Java Persistence API (JPA)
Mapeamento de Relacionamentos

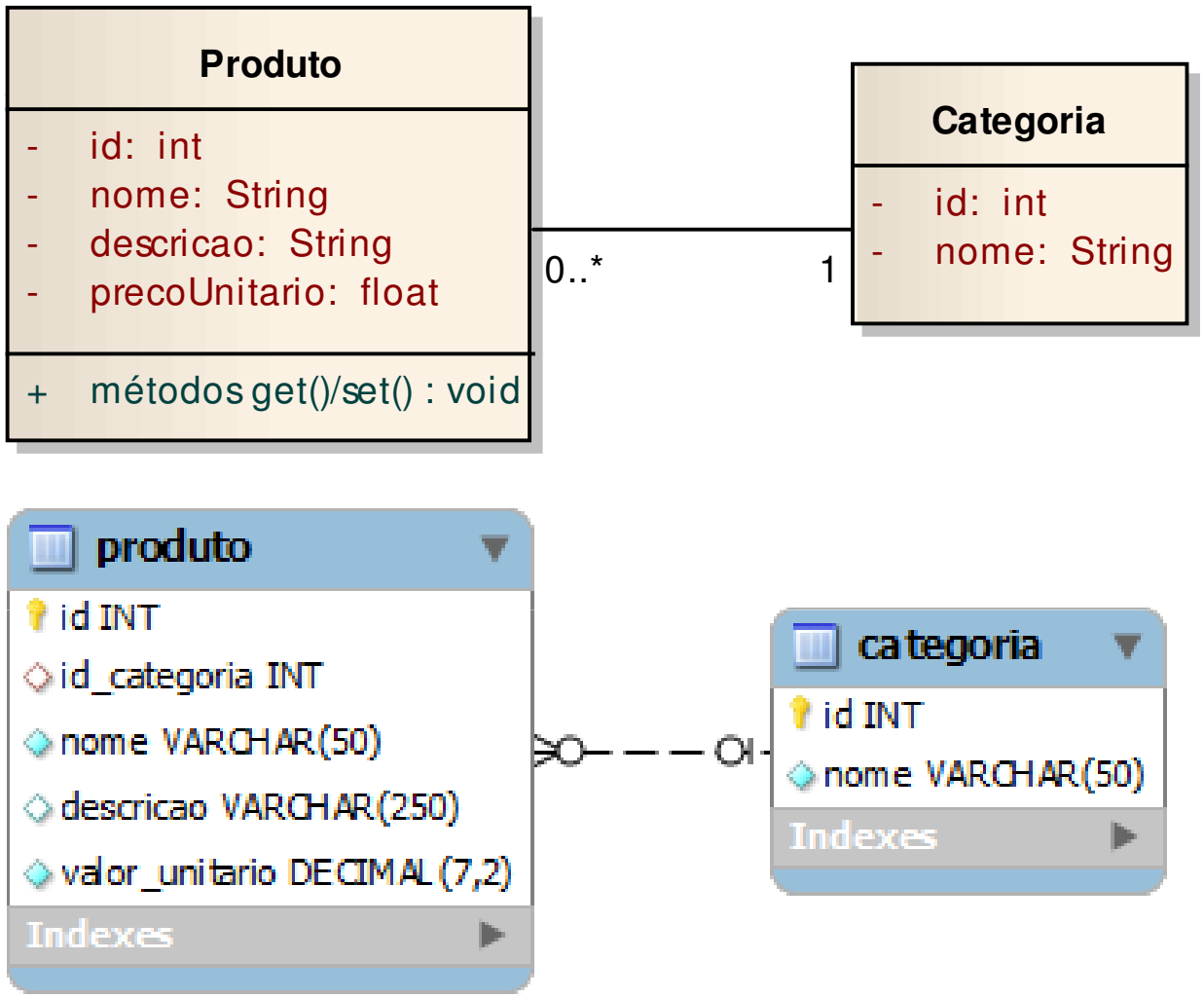
Fernando dos Santos
fernando.santos@udesc.br

