Mapeamento de Objetos para Tabelas Relacionais

- Sem dúvida os SGBDR dominam o mercado comercial.
  - Oracle
  - MS-SQLServer
  - IBM DB2
  - Informix
  - Sybase
  - Ingres
  - MySQL
  - PostgreSQL
  - Outros...

Banco de Dados OO

- Maiores informações: http://www.odbms.org/
- A informação a ser armazenada é o objeto
- Apenas para nichos específicos
  - Telecomunicações
  - espaço
  - saúde
  - Multimedia
  - CAD/CAM/CAE

Tipos de objetos

- Os objetos podem ser persistentes ou 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 e recuperados.
  - Tipicamente objetos de entidade.

Projeto de banco de dados

- Uma das primeiras atividades do projeto detalhado de um software é o desenvolvimento do banco de dados.
- Essa atividade corresponde ao ***projeto do banco de dados.*** Principais tarefas:
  - Construção do esquema do banco de dados
  - Armazenamento físico dos dados
  - Definição de visões sobre os dados armazenados.
  - Atribuição de direitos de acesso
  - Políticas de backup dos dados

Conceitos do modelo relacional

- O modelo relacional é fundamentado no conceito de ***relação.***
- Cada coluna de uma relação pode conter apenas ***valores atômicos.***
  - ***chave primária: colunas cujos valores podem ser utilizados para identificar unicamente cada linha de uma relação.***
- Associações entre linhas: valores de uma coluna fazem referência a valores de uma outra coluna. (***chave estrangeira) .***
  - Uma chave estrangeira também pode conter ***valores nulos , representados pela constante NULL.***
- O NULL é normalmente é usado para indicar que um valor não se aplica, ou é desconhecido, ou não existe.

Mapeamento de objetos para o modelo relacional

- É a partir do modelo de classes que o mapeamento de objetos para o modelo relacional é realizado.
  - Semelhante ao mapeamento do MER.
  - Diferenças em virtude de o modelo de classes possuir mais recursos de representação que o MER.
- MER e o modelo de classes não são equivalentes!
  - O MER é um modelo de dados;
  - O modelo de classes representa objetos (dados e comportamento).

Mapeamento de objetos para o modelo relacional

- Os exemplos dados a seguir utilizam 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

Mapeamento: Classes e seus atributos

- Classes são mapeadas para tabelas.
  - Caso mais simples: mapear cada classe como uma tabela, e cada atributo como uma coluna.
  - No entanto, pode não haver correspondência unívoca entre classes e tabelas.
- 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 de classes e seus atributos

