Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

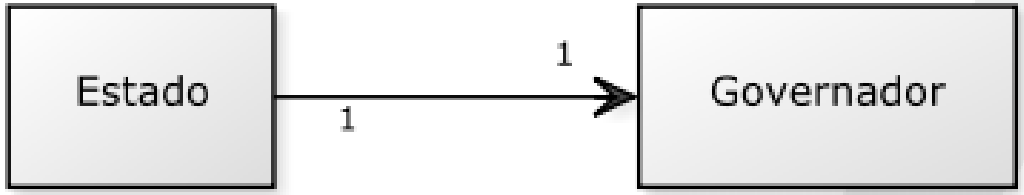
PROJETO – TIPOS DE RELACIONAMENTO

RELACIONAMENTOS

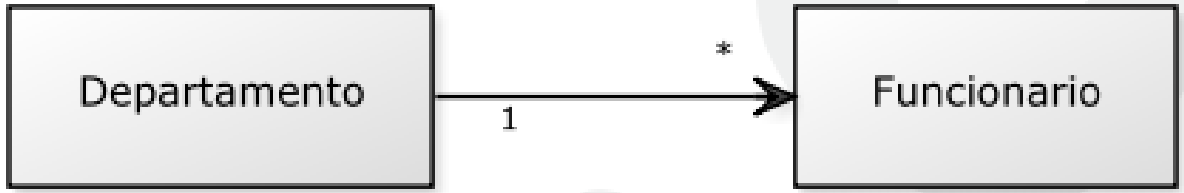
Spring

- Os relacionamentos entre as entidades de um domínio devem ser expressos na modelagem através de vínculos entre classes.
- De acordo com a JPA, podemos definir quatro tipos de relacionamentos de acordo com a cardinalidade (Multiplicidade)

One to One (Um para Um): Por exemplo, um estado é governado por apenas um governador e um governador governa apenas um estado.



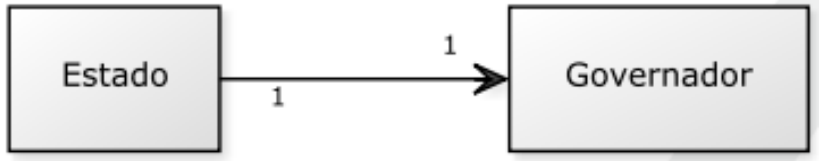
One to Many (Um para Muitos): Por exemplo, um departamento possui muitos funcionários e um funcionário trabalha em apenas em um departamento.



Many to One (Muitos para Um): Por exemplo, um pedido pertence a apenas um cliente e um cliente faz muitos pedidos.

Many to Many (Muitos para Muitos): Por exemplo, um livro possui muitos autores e um autor possui muitos livros.

One to One



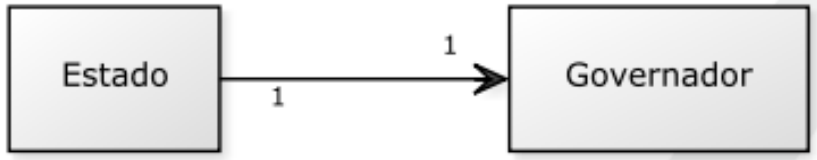
Suponha que em nosso domínio existam duas entidades: Estado e Governador. Devemos criar uma classe para cada entidade e aplicar nelas as anotações básicas de mapeamento.

```
1 @Entity
2 class Estado {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

Código Java 2.19: Estado.java

```
1 @Entity
2 class Governador {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

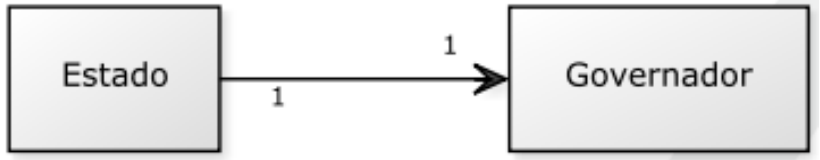
One to One



Como existe um relacionamento entre estados e governadores, devemos expressar esse vínculo através de um atributo que pode ser inserido na classe Estado.

```
1  @Entity
2  class Estado {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    private Governador governador;
8  }
```

