PUC Minas

Projeto de Software

Prof. Dr. João Paulo Aramuni

Unidade 3

Persistência

PDS - Manhã /Noite

Persistência

Mapeamento de objetos para o modelo relacional

Relevância do mapeamento de objetos para o modelo relacional:

- A tecnologia OO como forma usual de desenvolver sistemas de software.
- Sem dúvida os SGBDR dominam o mercado comercial.

Os princípios básicos do paradigma da orientação a objetos e do modelo relacional são bastante diferentes.

No modelo de objetos, os elementos (objetos) correspondem a abstrações de comportamento.

No modelo relacional, os elementos correspondem a dados no formato tabular.

Persistência

Mapeamento para BD Relacional

- O modelo Relacional tem características peculiares:

  ✓ Exigência de chaves (atributos que identificam elementos nas tabelas).
  ✓ Chave-primária e Chave-estrangeira (para relacionamento).

- Exige uma notação específica.

Persistência

Mapeamento de objetos para o modelo relacional

- Objetos de um SSOO podem ser classificados em persistentes e transientes.

- Objetos transientes existem somente na memória principal, durante uma sessão de uso do SSOO.
  - Objetos de controle e objetos de fronteira são tipicamente objetos transientes.

- Objetos persistentes têm uma existência que perdura durante várias execuções do sistema.
  - Precisam ser armazenados quando a sessão de uso do sistema termina, e restaurados quando uma outra sessão é iniciada.
  - Tipicamente objetos de entidade.

Persistência

Incompatibilidade entre POO e Banco de Dados Relacional

Problema da Granularidade

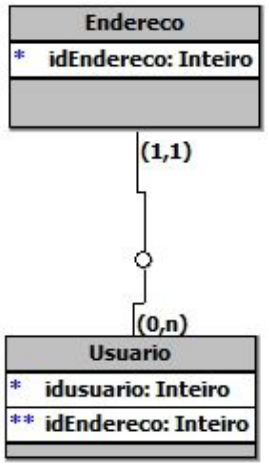
- Objetos podem ter vários níveis de granularidade.

- Na orientação a objetos pode-se pode definir uma classe Endereço e uma classe Usuário, sendo Endereço um objeto dentro da classe Usuário. Já no banco de dados o endereço pode vir como um campo ou atributo da tabela Usuário.

Persistência

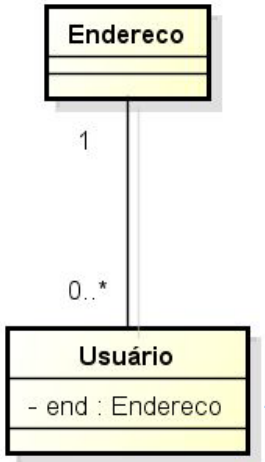
Incompatibilidade entre POO e Banco de Dados Relacional

Relacional



Coluna chave estrangeira

OO



Objeto

Persistência

Incompatibilidade entre POO e Banco de Dados Relacional

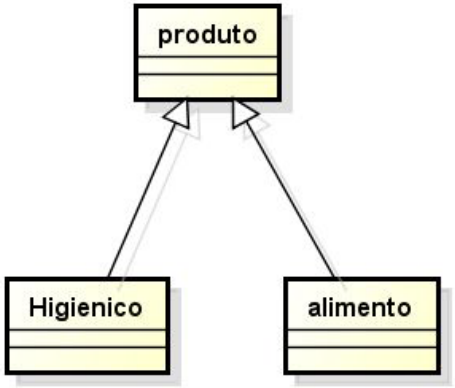
Problema dos Subtipos

- Herança e polimorfismo são características básicas e principais da programação orientada a objetos. Novas subclasses podem ser estendidas a partir de uma superclasse, os dados e métodos membros da superclasse podem ser visíveis nas subclasses, métodos podem ser herdados ou sobrecarregados e novos métodos e atributos podem ser adicionados nas subclasses.

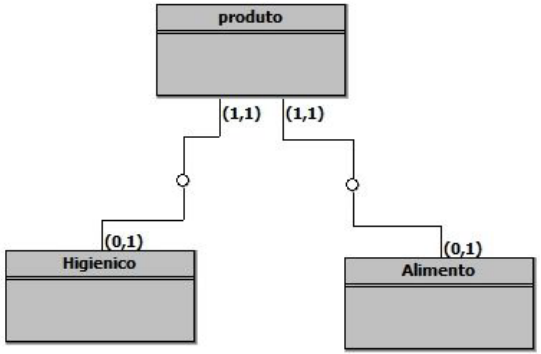
- Já no banco de dados relacional, as tabelas em geral utilizam linhas e colunas e não suportam herança de tipo.