Classe

Cliente
nome endereço

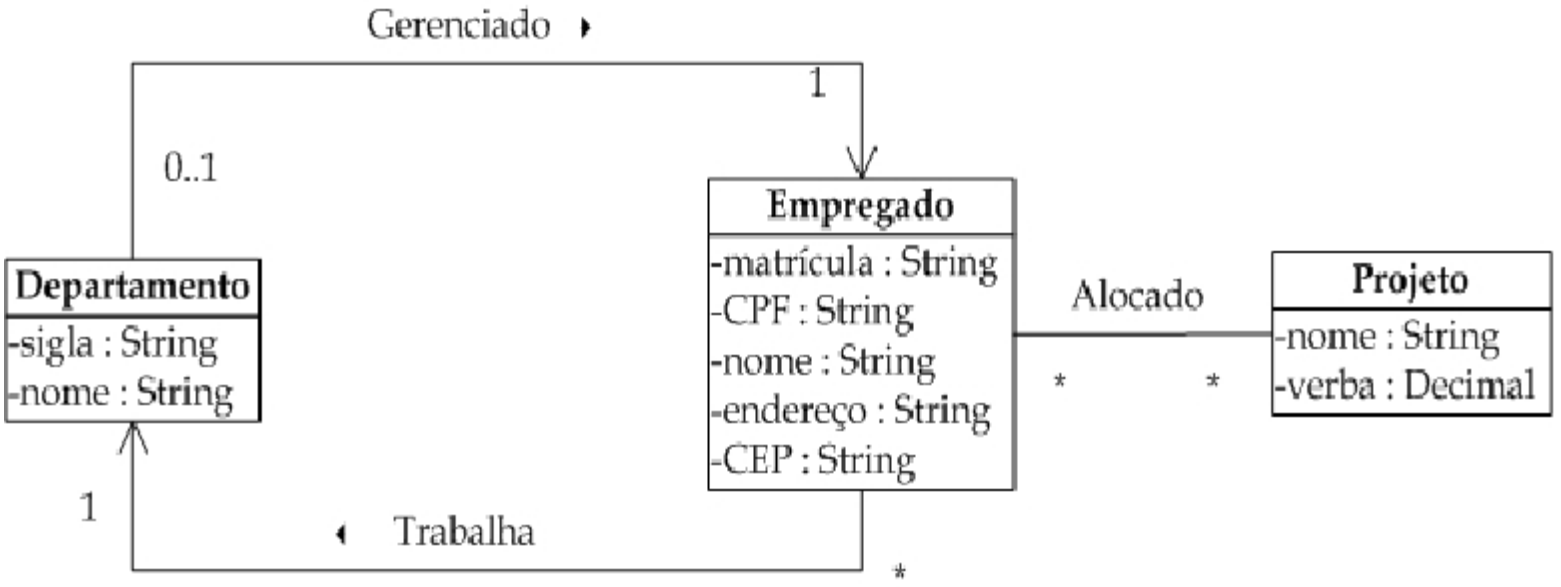
Tabela

idCliente	nome	endereço

```
CREATE TABLE Cliente
(  idCliente NUMBER(30) NOT NULL,
   NomeCliente VARCHAR (50) NOT NULL,
   EndereçoCliente VARCHAR (50) NOT NULL,
   CONSTRAINT pk_cliente PRIMARY KEY(idCliente));
```

Mapeamento de associações

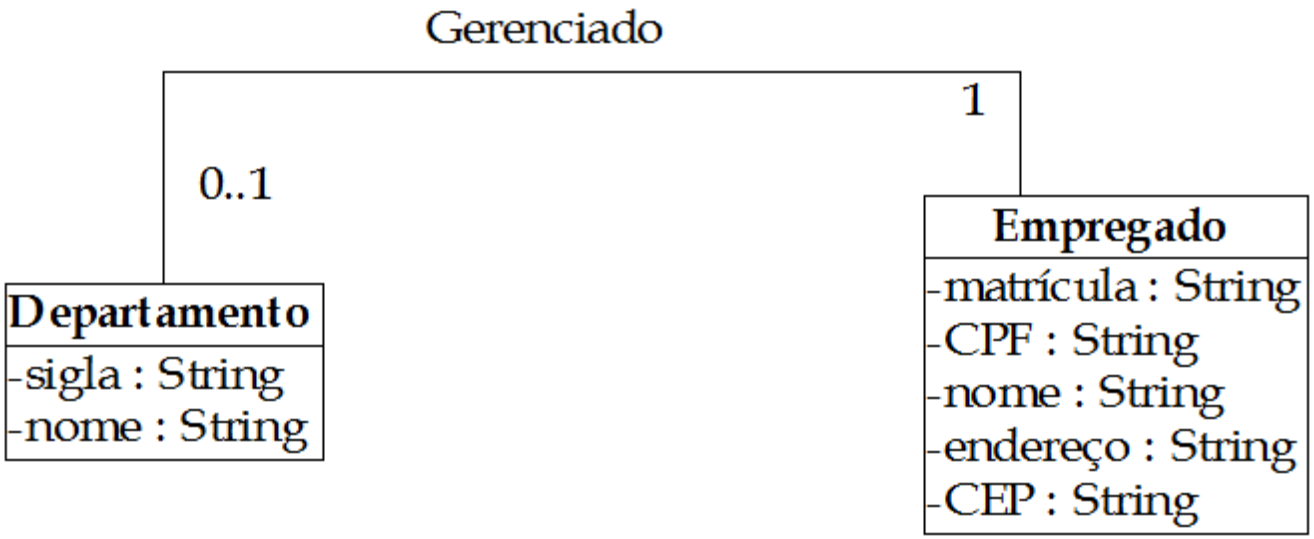
- O procedimento utiliza o conceito de ***chave estrangeira .***
- Há três casos, cada um correspondente a um tipo de ***conectividade.***
- Nos exemplos dados a seguir, considere o seguinte diagrama de classes:



