UnB Departamento de Ciência da Computação

# Mapeamento de Objetos para o Modelo Relacional

Na época Nixon estava normalizando as relações com a China.
Eu pensei que, se ele podia normalizar relações, eu também podia.
E. F. Codd

Edison Ishikawa, D. Sc.

# Introdução

- Objetivo
  - Apresentar método sistemático para mapear objetos para o modelo relacional
  - Apresentar Projeto Objeto-Relacional

# Sumário

- Introdução
- Desenvolvimento
- Referências

# Introdução

- Mapeamento de objetos para o modelo relacional é um problema relevante por dois motivos
  - Porque a tecnologia de orientação a objetos se consolidou como a forma usual de desenvolver sistemas de software
  - Porque a tecnologia de banco de dados relacionais é uma das tecnologias que tiveram maior êxito na área da computação e, sem dúvida, os Sistemas de Gerência de Banco de Dados Relacionais (SGBDR) dominam o mercado comercial

# Introdução

- Problema
  - Tecnologia OO e SGBDR nasceram de princípios teóricos bastante diferentes
  - Tecnologia OO
    - Encapsulamento
    - Sistemas OO são construídos com objetos, que encapsulam dados e operações
  - Tecnologia Relacional
    - Lida com armazenamento de dados tabulares

# Introdução

- Objetos de um sistema OO podem ser classificados em dois tipos
  - Objetos transientes
    - Existe somente durante uma sessão de uso do sistema
    - Ex: normalmente, objetos de controle e de fronteira são transientes
  - Objetos persistentes
    - Existência perdura por várias sessões do sistema
    - Ex: objetos de entidade são normalmente persistentes
    - Precisam ser armazenados quando uma execução do sistema termina, e restaurados quando uma outra execução é iniciada

# Introdução

- Nesta apresentação, quando nos referirmos a objetos, estaremos nos referindo a objetos persistentes.

# Introdução

- A rigor, quando uma sessão de uso de um sistema OO
  - é iniciada, todos os objetos tem que ser criados (alocar espaço em memória principal)
  - Termina, todos os objetos são destruídos ( o espaço em memória principal é devolvido ao S.O.)
- Objetos persistentes exigem algum mecanismo de armazenamento persistente, que armazena as informações dos objetos entre sessões de uso do sistema

# Introdução

- *Impedance mismatch*
  - O descasamento de informações é um termo utilizado para denotar o problema das diferenças entre as representações do Modelo OO e do Modelo Relacional
- Uma proporção significativa do esforço de desenvolvimento recai sobre a solução que o programador deve dar a este problema

# Projeto de Banco de Dados

- É uma das principais atividades do projeto detalhado, se não existir Banco de Dados
- Envolve
  - Construção do esquema do banco de dados
  - Criação de índices para agilizar  acesso aos dados armazenados
  - Definição das estruturas de dados a serem utilizadas no armazenamento físico dos dados
  - Definição de visões sobre os dados armazenados
  - Atribuição de direitos de acesso
  - Definição de políticas de backup dos dados
- Este tipo de projeto está fora do nosso escopo
- Foco apenas no mapeamento de informações entre as tecnologias OO e Relacional

# Mapeamento

- Mapeamento do modelo de classes para o modelo relacional resulta em um esquema de banco de dados, ou seja, na criação de um conjunto de representações que podem ser diretamente definidas no SGBD para a criação do Banco de Dados
- Existem ferramentas CASE que fazem este mapeamento de forma automática, mas é importante que o desenvolvedor tenha conhecimentos básicos dos procedimentos existentes para a realização do mapeamento.

# Conceitos do modelo de dados relacional

- Se fundamenta no conceito de relação
  - De forma simplista, uma relação é uma tabela composta de linhas e colunas
  - Cada relação possui um nome
  - Cada coluna de uma relação possui um nome e um domínio

**Departamento**

| id | sigla | nome | idGerente |
|---|---|---|---|
| 13 | RH | Recursos Humanos | 5 |
| 14 | INF | Informática | 2 |
| 15 | RF | Recursos Financeiros | 6 |

**Projeto**

| id | nome | Verba |
|---|---|---|
| 1 | PNADO | R$ 7.000,00 |
| 2 | BMMO | R$ 3.000,00 |
| 3 | SGI | R$ 6.000,00 |
| 4 | ACME | R$ 8.000,00 |

**Alocação**

| id | idProjeto | idEmpregado |
|---|---|---|
| 100 | 1 | 1 |
| 101 | 1 | 2 |
| 102 | 2 | 1 |
| 103 | 3 | 5 |
| 104 | 4 | 2 |

**Empregado**

| id | matrícula | CPF | nome | endereço | CEP | idDepto |
|---|---|---|---|---|---|---|
| 1 | 10223 | 038488847-89 | Carlos | SQN 301 A | 70568999 | 13 |
| 2 | 10490 | 024488847-67 | José | SCN 406 B | 70894561 | 13 |
| 3 | 10377 | NULL | Maria | SQS 501 F | NULL | NULL |
| 4 | 11057 | 034586378-20 | João | SBN QD 1 | NULL | 14 |
| 5 | 10922 | NULL | Aline | SQS 705 X | NULL | 14 |
| 6 | 11345 | 025464788-67 | Ana | SHN 780 D | NULL | 15 |

# Modelo Relacional

- Cada coluna pode conter apenas valores atômicos
  - Não pode conter informação estruturada
    - Exceção – tipo data – dia mês ano
- Chave primária
  - Coluna ou conjunto de colunas cujos valores são utilizados para identificar unicamente cada linha da relação
- Chave estrangeira
  - Deve existir, em uma das duas relações, uma coluna cujos valores fazem referência a valores de uma coluna de outra relação
  - Esta coluna de referência é denominada chave estrangeira
  - Pode conter  valor NULL
- NULL
  - Significa que um valor não se aplica, não existe ou é desconhecido

# Mapeamento de Objetos para o Modelo Relacional

- Realizado a partir do modelo de classes
- Semelhante ao mapeamento do Modelo Entidade Relacionamento (MER)
- No entanto, em virtude de o modelo de classes possuir mais recursos de representação que o MER, regras adicionais são definidas
- Mas, **MER e o Modelo de Classes não são equivalentes**
  - MER é um modelo de dados
  - Modelo de Classes moela objetos (dados e comportamento)

# Classes e seus atributos

- Classes são mapeadas para relações
  - Caso mais simples
    - Mapear cada classe como uma relação, e cada atributo desta classe como uma coluna da relação correspondente
  - No entanto, pode não haver uma correspondência unívoca entre classes e relações
    - Várias classes podem ser mapeadas em uma relação
    - Uma classe pode ser mapeada em vária relações

# Classes e seus atributos

- Para atributos, o que vale de forma geral é que um atributo será mapeado para uma ou mais colunas
  - Lembre-se que nem todos os atributos são persistentes. Ex: atributos derivados normalmente não são mapeados, mas por questão de desempenho...