Persistência

Incompatibilidade entre POO e Banco de Dados Relacional



OO



Relacional: uma possibilidade

Persistência

Incompatibilidade entre POO e Banco de Dados Relacional

Problema Relacionado à Associações

- Na POO as associações são representadas como referências ou coleções de referências para objetos.
- Para referências bidirecionais é necessário criar referências nos dois lados da associação, as referências nos dois lados podem ser N para M.
- No banco de dados relacional as associações são representadas por chaves estrangeiras, não são necessariamente direcionais, podem ser criadas associações arbitrárias como junções e projeções.
- Para se conseguir uma relação N para M é necessário ter uma tabela adicional.

Persistência

Incompatibilidade entre POO e Banco de Dados Relacional

Problema da Navegação de Dados

- O acesso e navegação entre os dados de um objeto para outro é diferente entre a POO e no banco de dados relacional.

- Para acessar e navegar entre os dados na POO é necessário utilizar métodos recuperadores, como por exemplo, GetItem(), getProduct(), etc.

- Para acessar dados no banco de dados relacional a partir de uma tabela para outra é necessário utilizar junções SQL entre as tabelas.

Persistência

Mapeamento de objetos para o modelo relacional

- Utilização de um SGBDR: necessidade do mapeamento dos valores de atributos de objetos persistentes para tabelas.

- É a partir do modelo de classes que o mapeamento de objetos para o modelo relacional é realizado.
  - Semelhante ao de mapeamento do MER.
  - Diferenças em virtude de o modelo de classes possuir mais recursos de representação que o MER.

- Importante: o MER e o modelo de classes não são equivalentes.
  - Esses modelos são frequentemente confundidos.
  - O MER é um modelo de dados; o modelo de classes representa objetos (dados e comportamento).

Persistência

Mapeamento de objetos para o modelo relacional

- Aqui, utilizamos a seguinte notação (simplificada):
  - Cada relação é representada através do seu nome e dos nomes de suas colunas entre parênteses.
  - Chaves primárias são sublinhadas
  - Chaves estrangeiras são tracejadas.

- Os exemplos dados a seguir utilizam sempre uma coluna de implementação como chave primária de cada relação.
  - Uma coluna de implementação é um identificador sem significado no domínio de negócio.
  - Essa abordagem é utilizada:
    - ✓ para manter uma padronização nos exemplos
    - ✓ e por ser uma das melhores maneiras de associar identificadores a objetos mapeados para tabelas.

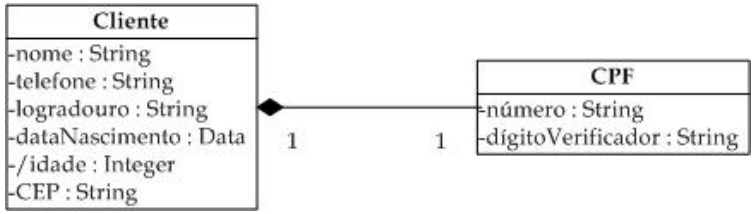
Persistência

Mapeamento: Classes e seus atributos

- Classes são mapeadas para relações.
  - Caso mais simples: mapear cada classe como uma relação, e cada atributo como uma coluna.
  - No entanto, pode não haver correspondência unívoca entre classes e relações.
- Para atributos o que vale de forma geral é que um atributo será mapeado para uma ou mais colunas.
- Cada instância de uma classe equivale a uma linha na tabela respectiva.
- Identificadores de objetos são representados como colunas indexadas que não admitem repetição de elementos, sendo, portanto, marcadas com a expressão unique.

