Certified Tech Developer
The Ultimate Degree

Relacionamentos um para muitos e muitos para um

Prof. Esp. Marcelo Gonçalves de Souza

Quando a entidade ter seu ID como chave estrangeira em outra entidade geralmente declaramos a relação como @OneToMany, ou seja, um para muitos:

```java
@Entity
@Table
public class Professor {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private String curso;
    @OneToMany(mappedBy = "professor", fetch = FetchType.LAZY)
    private Set<Aluno> alunos = new HashSet<>();
```

Basta pensar que neste caso a entidade Professor pode estar vinculado a um aluno ou vários alunos na entidade Alunos, por isso configuramos a coleção alunos neste mapeamento. Entretanto seus atributos ficarão assim: **ID, NOME E CURSO**.

Agora, quando a entidade receber a chave estrangeira, que foi o caso da entidade Aluno, declaramos a relação como @ManyToOne, ou seja, muitos para um:

```java
@Entity
@Table
public class Aluno {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private int idade;
    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
    @JoinColumn(name = "professor_id")
    private Professor professor;
```

Nesta situação, a entidade que terá o ID de outra entidade como chave estrangeira, logo será mapeada como muitos para um. Não podendo esquecer de declarar também o @JoinColumn(name = “nome_do_campo”) mais a declaração de um atributo do mesmo tipo da entidade estrangeira, ou seja, Professor. Portanto seus atributos ficarão assim: **ID, NOME, IDADE E PROFESSOR_ID**.

Bons estudos!