PROJETO LÓGICO DE BANCO DE DADOS

Prof. Ronaldo R. Goldschmidt

Baseado no livro "Princípios de Análise e Projeto Orientados a Objetos com UML

Prof. Eduardo Bezerra - Editora CAMPUS

Introdução

- Escopo: apenas os aspectos de mapeamento de informações entre as tecnologias de OO e relacional.
  - mapeamento do modelo de classes para o modelo relacional.
- Nota: ferramentas CASE e mapeamento automático; engenharia reversa.
- Nem sempre uma ferramenta CASE está disponível.
- Mesmo na existência de uma ferramenta, é importante um conhecimento básico dos procedimentos do mapeamento.

Introdução

- Os objetos de um sistema podem ser classificados em persistentes e transientes.
- ***Objetos transientes****:* existem somente na memória principal.
  - Objetos de controle e objetos de fronteira.
- ***Objetos persistentes***: têm uma existência que perdura durante várias execuções do sistema.
  - Precisam ser armazenados quando uma execução termina, e restaurados quando uma outra execução é iniciada.
  - Tipicamente objetos de entidade (negócios).

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.
- Os exemplos 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.

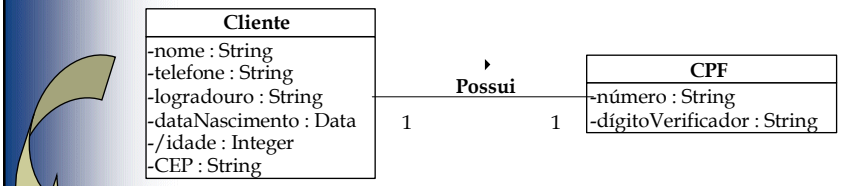
Mapeamento de objetos para o modelo relacional

- Exemplos 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.
  - Semelhante ao de mapeamento do MER (Modelo Entidade-Relacionamento), também chamado DER (Diagrama Entidade Relacionamento).
  - Diferenças em virtude de o modelo de classes possuir mais recursos de representação que o MER.

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.
- Nem todos os atributos de uma classe são persistentes (atributos derivados).

Mapeamento: Classes e seus atributos



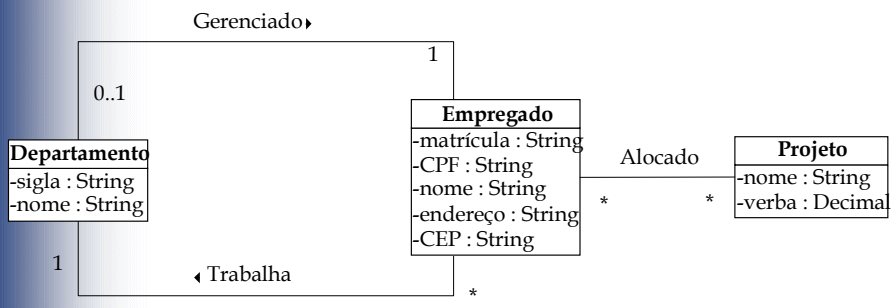

```
Cliente(id, nome, telefone, logradouro, dataNascimento,CEP, idCPF)
CPF(id, número, sufixo)
```

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

Mapeamento: Associações

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

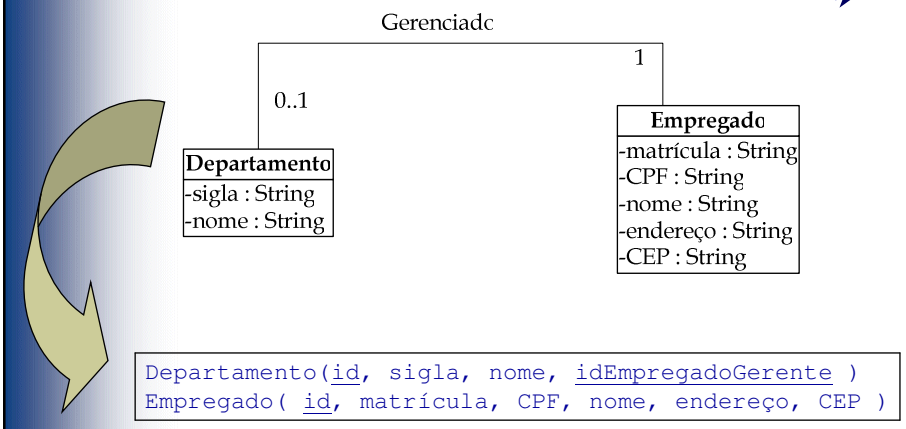
Mapeamento: Associações



Mapeamento: Associações 1:1

- 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:
  - Obrigatória em ambos os extremos.
  - Opcional em ambos os extremos.
  - Obrigatória em um extremo e opcional no outro extremo.