Persistência

Mapeamento de classes e seus atributos




```
Cliente(id,   CPF,   nome,   telefone,   logradouro,
   dataNascimento,idCEP)
CEP(id, número, sufixo)
```

```
Cliente(id, nome, telefone, logradouro, dataNascimento,
   CPF, CEP)
```

Persistência

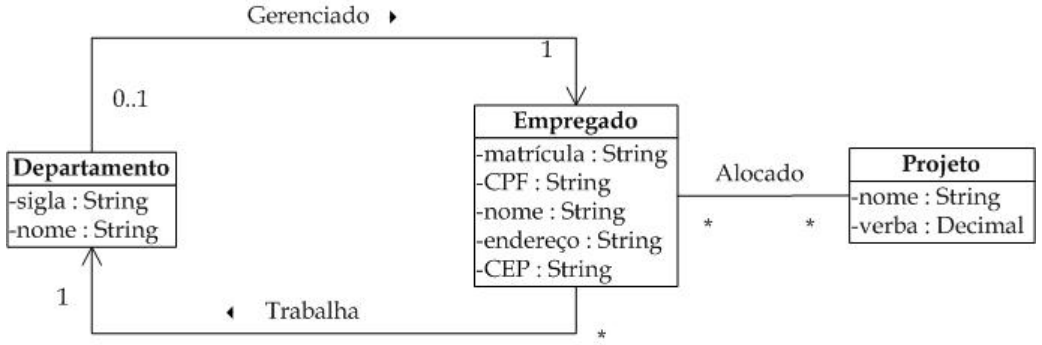
Mapeamento de associações

- O procedimento utiliza o conceito de chave estrangeira.
- Há três casos, cada um correspondente a um tipo de conectividade.
- Nos exemplos dados a seguir, considere, sem perda de generalidade, que:
  - há uma associação entre objetos de duas classes, Ca e Cb.
  - Ca e Cb foram mapeadas para duas relações separadas, Ta e Tb.

Persistência

Mapeamento de associações

- Considere também o seguinte diagrama de classes:



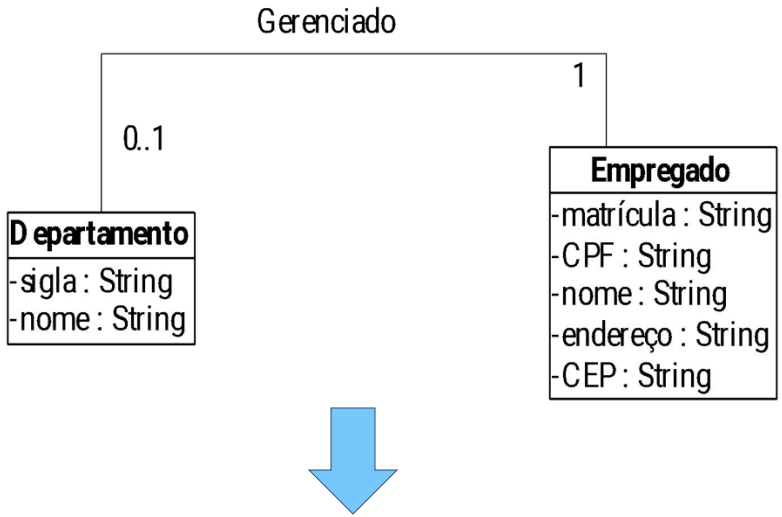
Persistência

Mapeamento de associações: um para um

- Deve-se adicionar uma chave estrangeira em uma das duas relações para referenciar a chave primária da outra relação.
- Escolha da relação na qual a chave estrangeira deve ser adicionada com base na participação.
- Há três possibilidades acerca da conectividade:
  - Obrigatória em ambos os extremos.
  - Opcional em ambos os extremos.
  - Obrigatória em um extremo e opcional no outro extremo.

Persistência

Mapeamento de associações: um para um




```
Departamento(id, sigla, nome, idEmpregadoGerente )
Empregado( id, matrícula, CPF, nome, endereço, CEP )
```

Persistência

Mapeamento de associações: um para muitos

- Seja Ca a classe na qual cada objeto se associa com muitos objetos da classe Cb.
- Sejam Ta eTb as relações resultantes do mapeamento de Ca e Cb, respectivamente.
- Neste caso, deve-se adicionar uma chave estrangeira em Ta para referenciar a chave primária de Tb.

Persistência

Mapeamento de associações: um para muitos


```
Departamento( id, sigla, nome, idEmpregadoGerente )
Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)
```

Persistência

Mapeamento de associações: muitos-muitos

- Seja Ca a classe na qual cada objeto se associa com muitos objetos da classe Cb.
- Sejam Ta eTb as relações resultantes do mapeamento de Ca e Cb, respectivamente.
- Uma relação de associação deve ser criada.
  - Uma relação de associação serve para representar a associação muitos para muitos entre duas ou mais relações.