UDESC
Ibirama

Mapeamento de Relacionamentos

Associação, Agregação, Composição

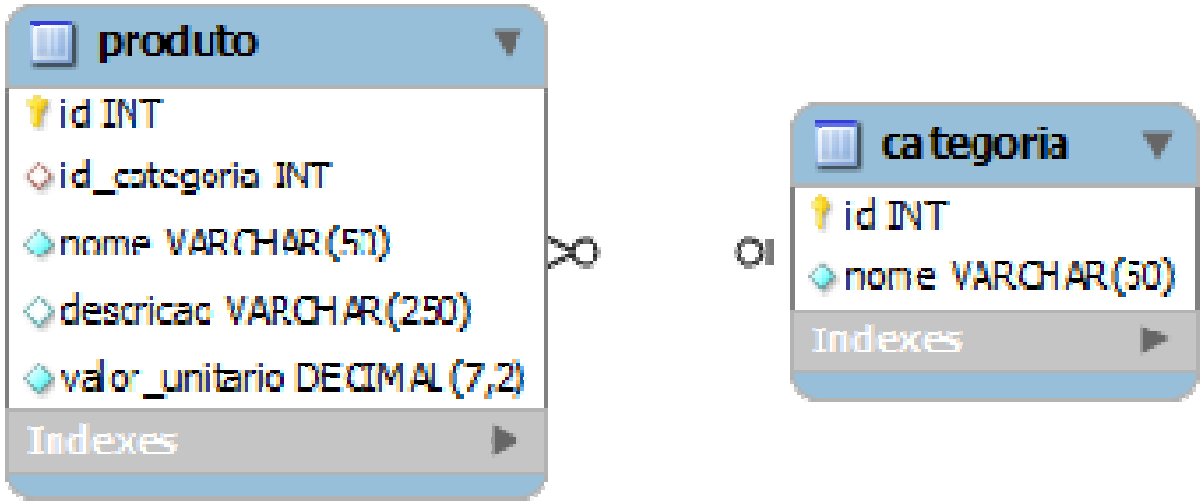
- Tipos de Relacionamentos mapeáveis:
  - Um para Um
  - **Muitos para Um**
  - Um para Muitos



Relacionamento **Muitos para Um**

Mapeamento

- Muitos produtos estão associados à uma categoria.



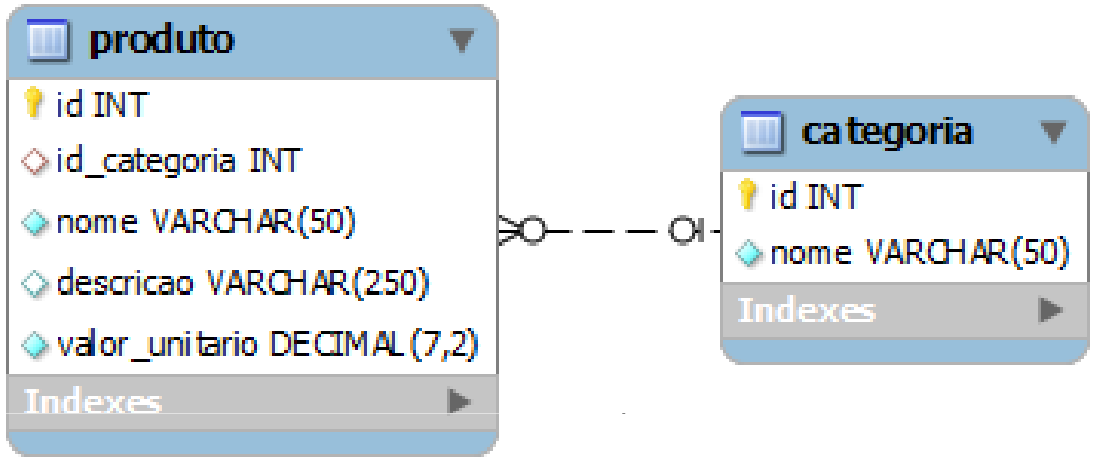
- Entidade Produto possui atributo para associar com Categoria
  - na tabela: coluna id_categoria INT (chave estrangeira)
  - na classe: atributo categoriaProduto do tipo Categoria (classe)
- O mapeamento é definido na entidade “muitos” (que possui a chave estrangeira) - Produto

