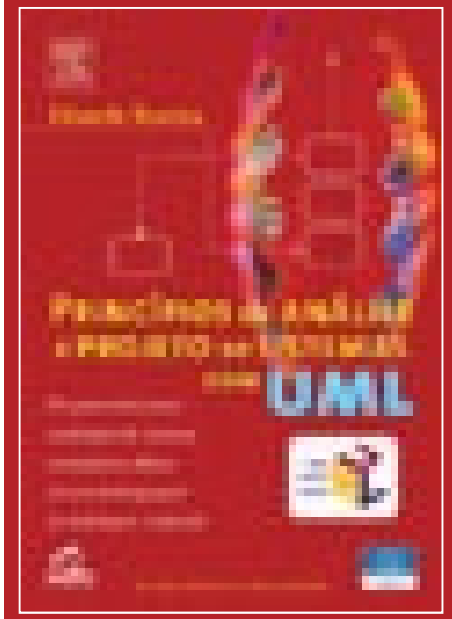
Princípios de Análise e Projeto de Sistemas com UML

2ª edição



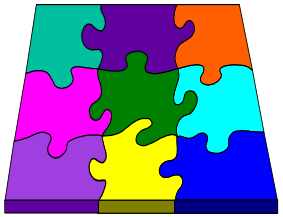
Eduardo Bezerra

Editora Campus/Elsevier

Capítulo 12
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

Tópicos



- Introdução
- Projeto de banco de dados
- Construção da camada de persistência

Introdução

- 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.

Introdução

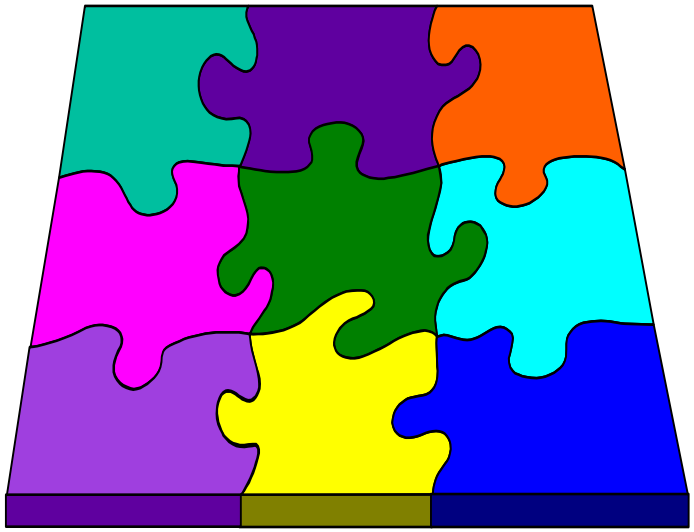
- Os objetos de um sistema podem ser classificados em persistentes e transientes.
- ***Objetos transientes****:* existem somente na memória principal.
  - Objetos de controle e objetos de fronteira.
- ***Objetos persistentes***: têm uma existência que perdura durante várias execuções do sistema.
  -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

Introdução

- 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.

Introdução

- Para objetos persistentes, surge o problema de conciliar as informações representadas pelo estado de um objeto e pelos dados armazenados em registros de uma tabela.
- O ***descasamento de informações*** (*impedance mismatch*) 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 desenvolvedor deve dar a este problema.



12.1 Projeto de banco de dados

Projeto de banco de dados

- Uma das primeiras atividades do projeto detalhado de um SSOO é o desenvolvimento do banco de dados a ser utilizado, se este não existir.
- Essa atividade corresponde ao ***projeto do banco de dados***.
- As principais tarefas no projeto do banco de dados são:
  - Construção do esquema do banco de dados
  - Criação de índices
  - Armazenamento físico dos dados
  - Definição de visões sobre os dados armazenados.
  - Atribuição de direitos de acesso
  - Políticas de backup dos dados

Projeto de banco de dados

- Restrição de escopo: apenas consideramos o aspecto de mapeamento de informações entre os modelos OO e relacional.
  - Ou seja, o mapeamento do modelo de classes para o modelo relacional.
  - Esse mapeamento possibilita a criação do ***esquema do banco de dados***.
- Atualmente, há diversas ferramentas que automatiza grande parte desse mapeamento (engenharias direta e reversa).
  - Mas, nem sempre uma ferramenta está disponível.
  - Mesmo na existência de uma ferramenta, é importante para o desenvolvedor um conhecimento básico dos procedimentos do mapeamento.

Conceitos do modelo relacional

- O modelo relacional é fundamentado no conceito de ***relação***.
- Cada coluna de uma relação pode conter apenas ***valores atômicos***.
- Uma ***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.

