CADERNO ESPECIALISTA JPA

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc126424049)

[1.1 O QUE É MAPEAMENTO OBJETO-RELACIONAL (ORM) 3](#_Toc126424050)

[1.2 Entendendo a diferença entre os métodos persist e merge 3](#_Toc126424051)

[2 MAPEAMENTO 4](#_Toc126424052)

[2.1 Mapeamento Básico 4](#_Toc126424053)

[2.2 Mapeamento de Relacionamentos 4](#_Toc126424054)

[2.2.1 Necessidade de mapear a volta em relacionamentos @ManyToOne, @OneToMany 5](#_Toc126424055)

[2.3 Conhecendo o EntityManager 5](#_Toc126424056)

[2.4 Mapeamento Avançado 5](#_Toc126424057)

[2.5 Mapeando entidades para geração do DDL 5](#_Toc126424058)

# INTRODUÇÃO

## O QUE É MAPEAMENTO OBJETO-RELACIONAL (ORM)

Mapeamento objeto-relacional é você representar uma tabela do banco de dados relacional usando uma classe de forma que possa automatizar a geração do SQL.

Exemplificando a comparação de modelagem de dados em um banco de dados de uma linguagem OO:

|  |  |
| --- | --- |
| Banco de Dados | Linguagem OO |
| Tabela | Classe |
| Coluna | Atributo |
| Linha | Objeto |

## Entendendo a diferença entre os métodos persist e merge

Uma das diferenças é que o persist serve apenas para inserir dados e o merge serve para inserir e fazer updates dos dados. O persist guarda a instância do objeto persistido o que pode ser útil para realizar alterações na instância do objeto. Já com o merge ele faz uma cópia do objeto e guarda a antiga na memória do Entity Manager, não guarda a atual na memória desse modo se você precisar realizar alterações no objeto que acabou de ser inserido ou atualizado você precisa capturar a instância do objeto no tempo do merge.

# MAPEAMENTO

## Mapeamento Básico

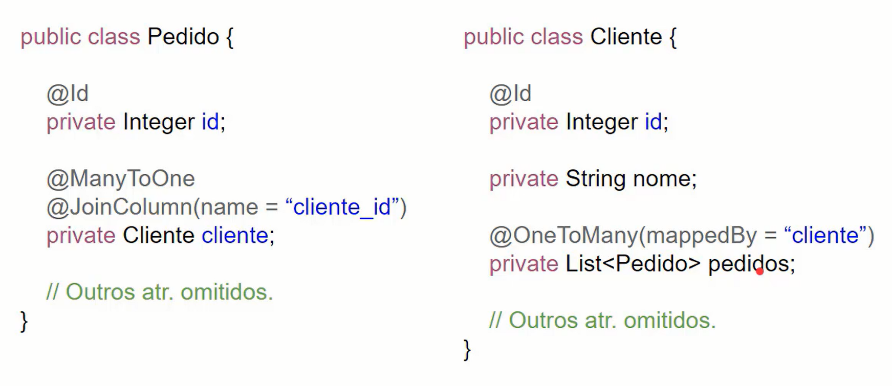
Lorem

## Mapeamento de Relacionamentos

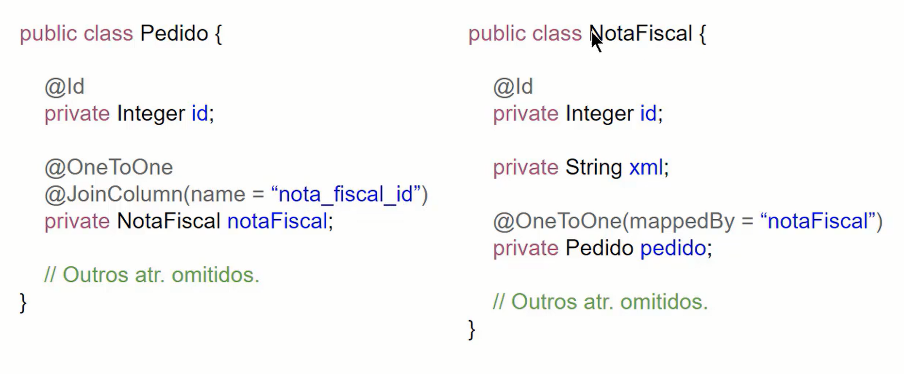
**@ManyToOne:** É quando temos a possibilidade de ter vários registros, de uma tabela qualquer, fazendo referência para um registro especifico que pode ser da mesma tabela ou, como na maioria das vezes, uma tabela diferente.