# Classes e seus atributos


Mapeamentos possíveis para a classe cliente

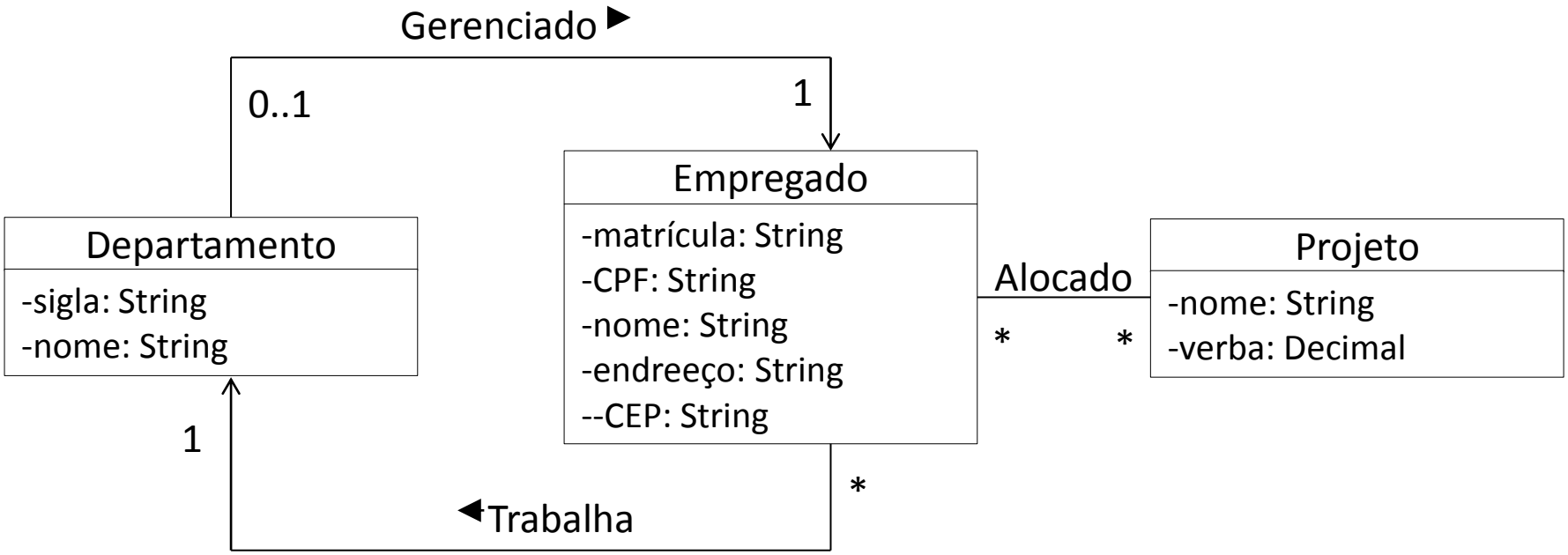
| | |
|---|---|
| 1ª | Cliente( id, CPF, nome, telefone, logradouro, dataNascimento, idCPF)<br>CPF(id, número, digitoVerificador) |
| 2ª | Cliente( id, nome, telefone, logradouro, dataNascimento, CPF, CEP) |

# Associações

- O mapeamento utiliza o conceito de chave estrangeira (utilizadas para relacionar linhas de relações diferentes ou linhas de uma mesma relação)
- Há 3 casos para mapeamento de associações, cada um corresponde a um tipo de *conectividade*
  - 1 para 1
  - 1 para muitos
  - Muitos para muitos

# Associações

- Considere, sem perda de generalidade, que há uma associação entre objetos de duas classes $C_a$ e $C_b$, e que estas duas classe foram mapeadas para duas relações separadas $T_a$ e $T_b$. Considere, ainda, o seguinte diagrama de classes a ser nos exemplos:



# Associações de conectividade um para um

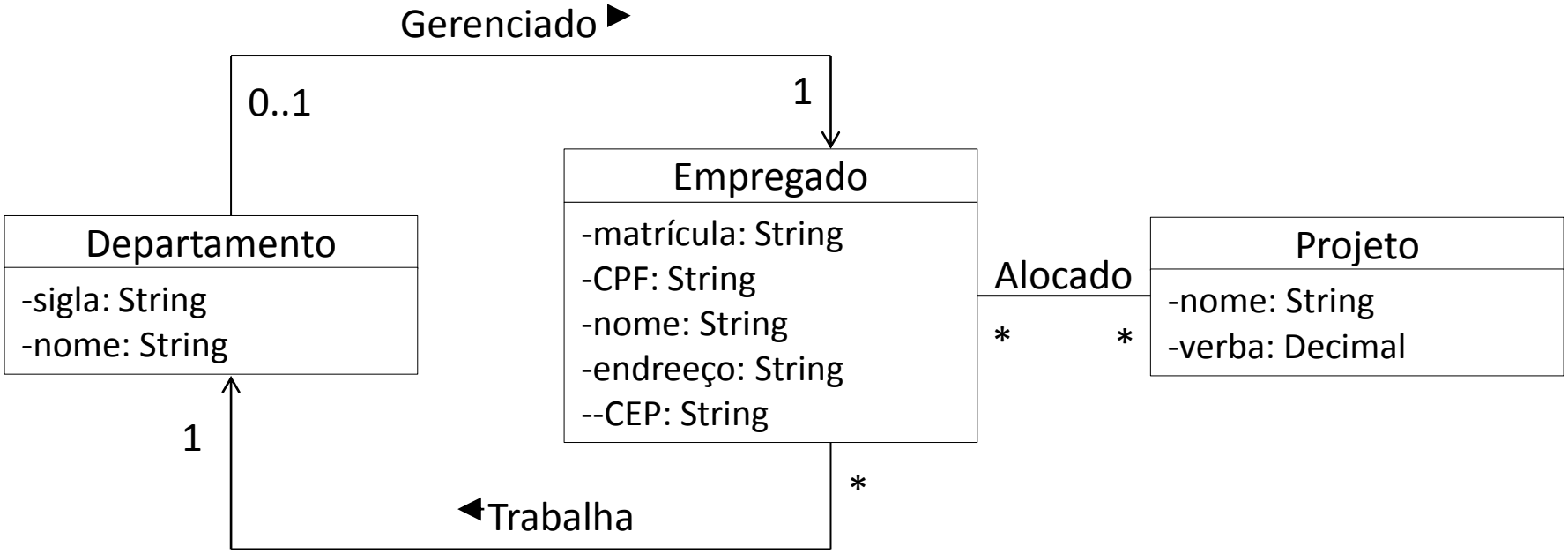
- Quando há uma associação um para um entre $C_a$ e $C_b$ deve-se adicionar uma chave estrangeira em uma das duas relações para referenciar a chave primária da outra relação
- Com respeito à escolha da relação na qual a chave estrangeira  deve ser adicionada, deve-se observar se a participação na associação é opcional ou obrigatória em ambos os extremos da associação. 3 possibilidades
  1. A associação é obrigatória em ambos os extremos
  2. A associação é opcional em ambos os extremos
  3. A associação é obrigatória em um extremo e opcional no outro extremo

# Associações de conectividade um para um

1. A associação é obrigatória em ambos os extremos
2. A associação é opcional em ambos os extremos
3. A associação é obrigatória em um extremo e opcional no outro extremo

- Nos casos 1 e 2, qualquer das relações pode receber a chave estrangeira
- No caso 3, deve-se optar pela relação que corresponde à classe de participação obrigatória
  - Dessa forma, esta coluna não terá valores NULL

# Associações de conectividade um para um




- A relação gerenciado é 1 para 1 caso 3

Departamento( id, sigla, nome, idEmpregadoGerente )

Empregado( id, matrícula, CPF, nome, endereço, CEP )

# Associações de conectividade um para um

- Com o objetivo de aumentar  desempenho do processamento, as classes que participam de uma associação um para um também podem ser mapeadas para uma única relação
  - Nesse caso, não há necessidade do uso de uma chave estrangeira

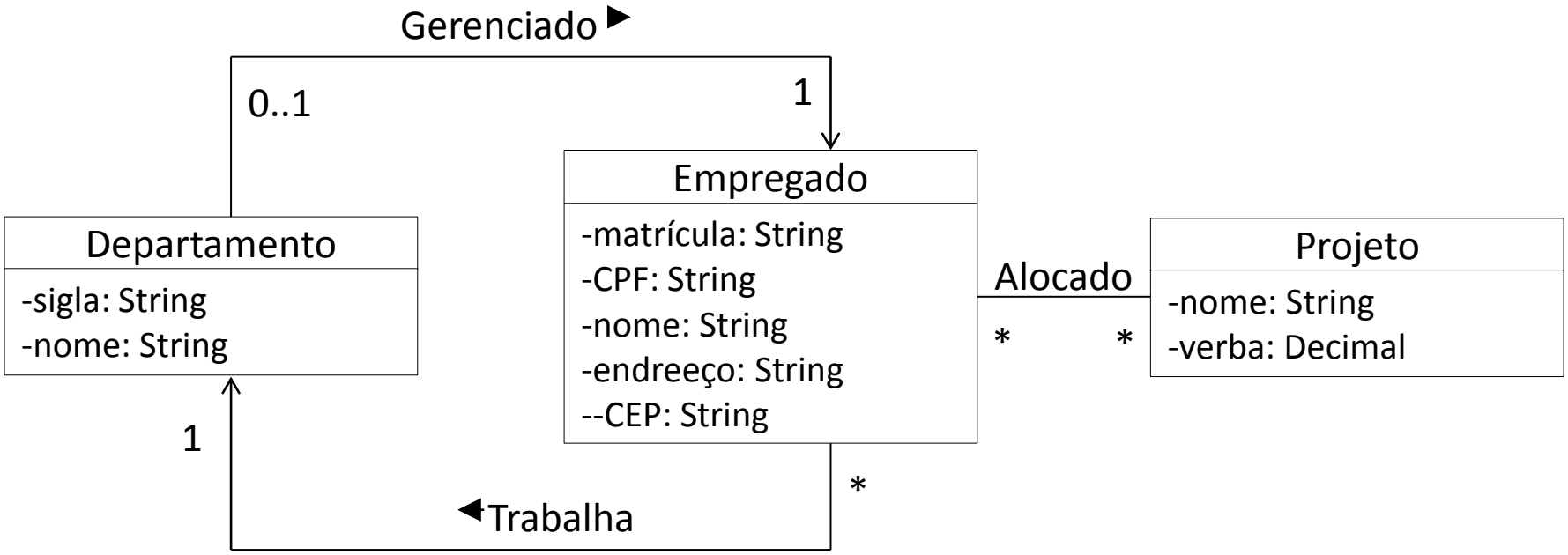
Departamento( id, sigla, nome, gerente, matrícula, CPF, nome, endereço, CEP )