Conceitos do modelo relacional

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

Alocação		
id	**idProjeto**	**idEmpregado**
100	1	1
101	1	2
102	2	1
103	3	5
104	4	2

Projeto		
id	**nome**	**verba**
1	PNADO	R$ 7.000
2	BMMO	R$ 3.000
3	SGILM	R$ 6.000
4	ACME	R$ 8.000

Conceitos do modelo relacional

Empregado						
id	**matrícula**	**CPF**	**nome**	**endereço**	**CEP**	**idDepartamento**
1	10223	038488847-89	Carlos	Rua 24 de Maio,40	22740-002	13
2	10490	024488847-67	Marcelo	Rua do Bispo, 1000	22733-000	13
3	10377	NULL	Adelci	Av. Rio Branco, 09	NULL	NULL
4	11057	0345868378-20	Roberto	Av. Apiacás, 50	NULL	14
5	10922	NULL	Aline	R. Uruguaiana, 50	NULL	14
6	11345	0254647888-67	Marcelo	NULL	NULL	15

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 freqüentemente confundidos.
  - O MER é um modelo de dados; o modelo de classes representa objetos (dados e comportamento).

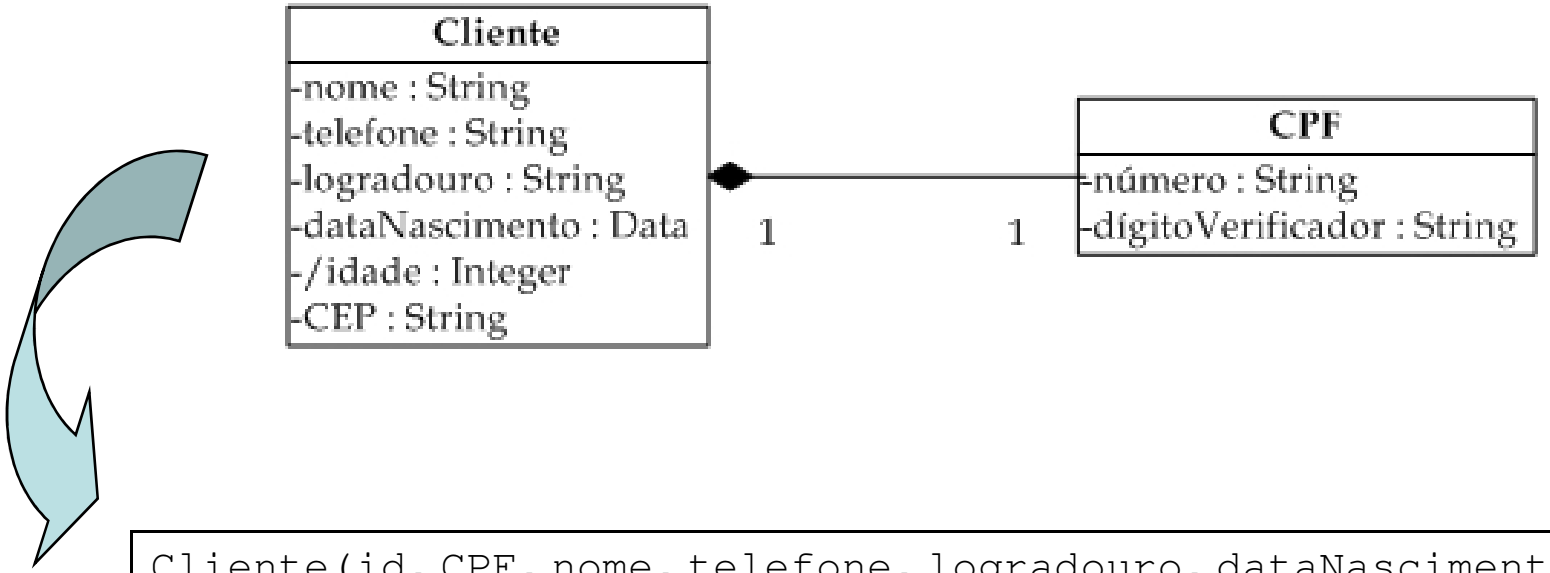
Mapeamento de objetos para o modelo relacional

- 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.

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).