Persistência

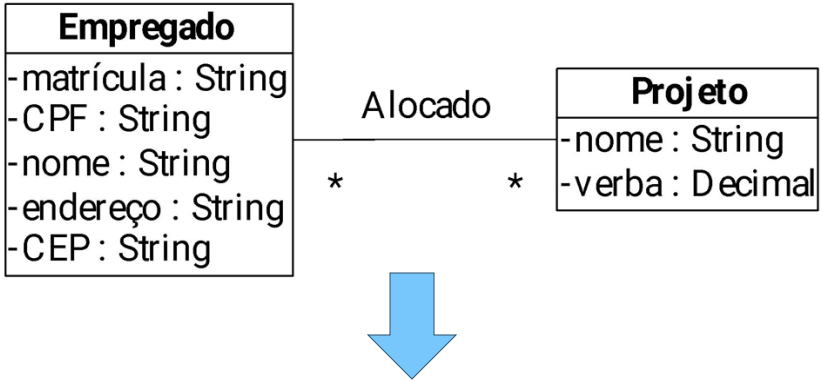
Mapeamento de associações: muitos-muitos

- Alternativas para definir a chave primária na relação de associação
  - definir uma chave primária composta.
  - criar uma coluna de implementação que sirva como chave primária simples da relação de associação.
  - As chaves primárias das tabelas Ta e Tb serão chaves estrangeiras

- A relação de associação pode ter então três tipos de colunas:
  - A sua própria chave primária simples.
  - Duas colunas com valores tomados das chaves primárias das tabelas associadas, correspondendo a uma chave candidata.
  - Os atributos da classe de associação.

Persistência

Mapeamento de associações: muitos-muitos



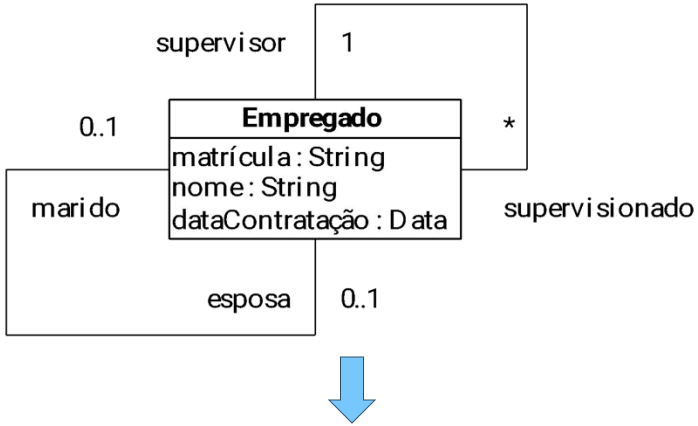

```
Departamento(id, sigla, nome, idEmpregadoGerente)
Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)
Alocação(idProjeto, idEmpregado, nome, verba)
Projeto(id, nome, verba)
```

```
Departamento(id, sigla, nome, idEmpregadoGerente)
Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)
Alocação(id, idProjeto, idEmpregado, nome, verba)
Projeto(id, nome, verba)
```

Persistência

Mapeamento de associações: reflexivas

- Forma especial de associação
  - mesmo procedimento para realizar o mapeamento de associações pode ser utilizado.
- Em particular, em uma associação reflexiva de conectividade muitos para muitos, uma relação de associação deve ser criada.



Empregado(id, matrícula, nome, dataContratação, idCônjunge, idSupervisor)