# Associação de conectividade um para muitos

- Em uma associação um para muitos entre objetos de $C_a$ e de $C_b$ , seja $C_a$ a classe na qual cada objeto se associa com muitos objetos da classe $C_b$ . Neste caso, deve-se adicionar uma chave estrangeira em $T_a$ para referenciar a chave primária de $T_b$.

# Associação de conectividade um para muitos

- Como exemplo considere a associação Trabalha




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

# Associação de conectividade muitos para muitos

- Quando há uma associação muitos para muitos entre objetos de $C_a$ e de $C_b$ , uma relação de associação deve ser criada.
- Uma relação de associação tem o objetivo de representar a associação muitos para muitos entre duas ou mais relações
- Este tipo de conectividade exige a criação de uma nova relação

# Associação de conectividade muitos para muitos

- Seja $T_{assoc}$ o nome da relação de associação. O mapeamento de uma associação muitos para muitos é feito aplicando-se a regra para o mapeamento um para muitos duas vezes, considerando-se separadamente os pares de relações ($T_a$, $T_{assoc}$ ) e ($T_b$, $T_{assoc}$ )
- Há duas alternativas para se definir a chave primária de $T_{assoc}$ .
  - Primeiramente, pode-se definir uma chave primária composta para $T_{assoc}$ . Uma chave primária composta é uma chave primária composta de mais de uma coluna.
  - A segunda alternativa é criar uma coluna de implementação que sirva como chave primária simples da relação de associação.

# Associação de conectividade muitos para muitos

| | |
|---|---|
| 1ª | Departamento( id, sigla, nome, idEmpregadoGerente )<br>Empregado( id, matrícula, CPF, nome, endereço, CEP, idDepartamento)<br>Alocação( idProjeto, idEmpregado)<br>Projeto( id, nome, verba) |
| 2ª | Departamento( id, sigla, nome, idEmpregadoGerente )<br>Empregado( id, matrícula, CPF, nome, endereço, CEP, idDepartamento)<br>Alocação( id, idProjeto, idEmpregado)<br>Projeto( id, nome, verba) |

# Agregações e Composições

- Uma agregação (ou composição) é uma forma especial de associação
  - Portanto, o mesmo procedimento para realizar o mapeamento de associações pode ser utilizado para agregações ou composições
- No entanto, a diferença semântica entre uma associação e uma agregação (composição) influi na forma como o SGBDR deve agir quando um registro da relação correspondente ao todo deve ser excluído ou atualizado

# Agregações e Composições

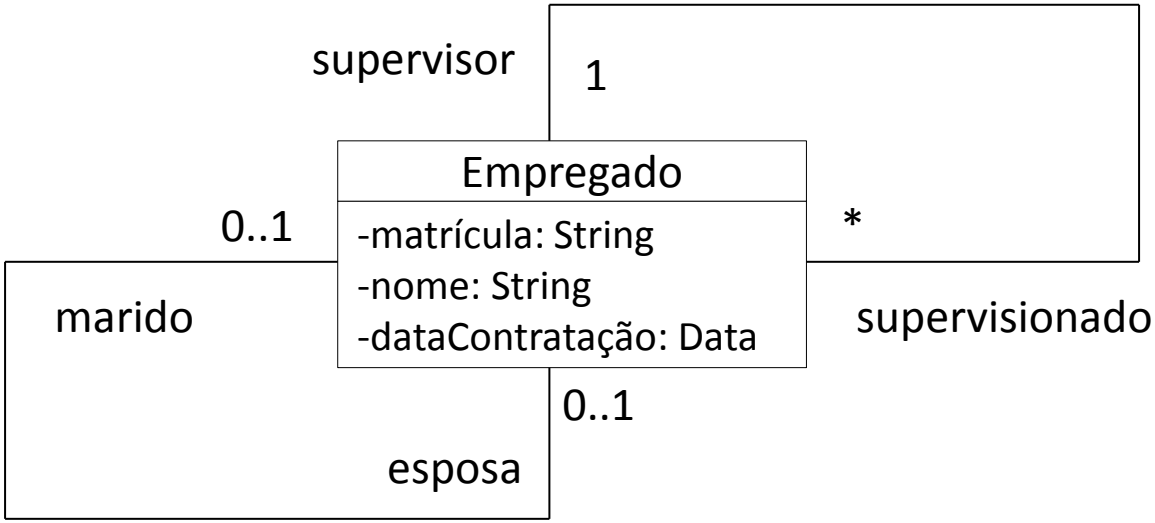
- Exemplo
  - Pode ser necessário que, quando um objeto todo for removido, os objetos parte também sejam removidos
    - Ex: não faz sentido ficar armazenando itens de um pedido, de um pedido que já foi removido
    - Isto pode ser implementado utilizando-se recursos do SGBDR, como gatilhos (triggers) e procedimentos armazenados (stored procedures)

# Agregações e Composições

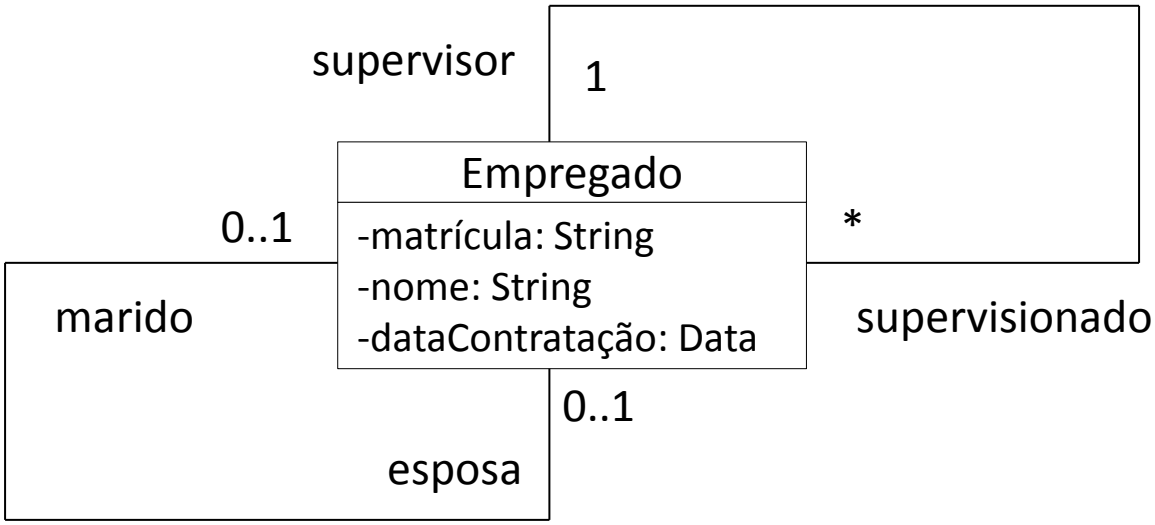
- O padrão de acesso em agregações (composições) também é diferente do encontrado nas associações
  - Usualmente, quando um objeto todo deve ser restaurado, é natural restaurar também os objetos parte. Em associações, isso nem sempre é o caso.
  - Nesta situação, a definição de índices adequados no SGBDR é importante para que o acesso aos objetos parte seja feito da forma mais eficiente possível.

