# Projeto da Persistência de Dados

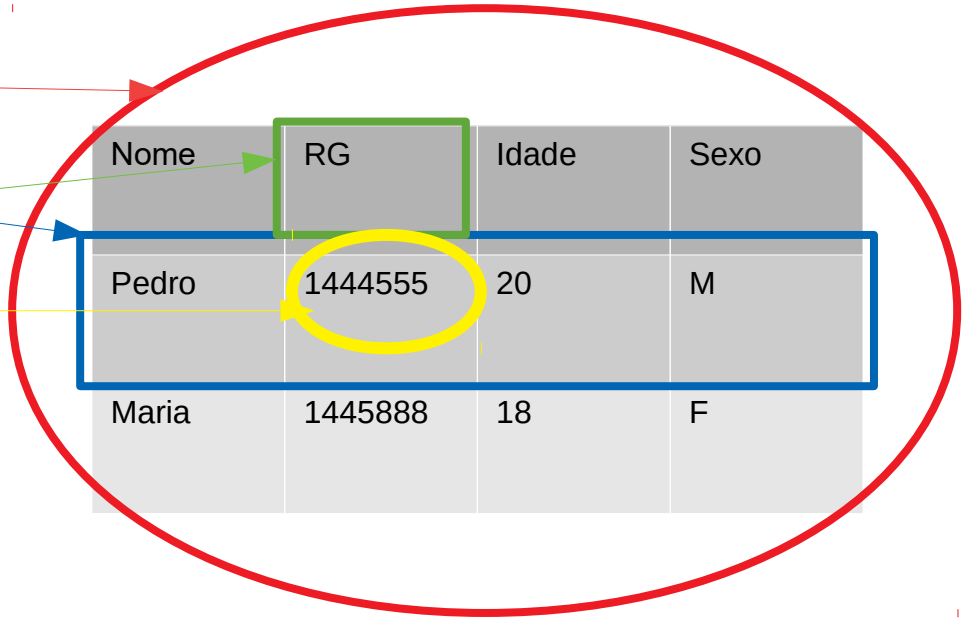
Instituto Federal do Espirito Santo – Campus Colatina
Disciplina: Projeto de Sistemas
Professor: Giovany Frossard Teixeira

# Introdução

- Tipos de armazenamento dos dados:
  - Arquivos
  - Bancos de Dados Relacionais. Exemplos: Oracle, MySQL, PostgreSQL, etc.
  - Bancos de Dados Orientados a Objetos. Exemplos: DB4Objects, Zope Object Data Base (ou ZODB), etc.
  - Bancos de Dados Não Relacionais (NoSQL). Exemplos: MongoDB, Cassandra, CouchDB, etc.

# O Modelo Relacional

- Em um modelo de dados relacional, os conjuntos de dados são representados por tabelas de valores. Cada tabela, denominada de relação, é bidimensional, sendo organizada em linhas e colunas. Esse modelo está fortemente baseado na teoria matemática sobre relações, daí o nome relacional. Os principais conceitos do modelo relacional são:
  - Tabela ou relação
  - Linha
  - Coluna
  - Célula
  - Chave Primária
  - Chave Estrangeira
  - Tabela Associativa



# O Modelo Relacional

- Chave Primária: coluna ou combinação de colunas que possui a propriedade identificar de forma única uma linha da tabela e que é utilizada para estabelecer associações entre entidades via transposição de chave.
- Chave Estrangeira ou Transposta: é a forma utilizada para associar linhas de tabelas distintas. A chave primária de uma tabela é transposta como uma coluna na outra tabela, onde é considerada uma chave estrangeira.

*Departamentos*

| Código | Nome |
|---|---|
| INF | Informática |
| LET | Letras |
| MAT | Matemática |

*Funcionários*

| Matrícula | Nome | Cod-Depto |
|---|---|---|
| 0158 | José | MAT |
| 5295 | Ricardo | INF |
| 7712 | Rosane | INF |

Chave Estrangeira

# O Modelo Relacional

- Tabela Associativa:  usadas para representar relacionamentos n-para-n entre tabelas.