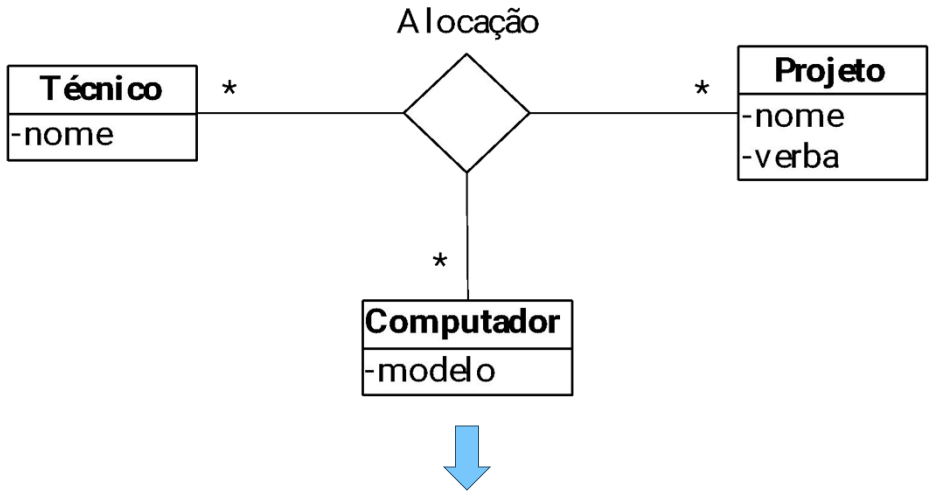
Persistência

Mapeamento de associações: n-árias

- Associações n-árias (n≥3): procedimento semelhante ao utilizado para associações binárias de conectividade muitos para muitos.
  - Uma relação para representar a associação é criada.
  - São adicionadas nesta relação chaves estrangeiras.
  - Se a associação n-ária possuir uma classe associativa, os atributos desta são mapeados como colunas da relação de associação.

Persistência

Mapeamento de associações: n-árias




```
Técnico( id, nome )
Projeto( id, nome, verba )
Computador( id,  modelo )
Alocação( id, idProjeto, idTécnico, idComputador )
```

Persistência

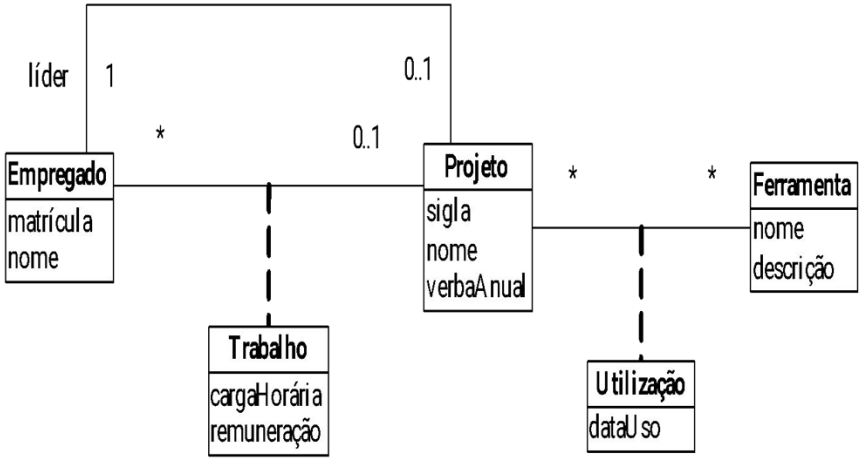
Mapeamento de classes associativas

- Para cada um dos casos de mapeamento de associações, há uma variante onde uma classe associativa é utilizada.

- Mapeamento é feito através da criação de uma relação para representá-la.
  - Os atributos da classe associativa são mapeados para colunas dessa relação.
  - Essa relação deve conter chaves estrangeiras que referenciem as relações correspondentes às classes que participam da associação.

Persistência

Mapeamento de classes associativas




```
Empregado(id, matrícula, nome)
Projeto(id, sigla, nome, verbaAnual, idEmpregadoLíder)
Ferramenta(id, nome, descrição)
Utilização(id, idFerramenta, idProjeto, dataUso )
Trabalho(id, idEmpregado, idProjeto, cargaHorária, remuneração)
```

Persistência

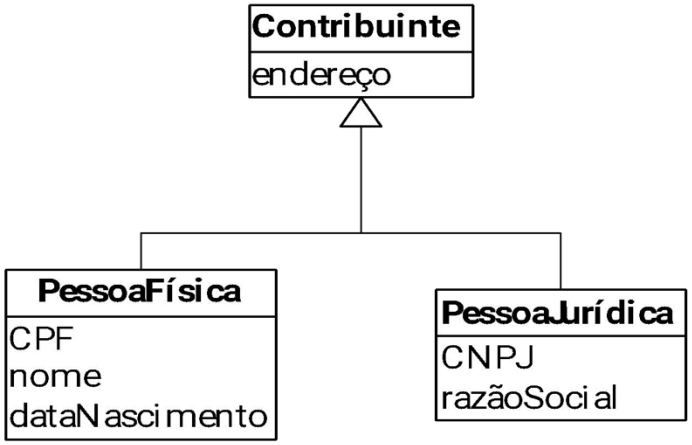
Mapeamento de generalizações

- Três formas alternativas de mapeamento:
  - Uma relação para cada classe da hierarquia
  - Uma relação para toda a hierarquia
  - Uma relação para cada classe concreta da hierarquia

- **Nenhuma** das alternativas de mapeamento de generalização pode ser considerada a melhor dentre todas.
  - Cada uma delas possui vantagens e desvantagens.
  - Escolha de uma delas depende das do sistema sendo desenvolvido.
  - A equipe de desenvolvimento pode decidir implementar mais de uma alternativa.

