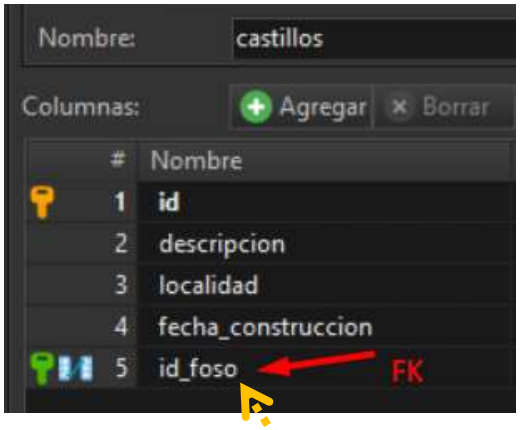
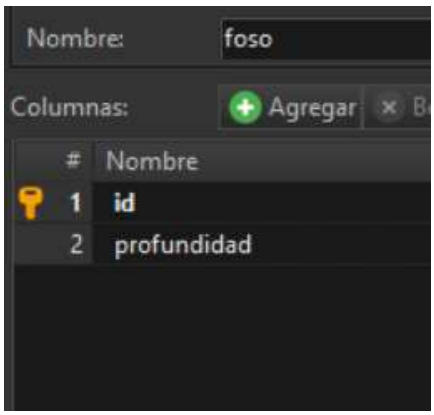
ONE TO ONE

TABLAS

Un Castillo tiene un Foso y un Foso pertenece a un Castillo





ENTIDADES (Clases)

Unidireccional Castillo → Foso (Castillo tiene un Foso):

```java
public class Castillo {

    @OneToOne
    @JoinColumn(name = "id_foso")
    private Foso foso;
```

```java
public class Foso {

    // No hay ningún atributo
    // de tipo Castillo
```

Unidireccional Foso → Castillo (Foso tiene un Castillo):

NO ES POSIBLE ESTA RELACIÓN, porque la columna FK del JOIN está en la tabla CASTILLOS

Bidireccional Castillo ←→ Foso (Castillo tiene un Foso y Foso tiene un Castillo):

```java
public class Castillo {

    @OneToOne
    @JoinColumn(name = "id_foso")
    private Foso foso;
```

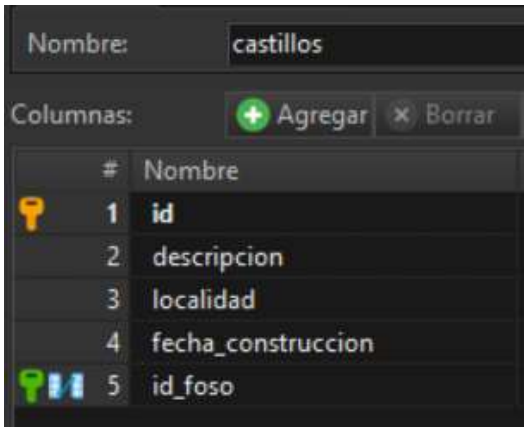
```java
public class Foso {

    @OneToOne(mappedBy = "foso")
    private Castillo castillo;
```

ONE TO MANY y MANY TO ONE

TABLAS

Un Castillo tiene muchos Soldados y un Soldado pertenece a un único Castillo



ENTIDADES (Clases)

Unidireccional Castillo → Soldado (Castillo tiene una lista de Soldado):

```java
public class Castillo {

    @OneToMany
    @JoinColumn(name = "id_castillo")
    private List<Soldado> soldados;
```

```java
public class Soldado {

    // No tiene ningún atributo
    // de tipo Castillo
```

Unidireccional Soldado → Castillo (Soldado tiene un Castillo):

```java
public class Castillo {

    // No tiene ningún atributo
    // de tipo Soldado
```

```java
public class Soldado {

    @ManyToOne
    @JoinColumn(name = "id_castillo")
    private Castillo castillo;
```

Bidireccional Castillo ←→ Soldado (Castillo tiene una lista de Soldado y Soldado tiene un Castillo):

```java
public class Castillo {

    @OneToMany(mappedBy = "castillo")
    private List<Soldado> soldados;
```

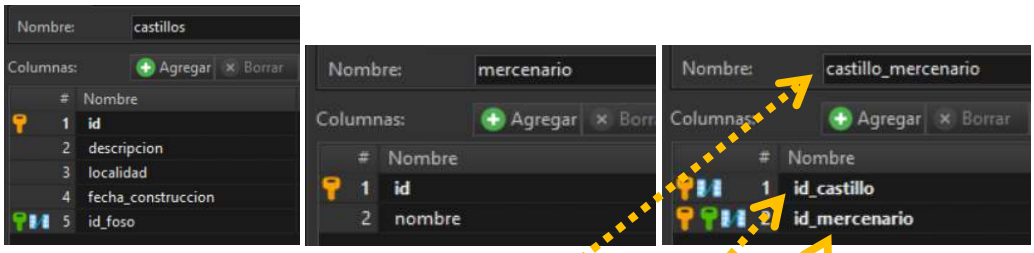
```java
public class Soldado {

    @ManyToOne
    @JoinColumn(name = "id_castillo")
    private Castillo castillo;
```

MANY TO MANY

TABLAS

Un Castillo tiene muchos Mercenarios y un Mercenario pertenece a muchos Castillos



ENTIDADES (Clases)

Unidireccional Castillo → Mercenario (Castillo tiene una lista de Mercenario):

```java
public class Castillo {

    @ManyToMany
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_castillo") },
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })
    private List<Mercenario> mercenarios;
```

```java
public class Mercenario {

    // No tiene ningún atributo
    // de tipo Castillo
```

Unidireccional Mercenario → Castillo (Mercenario tiene una lista de Castillo):

```java
public class Castillo {

    // No tiene ningún atributo
    // de tipo Mercenario
```

```java
public class Mercenario {

    @ManyToMany
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_castillo") },
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })
    private List<Castillo> castillos;
```

Bidireccional Castillo ←→ Mercenario (Castillo tiene una lista de Mercenario y Mercenario tiene una lista de Castillo):

Hay varias opciones. La más sencilla es poner JoinTable en las dos entidades:

```java
public class Castillo {

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_castillo") },
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })
    private List<Mercenario> mercenarios;
```

```java
public class Mercenario {

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_mercenario") },
        inverseJoinColumns = {@JoinColumn(name = "id_castillo") })
    private List<Castillo> castillos;
```

NOTA: Ten en cuenta que en joinColumns ponemos la FK perteneciente a la entidad donde ponemos el JoinTabla, y en inverseJoinColumns ponemos la otra.

Pero esta opción es redundante. Normalmente se pone JoinTable sólo en la entidad principal, y en la otra un mappedBy.

Así, si nuestra entidad principal es Castillo, pondríamos:

```java
public class Castillo {

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_castillo") },
        inverseJoinColumns = {@JoinColumn(name = "id_mercenario") })
    private List<Mercenario> mercenarios;
```

```java
public class Mercenario {

    @ManyToMany(mappedBy = "mercenarios")
    private List<Castillo> castillos;
```

O al revés:

```java
public class Castillo {

    @ManyToMany(mappedBy = "castillos")
    private List<Mercenario> mercenarios;
```

```java
public class Mercenario {

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "castillo_mercenario",
        joinColumns = {@JoinColumn(name = "id_mercenario") },
        inverseJoinColumns = {@JoinColumn(name = "id_castillo") })
    private List<Castillo> castillos;
```