public void setDespacho(Despacho despacho) {
      this.despacho = despacho;
   }
}

@Entity
public class Despacho {
   @OneToOne(mappedBy="despacho")
   private Empleado empleado;
   // ...

   public Empleado setEmpleado(Empleado emp) {
      this.empleado = emp;
   }
}
```

One-to-many (Many-to-one) bidireccional

Entidad propietaria de la relación

Clave ajena



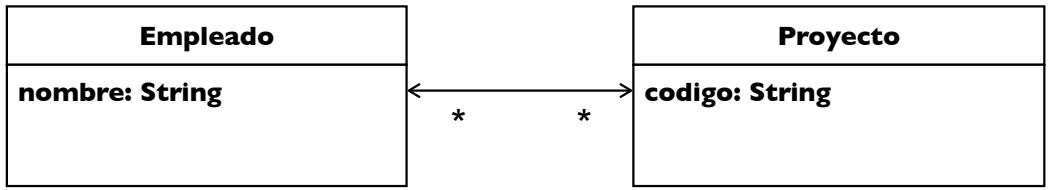

```
@Entity
public class Empleado {
    @ManyToOne
    private Departamento depto;
    // ...

    public void setDepartamento(Departamento depto) {
       this.depto = depto;
    }
}

@Entity
public class Departamento {
    @OneToMany(mappedBy="despacho")
    private Collection<Empleado> emp =
                          new HashSet<Empleado>();
    //...
}
```

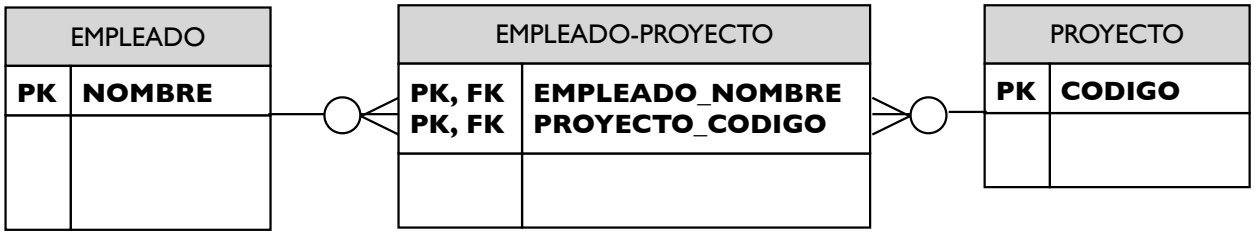
Many-to-many bidireccional (anotaciones)

```
@Entity
public class Empleado {
   @Id String nombre;
   @ManyToMany
   private Collection<Projecto> proyectos = new HashSet();

   public Collection<Proyecto> getProyectos() {
      return this.proyectos;
   }

   public void setProyectos(Collection<Proyecto> proyectos) {
      this.proyectos = proyectos;
   }

   // ...
}
```

```
@Entity
public class Proyecto {
   @Id String codigo;
   @ManyToMany(mappedBy="proyectos");
   private Collection<Empleado> empleados = new HashSet();

   public Collection<Empleado> getEmpleados() {
      return empleados;
   }
   // ...
}
```

Many-to-many bidireccional (mapeo)





Actualización de relaciones (1)

- Siempre se sincroniza con la base de datos la entidad que contiene la clave ajena
- La otra entidad conviene actualizarla también para mantener la relación consistente en memoria
- En una relación one-to-many se elimina una entidad de una relación poniendo a null la clave ajena
- Cambio de departamento de un empleado

```
Empleado empleado = em.find(Empleado.class, "Miguel Garcia");
Departamento deptoBaja = em.find(Depto.class, "D1");
Departamento deptoAlta = em.find(Depto.class, "D2");
empleado.setDepartamento(deptoAlta);
deptoBaja.getEmpleados().remove(empleado);
```

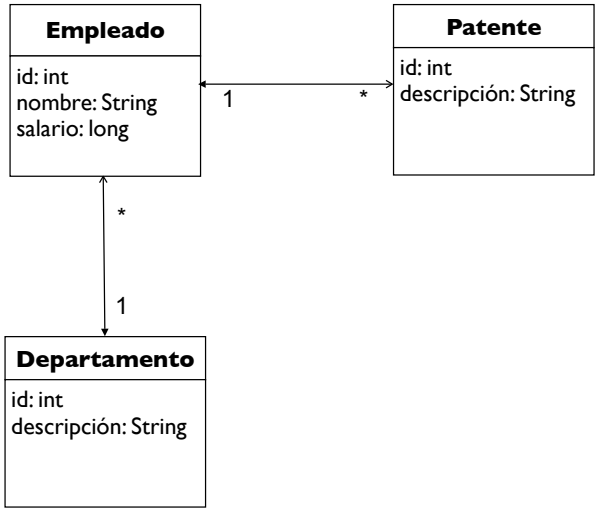
Actualización de relaciones (2)

- En el caso de una relación many-to-many, hay que actualizar (añadir o eliminar) el lado propietario de la relación

```
Empleado empleado = em.find(Empleado.class, "Miguel Garcia");
Proyecto proyectoBaja = em.find(Proyecto.class, "P04");
Proyecto proyectoAlta = em.find(Proyecto.class, "P01");
empleado.getProyectos().remove(proyectoBaja);
empleado.getProyectos().add(proyectoAlta);
```

Carga perezosa

- Al recuperar una colección de empleados de un departamento sería muy costoso recuperar también todas las patentes de cada uno de ellos
- En las relaciones JPA utiliza la cara perezosa por defecto
- Es posible desactivar este comportamiento con la opción `fetch=FetchType.EAGER` en el tipo de relación




```
@Entity
public class Empleado {
   @Id String nombre;
   @OneToMany(fetch=FetchType.EAGER)
   private Collection<Patentes> patentes = new HashSet();
   // ...
}
```

¿Preguntas?