Persistência

Mapeamento de generalizações




```
Contribuinte(id, endereço)
PessoaFísica(id, nome, dataNascimento, CPF, idContribuinte)
PessoaJurídica(id, CNPJ, razãoSocial, idContribuinte)
```

Persistência

Mapeamento de generalizações

- A 1ª alternativa (uma relação para cada classe da hierarquia) é a que melhor reflete o modelo OO.
  - classe é mapeada para uma relação
  - as colunas desta relação são correspondentes aos atributos específicos da classe.
  - Desvantagem: desempenho da manipulação das relações.
    - ✓ Inserções e remoções e junções.

Persistência

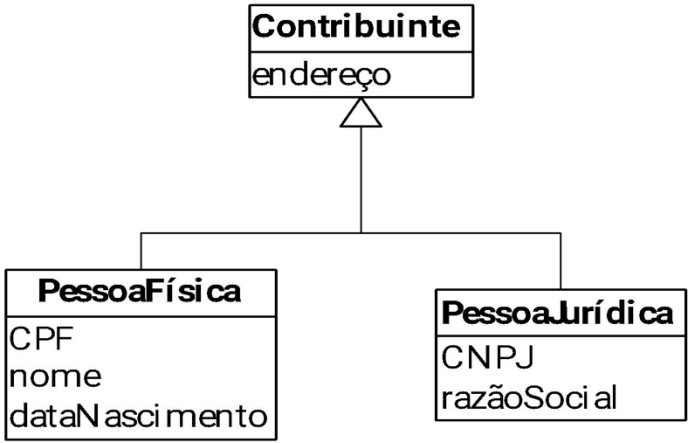
Mapeamento de generalizações

- A 2ª alternativa (Uma relação para toda a hierarquia) de implementação é bastante simples, além de facilitar situações em que objetos mudam de classe.

- Desvantagem: alteração de esquema
  - ✓ Adição ou remoção de atributos.
  - ✓ tem o potencial de desperdiçar bastante espaço de armazenamento:
    - hierarquia com várias classes “irmãs”
    - objetos pertencem a uma, e somente uma, classe da hierarquia.

Persistência

Mapeamento de generalizações




```
Pessoa(id, nome, endereço, dataNascimento, CPF, CNPJ, razãoSocial, tipo)
```

Persistência

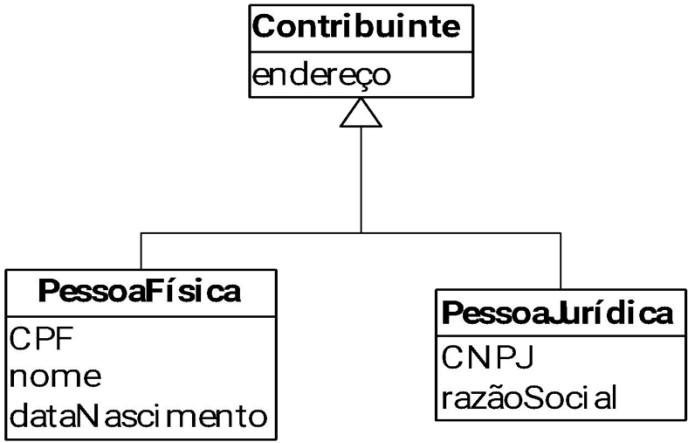
Mapeamento de generalizações

- A 3ª alternativa (Uma relação para cada classe concreta da hierarquia) apresenta a vantagem de agrupar os objetos de uma classe em uma única relação.

- Desvantagem: quando uma classe é modificada, cada uma das relações correspondentes as suas subclasses deve ser modificada.
    - Todas as relações correspondentes a subclasses devem ser modificadas quando a definição da superclasse é modificada.