*Pessoas*

| CPF | Nome |
|---|---|
| 96100199 | José |
| 83467187 | Maria |
| 02765140 | Luiza |

*Interesses*

| CPF-Pessoa | Código-Assunto |
|---|---|
| 96100199 | COMP |
| 96100199 | MUS |
| 02765140 | ENG |

*Assuntos*

| Código | Nome |
|---|---|
| ENG | Engenharia |
| COMP | Computação |
| MUS | Música |

# O Modelo Relacional - Propriedades

1) Cada tabela possui um nome único.
2) Nenhum campo parte de uma chave primária pode ser nulo.
3) Cada célula pode ser vazia ou conter no máximo um único valor.
4) Não há duas linhas iguais.
5) A ordem das linhas é irrelevante.
6) Cada coluna tem um nome único na mesma tabela.
7) Usando-se os nomes para referência às colunas, a ordem destas torna-se irrelevante.
8) Os valores de uma coluna são do mesmo tipo.
9) Colunas diferentes podem possuir o mesmo tipo.
10) Um campo que seja chave estrangeira só pode assumir valor nulo ou um valor para o qual exista um registro na tabela onde ele é chave primária.

---

# Mapeamento Objeto-relacional

- Diferenças:
  - objetos armazenam referências a outros objetos (p.ex., endereços de memória), enquanto bancos de dados relacionais ligam tabelas por meio de chaves transpostas.
  - objetos usam coleções para tratar relacionamentos e atributos multivalorados, enquanto células de uma tabela só podem ter no máximo um valor.
  - existe herança no modelo orientado a objetos e não há suporte a herança no modelo relacional.
  - tabelas têm de ter uma chave primária, enquanto objetos são únicos por essência (endereço de memória único), ficando transparente para o desenvolvedor a existência de identificadores.
- No mapeamento O/R, as seguintes questões devem ser abordadas: (i) mapeamento de classes e objetos; (ii) mapeamento de herança; e (iii) mapeamento de associações entre objetos.

# Mapeamento de classes e objetos

- Quando não há herança 1 classe é mapeada para 1 tabela. Ou seja, cada objeto de uma classe vira uma linha dessa tabela.
- Uma questão importante que surge é a chave primária. Uma solução possível é eleger um campo como chave (um campo que não se repita, que seja único, que possa atuar como chave primária, como por exemplo rg ou cpf).
- Uma outra possibilidade é a criação de uma chave primária artificial, ou seja, a criação de um identificador de objeto (id) e que não faça parte da lógica do negócio. Essa abordagem facilita a construção de componentes mais genéricos de persistência.

# Mapeamento de herança

**Existem 3 formas principais de Mapear Herança:**

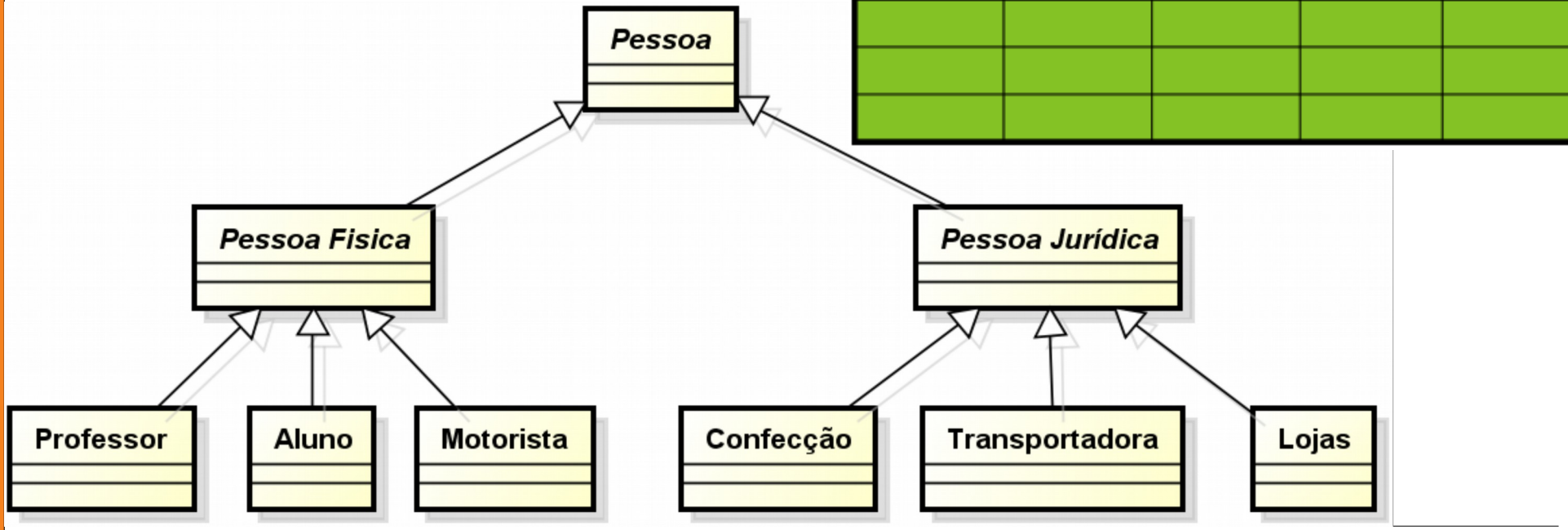
- Utilizar uma tabela para toda a hierarquia
- Utilizar uma tabela por classe concreta na hierarquia
- Utilizar uma tabela por classe na hierarquia