Mapeamento de associações 1..1

- Raramente acontecem
- Deve-se adicionar uma chave estrangeira em uma das duas tabelas.

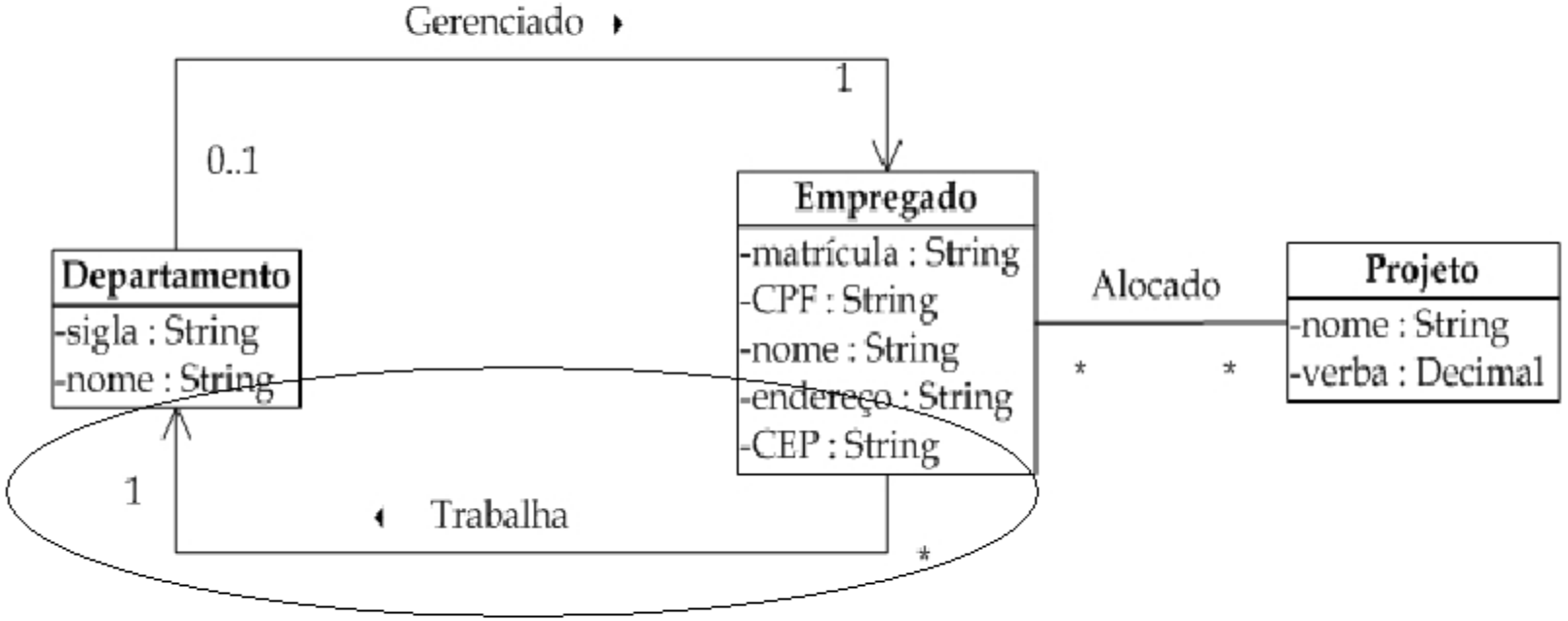
Exemplo de mapeamento de associação 1..1




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

Mapeamento de associações 1-muitos

- Adicionar uma chave estrangeira para referenciar a chave primária na parte "muitos".