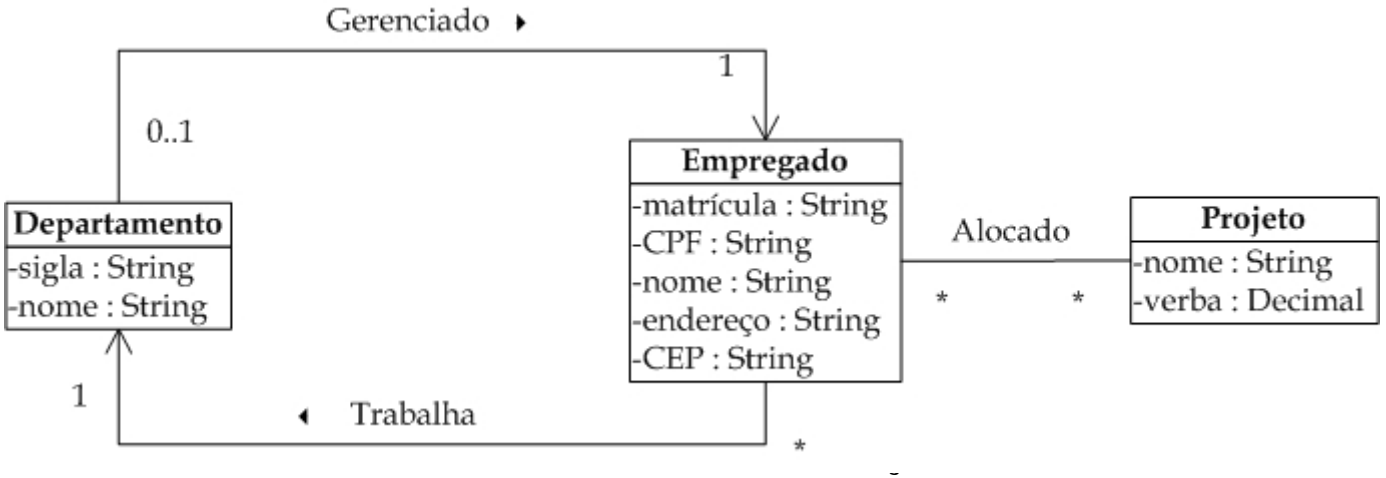
Mapeamento de classes e seus atributos



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

Mapeamento de associações

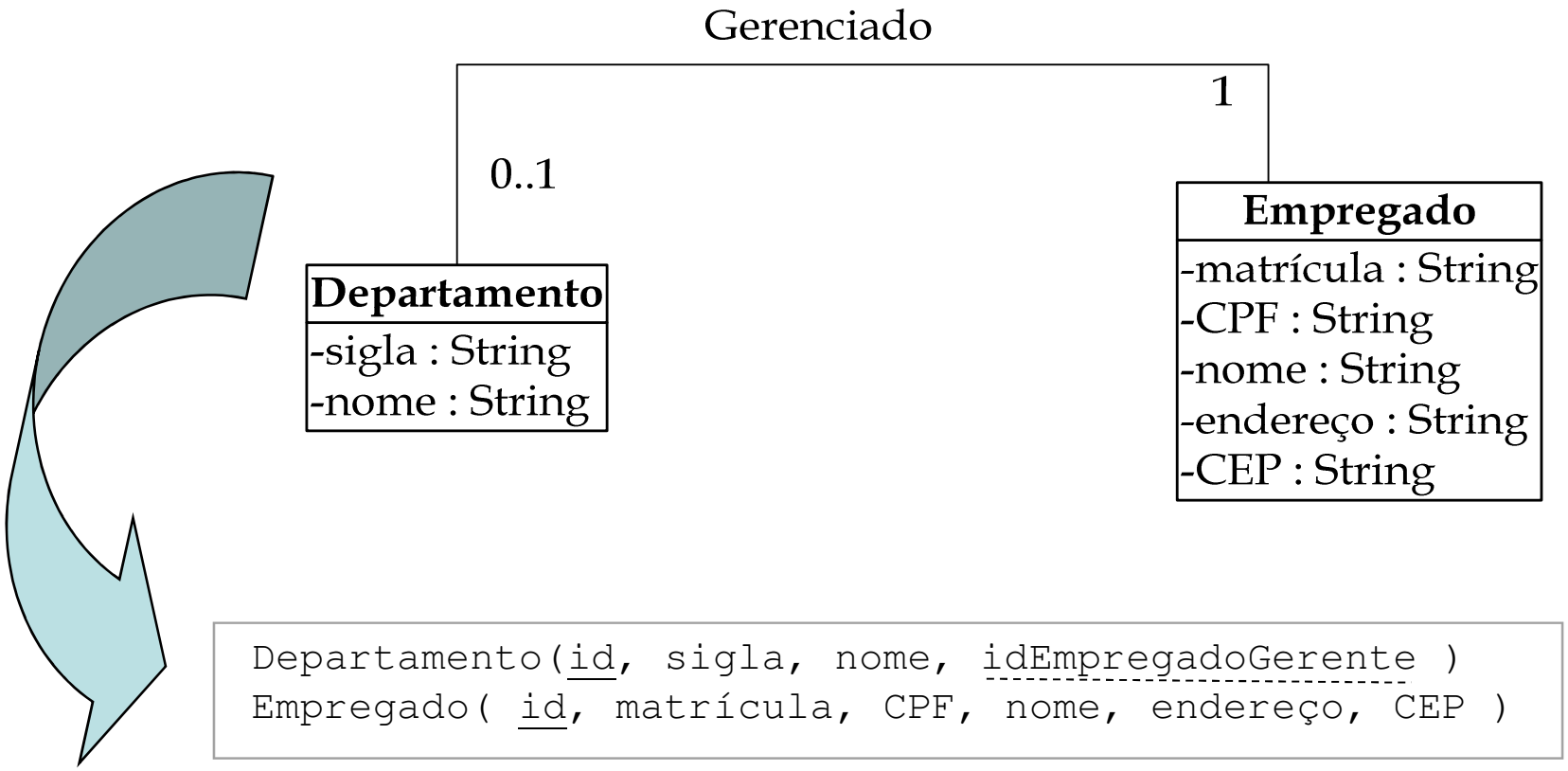
- O procedimento utiliza o conceito de ***chave estrangeira***.
- Há três casos, cada um correspondente a um tipo de ***conectividade***.
- Nos exemplos dados a seguir, considere, sem perda de generalidade, que:
  - há uma associação entre objetos de duas classes, Ca e Cb.
  - Ca e Cb foram mapeadas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



Mapeamento de 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 acerca da conectividade:
  - Obrigatória em ambos os extremos.
  - Opcional em ambos os extremos.
  - Obrigatória em um extremo e opcional no outro extremo.