# Uma tabela para toda a hierarquia

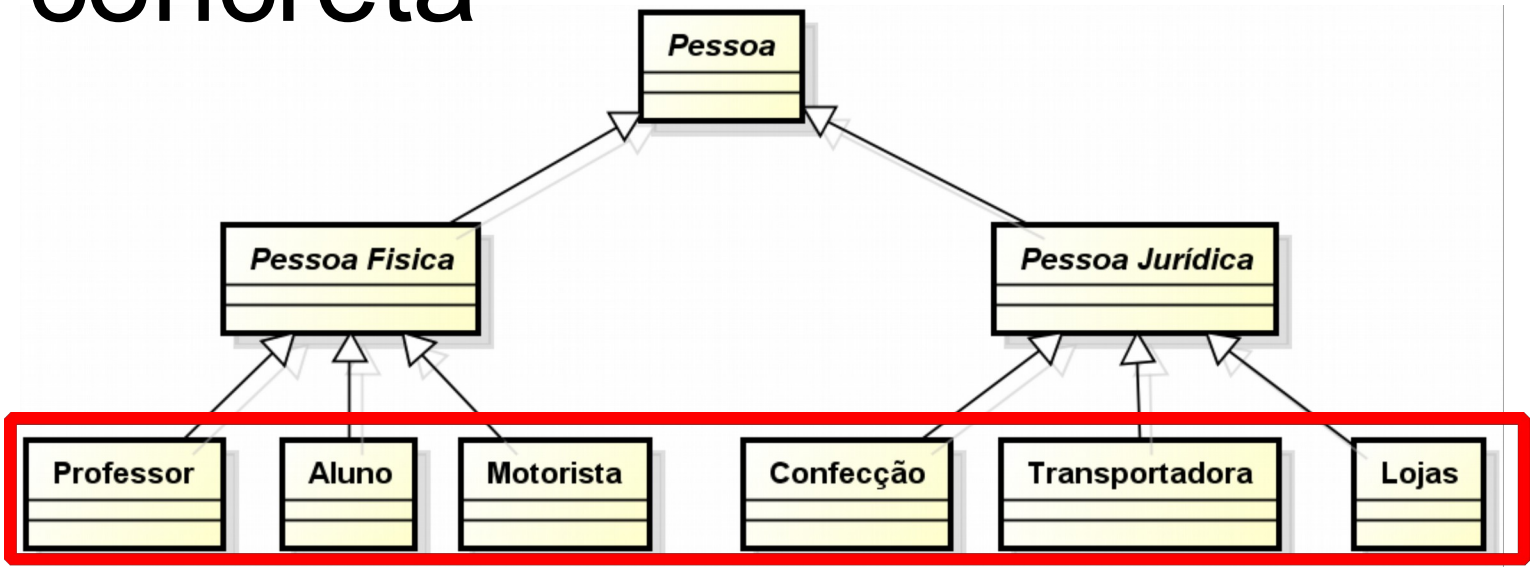
- Todos os campos de todas as classes da hierarquia são mapeados numa única tabela.

**Vantagem: Simplicidade, polimorfismo e Ids.**
**Desvantagem: Muitas colunas vazias.**

| Pessoas | | | | |
|---|---|---|---|---|
| id | Campo1 | Campo2 | ... | Campo N |
| | | | | |
| | | | | |
| | | | | |