**@OneToMany:** É quando temos a possibilidade de um registro especifico ser referenciado por vários outros. Basicamente, o contrário do que foi descrito para @ManyToOne. Só que agora estamos falando de coleções.



**@OneToOne:** É quando uma tabela restringe seus registros a não fazerem referências repetidas. Uma mesma referência não pode ser feita duas vezes.



**@ManyToMany:** É quando temos a possibilidade de ter um registo referenciando vários outros, ao mesmo tempo que pode ser referenciado várias vezes. Como estamos falando de múltiplas referências de e para um mesmo registro, lidamos também com coleções.



### Necessidade de mapear a volta em relacionamentos @ManyToOne, @OneToMany

Você não precisa mapear a volta quando criar relacionamentos @ManyToOne, apenas analise se você precisa da volta, e caso você queira utilizar a volta você vai na outra classe e mapeia o atributo com o @OneToMany(mappedBy = “own\_class”) e o atributo do mapeamento sempre vai ter que ser uma collection, porque é ”UM” para “MUITOS”.

### Remoção e entidade já referenciadas

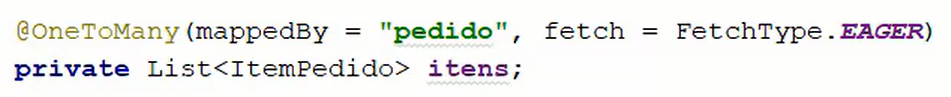
Quando uma entidade possui relacionamentos com outras entidades, mapeamentos, e você vai tentar remover essa entidade vai ocorrer exceções de foreign key, para resolver esse problema é necessário utilizar operações em cascata. Ou outra forma é remover cada dependência do relacionamento antes de remover a entidade.

### Entendendo o funcionamento de Eager e Lazy Loading

Eager (Ansioso) e Lazy (Lerdo) são utilizados pelo Hibernate para manipular a forma com que os atributos são retornados pela consulta no banco de dados. Por padrão atributos únicos são Eager, ou seja, eles são retornados assim que a classe é instanciada, de forma automática. Pelo contrário atributos de collections, como listas tem como padrão Lazy que significa que o hibernate não retorna eles automaticamente na instância da classe, eles só serão carregados caso você utilize os objetos da collection, mas não basta só chamar o atributo, você tem que utilizar um método para o JPA incluir ele na consulta. O JPA não retorna os dados de consulta de atributos Lazy.

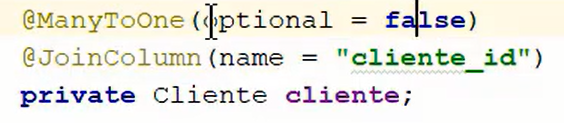
Você pode modificar esse comportamento caso necessite que um atributo de lista seja incluído na consulta imediatamente assim que a classe é chamada, e também pode omitir valores únicos de serem carregados imediatamente com Lazy.

Para mudar você utiliza a parâmetro fetch na annotation de relacionamento:



### Atributo Optional

Sempre que precisarmos definir que um atributo seja obrigatório nós utilizamos o atributo optional na @annotation de relacionamento. Caso necessitemos que seja obrigatório utilizamos:



Por padrão todos os atributos são opcionais, para mudar precisa setar manualmente para false.

## Conhecendo o EntityManager

### Estados e ciclo de vida dos objetos

S

## Mapeamento Avançado

Lorem

## Mapeando entidades para geração do DDL

Lorem