Mapeamento: Associações 1-1

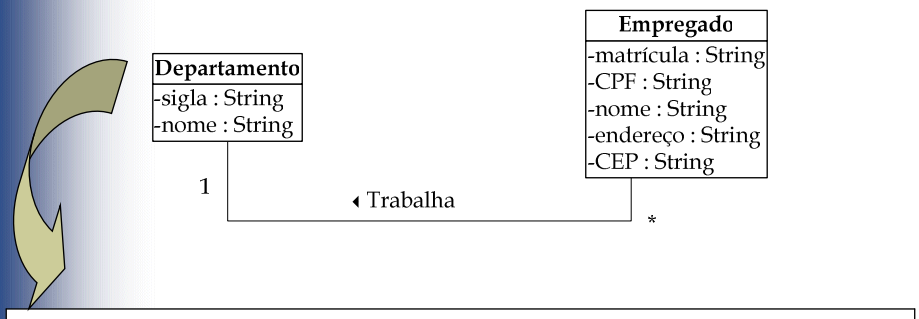



```
Departamento(id, sigla, nome, idEmpregadoGerente )
Empregado( id, matrícula, CPF, nome, endereço, CEP )
```

Mapeamento: Associações 1-muitos

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

Mapeamento: Associações 1-muitos




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

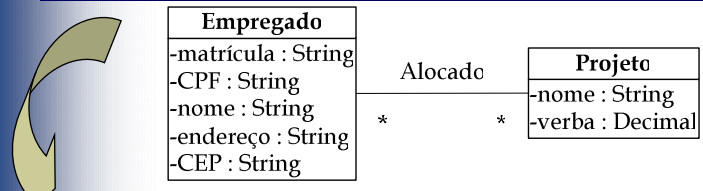
Mapeamento: Associações muitos-muitos

- 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 de conectividade muitos para muitos entre duas ou mais relações.

Mapeamento: Associações muitos-muitos

- Equivalente à aplicação do mapeamento *um para muitos* duas vezes, considerando-se os pares (Ta, Tassoc) e (Tb, Tassoc).
- Alternativas para definir a chave primária de Tassoc.
  - definir uma ***chave primária composta***.
  - criar uma coluna de implementação que sirva como chave primária simples da relação de associação.

Mapeamento: Associações muitos-muitos




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

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

Mapeamento: Agregações

- Forma especial de associação → *mesmo* procedimento para realizar o mapeamento de associações pode ser utilizado.
- No entanto, a diferença semântica influi na forma como o SGBDR deve agir quando um registro da relação correspondente ao *todo* deve ser excluído ou atualizado.
  - Remoção ou atualização em cascata.
  - Pode ser implementado como *gatilhos* e *procedimentos armazenados*.

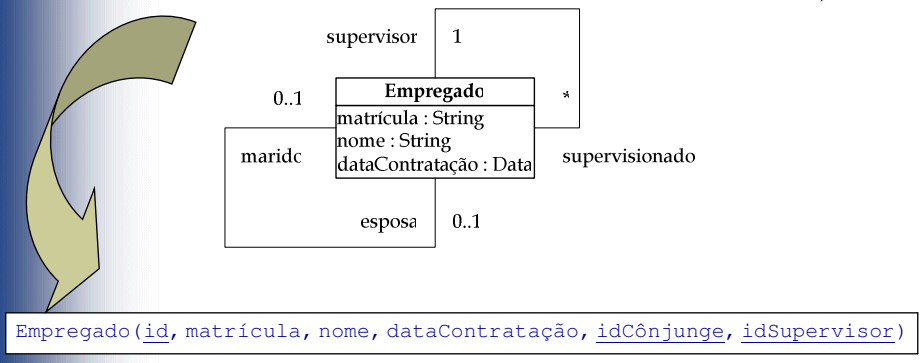
Mapeamento: Agregações

- O padrão de acesso em agregações (composições) também é diferente do encontrado nas associações.
  - Quando um objeto todo deve ser restaurado, é natural restaurar também os objetos parte.
  - Em associações, isso nem sempre é o caso.
  - Definição de *índices* adequados é importante para acesso eficiente aos objetos *parte*.