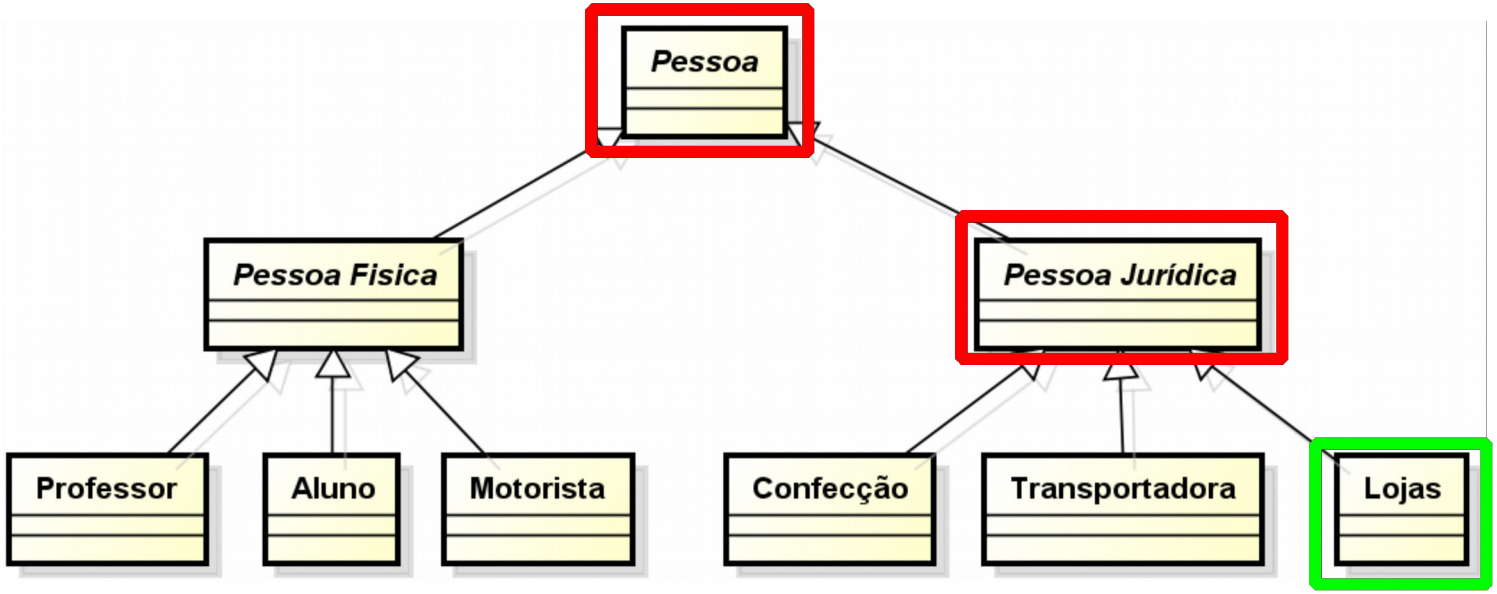


# Uma tabela para cada classe concreta




**Vantagem: Facilidade de processamento sobre as subclasses concretas, Ids.**
**Desvantagem: Alteração da superclasse, muito processamento na superclasse há queda de desempenho.**

# Uma tabela por classe da hierarquia




**Vantagem: Fácil modificar superclasse e acrescentar subclasses.**
**Desvantagem: Grande numero de tabelas, mais tempo para acessar dados de uma classe pois há a necessidade acessar várias tabelas.**

# Mapeando associações

- 1 : 1

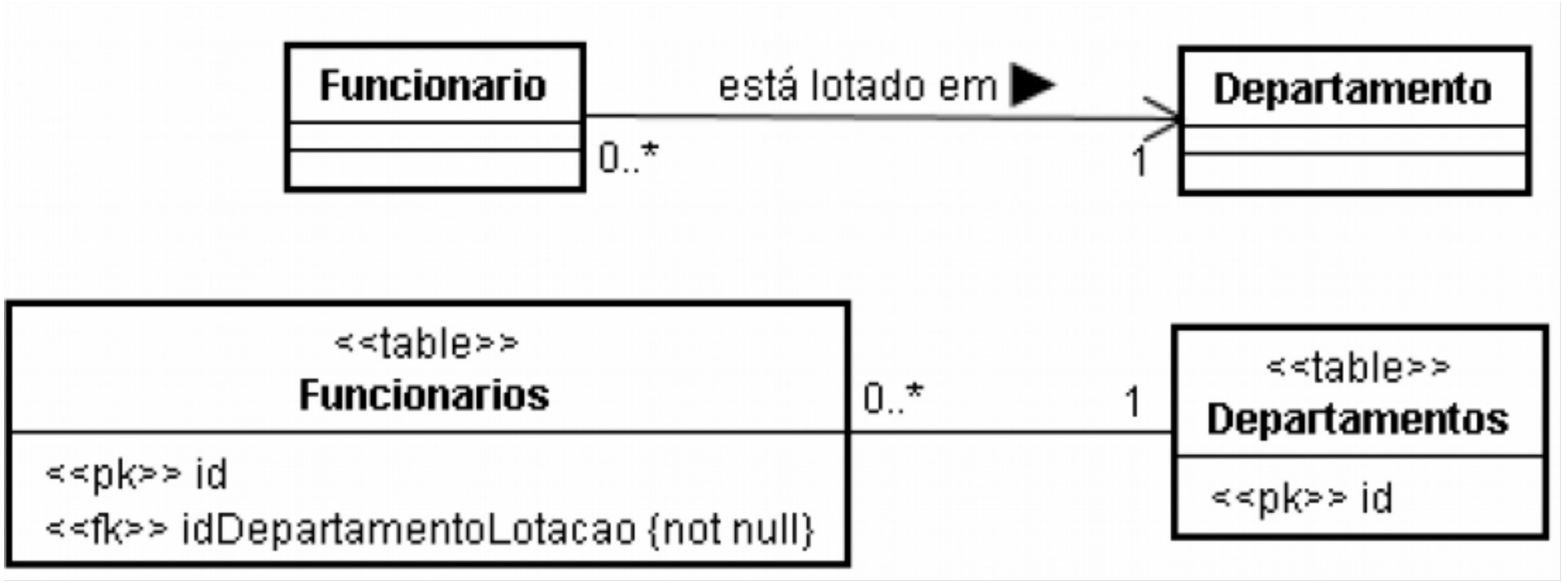
**É feito transpondo uma chave primária de uma tabela para outra.**