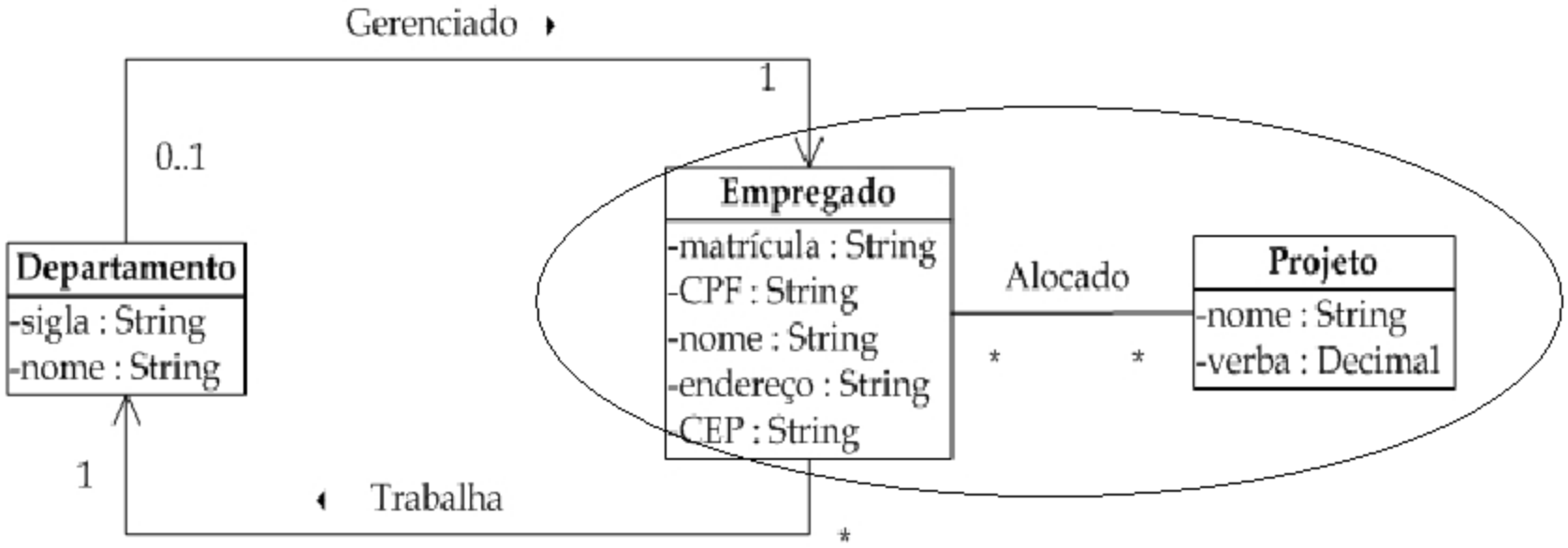


Empregado (**idEmpregado**, CPF, nome, endereço, CEP, idDepto)

Mapeamento de associações muitos-muitos

- Uma tabela ***de associação deve ser criada.***
- Alternativas para definir a chave primária.
  - definir uma ***chave primária composta, combinando as chaves primárias das tabelas.***
  - criar uma coluna de implementação que sirva como chave primária simples da relação de associação.