# Associações Reflexivas

- É um tipo especial de associação.
- O mapeamento aplicado a associações se aplica da mesma forma aqui

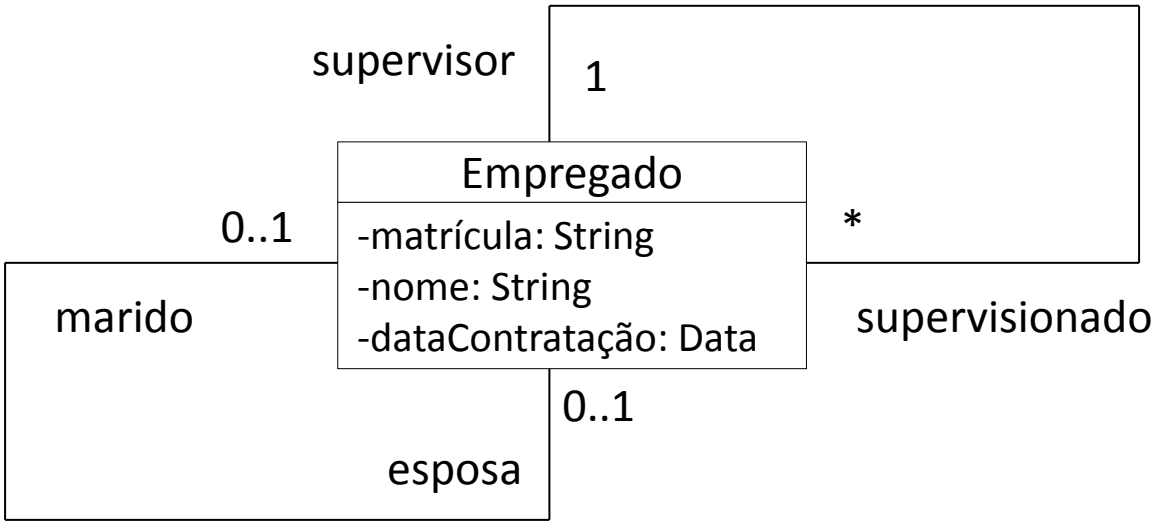


# Associações Reflexivas



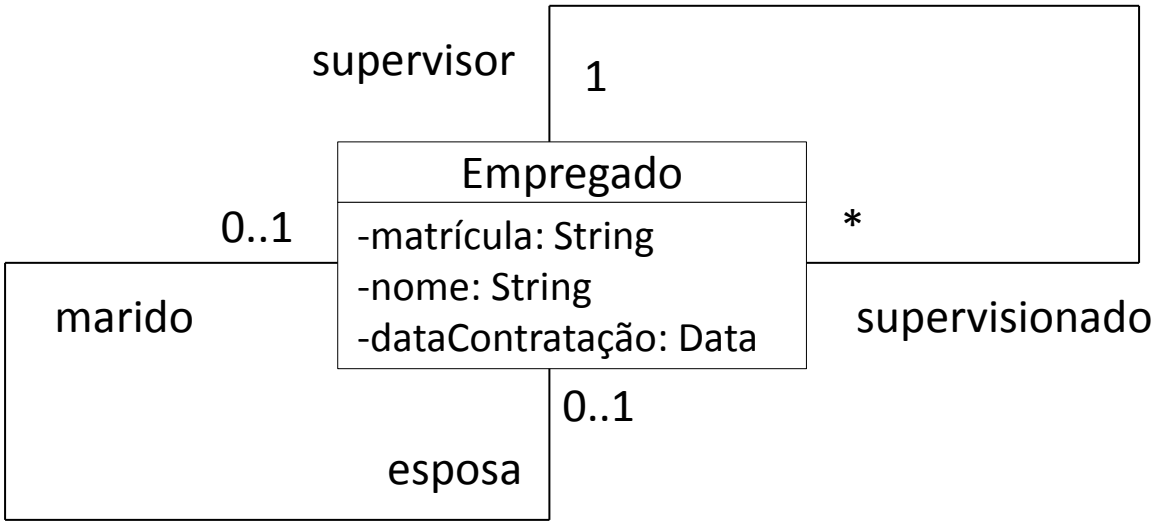

- Cada empregado tem um supervisor, sendo o próprio supervisor um empregado
- O diagrama também representa relacionamentos marido/esposa entre os empregados

# Associações Reflexivas




- No mapeamento a seguir, as chaves estrangeiras idCônjuge e idSupervisor foram definidas na relação Empregado, visto que ambas as relações são reflexivas
- Empregado( id, matrícula, nome, endereço, dataContratação, idCônjuge, idSupervisor)

# Associações Reflexivas

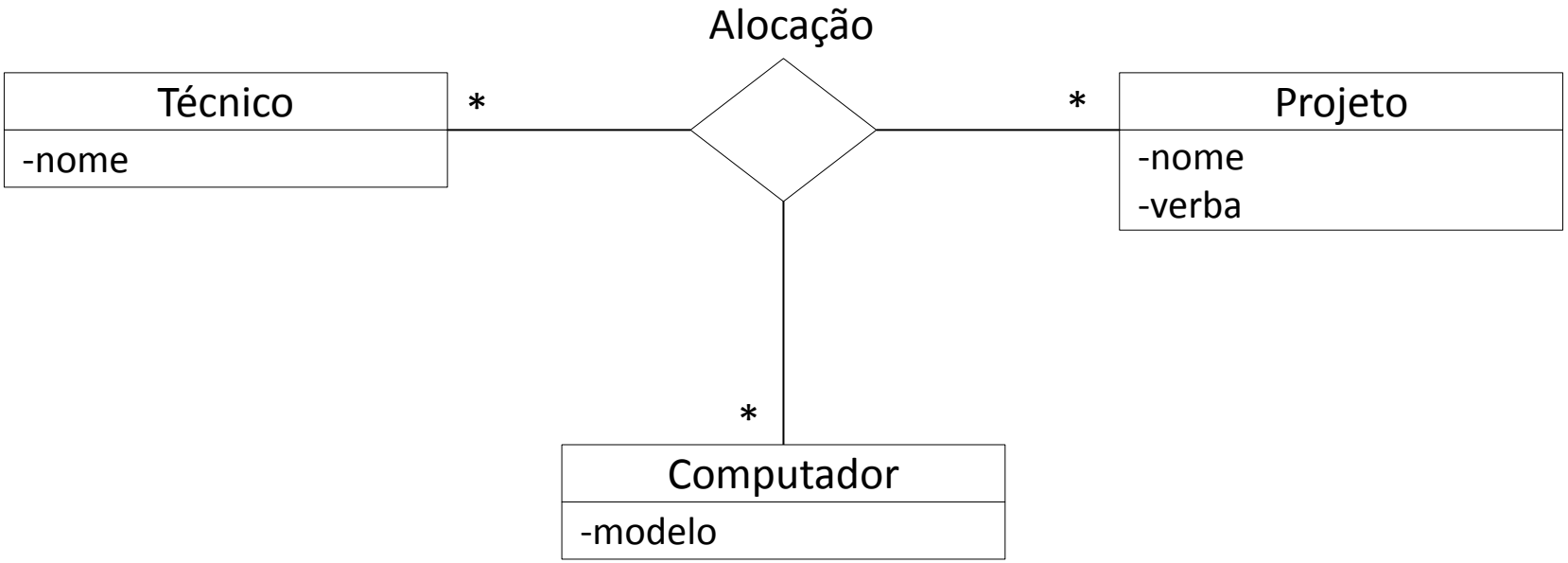



- No caso de uma associação reflexiva de conectividade muitos para muitos, uma relação de associação deve ser criadas para implementar o mapeamento, no mesmo molde das associações muito para muitos não-reflexivas

# Associações Ternárias

- Associações ternárias e, de modo geral, associações n-árias, podem ser mapeadas através de um procedimento semelhante ao utilizado para associações binárias muitos para muitos
- Uma relação para representar a associação é criada e são adicionadas nesta relação chaves estrangeiras para as n classes participantes da associação
- Se a associação ternária possuir uma classe associativa, os atributos destas são mapeadas como colunas da associação