- Quando a associação for obrigatória nas duas extremidades (multiplicidade mínima 1 em ambas as extremidades), pode-se escolher qualquer das chaves para transpor.
- Quando a associação for opcional em pelo menos uma das duas extremidades (multiplicidade mínima 0), é melhor transpor a chave que dará origem a uma coluna mais densa, isto é, que terá menos valores nulos.
- Sempre que possível, deve-se transpor a chave que facilite a navegabilidade escolhida.

# Mapeando associações

- 1: N

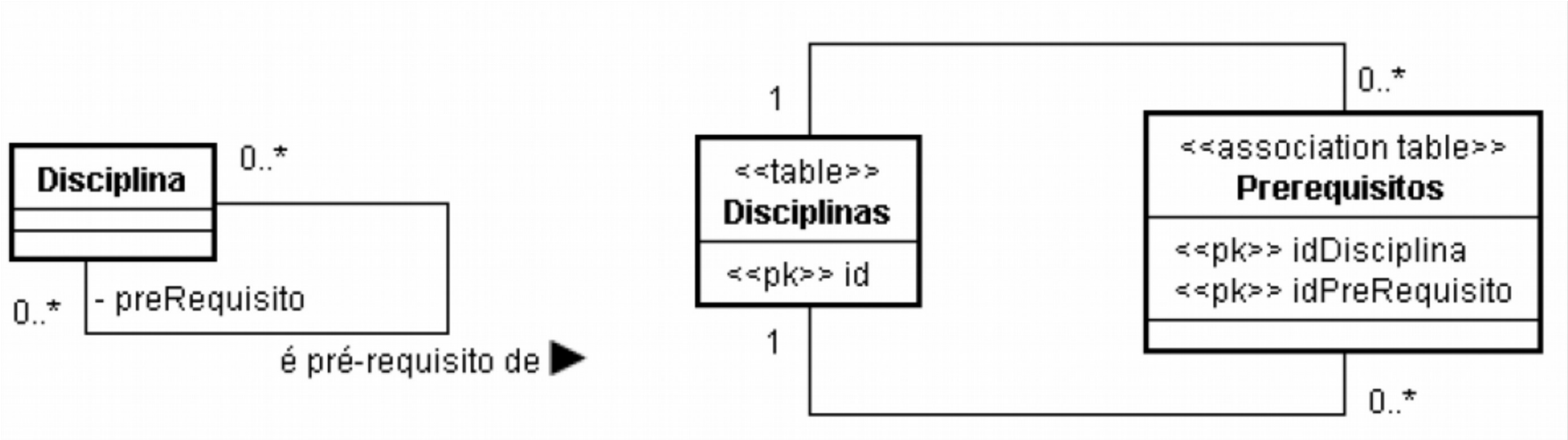
  O mapeamento de associações 1:N é feito transpondo-se a chave primária da tabela correspondente à classe cuja extremidade da associação tem multiplicidade máxima 1 para a tabela que corresponde à classe cuja extremidade da associação tem multiplicidade máxima n