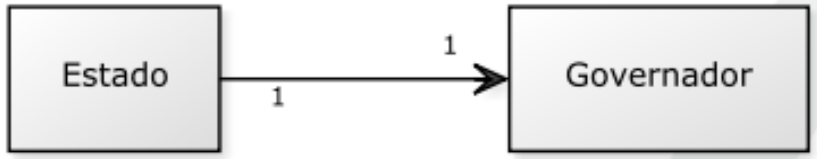
One to One



Como existe um relacionamento entre estados e governadores, devemos expressar esse vínculo através de um atributo que pode ser inserido na classe Estado.

```
1  @Entity
2  class Estado {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    private Governador governador;
8  }
```

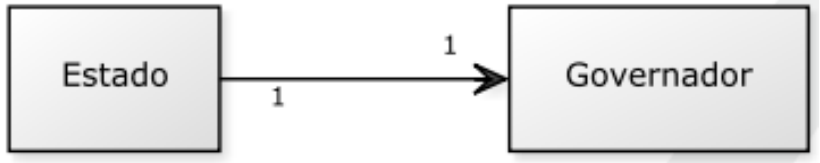
One to One



Além disso, devemos informar ao provedor JPA que o relacionamento que existe entre um estado e um governador é do tipo One to One. Fazemos isso aplicando a anotação **@OneToOne** no atributo que expressa o relacionamento.

```
1  @Entity
2  class Estado {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    @OneToOne
8    private Governador governador;
9  }
```

One to One



No banco de dados, a tabela referente à classe `Estado` possuirá uma coluna de relacionamento chamada de **join column**. Em geral, essa coluna será definida como uma chave estrangeira associada à tabela referente à classe `Governador`.

Por padrão, o nome da coluna de relacionamento é formado pelo nome do atributo que estabelece o relacionamento, seguido pelo caractere "_" e pelo nome do atributo que define a chave primária da entidade alvo. No exemplo de estados e governadores, a join column teria o nome **governador_id**.

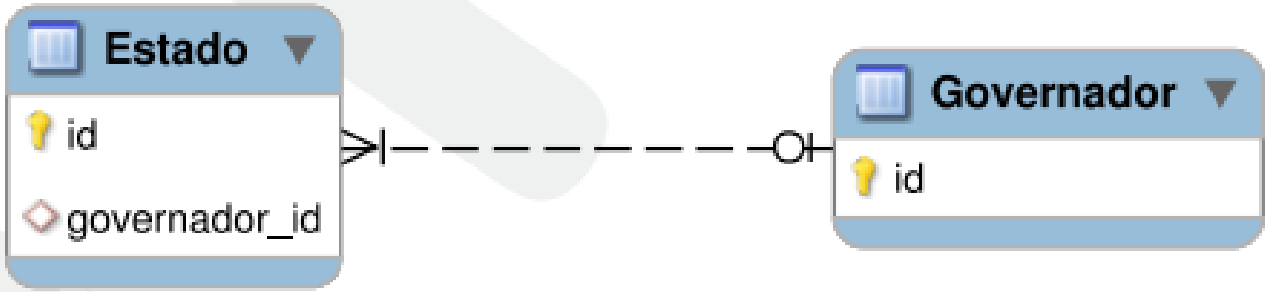
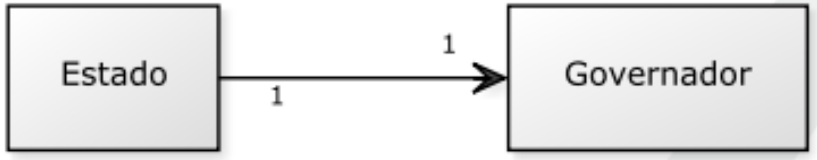