# Associações Ternárias




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

# Classes Associativas

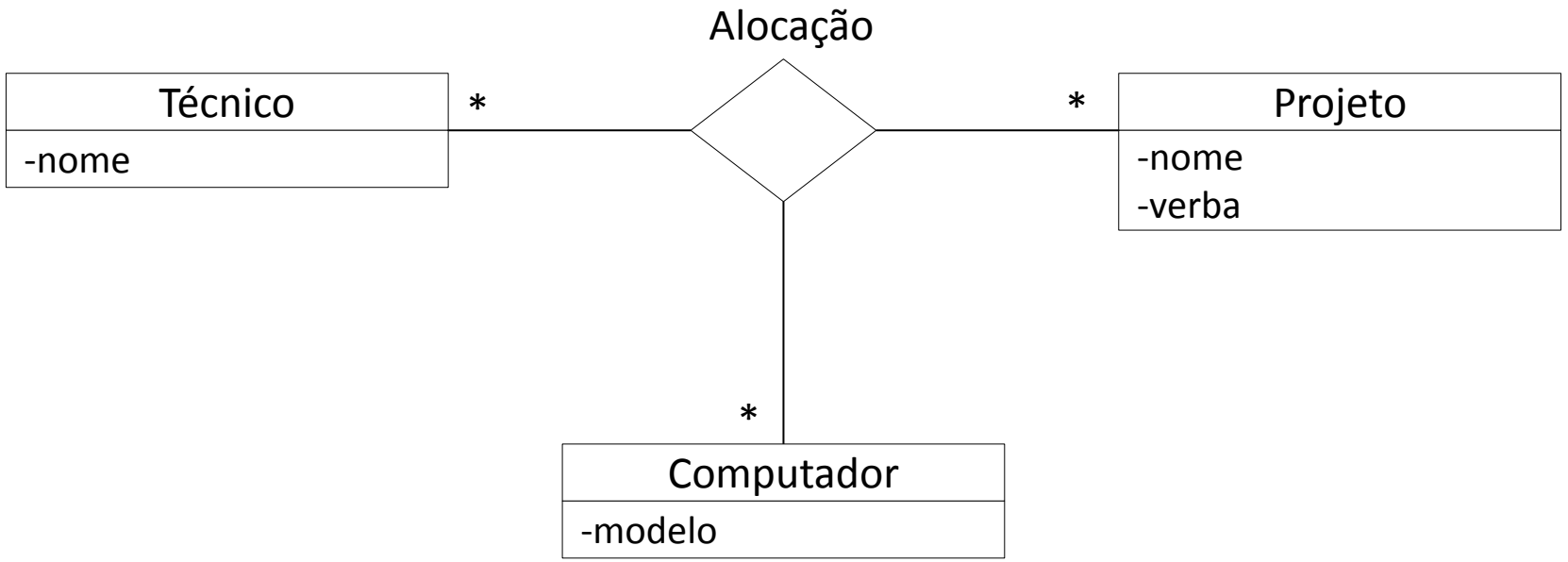
- Uma classe associativa é geralmente encontrada em associações de conectividade muitos para muitos
- Mas nada impede em que ela seja utilizada em associações de conectividade um para muitos ou um para um
- Logo, para cada um dos três casos de mapeamento de associações entre objetos, há uma variante em que uma classe associativa é utilizada

# Classes Associativas

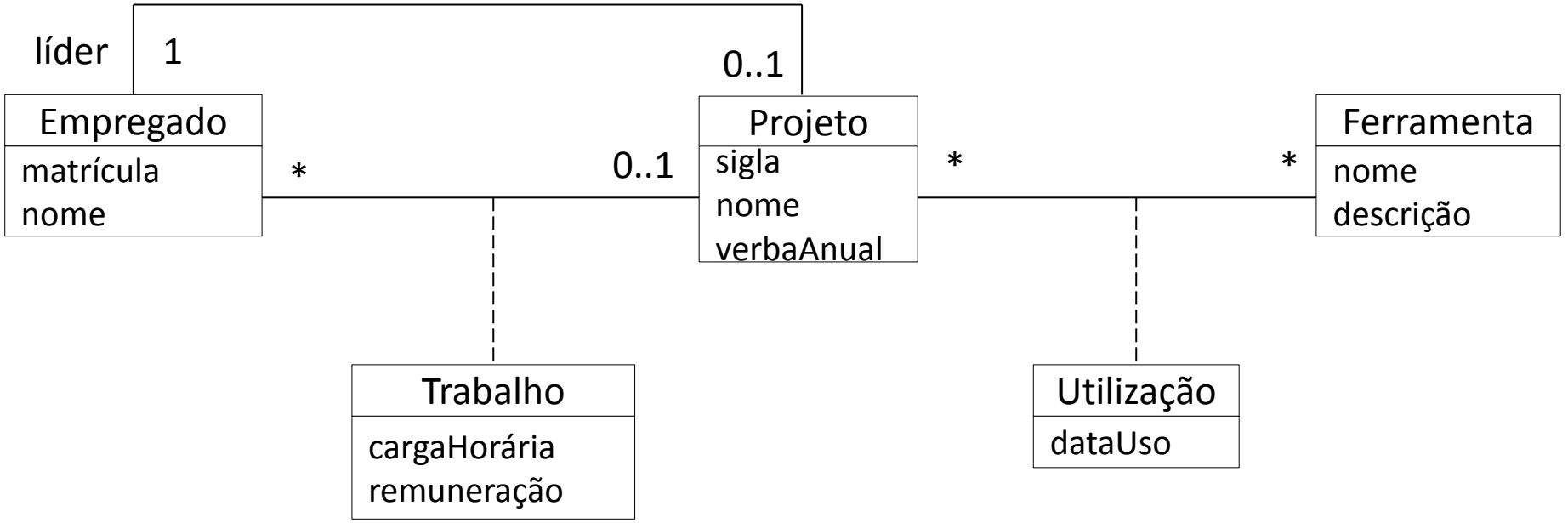
- De uma forma geral, o mapeamento de ma classe associativa é feito através da criação de uma relação para representá-la.
- Os atributos da classe associativa são mapeados para colunas dessa relação
- Além disso, a relação deve conter chaves estrangeiras que referenciam as relações correspondentes às classes que participam da associação

# Classes Associativas

- Exemplo: estendendo o diagrama de classe visto anteriormente



# Classes Associativas



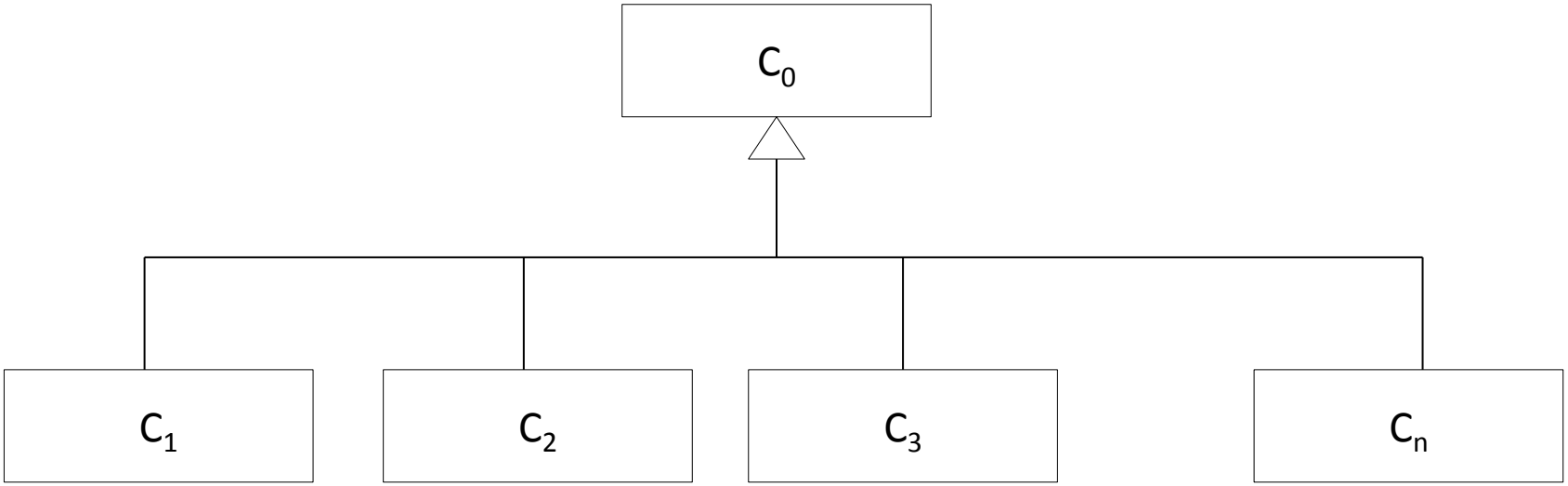

Empregado( id, matrícula, nome)
Projeto( id, sigla, nome, verbaAnual)
Ferramenta( id, nome, descrição)
Utilização( id, idProjeto, idFerramenta, dataUso)
Trabalho( id, idEmpregado, idProjeto, cargaHorária, remuneração)

# Classes Associativas

- Há uma outra forma de realizar o mapeamento da classe associativa Trabalho, a saber, não criar uma relação, mas mapear os atributos dessa classe na relação correspondente à classe Empregado
- Essa alternativa tem a desvantagem de apresentar um potencial desperdício de espaço, pois as colunas cargaHorária e remuneração não seriam utilizadas para empregados que não estivessem trabalhando em projeto algum.