Relacionamento **Muitos para Um**

Mapeamento com anotações @ManyToOne e @JoinColumn

```
@Entity
@Table(name="produto")
public class Produto implements Serializable {
    @Id
    @GeneratedValue
    private int id;
    @Column(name="nome")
    private String nome;
    @Column(name="descricao")
    private String senha;
    @Column(name="valor_unitario")
    private float precoUnitario;

    @ManyToOne
    @JoinColumn(name="id_categoria", nullable=false)
    private Categoria categoriaProduto;
}
```



Relacionamento **Muitos para Um**

Anotações @ManyToOne e @JoinColumn - OBSERVAÇÕES

- **Não** se deve criar um atributo para o valor da chave estrangeira...
  - exemplo: **private int idCategoria [errado !]**
- Deve-se criar um atributo para o objeto relacionado...
  - exemplo: **private Categoria categoriaProduto [correto !]**

- A entidade alvo (Categoria) já deve estar mapeada
  - deve existir uma classe Categoria, com atributos e métodos get/set;
  - a classe Categoria deve estar mapeada - **@Entity**
  - os atributos da classe Categoria devem estar mapeados - **@Column**

Relacionamento **Muitos para Um**

Manipulações (1)

- Criar novo produto **(1)** e nova categoria **(2)**;
- Associar categoria ao produto **(3).**

```
Produto prod = new Produto();  // (1)
prod.setNome("TV");
prod.setDescricao("TV LCD 40 Polegadas");
prod.setPrecoUnitario(999);

Categoria cat = new Categoria();  // (2)
cat.setNome("Eletrônicos");

prod.setCategoriaProduto(cat);   // (3)

em.getTransaction().begin();
em.persist(cat);    // (4)
em.persist(prod);
em.getTransaction().commit();
```

- **É necessário persistir a categoria e o produto (4)**

Relacionamento **Muitos para Um**

Manipulações (2)

- Criar novo produto **(1)**; Associar categoria já existente **(2)**;

```
Produto prod = new Produto();    // (1)
prod.setNome("Blu-Ray");
prod.setDescricao("Blu-Ray Player");
prod.setPrecoUnitario(150);

Categoria cat = em.find(Categoria.class, 1);

prod.setCategoriaProduto(cat);     // (2)

em.getTransaction().begin();
em.persist(prod);     // (3)
em.getTransaction().commit();
```

- **É necessário persistir somente o produto (3)**

Relacionamento **Muitos para Um**
Manipulações (3)

- Trocar a categoria de um produto.

```
// No banco de dados, o produto 1 (TV)
// possui a categoria 1 (Eletrônicos)
Produto prod = em.find(Produto.class, 1);

// Buscar outra categoria (2-Informática)
Categoria cat = em.find(Categoria.class, 2);

em.getTransaction().begin();
// Trocar a categoria do produto
prod.setCategoriaProduto(cat);   // (1)

em.getTransaction().commit();
```

- **O update será feito automáticamente no banco de dados (1)**

Relacionamento **Muitos para Um**
Manipulações (4)

- Buscar um produto **(1)**

```
Produto prod = em.find(Produto.class, 1);  // (1)

System.out.println("Produto: "+prod.getNome());
System.out.println("Categoria do Produto: "+prod.getCategoriaProduto().getNome());// (2)
```

- **A categoria é recuperada junto com o produto (2)**
  - será impresso:

```
Produto: TV
Categoria do Produto: Informática
```

Relacionamento **Muitos para Um**

Manipulações (5)

- Buscar uma categoria que possui produtos **(1)**
- Tentar remover a categoria **(2)**

```
Categoria cat = em.find(Categoria.class, 1);  // (1)

em.getTransaction().begin();
em.remove(cat);   // (2)

em.getTransaction().commit(); // (3)
```

- **O commit falhará, e será gerada exceção RollbackException (3)**

```
Exception in thread "main" javax.persistence.RollbackException: Error while commiting the transaction
        at org.hibernate.ejb.TransactionImpl.commit(TransactionImpl.java:71)
        at SistemaVendasManyToOneTestes.main(SistemaVendasManyToOneTestes.java:17)
```