# Mapeando associações

- N : N

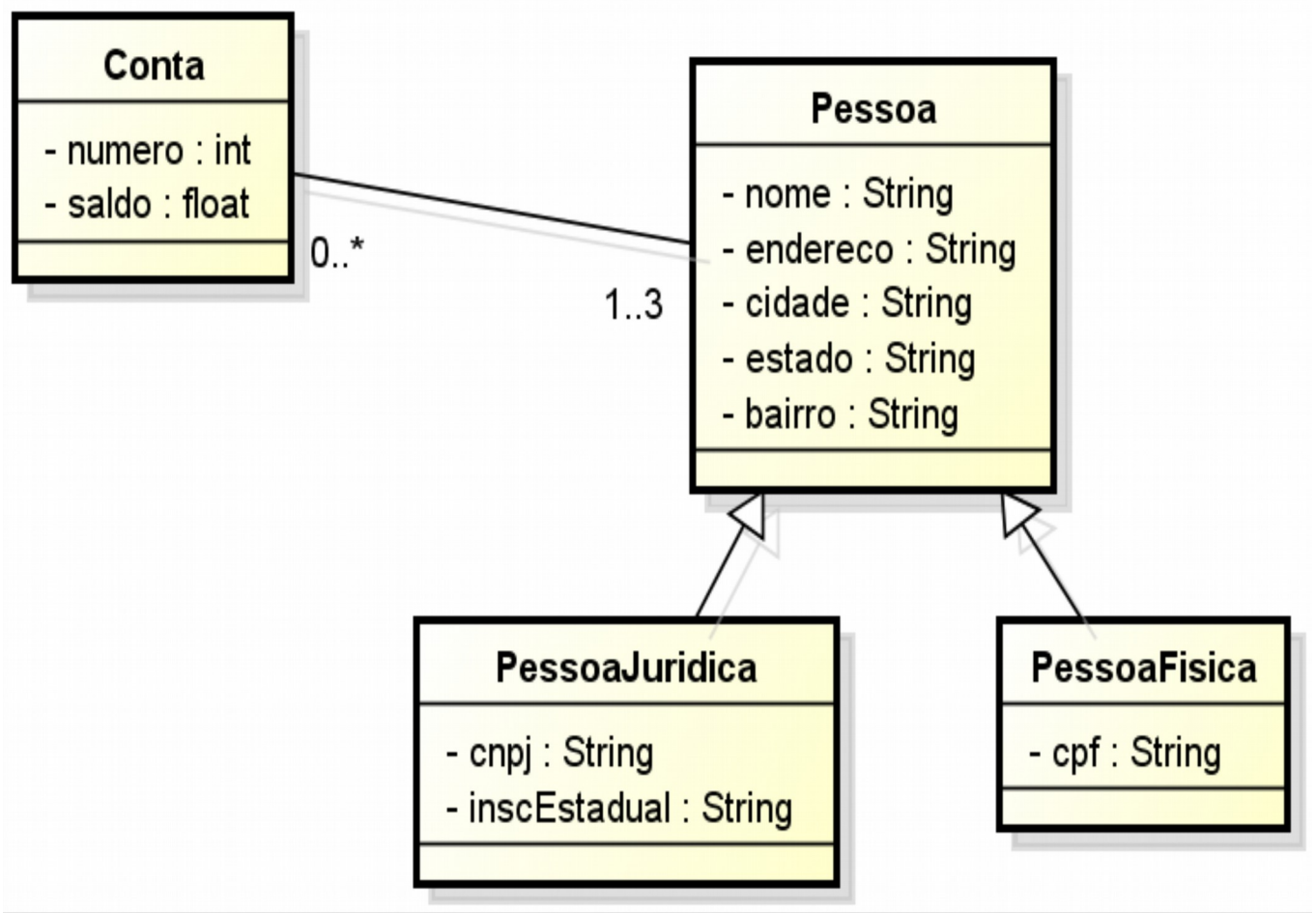
O mapeamento de associações N:N é feito utilizando-se uma tabela associativa, uma vez que bancos de dados relacionais não são capazes de manipular diretamente relacionamentos N:N.