Mapeamento de associações muitos-muitos



Alocado (**idAlocado**, idProjeto, idEmpregado, nome, verba)

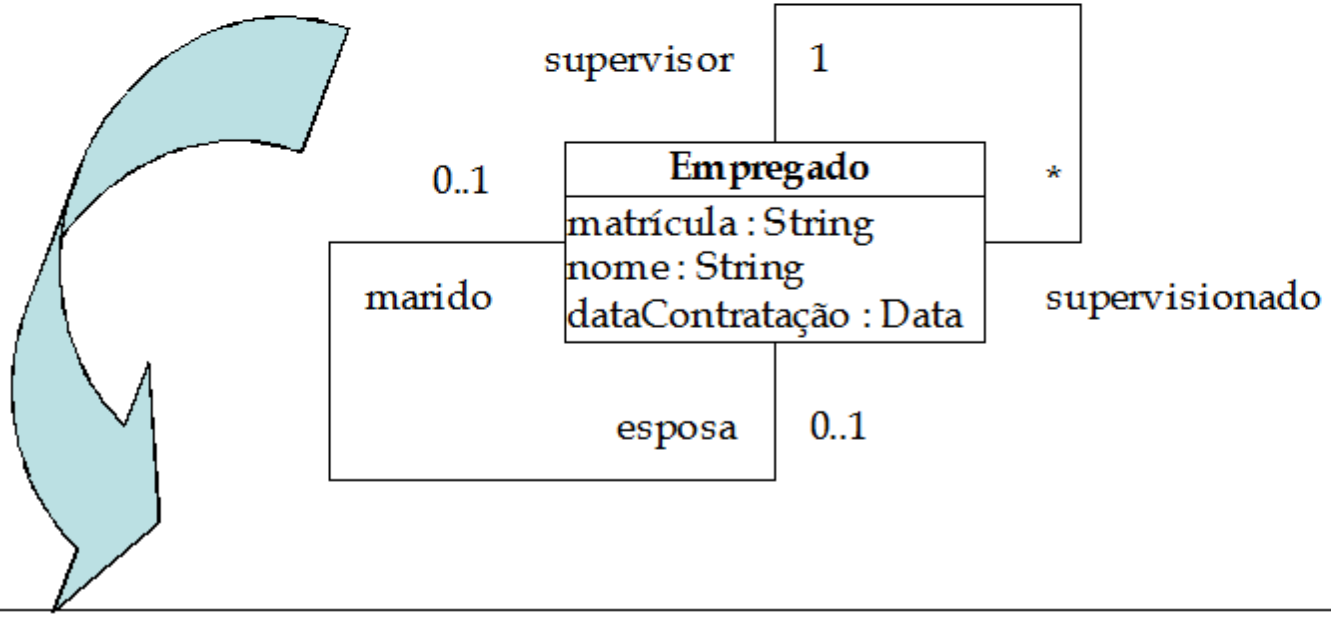
→ Alocado (**idProjeto**, **idEmpregado**, nome, verba)

Mapeamento de agregações

- Forma especial de associação → *mesmo procedimento para realizar o mapeamento de associações pode ser utilizado.*
- 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 (triggers) e procedimentos armazenados (stored procedures)* .

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 tabela de associação deve ser criada.