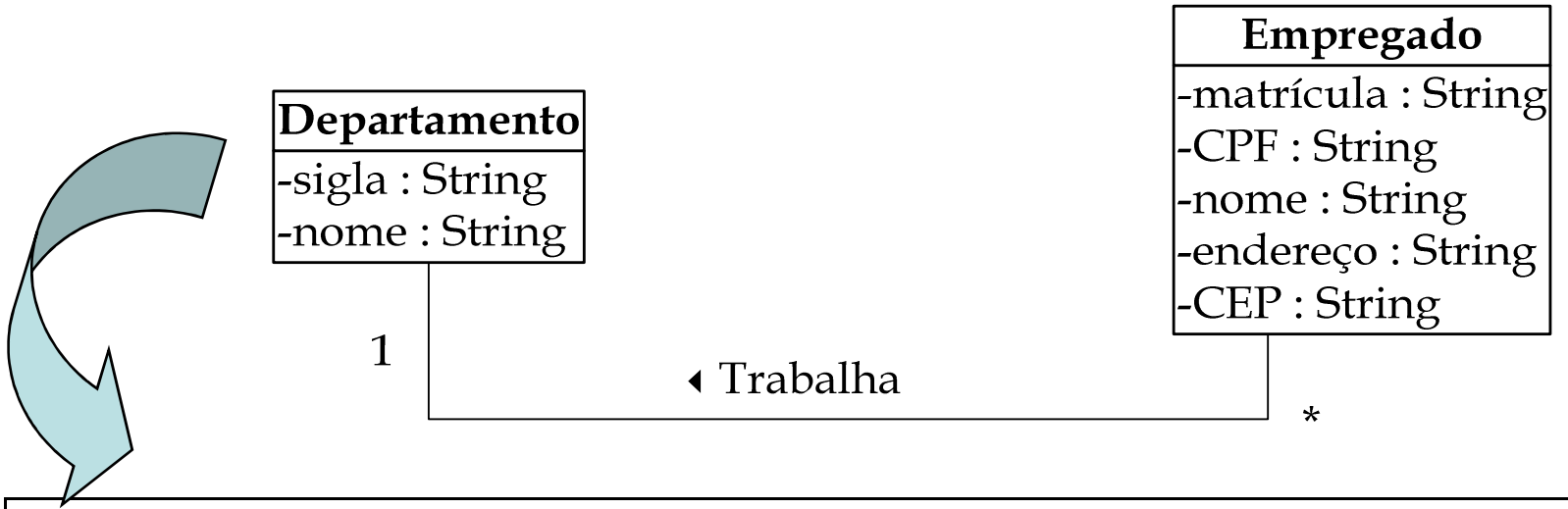
Mapeamento de associações 1:1




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

Mapeamento de associações 1-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 T*a* para referenciar a chave primária de Tb.

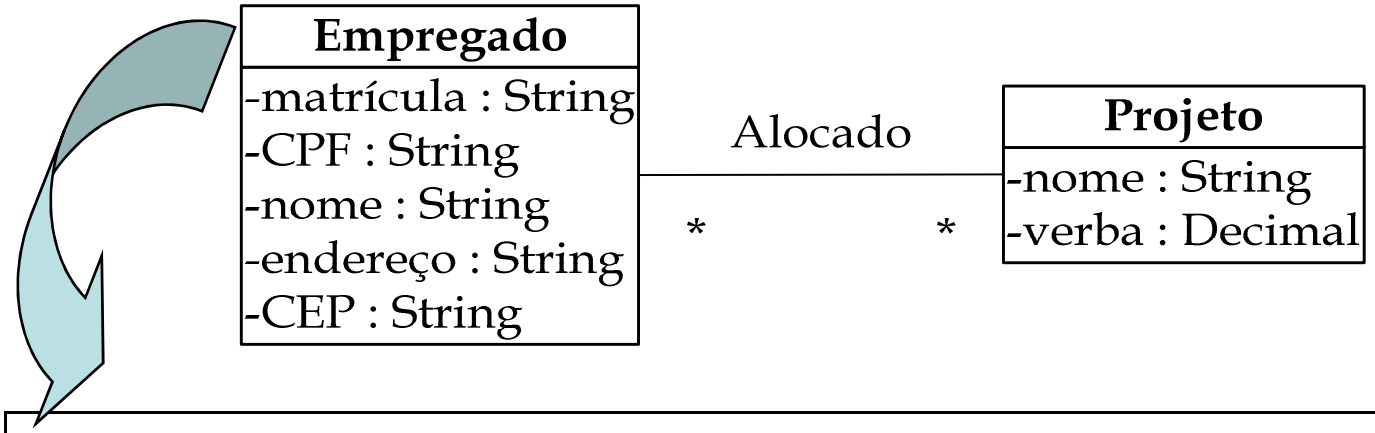



```
Departamento( id, sigla, nome, idEmpregadoGerente )
Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)
```

