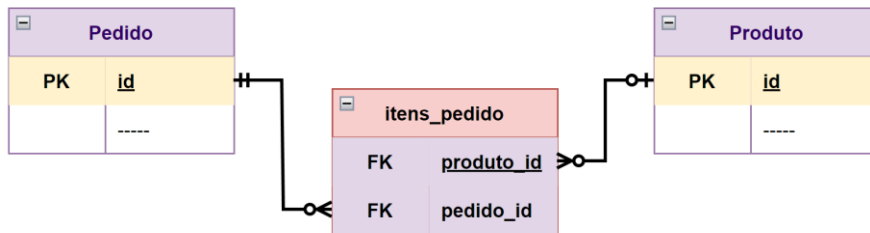




Quando é preciso relacionar **muitos para muitos** é necessário a criação de uma tabela intermediária, essa que é comumente chamada de **“tabela de join”**, onde ela é **composta** apenas pelos **ID’s das tabelas** participantes do **relacionamento**:



****Um pedido pode ter muitos produtos, um produto pode estar presente em vários pedidos****

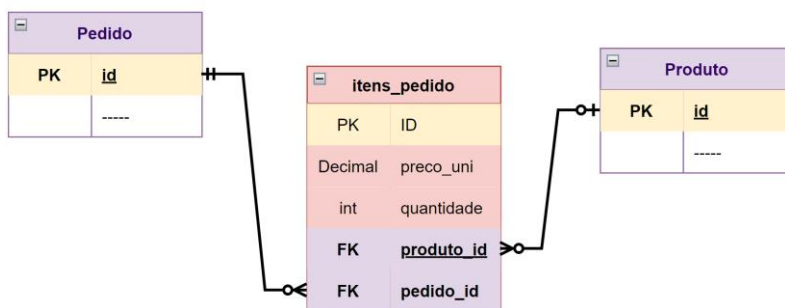
Uma **tabela de join**, **simples nesse nível** de apenas **relacionar os ID’s** pode ser mapeada com **@ManyToMany**, e renomeada (se quiser) usando **@JoinTable** com o nome das colunas configuradas nos parâmetros:

```
public class Pedido {

    @ManyToMany
    @JoinTable(name = "itens_pedidos",
        joinColumns = @JoinColumn(name = "pedido_id"),
        inverseJoinColumns = @JoinColumn(name = "produto_id"))
    private List<Produto> produtos;

}
```

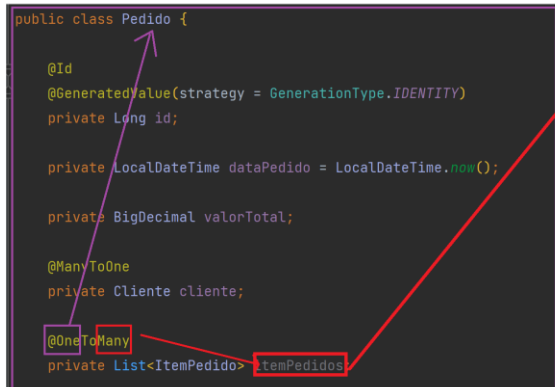
Só que nem sempre as **“join tables”** são simples, as vezes **são precisos mais atributos para representar esse tipo de relacionamento além dos ID’s**. Por exemplo, eu **quero saber o preço unitário do produto QUANDO** ele foi comprado além da **quantidade de unidades** naquele **item de pedido**:



Quando existe essa complexidade, é **necessário a criação de outra entidade** que a **represente**, nesse caso uma entidade de **ItemPedido** que vai ter **relacionamento de muitos para um produto** e de **muitos para um pedido**.

O que vai acontecer na prática é que o **Pedido** vai ter **muitos ItemPedido**, esses **ItemPedido** vão ter **muitos produtos** e os **itemPedido** podem fazer parte de vários **pedidos**:

```
public class Pedido {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private LocalDateTime dataPedido = LocalDateTime.now();  
  
    private BigDecimal valorTotal;  
  
    @ManyToOne  
    private Cliente cliente;  
  
    @OneToMany  
    private List<ItemPedido> itemPedidos;  
}
```



```
public class ItemPedido {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private BigDecimal precoUnitario;  
  
    private int quantidade;  
  
    @ManyToOne  
    private Pedido pedido;  
  
    @ManyToOne  
    private Produto produto;  
}
```