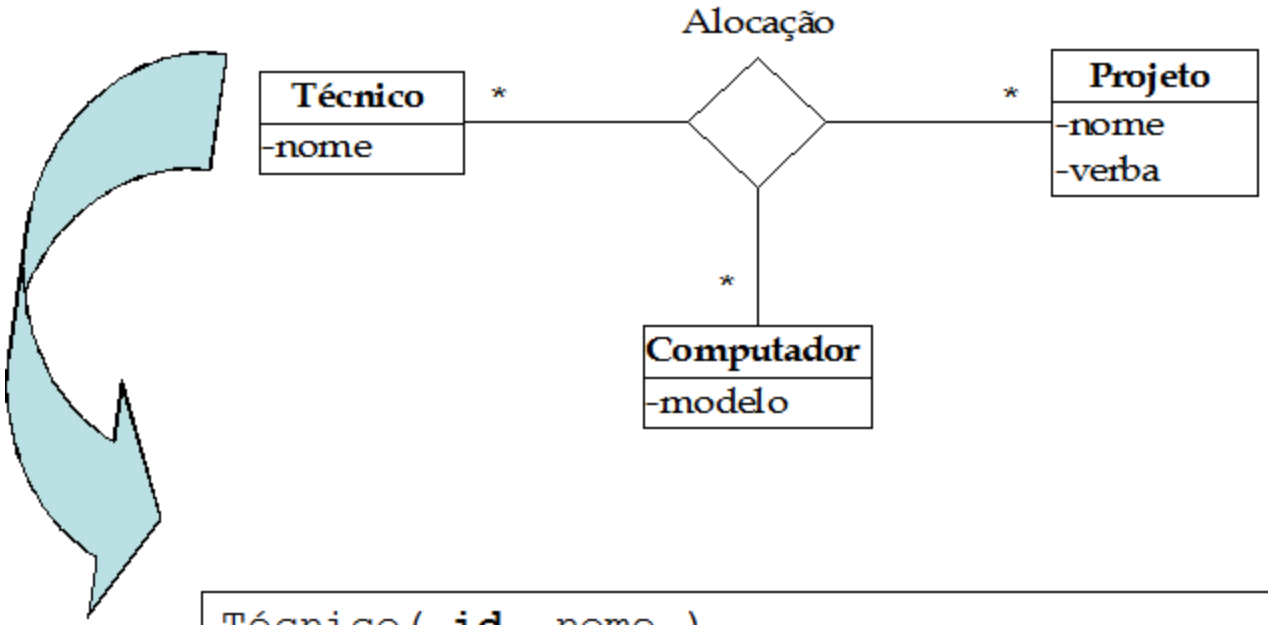


Empregado(**id**, matrícula, nome, dataContratação, idCônjunge, idSupervisor)

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 tabela de associação.

Mapeamento de associações n-árias

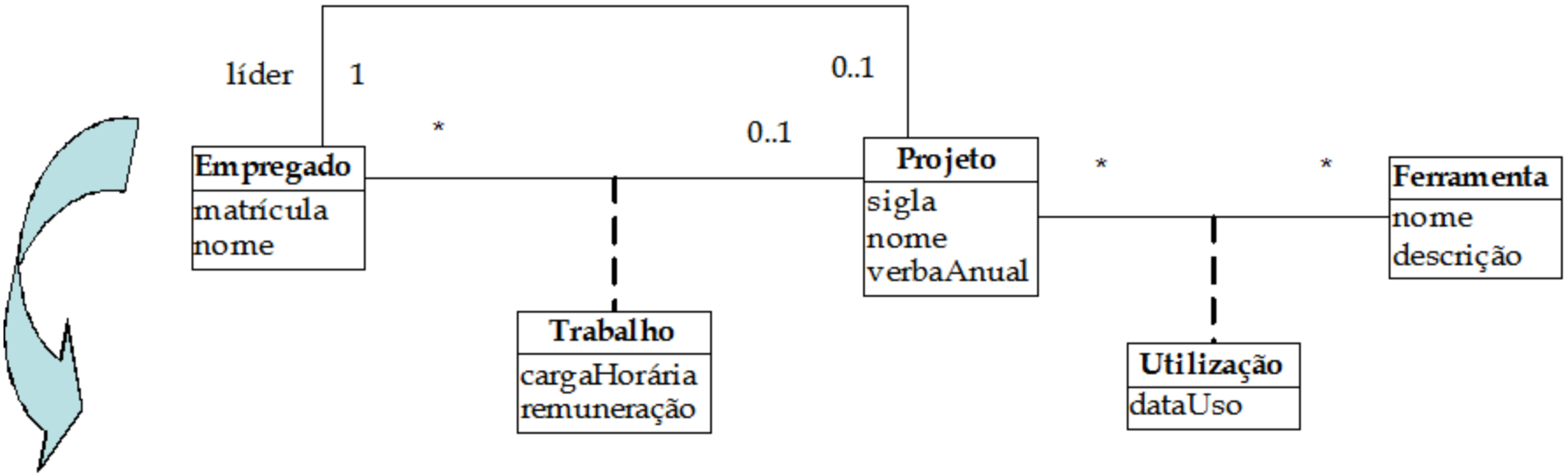



```
Técnico( id, nome )
Projeto( id, nome, verba )
Computador( id,  modelo )
Alocação( idProjeto, idTécnico, idComputador )
```