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.
- 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 de associações muitos-muitos




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

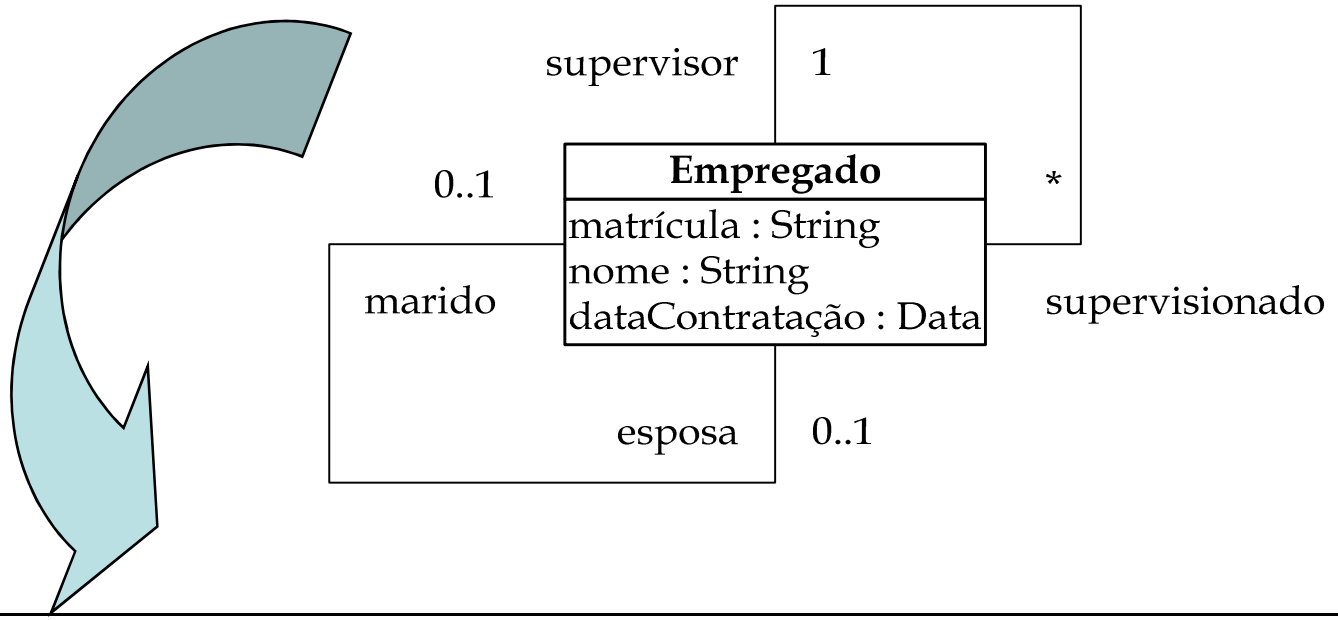
```
Departamento(id, sigla, nome, idEmpregadoGerente)
Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)
Alocação(id, idProjeto, idEmpregado)
Projeto(id, nome, verba)
```

Mapeamento de 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*.
- 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 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.

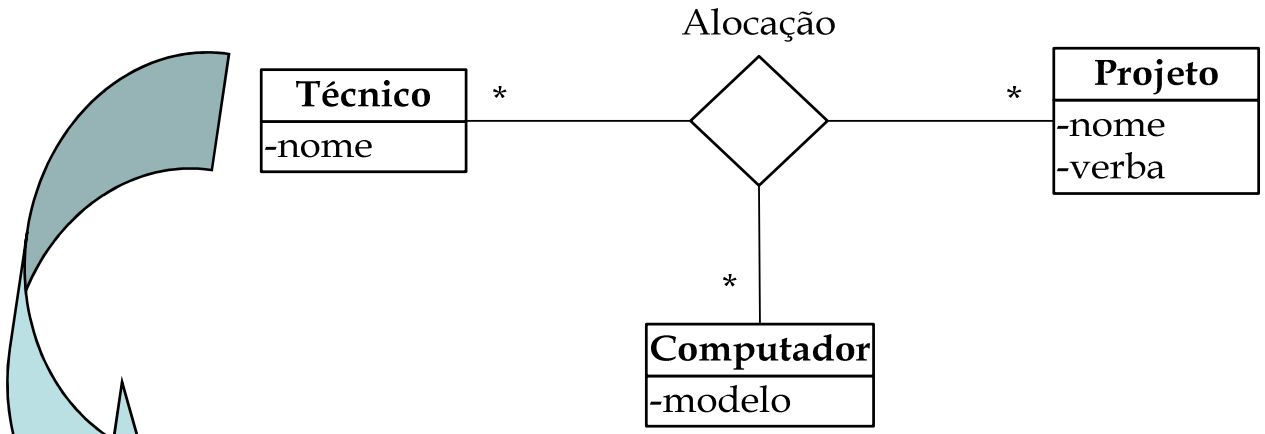


`Empregado(id, matrícula, nome, dataContratação, idCônjunge, idSupervisor)`

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.