Persistência

Mapeamento de generalizações




```
PessoaFísica(id, dataNascimento, nome, endereço, CPF)
PessoaJurídica(id, CNPJ, endereço, razãoSocial)
```

Projeto de Software

Referências básicas:

- **ACM TRANSACTIONS ON SOFTWARE ENGINEERING AND METHODOLOGY**. New York, N.Y., USA: Association for Computing Machinery, 1992-. Trimestral. ISSN 1049-331X. Disponível em: https://dl.acm.org/toc/tosem/1992/1/2. Acesso em: 19 jul. 2024. (Periódico On-line).

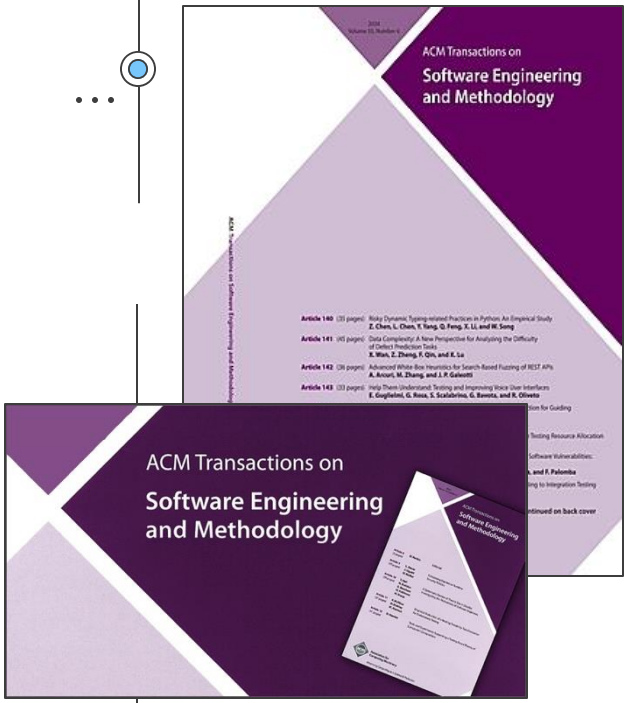
- LARMAN, Craig. **Utilizando UML e padrões**: uma introdução á análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007. E-book. ISBN 9788577800476. (Livro Eletrônico).

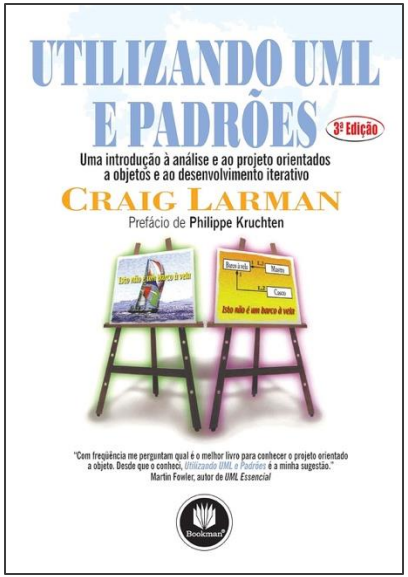
- SILVEIRA, Paulo et al. **Introdução à arquitetura e design de software**: uma visão sobre a plataforma Java. Rio de Janeiro, RJ: Elsevier, Campus, 2012. xvi, 257 p. ISBN 9788535250299. (Disponível no Acervo).

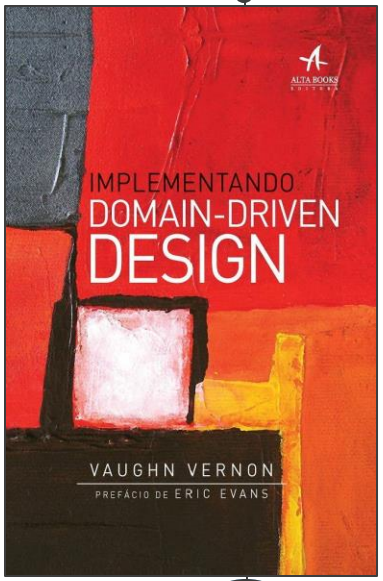
- VERNON, Vaughn. **Implementando o Domain-Driven Design**. Rio de Janeiro, RJ: Alta Books, 2016. 628 p. ISBN 9788576089520. (Disponível no Acervo).

