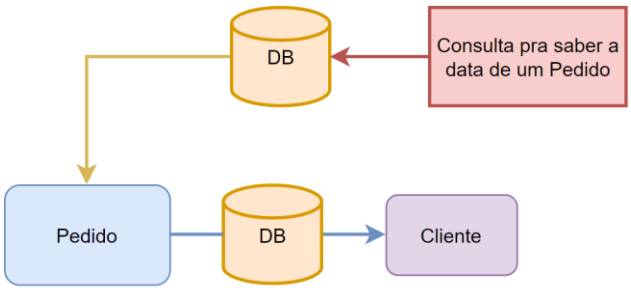
Na **JPA** existe **duas estratégias de carregamento de relacionamentos**, **EAGER** e **LAZY** ***(Ansioso/preguiçoso)***. Por padrão **todo relacionamento terminado em ToOne** usa **EAGER**, enquanto **relacionamentos terminados em ToMany** usam **LAZY**.

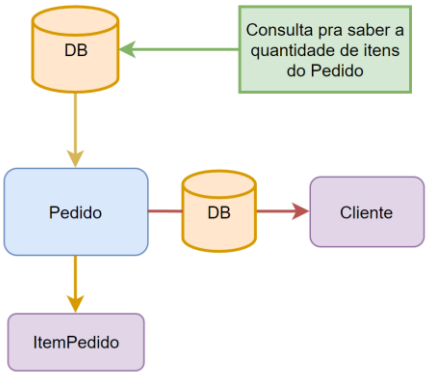
EAGER: significa que todas as **entidades relacionadas** com a **Entidade buscada serão carregadas para a memória da aplicação mesmo que esses** relacionamentos **não estejam sendo utilizados.** Um exemplo é uma **consulta** em um **Pedido** só pra saber a data dele, mas mesmo assim ele busca o Cliente associado, mesmo que eu não use:




```
Pedido pedido = entityManager.find(Pedido.class, 1L);
System.out.println(pedido.getDataPedido());
```

```
select
    pedido0_.id as id1_3_0_,
    pedido0_.cliente_id as cliente_4_3_0_,
    pedido0_.dataPedido as datapedi2_3_0_,
    pedido0_.valorTotal as valortot3_3_0_,
    cliente1_.id as id1_1_1_,
    cliente1_.cpf as cpf2_1_1_,
    cliente1_.nome as nome3_1_1_
from
    pedidos pedido0_
left outer join
    clientes cliente1_
        on pedido0_.cliente_id=cliente1_.id
where
    pedido0_.id=?
```

Lazy: Em contrapartida o relacionamento de **itemPedido** **não** executou uma busca, isso porque ele é termina em **toMany** e por padrão é **Lazy**, o que significa que ele só faria a busca se eu **ENTRASSE NESSE RELACIONAMENTO EXPLICITAMENTE**




```
Pedido pedido = entityManager.find(Pedido.class, 1L);
System.out.println(pedido.getItens().size());
```

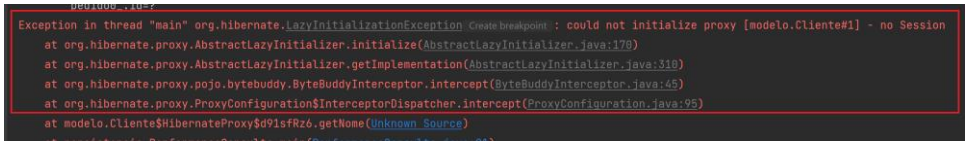
```
    produto1_.categoria_id as categori6_4_2_,
    produto1_.dataCadastro as datacada2_4_2_,
    produto1_.descricao as descrica3_4_2_,
    produto1_.nome as nome4_4_2_,
    produto1_.preco as preco5_4_2_,
    categoria2_.id as id1_0_3_,
    categoria2_.nome as nome2_0_3_
from
    itens_pedidos itens0_
left outer join
    produtos produto1_
        on itens0_.produto_id=produto1_.id
left outer join
    categorias categoria2_
        on produto1_.categoria_id=categoria2_.id
where
    itens0_.pedido_id=?
```

Mesmo **que os relacionamentos possuam seus padrões não significa que não podem ser especificados com o comportamento que a gente quiser**, então eu posso **explicitar pro** meu **pedido** que ele vai usar a **ESTRATÉGIA LAZY** para **carregar o relacionamento** do **Cliente**:

```
@ManyToOne(fetch = FetchType.LAZY)
private Cliente cliente;
```

Colaterais de Lazy

Pode acontecer de algumas consultas de relacionamentos Lazy darem exception de **LazyInitialization**:



Essa **Exception** acontece quando **fazemos uma consulta** em **alguma Entidade** e p**recisamos usar o relacionamento dela que está como Lazy**, porém quando o **entityManager chega no ponto do código que precisar buscar esse relacionamento**, **ele** já está fechado, o que causa a **exceção**.

Para prevenir isso são usadas “**querys planejadas**”, caso você tenha uma **situação que você precisa carregar um determinado relacionamento de maneira Eager,** mas que está mapeado como **Lazy** você monta uma **“query planeja”** usando **“JOIN FETCH”:**

```
public Pedido buscarClienteDoPedido(long id) {
    String jpql = "SELECT p FROM Pedido p JOIN FETCH p.cliente WHERE p.id = :id";

    return entityManager.createQuery(jpql, Pedido.class).setParameter( s: "id", id).getSingleResult();
}
```

A ideia é que **todas as consultas** que precisam usar o Relacionamento mapeado com Lazy de uma certa Entidade, mas com o comportamento de Eager, **façam isso de maneira isolada do resto da Aplicação.** Assim **teremos por padrão comportamento Lazy onde não precisamos carregar os dados** relacionados, e o comportamento Eager **específico em querys específicas.**