Mapeamento de associações n-árias

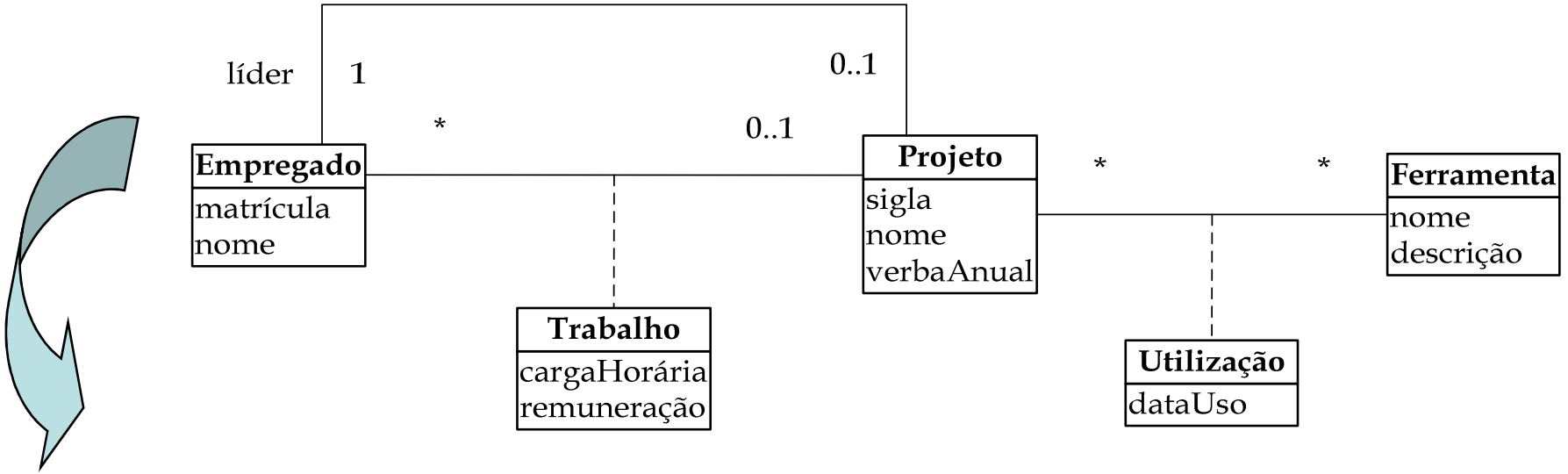



```
Técnico( id, nome )
Projeto( id, nome, verba )
Computador( id,  modelo )
Alocação( id, idProjeto, idTécnico, idComputador )
```

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.

Mapeamento de classes associativas

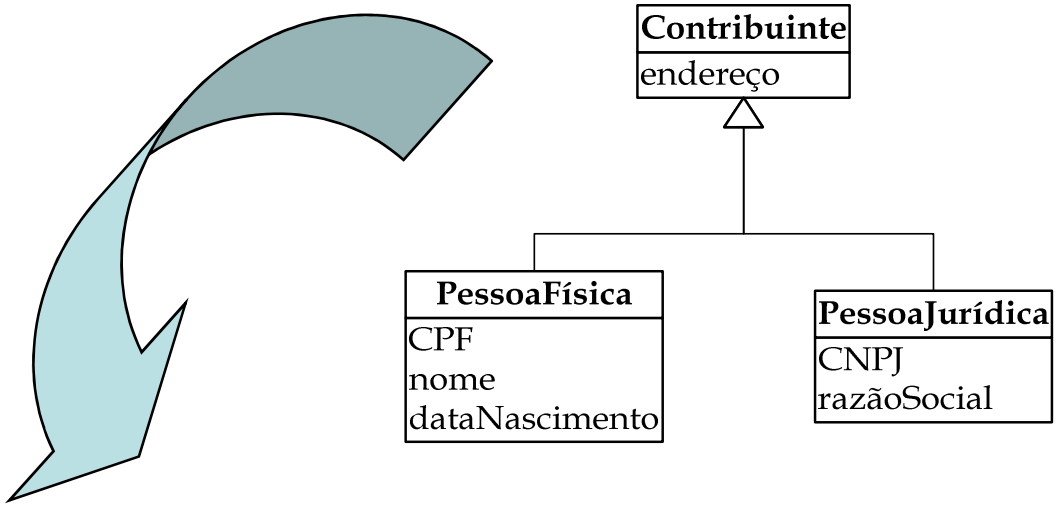



```
Empregado(id, matrícula, nome)
Projeto(id, sigla, nome, verbaAnual, idEmpregadoLíder)
Ferramenta(id, nome, descrição)
Utilização(id, idFerramenta, idProjeto, dataUso )
Trabalho(id, idEmpregado, idProjeto, cargaHorária, remuneração)
```

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.