Figura 2.5: Tabelas correspondentes às classes `Estado` *e* `Governador`

One to One



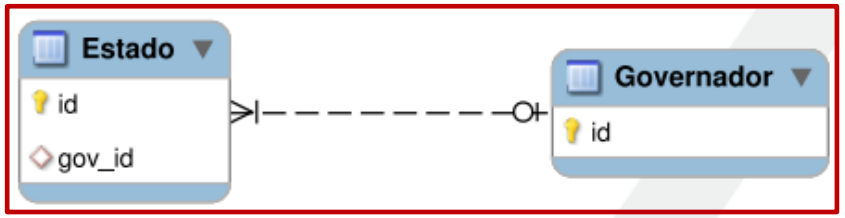
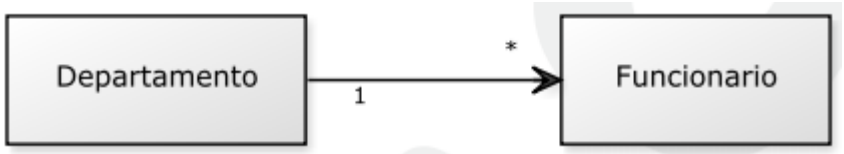


Figura 2.6: Personalizando o nome da coluna de relacionamento

Podemos alterar o nome padrão das join columns aplicando a anotação **@JoinColumn**, conforme apresentado no exemplo abaixo.

```
1  @Entity
2  class Estado {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    @OneToOne
8    @JoinColumn(name="gov_id")
9    private Governador governador;
```

One to Many

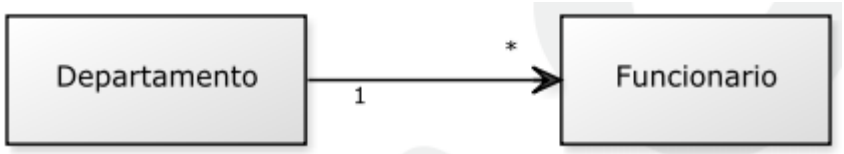


Suponha que em nosso domínio existam as entidades Departamento e Funcionário. Criaríamos duas classes com as anotações básicas de mapeamento.

```
1 @Entity
2 class Departamento {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

```
1 @Entity
2 class Funcionario {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

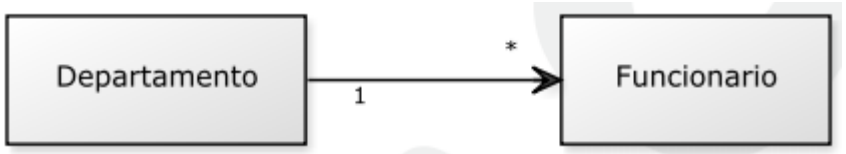
One to Many



Como existe um relacionamento entre departamentos e funcionários, devemos expressar esse vínculo através de um atributo que pode ser inserido na classe `Departamento`. Supondo que um departamento possa ter muitos funcionários, devemos utilizar uma coleção para expressar esse relacionamento.

```
1  @Entity
2  class Departamento {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    private Collection<Funcionario> funcionarios;
8  }
```

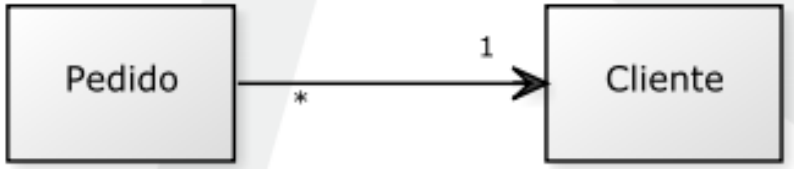
One to Many



Para informar a cardinalidade do relacionamento entre departamentos e funcionários, devemos utilizar a anotação **@OneToMany** na coleção.

```
1  @Entity
2  class Departamento {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    @OneToMany
8    private Collection<Funcionario> funcionarios;
9  }
```

Many to One



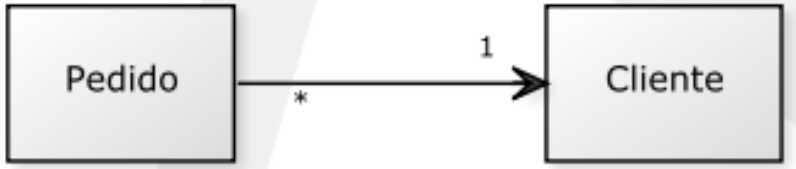
Suponha que em nosso domínio existam as entidades Pedido e Cliente. As duas classes que modelariam essas entidades seriam definidas com as anotações principais de mapeamento.

```
1 @Entity
2 class Pedido {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

Código Java 2.36: Pedido.java