Mapeamento de classes associativas

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

Mapeamento de classes associativas

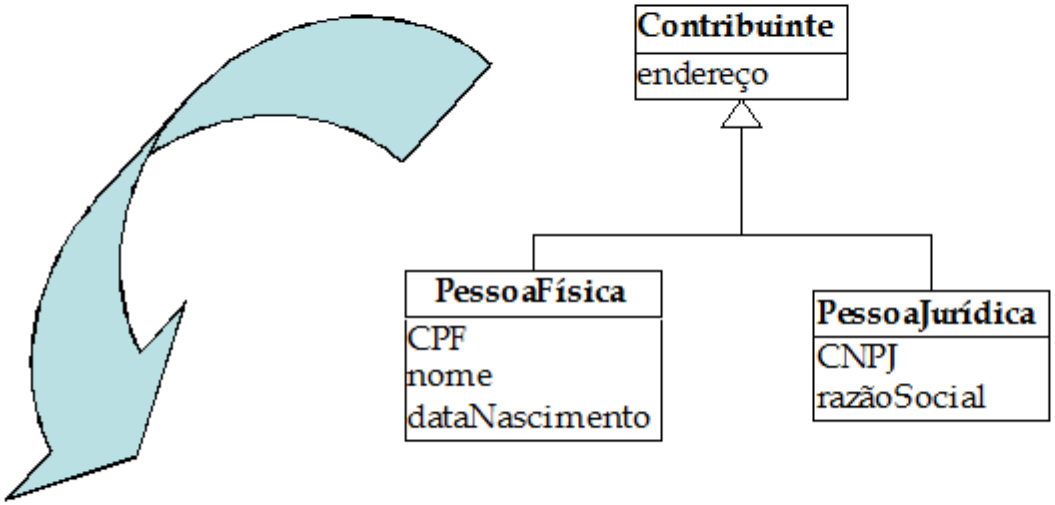



```
Empregado(id, matrícula, nome)
Projeto(id, sigla, nome, verbaAnual, idEmpregadoLíder)
Ferramenta(id, nome, descrição)
Utilização(idFerramenta, idProjeto, dataUso )
Trabalho(idEmpregado, idProjeto, cargaHorária, remuneração)
```

Mapeamento de Herança

- Três formas alternativas de mapeamento:
  - Uma tabela para cada classe da hierarquia
  - Uma tabela para toda a hierarquia
  - Uma tabela 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 do software sendo desenvolvido.
  - A equipe de desenvolvimento pode decidir implementar mais de uma alternativa.