# Exemplo de Mapeamento Objeto-relacional

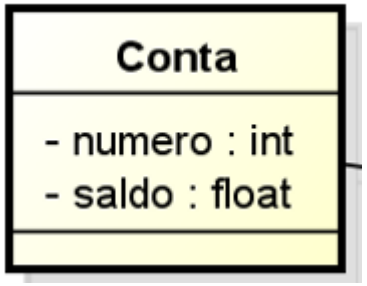
- Para fazer o mapeamento objeto-relacional e aplicar num banco precisaríamos de um SGBD configurado e uma ferramenta visual para comunicar com ele.
- Optaremos por uma opção mais simples: Utilizaremos o software BrModelo. Com esse software podemos desenhar o Modelo Lógico do banco sem nos preocuparmos diretamente com a tecnologia do banco em si (por exemplo: o tipo inteiro será INTEGER, NUMBER ou será mapeado em algum outro tipo? No BrModelo podemos convencionar o que quisermos).
- É importante notar que o projeto gerado no BrModelo não é executado diretamente num SGBD qualquer, ele é apenas um modelo.

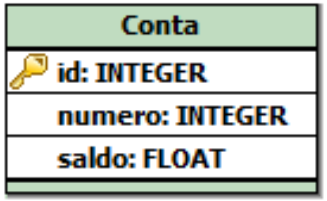
# Exemplo de Mapeamento Objeto-relacional



# Mapeando a classe Conta

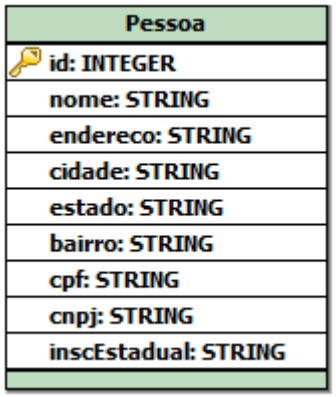
- A classe Conta não herda de ninguém, nesse caso uma tabela é o suficiente.
- Também optaremos pela solução de criação de uma chave artificial id. Nesse contexto tivemos o  mapeamento da classe Conta na tabela Conta:

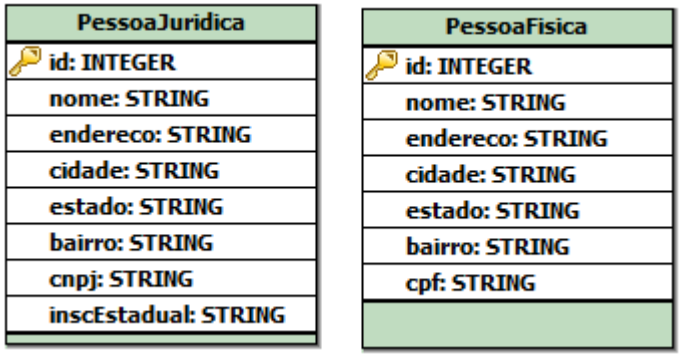


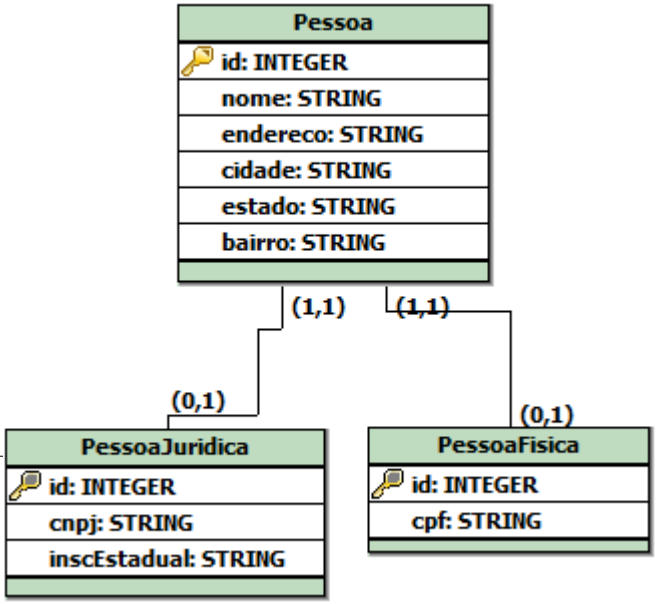
# Mapeamento da classe Pessoa

- Podemos mapear a herança de 3 formas:
  - Utilizar uma tabela para toda a hierarquia