Mapeamento: 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.

Mapeamento: Associações Reflexivas

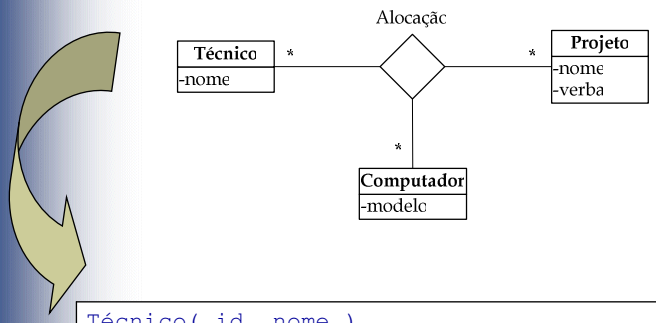


`Empregado(id, matrícula, nome, dataContratação, idCônjunge, idSupervisor)`

Mapeamento: 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.

Mapeamento: Associações n-árias

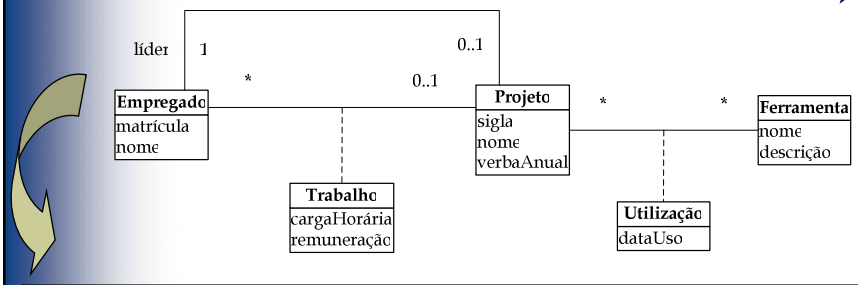



```
Técnico( id, nome )
Projeto( id, nome, verba )
Computador( id,  modelo )
Alocação( id, idProjeto, idTécnico, idComputador )
```

Mapeamento: 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.

Mapeamento: Classes Associativas

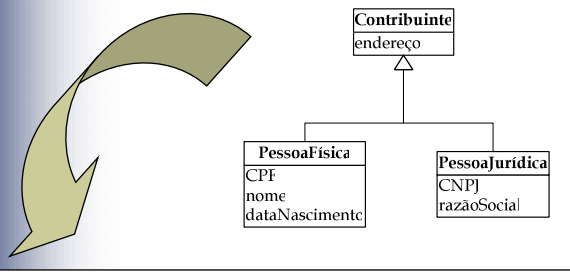



```
Empregado(id, matrícula, nome)
Projeto(id, sigla, nome, verbaAnual, idEmpregadoLíder)
Ferramenta(id, nome, descrição)
Utilização(id, idFerramenta, idProjeto, dataUso )
Trabalho(id, idEmpregado, idProjeto, cargaHorária, remuneração)
```

Mapeamento: 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 é a melhor.
  - Cada uma delas possui vantagens e desvantagens.
  - Escolha de uma delas depende das características do sistema em desenvolvimento.
  - A equipe de desenvolvimento pode decidir implementar mais de uma alternativa.

Mapeamento: Generalizações




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

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

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

Mapeamento: 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*.

Mapeamento: Generalizações

- A 2ª alternativa 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.

Mapeamento: Generalizações

- A 3ª alternativa 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.

Tarefas

Leitura Recomendada

- **Cap. 13 – Eduardo Bezerra**

Atividades Práticas

- **Todos os exercícios do cap. 13 – Eduardo Bezerra**
- **Projeto Lógico dos Modelos de Dados Conceituais de todos os exercícios de modelagem**