Projeto de Software

Referências básicas:







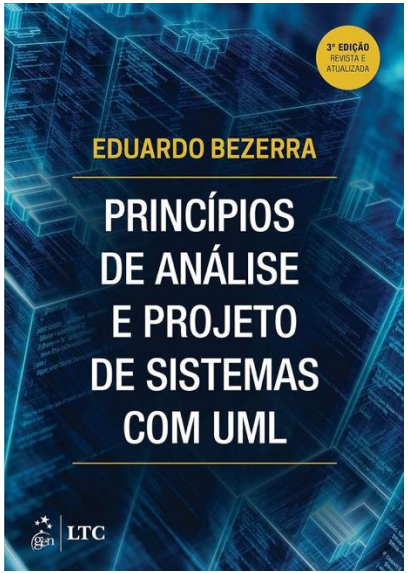
Projeto de Software

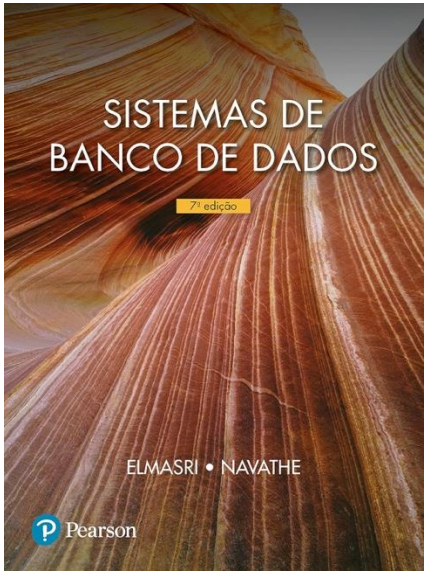
Referências complementares:

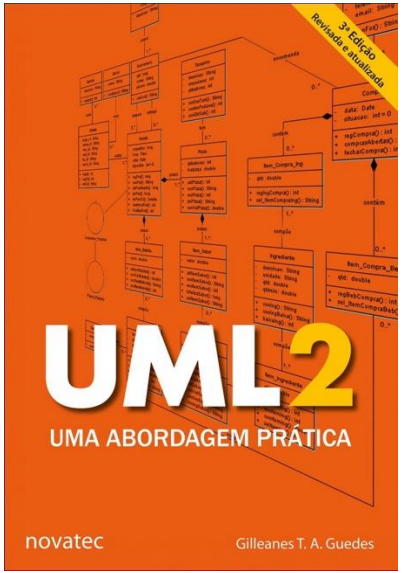
- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 3. ed. rev. e atual. Rio de Janeiro: Elsevier, 2015. xvii, 398 p. ISBN 9788535226263. (Disponível no Acervo).
- ELMASRI, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados**, 7ª ed. Editora Pearson 1152 ISBN 9788543025001. (Livro Eletrônico).
- GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, c2011. 484 p. ISBN 9788575222812. (Disponível no Acervo).
- **IEEE TRANSACTIONS ON SOFTWARE ENGINEERING**. New York: IEEE Computer Society,1975-. Mensal,. ISSN 0098-5589. Disponível em: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32. Acesso em: 19 jul. 2024. (Periódico On-line).
- SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson Education do Brasil, c2019. xii, 756 p. ISBN 9788543024974. (Disponível no Acervo).
- WAZLAWICK, Raul Sidnei. **Análise e design orientados a objetos para sistemas de informação**: modelagem com UML, OCL e IFML. 3. ed. Rio de Janeiro, RJ: Elsevier, Campus, c2015. 462 p. ISBN 9788535279849. (Disponível no Acervo).

Projeto de Software

Referências complementares:

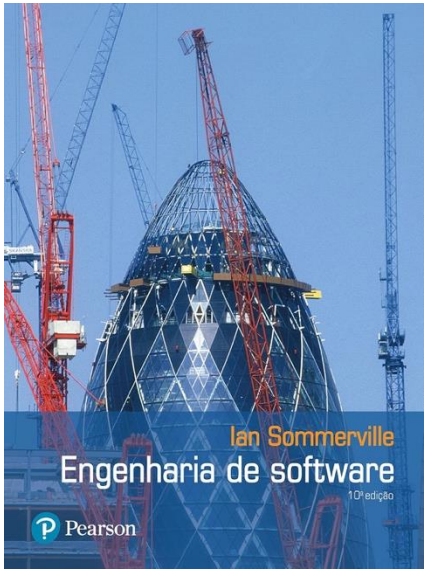






Projeto de Software

Referências complementares:



Obrigado!

Dúvidas?

joaopauloaramuni@gmail.com



GitHub LinkedIn Lattes