```
1 @Entity
2 class Cliente {
3   @Id
4   @GeneratedValue
5   private Long id;
6 }
```

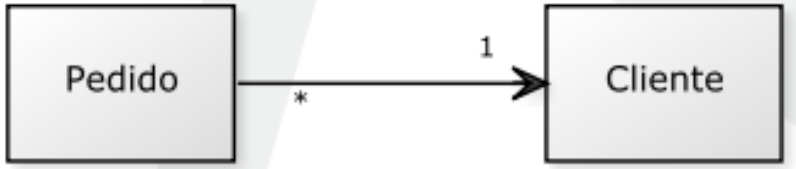
Many to One



Como existe um relacionamento entre pedidos e clientes, devemos expressar esse vínculo através de um atributo que pode ser inserido na classe Pedido. Supondo que um pedido pertença a um único cliente, devemos utilizar um atributo simples para expressar esse relacionamento.

```
1  @Entity
2  class Pedido {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    private Cliente cliente;
8  }
```

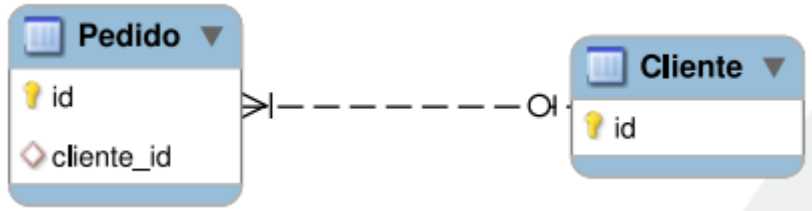
Many to One



Para informar a cardinalidade do relacionamento entre pedidos e clientes, devemos utilizar a anotação **@ManyToOne** no atributo.

```
1  @Entity
2  class Pedido {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    @ManyToOne
8    private Cliente cliente;
9  }
```

Many to One



No exemplo de pedidos e clientes, o nome da join column seria **cliente_id**. Podemos alterar o nome padrão das join columns aplicando a anotação **@JoinColumn**.

```
1  @Entity
2  class Pedido {
3    @Id
4    @GeneratedValue
5    private Long id;
6
7    @ManyToOne
8    @JoinColumn(name="cli_id")
9    private Cliente cliente;
10 }
```

Entidade Lançamento

Spring

```
public class Lancamento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long codigo;

    private String descricao;

    @Column(name = "data_vencimento")
    private LocalDate dataVencimento;

    @Column(name = "data_pagamento")
    private LocalDate dataPagamento;

    private BigDecimal valor;

    private String observacao;

    @Enumerated(EnumType.STRING)
    private TipoLancamento tipo;

    @ManyToOne
    @JoinColumn(name = "codigo_categoria")
    private Categoria categoria;

    @ManyToOne
    @JoinColumn(name = "codigo_pessoa")
    private Pessoa pessoa;
```

Migration

Spring

```
public class Lancamento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long codigo;

    private String descricao;

    @Column(name = "data_vencimento")
    private LocalDate dataVencimento;

    @Column(name = "data_pagamento")
    private LocalDate dataPagamento;

    private BigDecimal valor;

    private String observacao;

    @Enumerated(EnumType.STRING)
    private TipoLancamento tipo;

    @ManyToOne
    @JoinColumn(name = "codigo_categoria")
    private Categoria categoria;

    @ManyToOne
    @JoinColumn(name = "codigo_pessoa")
    private Pessoa pessoa;
```

- **Categoria**

```
CREATE TABLE categoria (
    codigo BIGINT(20) PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL
)
INSERT INTO categoria (nome) values ('Lazer');
INSERT INTO categoria (nome) values ('Alimentação');
INSERT INTO categoria (nome) values ('Supermercado');
INSERT INTO categoria (nome) values ('Farmácia');
INSERT INTO categoria (nome) values ('Outros');
```

- **Pessoa**