# Generalização

- 3 alternativas de se mapear relacionamentos de generalização
- Considere a seguinte hierarquia de generalização, onde a superclasse C0, possui diversas subclasses Ci, $1 \leq i \leq n$.



# 1. Uma relação para cada classe da hierarquia

- Uma relação é criada para cada uma das classes ($C_0$, $C_1$, $C_2$, . . . , $C_n$)
- A seguir, considera-se que há uma correspondência unívoca entre os objetos de $C_0$ e objetos de $C_1$.
  - Na verdade, essa correspondência entre objetos no relacionamento de herança não existe. O relacionamento de herança não é um relacionamento entre objetos, mas sim entre classes de objetos. Esta considerção é apenas um artifício
- Feita esta consideração de correspondência, aplica-se a mesma regra de mapeamento para associações de conectividade “um para um”, seno a chave estrangeira  adicionada na relação que implementa a subclasse

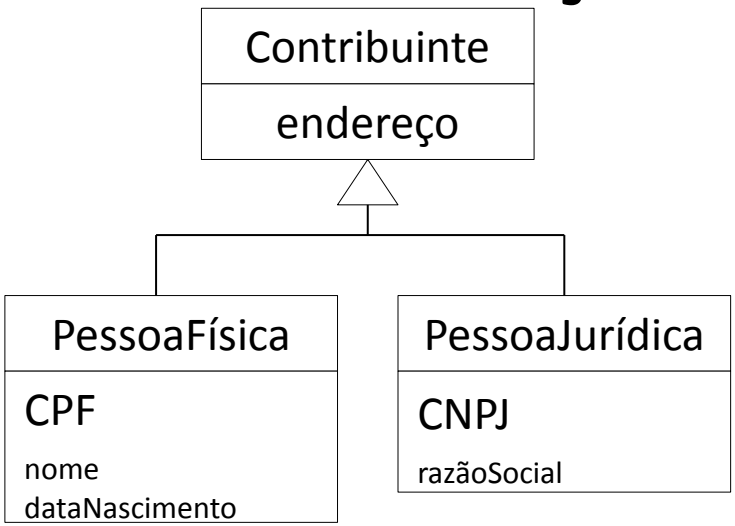
# 2. Uma relação para toda a hierarquia

- Nesta alternativa, cria-se uma única relação com colunas para representar os atributos da super classe e também os atributos de todas as subclasses.
- Além disso, a relação deve conter mais uma coluna cujos valores servem para identificar, dado um objeto (linha) desta relação, a qual classe da hierarquia ela pertence.

# 3. Uma relação para cada classe concreta da hierarquia

- Nesta alternativa, para cada subclasse é criada uma relação.
- Os atributos específicos de cada subclasse são mapeadas nas colunas na relação correspondente.
- Cada relação que implementa uma subclasse $C_i$, também possui colunas para cada atributo herdado da superclasse $C_0$. Dessa forma, os atributos da superclasse são replicados pelas *n* relações que implementam as subclasses.

# Exemplo de Mapeamento de Generalização




| | |
|---|---|
| 1ª alternativa | Contribuinte ( id, endereço)<br>PessoaFísica( id, nome, dataNascimento, CPF, idContribuinte)<br>PessoaJurídica( id, CNPJ, razãoSocial, idContribuinte) |
| 2ª alternativa | Contribuinte( id, nome, endereço, dataNascimento, CPF, CNPJ, razãoSocial, tipo) |
| 3ª alternativa | PessoaFísica( id, nome, dataNascimento, CPF, endereço)<br>PessoaJurídica( id, CNPJ, razãoSocial, endereço) |

# Comparação entre as estratégias de mapeamento

1. Uma relação para cada classe da hierarquia
   - É a que melhor reflete o modelo orientado a objetos
   - Tem a desvantagem em relação ao desempenho da manipulação das relações
     - Sempre que for inserir(remover) um objeto da subclasse, deverá haver inserção(remoção) de dois registros, um em cada relação.
     - É desvantajosa em consultas que precisem processar dados em que as colunas necessárias estão em mais de uma relação. Neste caso uma operação de junção (bastante cara do ponto de vista de desempenho) sobre as relações envolvidas deverá ser realizada.

# Comparação entre as estratégias de mapeamento

2. Uma relação para toda a hierarquia
   - Bastante simples, além de facilitar situações em que objetos mudam de classe
   - Desvantagem
     - um novo atributo deve ser adicionado ou removido de uma das classes da hierarquia. Neste caso, não importa qual seja a classe, a relação que mapeia toda a hierarquia é sempre modificada
     - Tem o potencial de desperdiçar bastante espaço de armazenamento, principalmente no caso em que a hierarquia tem uma largura grande (várias classes “irmãs”) e os objetos pertencem a uma e somente uma, classe da hierarquia

# Comparação entre as estratégias de mapeamento

3. Uma relação para cada classe concreta da hierarquia
   - Vantagem
     - Agrupa os objetos de uma classe em um única relação
   - Desvantagem
     - Quando uma classe é modificada, cada uma das relações correspondentes às suas subclasses deve ser modificada
       - Ex: considere a quantidade de trabalho a ser realizada quando um atributo for adicionado à classe Contribuinte. As relações PessoaFísica e PessoaJurídica devem ser alteradas.

# Comparação entre as estratégias de mapeamento

- Conclusão
  - Nenhuma das alternativas de mapeamento de generalização é a melhor
  - Cada uma possui vantagens e desvantagens
  - A escolha da alternativa a ser usada depende das características do sistema de software sendo desenvolvido
  - Mais que isso, a equipe de desenvolvimento pode decidir implementar mais de uma alternativa
    - Isto pode ser feito por meio do conceito de visões em um SGBDR

# Construção da camada de persistência

- Acesso direto a BD
  - Código SQL para a restauração, atualização, remoção e consultas das tabelas onde estão armazenados os objetos
  - Solução de fácil implementação em linguagens de Quarta Geração (Visual Basic, Power Builder e Delphi) que possuem os controles *data aware*
    - Desvantagem
      - Classes relativas à lógica do negócio ficam muito acopladas às classes relativas à interface e ao acesso ao BD,
        - Torna complexa a migração de um SGBD para outro
        - Lógica da aplicação fica desprotegida de eventuais modificações na estrutura do SGBD
      - Diminui coesão das classes. Cada classe possui responsabilidades relativas ao armazenamento, materialização dos objetos e ao negócio
      - A dificuldade de manutenção e extensão do código fonte particamente proíbe a utilização desta abordagem para sistemas complexos

# Construção da camada de persistência

- Uso de um SGBDOO ou SGBDOR
  - SGBDOO
    - ODMG
      - ODL
      - OQL
    - Não teve êxito comercial
  - SGBDOR
    - SGBDR incorporaram características OO
      - Oracle9i
      - DB2 Unversal Server
    - É o  adotado pelas empresas
    - Necessidade de estratégias para mapeamento de objetos mais realistas
      - Padrão DAO
      - Framework ORM

# Construção da camada de persistência

- Padrão DAO
  - Data Access Object
  - Desacopla as classes de negócio dos aspectos relativos aos mecanismos de acesso ao armazenamento persistente
  - Substitui com vantagens API (JDBC, ADO.NET) que apresentam o risco de acoplamento e, em consequência, são menos portáveis e desnecessariamente atrelados a uma tecnologia específica de armazenamento.
  - Embora o padrão DAO targa maior flexibilidade para a aplicação, ele aumenta a complexidade da mesma pelo simples fato de a quantidade de classes aumentar