Mapeamento de herança




```
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 de herança

- A 1ª alternativa (uma relação para cada classe da hierarquia) é a que melhor reflete o modelo OO.
  - Desvantagem: desempenho da manipulação das relações.
    - Inserções, remoções e *junções.*

Mapeamento de herança

- 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

Mapeamento de herança

- 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 tabelas correspondentes as suas subclasses deve ser modificada.
  - Todas as tabelas correspondentes a subclasses devem ser modificadas quando a definição da superclasse é modificada.

Camada de Persistência

- Para isolar os objetos do negócio de detalhes de comunicação com o SGBD, uma ***camada de persistência pode ser utilizada.***
- O objetivo de uma camada de persistência é isolar os objetos do software de mudanças no mecanismo de armazenamento.
  - Se um SGBD diferente tiver que ser utilizado pelo sistema, somente a camada de persistência é modificada;
  - Os objetos da camada de negócio permanecem intactos.
- Diminuição do acoplamento entre os objetos e a estrutura do banco de dados torna o software mais *flexível e mais portável.*

Camada de persistência

- No entanto, as vantagens de uma camada de persistência não vêm de graça.
  - A intermediação feita por essa camada entre os objetos do domínio e o SGBD traz uma sobrecarga de processamento .
  - A camada de persistência pode aumentar a complexidade computacional da realização de operações que seriam triviais com o uso direto de SQL.
- As vantagens adquiridas pela utilização de uma camada de software, principalmente em sistemas complexos , geralmente compensam a perda no desempenho e a dificuldade de implementação.

Estratégias de persistência

- Há diversas estratégias que podem ser utilizadas para definir a camada de persistência de um software:
  - Acesso direto ao banco de dados
  - Uso de um SGBDOO ou de um SGBDOR
  - Uso do padrão DAO (*Data Access Object)*
  - Uso de um frameworks

Acesso direto

- Uma estratégia simples para o mapeamento objeto-relacional é cada objeto persistente possuir comportamento que permita a sua restauração, atualização ou remoção.
  - Há código escrito em SQL para realizar a inserção, remoção, atualização e consulta das tabelas onde estão armazenados os objetos.
- Essa solução é de fácil implementação em Linguagens de quarta geração, como o Visual Basic, e o Delphi.
- Essa estratégia de mapeamento objeto-relacional é justificável para sistemas simples.

Acesso direto - desvantagens

- Classes relativas à lógica do negócio ficam muito acopladas às classes relativas à interface gráfica e ao acesso ao banco de dados.
- Mais complicado migrar o software de um SGBD para outro.
- A lógica da aplicação fica desprotegida de eventuais modificações na estrutura do banco de dados.
- A coesão das classes diminui
  - cada classe deve possuir responsabilidades relativas ao armazenamento de seus objetos, além de ter responsabilidades inerentes ao negócio.
- Dificuldades de manutenção e extensão do código fonte praticamente proíbe a utilização desta estratégia em sistemas complexos.

Uso de SGBDOO ou SGBDOR

- Um SGBDOO permite a definição de estruturas de dados arbitrariamente complexas (classes) no SGBDOO.
- Nesse modelo, atributos de um objeto podem conter valores de tipos de dados estruturados
- No modelo relacional, as tabelas só armazenam itens atômicos.
- É possível definir hierarquias de herança entre classes.
- A linguagem de consulta para SGBDOO, OQL (Object Query Language), permite consultar e manipular objetos armazenados em um banco de dados.
  - Também possui extensões para identidade de objetos, objetos complexos, chamada de operações e herança.

Uso de SGBDOO ou SGBDOR

- Os principais SGBDR começaram a incorporar características de orientação a objetos.
- Esses SGBD passaram a adotar o ***modelo de dados objeto-relacional, (extensão do modelo relacional), onde são adicionadas características da orientação a objetos.***
- Hoje em dia os principais SGBD são ***sistemas de gerência de bancos de dados objeto-relacionais (SGBDOR).***

Uso do padrão DAO

- O padrão DAO é uma forma de desacoplar as classes do negócio dos aspectos relativos ao acesso ao armazenamento persistente.
  - DAO: *Data Access Object (Objeto de Acesso a Dados).*
- Nessa estratégia, um software obtém acesso a objetos de negócio através de uma interface, a chamada ***interface DAO.***
- O software interage com o ***objeto DAO através de uma interface.***
- O objeto DAO isola completamente os seus clientes das particularidades do mecanismo de armazenamento (fonte de dados) sendo utilizado.
  - Ex. JDBC, ODBC, ADO.NET