Relacionamento **Muitos para Um**

Consultas (1)

- Buscar todos os produtos de uma categoria (informada por parâmetro)
- Situação no banco de dados:

produto

ID	Descrição	ID Categoria
1	TV LCD 40 Polegadas	1
2	Blu-Ray Player	1
3	Notebook HP Pavilion	2

categoria

ID	Nome
1	Eletrônicos
2	Informática
3	Livros

- Consulta:

```
Query cons =  em.createQuery("select p from Produto p where p.categoriaProduto.id = :idCat");
cons.setParameter("idCat", 1);

List<Produto> produtos = cons.getResultList();
for(Produto prod : produtos){
   System.out.println("Cód: "+prod.getId()+" Descr: "+prod.getDescricao());
}
```

O mapeamento pode ser utilizado para fazer JOIN em consultas.

- Resultado:

ID	Descrição	ID Categoria
1	TV LCD 40 Polegadas	1
2	Blu-Ray Player	1

Relacionamento **Muitos para Um**

Consultas (2) - OBSERVAÇÕES

- Na query, usa-se o **atributo** da classe Produto
  - select p from Produto p where p.**categoriaProduto**.id ...

- Não se usa o **campo da tabela**
  - select p from Produto p where p.**id_categoria** ... **[errado!]**

modo correto!
usar o atributo

```
Query cons =  em.createQuery("select p from Produto p where p.categoriaProduto.id = :idCat");
cons.setParameter("idCat", 1);

List<Produto> produtos = cons.getResultList();
for(Produto prod : produtos){
    System.out.println("Cód: "+prod.getId()+" Descr: "+prod.getDescricao());
}
```

Relacionamento **Muitos para Um**

Consultas (3)

- Buscar quantidade de produtos por categoria
- Situação no banco de dados:

produto

ID	Descrição	ID Categoria
1	TV LCD 40 Polegadas	1
2	Blu-Ray Player	1
3	Notebook HP Pavilion	2

categoria

ID	Nome
1	Eletrônicos
2	Informática
3	Livros

- Consulta:

```
Query cons = em.createQuery("select p.categoriaProduto,
                                    count(p)
                             from Produto p
                             group by p.categoriaProduto");
List<Object[]> resultados = cons.getResultList(); //  objetos retornados em um array
for (Object[] result : resultados) {
    Categoria cat = (Categoria) result[0];   //  posição [0] é a categoria
    Long qtd = (Long) result[1];             //  posição [1] é a quantidade
    System.out.println("Categoria: " + cat.getNome() + ", Qtd: " + qtd);
}
```

- Resultado:

```
Categoria: Eletrônicos, Qtd: 2
Categoria: Informática, Qtd: 1
```

E as categorias que não possuem produtos?

Relacionamento **Muitos para Um**

Consultas (4)

- Buscar quantidade de produtos por categoria (**TODAS**)
- Situação no banco de dados:

produto

ID	Descrição	ID Categoria
1	TV LCD 40 Polegadas	1
2	Blu-Ray Player	1
3	Notebook HP Pavilion	2

categoria

ID	Nome
1	Eletrônicos
2	Informática
3	Livros

- Consulta:

```
Query cons = em.createQuery("select c,
                                    (select count(p)
                                       from Produto p
                                       where p.categoriaProduto = c)
                                from Categoria c");
List<Object[]> resultados = cons.getResultList(); //  retornados em um array
for(Object[] result : resultados){
    Categoria cat = (Categoria)result[0];    //  posição [0] é a categoria
    Long qtd = (Long)result[1];              //  posição [1] é a quantidade
    System.out.println("Categoria: "+cat.getNome()+", Qtd: "+qtd);
}
```

- Resultado:

```
Categoria: Eletrônicos, Qtd: 2
Categoria: Informática, Qtd: 1
Categoria: Livros,      Qtd: 0
```

Relacionamento **Muitos para Um**

Consultas (5)