# Construção da camada de persistência

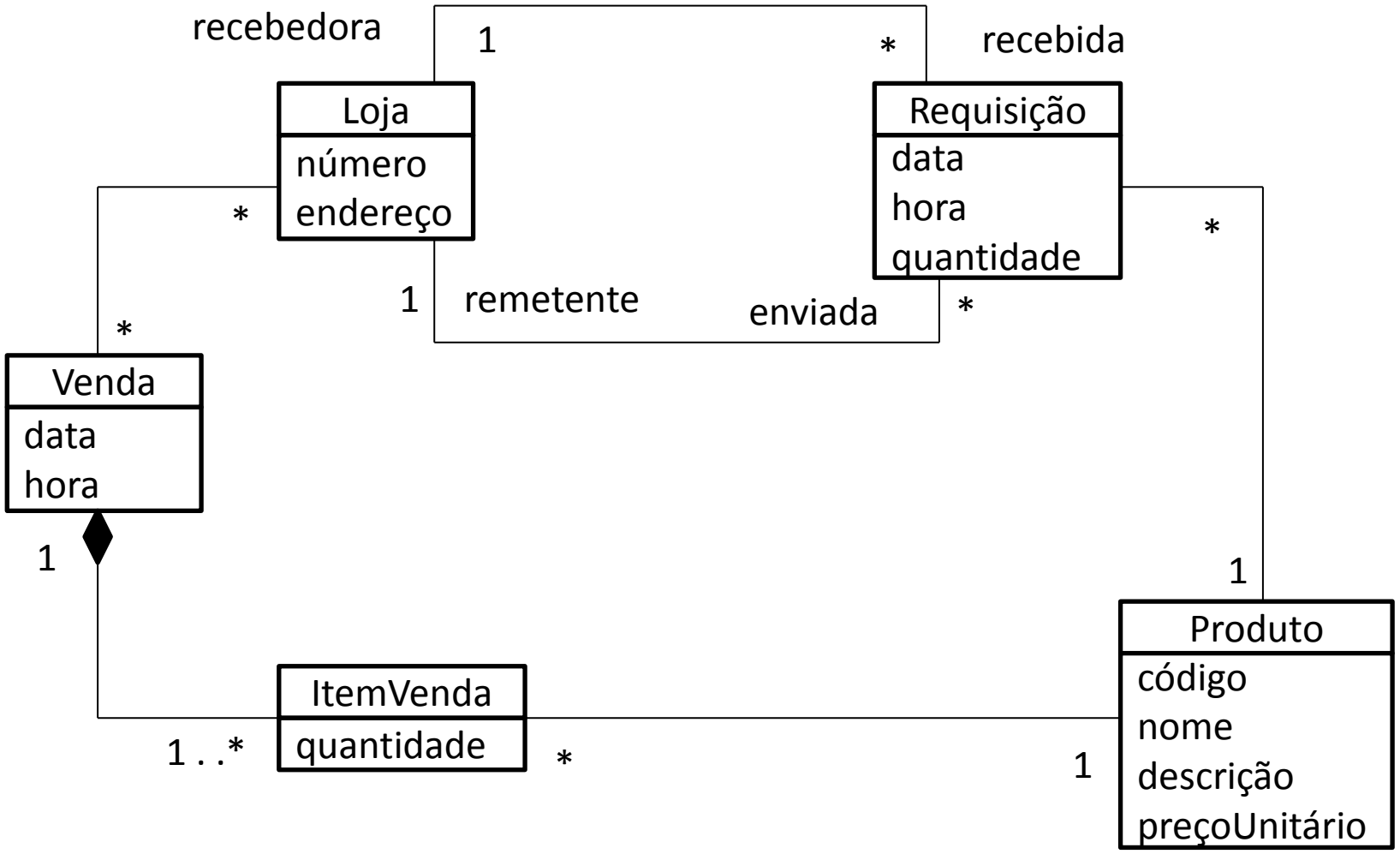
- Framework ORM
  - Object-Relational Mapping
  - Conjunto de classes que realiza o mapeamento transparente entre objetos da aplicação e tabelas em um SGBDR
  - Tenta resolver o problema de descasamento de informações fornecendo classes que o realizam de forma transparente para o programador
  - Desenvolvedor fornece a correspondência entre a estrutura de objetos da aplicação e o esquema relacional do BD, por meio de um arquivo de configuração, denominado arquivo de mapeamento

# Construção da camada de persistência

- Framework ORM
  - Exemplo: objeto Aluno recebeu uma mensagem para informar as disciplinas que já cursou. Essa mensagem ocasiona uma junção entre a tabela que guarda os registros dos alunos e a tabela que guarda registros e disciplinas. Com o uso do framework ORM, tudo o que o programador precisa fazer é implementar um método obterDisciplinasCursadas na classe Aluno; quando uma mensagem é recebida para ativar este método, o Framework ORM é responsável por mapear essa chamada  em uma expressão SQL a ser enviada ao SGBDR, receber o resultado, e criar uma lsita de objetos Disciplina para ser retornada pelo método. Tudo é fito transparentemente, uma vez pelo ORM.

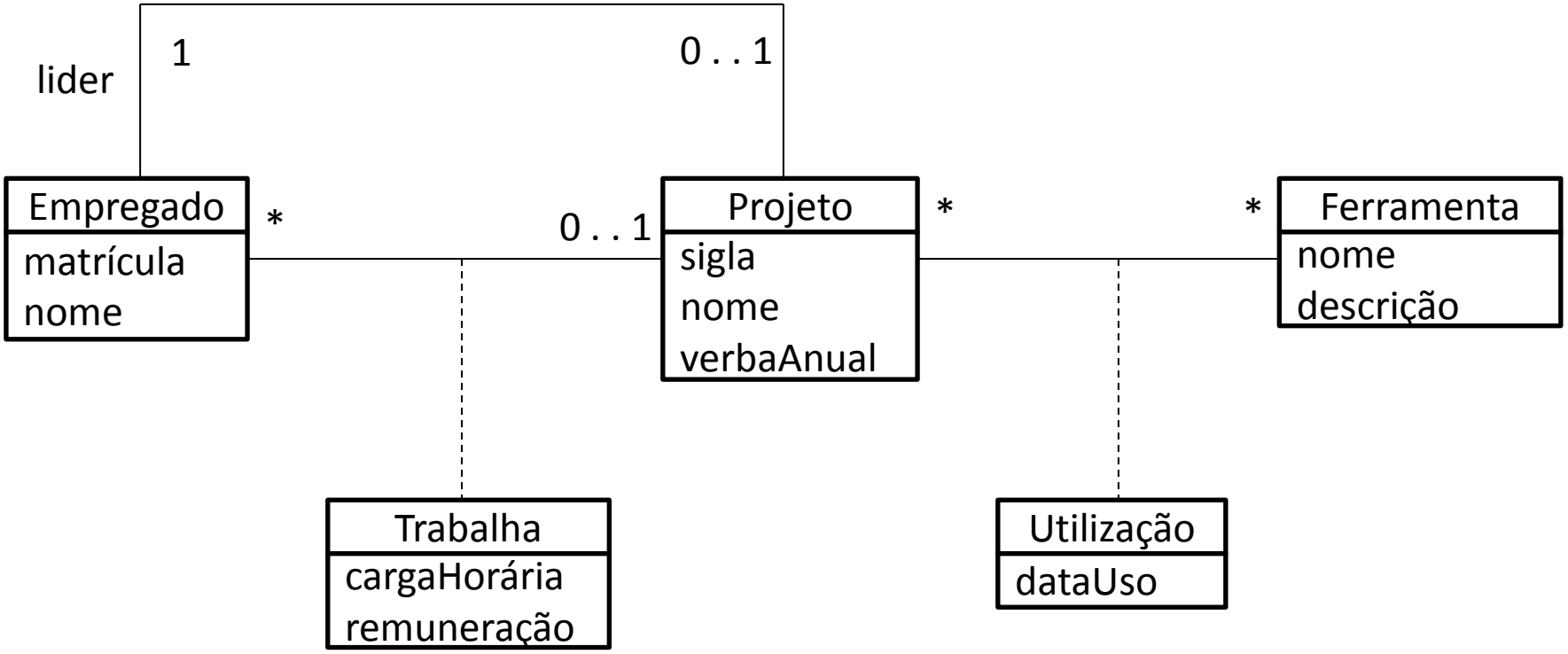
# Exercício 1

Dado o diagrama de classes a seguir, mapeie o diagrama para o modelo relacional.