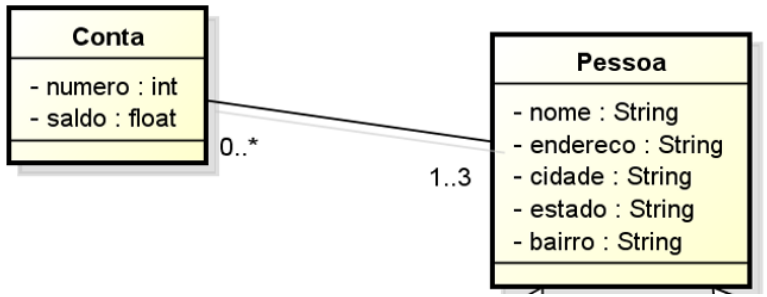


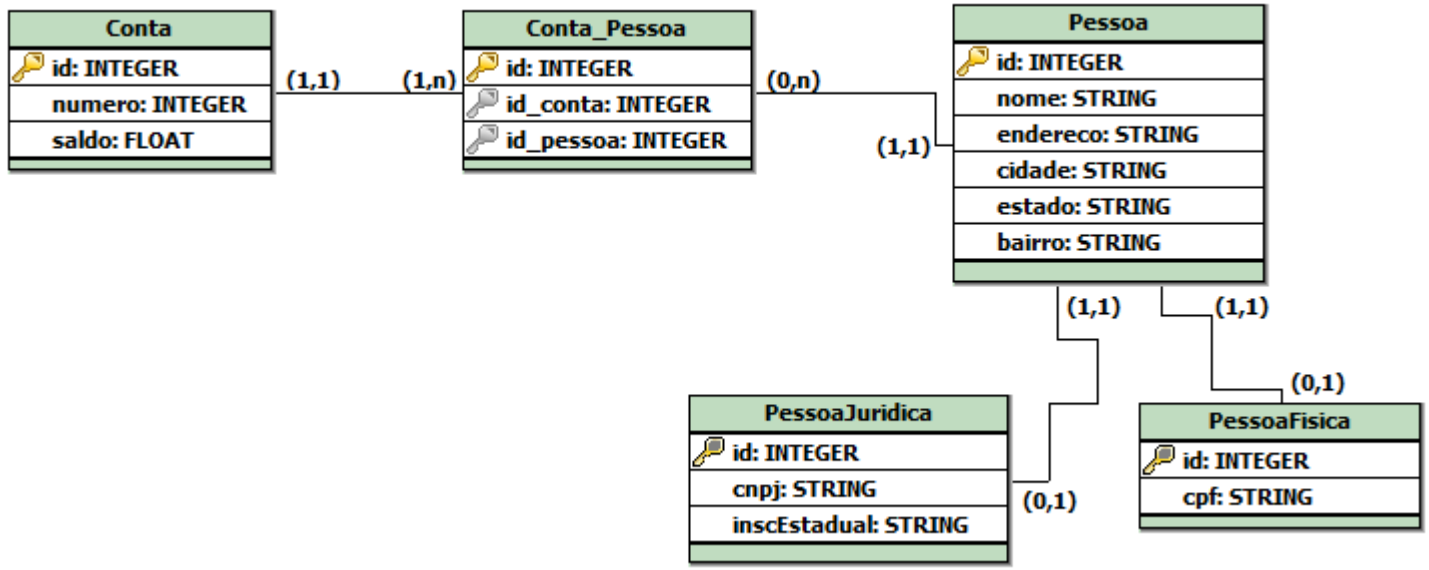

  - Utilizar uma tabela por classe concreta na hierarquia
  - Utilizar uma tabela por classe na hierarquia



# Mapeando a relação




- **É importante notar que se usássemos o modelo de uma tabela por classe concreta da hierarquia teríamos PessoaJuridica e PessoaFisica se relacionando diretamente com Conta_Pessoa.**
- **Note também que a chave Primária de Pessoa é a chave primária (e também estrangeira) de PessoaJuridica e PessoaFisica**



# Camada de Persistência

A Camada de Persistência (ou Componente de Gerência de Dados - CGD) provê a infraestrutura básica para o armazenamento e a recuperação de objetos no sistema.

Sua finalidade é isolar os impactos da tecnologia de gerenciamento de dados sobre a arquitetura do software (COAD; YOURDON, 1993).

# Camada de Persistência

- É possível implementar classes (manualmente) para fazer o trabalho de persistência ou usar frameworks para facilitar o trabalho de implementação.
- Frameworks como o Hibernate já dispõe de linguagem própria para escrita de consultas com desempenho próximo a implementação direta de classes para persistência.
- A escolha da abordagem a ser utilizada fica a cargo da equipe de desenvolvimento. Em geral é recomendado o uso de frameworks dado seu atual grau de maturidade e, em situações específicas, pode-se se utilizar a codificação mais manual por acessar de forma mais direta a tecnologia de persistência.

# Dúvidas?