- Buscar categorias que não possuem produtos (subconsulta)
- Situação no banco de dados:

produto

ID	Descrição	ID Categoria
1	TV LCD 40 Polegadas	1
2	Blu-Ray Player	1
3	Notebook HP Pavilion	2

categoria

ID	Nome
1	Eletrônicos
2	Informática
3	Livros

- Consulta:

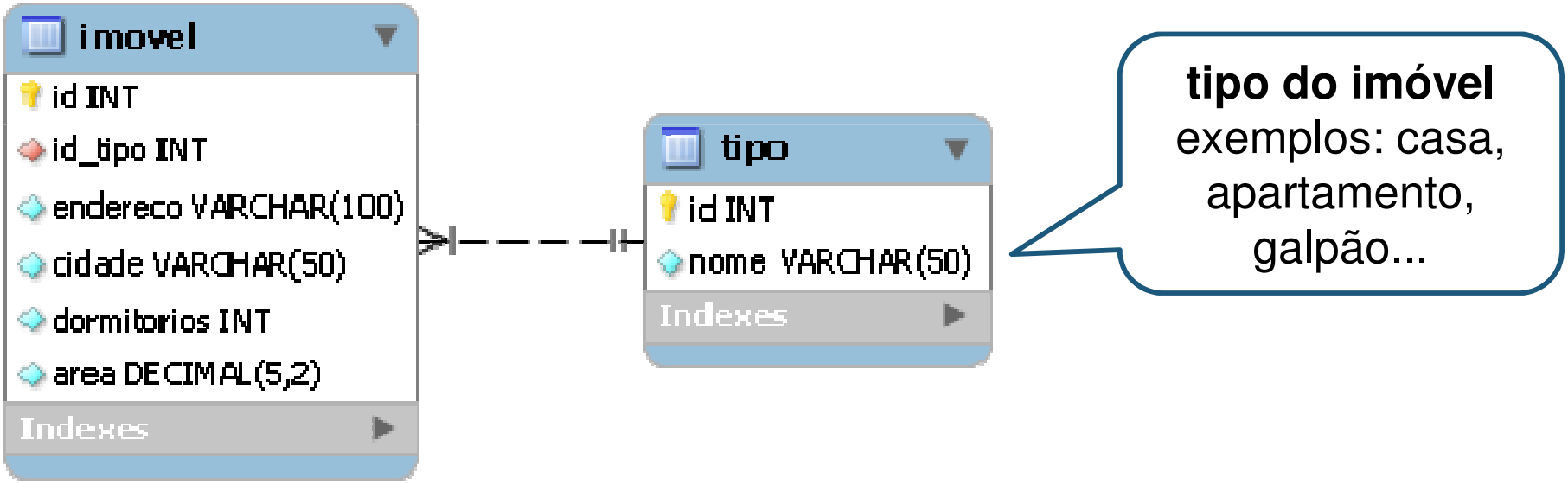
```
Query cons = em.createQuery("select c
                                from Categoria c
                               where c not in ( select p.categoriaProduto from Produto p )");
List<Categoria> categorias = cons.getResultList();
for(Categoria cat : categorias){
    System.out.println("Cód: "+cat.getId()+" Nome: "+cat.getNome());
}
```

- Resultado:

ID	Nome
3	Livros

Exercício

- Criar o seguinte relacionamento Many-to-One:



- Criar opções para:
  - incluir/editar/remover tipos de imóvel
  - ao cadastrar um imóvel, informar qual é o seu tipo
  - consultas:
    - quantidade de imóveis por tipo (todos os tipos)
    - tipos que não possuem imóveis
    - todos os tipos, e para cada tipo, mostrar todos os imóveis do tipo

Bibliografia

- BAUER, Christian; KING, Gavin. **Java Persistence com Hibernate.** Rio de Janeiro: Ciência Moderna, 2007. 844 p.

- BURKE, Bill; MONSON-HAEFEL, Richard. **Enterprise JavaBeans 3.0**. 5.ed. São Paulo: Prentice Hall, 2007. 538 p.

- **The Java EE 6 Tutorial**, parte VI (Persistence)
  - http://download.oracle.com/javaee/6/tutorial/doc/