# Exercício 2

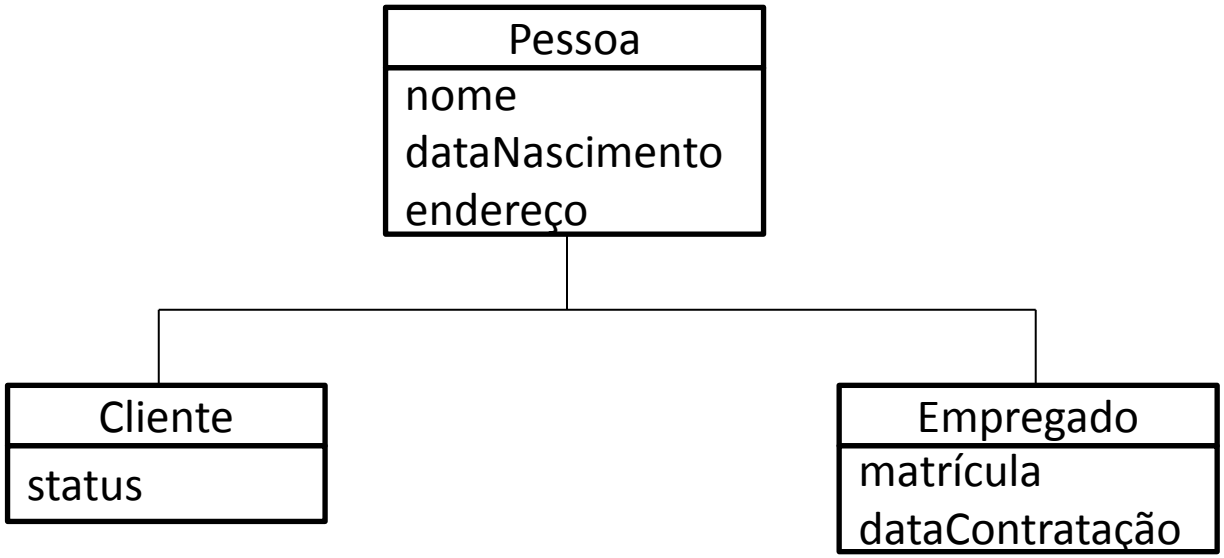
O diagrama de classes a seguir informa que empregados devem liderar um projeto por vez, enquanto cada projeto tem, necessariamente, um só líder. O diagrama também informa que pode haver diversos empregados trabalhando em um projeto, embora não possa haver um empregado trabalhando em mais de um projeto. Finalmente, projetos utilizam determinadas ferramentas, e uma ferramenta pode ser utilizada por vários projetos. Realize o mapeamento deste diagrama para o modelo relacional.



# Exercício 3

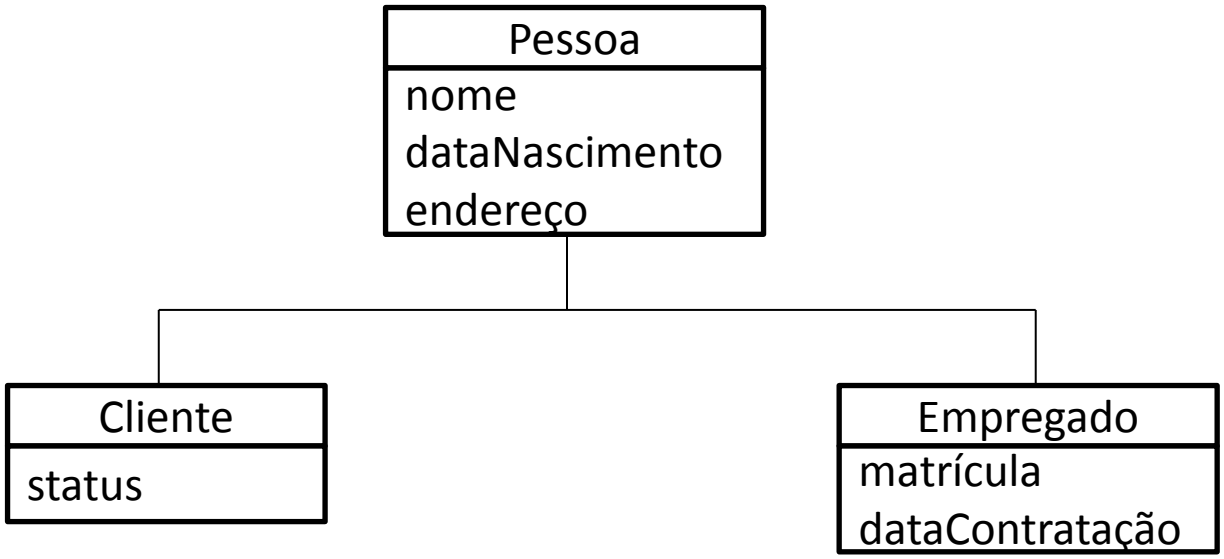
Considere a hierarquia de generalização a seguir, onde empregado e Cliente são subclasses de pessoa. Discuta o que acontece em termos dos objetos armazenados em relações em cada uma das alternativas de mapeamento de generalização nas situações a seguir:

a) Um empregado se torna uma cliente da loja
b) Um cliente se torna empregado da loja
c) Uma pessoa é tanto um empregado quanto um cliente da loja



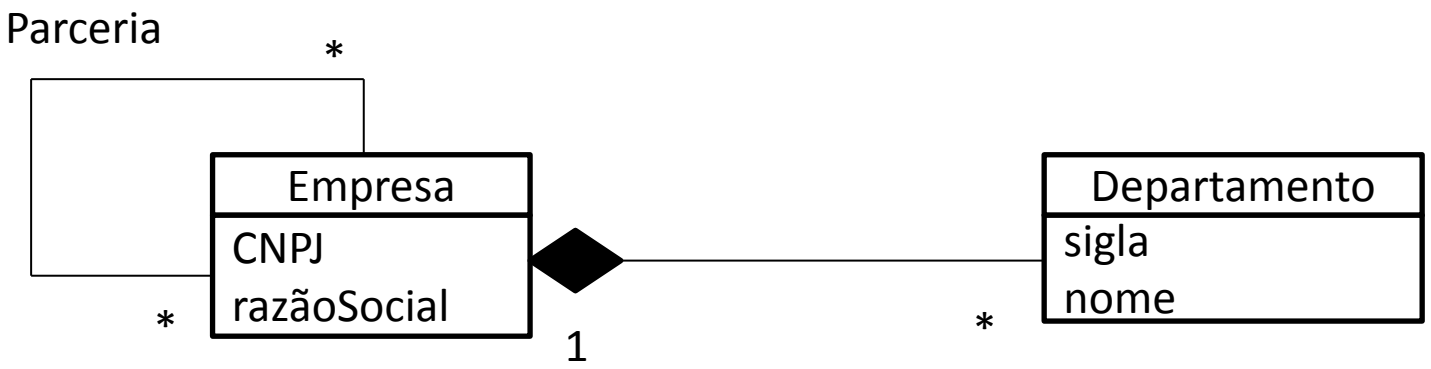
# Exercício 4

Considere a mesma hierarquia de generalização da questão anterior. Suponha que uma nova classe, Gerente, é adicionada à hierarquia como subclasse de Empregado, Suponha, ainda, que esta nova classe tem um atributo que informa a comissão para gerentes (considere que só empregados gerentes têm comissão). Reflita sobre as modificações que devem ser feitas ao esquema de banco de dados em cada uma das alternativas de mapeamento de generalização



# Exercício 5

Realize o mapeamento para o modelo relacional do seguinte diagrama de classe



# Referências

- Ferramentas de modelagem visual
  - Rational Rose (www.rational.com)
  - ASTAH Community (astah.net/editions/community)
- Livros
  - The Unified Modeling Language User Guide, Grady Booch et al
  - Engenharia de software – uma abordagem profissional, Roger S. Pressman
  - Princípios de análise e projetos de sistemas com UML, Eduardo Bezerra
  - Projeto e modelagem de Banco de Dados, T. Teorey e outros
- Especificações
  - www.omg.org

# Dúvidas