Mapeamento de generalizações



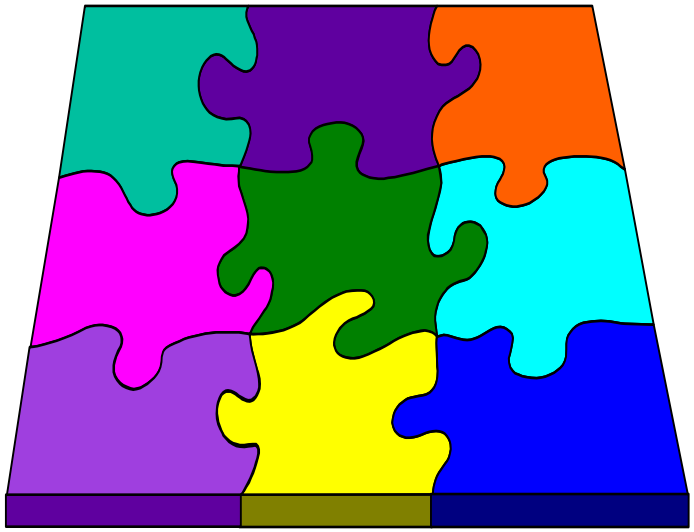
Contribuinte(id, endereço) PessoaFísica(id, nome, dataNascimento, CPF, idContribuinte) PessoaJurídica(id, CNPJ, razãoSocial, idContribuinte)
Pessoa(id, nome, endereço, dataNascimento, CPF, CNPJ, razãoSocial, tipo)
PessoaFísica(id, dataNascimento, nome, endereço, CPF) PessoaJurídica(id, CNPJ, endereço,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

Mapeamento 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*.
- 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 de 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.



12.2 Construção da camada de persistência

Camada de persistência

- Além da construção do esquema de banco de dados, outros aspectos importantes e relativos ao armazenamento de objetos em um SGBDR devem ser definidos.
- Alguns desses aspectos são enumerados a seguir.
  - ***Materialização***: restaurar um objeto a partir do banco de dados, quando necessário.
  - ***Atualização***: enviar modificações sobre um objeto para o banco de dados.
  - ***Remoção***: remover um objeto do armazenamento persistente.
- Esses aspectos estão relacionados a funcionalidades que implementam o transporte de objetos da memória principal alocada ao SSOO para um SGBD e vice-versa.

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 SSOO de mudanças no mecanismo de armazenamento.
  - Se um SGBD diferente tiver que ser utilizado pelo sistema , por exemplo, somente a camada de persistência é modificada;
  - Os objetos da camada de negócio permanecem intactos.
- Aa diminuição do acoplamento entre os objetos e a estrutura do banco de dados torna o SSOO 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.
  - Outra desvantagem é que a camada de persistência pode aumentar a complexidade computacional da realização de certas operações, que seriam triviais com o uso direto de SQL.
- Entretanto, 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 SSOO:
  - Acesso direto ao banco de dados
  - Uso de um SGBDOO ou de um SGBDOR
  - Uso do padrão DAO (*Data Access Object*)
  - Uso de um framework ORM

Acesso direto

- Uma estratégia simples para o mapeamento objeto-relacional é fazer com que cada objeto persistente possua 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, o PowerBuilder e o Delphi.
  - Uso de controles *data aware*.
- Essa estratégia de mapeamento objeto-relacional é justificável para sistemas simples.

Acesso direto

- No entanto, a solução de acesso direto apresenta algumas desvantagens para sistemas mais complexos.
  - Classes relativas à lógica do negócio ficam muito acopladas às classes relativas à interface gráfica e ao acesso ao banco de dados.
    - Por vezes, essas classes sequer são criadas.
  - Mais complicado migrar o SSOO 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, porque cada classe deve possuir responsabilidades relativas ao armazenamento e materialização 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

- Na metade dos anos 1980, começou-se a falar em um novo modelo para SGBDs, o orientado a objetos.
- Nesse modelo, em vez de tabelas, os conceitos principais eram classes e objetos.
- No início da década de 1990, foram criados alguns produtos comerciais de ***sistemas de gerência de bancos de dados orientados a objetos*** (SGBDOO).
  - ORION (MOC), OPENOODB (Texas Instruments), Iris (HP), GEMSTONE (GEMSTONE Systems), ONTOS (Ontos), Objectivity (Objectivity Inc.), ARDENT (ARDENT software), POET (POET Software).

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, diferente do modelo relacional, onde as tabelas só podem armazenar itens atômicos.
- Também 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, expressões de caminho, chamada de operações e herança.

Uso de SGBDOO ou SGBDOR

- Algumas pessoas pensavam que a tecnologia de SGBDOO suplantaria a velha tecnologia relacional.
- No entanto, os principais SGBDR começaram a incorporar características de orientação a objetos.
- Esses SGBD passaram a adotar o ***modelo de dados objeto-relacional***, que é uma 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).
  - Um SGBDOR é também conhecido por SGBD relacional-estendido.
- Exemplos: Oracle 9iTM e o DB2 Universal ServerTM.