```
CREATE TABLE pessoa (
    codigo BIGINT(20) PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    logradouro VARCHAR(30),
    numero VARCHAR(30),
    complemento VARCHAR(30),
    bairro VARCHAR(30),
    cep VARCHAR(30),
    cidade VARCHAR(30),
    estado VARCHAR(30),
    ativo BOOLEAN NOT NULL )
```

Spring

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('João Silva', 'Rua do Abacaxi', '10', null, 'Brasil', '38.400-12', 'Uberlândia', 'MG', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Maria Rita', 'Rua do Sabiá', '110', 'Apto 101', 'Colina', '11.400-12', 'Ribeirão Preto', 'SP', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Pedro Santos', 'Rua da Bateria', '23', null, 'Morumbi', '54.212-12', 'Goiânia', 'GO', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Ricardo Pereira', 'Rua do Motorista', '123', 'Apto 302', 'Aparecida', '38.400-12', 'Salvador', 'BA', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Josué Mariano', 'Av Rio Branco', '321', null, 'Jardins', '56.400-12', 'Natal', 'RN', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Pedro Barbosa', 'Av Brasil', '100', null, 'Tubalina', '77.400-12', 'Porto Alegre', 'RS', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Henrique Medeiros', 'Rua do Sapo', '1120', 'Apto 201', 'Centro', '12.400-12', 'Rio de Janeiro', 'RJ', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Carlos Santana', 'Rua da Manga', '433', null, 'Centro', '31.400-12', 'Belo Horizonte', 'MG', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Leonardo Oliveira', 'Rua do Músico', '566', null, 'Segismundo Pereira', '38.400-00', 'Uberlândia', 'MG', true);

INSERT INTO pessoa (nome, logradouro, numero, complemento, bairro, cep, cidade, estado, ativo) values ('Isabela Martins', 'Rua da Terra', '1233', 'Apto 10', 'Vigilato', '99.400-12', 'Manaus', 'AM', true);

- **Lancamento**

```
CREATE TABLE lancamento (
    codigo BIGINT(20) PRIMARY KEY AUTO_INCREMENT,
    descricao VARCHAR(50) NOT NULL,
    data_vencimento DATE NOT NULL,
    data_pagamento DATE,
    valor DECIMAL(10,2) NOT NULL,
    observacao VARCHAR(100),
    tipo VARCHAR(20) NOT NULL,
    codigo_categoria BIGINT(20) NOT NULL,
    codigo_pessoa BIGINT(20) NOT NULL,
    FOREIGN KEY (codigo_categoria) REFERENCES categoria(codigo),
    FOREIGN KEY (codigo_pessoa) REFERENCES pessoa(codigo)
)
```

Spring

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Salário mensal', '2017-06-10', null, 6500.00, 'Distribuição de lucros', 'RECEITA', 1, 1);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Bahamas', '2017-02-10', '2017-02-10', 100.32, null, 'DESPESA', 2, 2);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Top Club', '2017-06-10', null, 120, null, 'RECEITA', 3, 3);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('CEMIG', '2017-02-10', '2017-02-10', 110.44, 'Geração', 'RECEITA', 3, 4);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('DMAE', '2017-06-10', null, 200.30, null, 'DESPESA', 3, 5);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Extra', '2017-03-10', '2017-03-10', 1010.32, null, 'RECEITA', 4, 6);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Bahamas', '2017-06-10', null, 500, null, 'RECEITA', 1, 7);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Top Club', '2017-03-10', '2017-03-10', 400.32, null, 'DESPESA', 4, 8);

Spring

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Despachante', '2017-06-10', null, 123.64, 'Multas', 'DESPESA', 3, 9);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Pneus', '2017-04-10', '2017-04-10', 665.33, null, 'RECEITA', 5, 10);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Café', '2017-06-10', null, 8.32, null, 'DESPESA', 1, 5);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Eletrônicos', '2017-04-10', '2017-04-10', 2100.32, null, 'DESPESA', 5, 4);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Instrumentos', '2017-06-10', null, 1040.32, null, 'DESPESA', 4, 3);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Café', '2017-04-10', '2017-04-10', 4.32, null, 'DESPESA', 4, 2);

INSERT INTO lancamento (descricao, data_vencimento, data_pagamento, valor, observacao, tipo, codigo_categoria, codigo_pessoa) values ('Lanche', '2017-06-10', null, 10.20, null, 'DESPESA', 4, 1);

Spring