Uso de SGBDOO ou SGBDOR

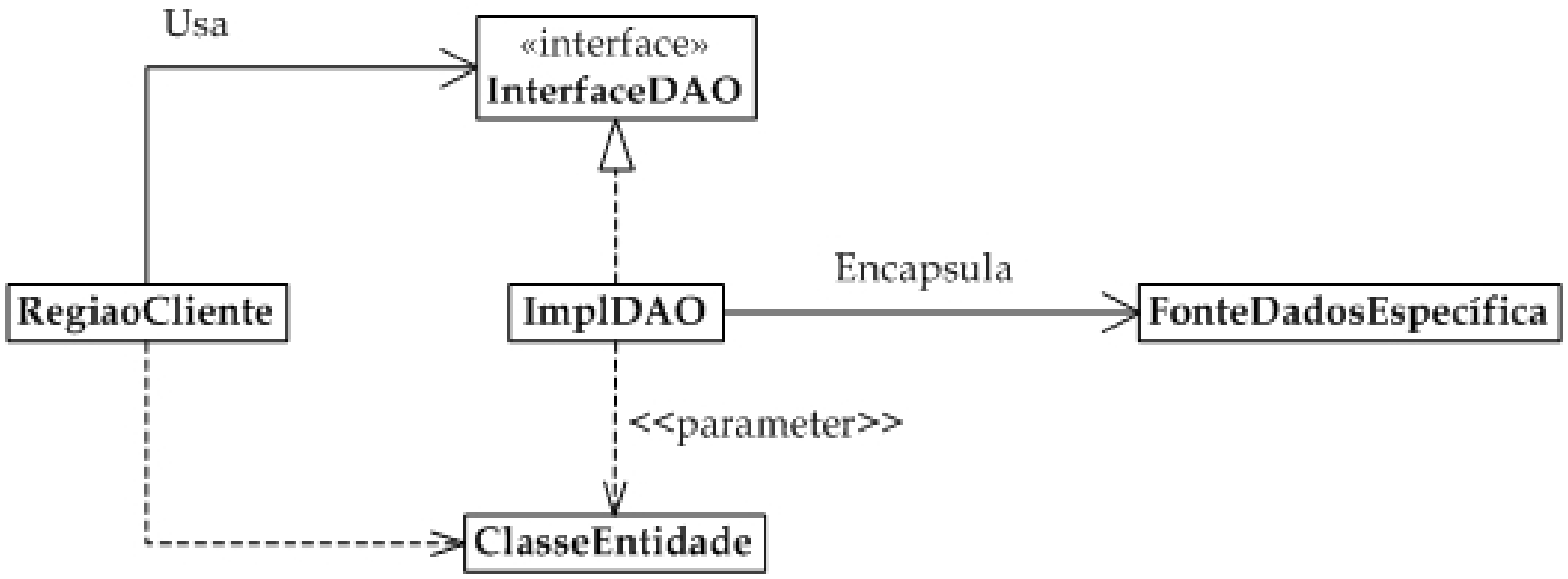
- Os modelos de dados usados por SGBDOR e SGBDOO são mais adequados para realizar o mapeamento de objetos.
- Mas, o fato é que existe uma plataforma imensa de sistemas que usam o modelo relacional puro.
  - De fato, existe uma grande resistência em substituir esses sistemas.
- Isso leva a crer que o mapeamento de objetos para o modelo relacional ainda irá durar por muitos anos.

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 SSOO obtém acesso a objetos de negócio através de uma interface, a chamada ***interface DAO***.
  - Classes que implementam essa interface transformam informações provenientes do mecanismo de armazenamento em objetos de negócio, e vice-versa.
- O SSOO interage com o ***objeto DAO*** através de uma interface.
  - A implementação desse objeto simplesmente não faz diferença para a aplicação.
  - O objeto DAO isola completamente os seus clientes das particularidades do mecanismo de armazenamento (fonte de dados) sendo utilizado.

Uso do padrão DAO

- Estrutura do padrão DAO



Uso do padrão DAO

- Exemplo em linguagem Java de uma InterfaceDAO

```
public interface AlunoDAO
{
  public void inserir(Aluno aluno)
   throws AlunoDAOException;

  public void atualizar(Aluno aluno)
   throws AlunoDAOException;

  public void remover(Aluno aluno)
   throws AlunoDAOException;

  public List<Aluno> encontrarTodos() throws AlunoDAOException;

  public Aluno encontrarPorMatricula(String matricula) throws AlunoDAOException;

  public List<Aluno> encontrarPorTurma(int idTurma) throws AlunoDAOException;

  public HistoricoEscolar obterHistoricoEscolar() throws AlunoDAOException;
}
```

Uso de um framework ORM

- Um framework ORM é um conjunto de classes que realiza o mapeamento objeto-relacional de forma transparente.
  - ORM: *Object-Relational Mapping* (mapeamento objeto-relacional).
- Os frameworks ORM tentam resolver o problema do mapeamento objeto relacional através de classes que o realizam de forma transparente.
- Normalmente, um framework ORM demanda a definição da correspondência entre a estrutura de objetos da aplicação e o esquema relacional do banco de dados.
  - Essa correspondência é fornecida através de um arquivo de configuração, denominado ***arquivo de mapeamento***.
  - De posse dessa correspondência, o framework está apto a mapear qualquer requisição por uma informação armazenada no SGBDR.