

# Sumário

1. Introdução à Aplicações Não-Convencionais
2. Revisão e Dicas de Modelagem Conceitual
- 3. BD Orientado a Objetos (BDOO)**
4. BD Temporal (BDT)
5. BD Geográfico (BDG)
6. XML & BD

# BD Orientado a Objetos

- **BDOO**: paradigma OO + SGBD
  - orientação a objetos
    - encapsulamento: objeto “encapsula” uma *estrutura* (atributos) e um *comportamento* (métodos)
      - vantagem: cada aplicação com sua interface (“visão”) particular dos dados
    - reusabilidade: novos objetos podem aproveitar propriedades já definidas em outros objetos
      - vantagem: redefinições de dados são evitadas
  - SGBD
    - gerenciamento eficiente de dados operacionais (persistentes)
      - acesso otimizado e concorrente; segurança; integridade; ...
- **SGBDOO**: gerenciamento de objetos persistentes

# Modelo de Dados OO

- BDR
  - modelo formalmente definido e com um conjunto fixo de conceitos
- BDOO
  - falta de consenso sobre um padrão (conjunto de conceitos)
    - SGBDOOs com modelos heterogêneos
  - carência de uma base formal
    - início das pesquisas em BDOO
      - muita atividade experimental, voltada às necessidades das aplicações
    - tentativa de padronização: [ODMG](#)

# Modelo de Dados OO - Conceitos

1. Identidade de objeto (OID)
2. Métodos
3. Classes
4. Estruturas complexas
5. Herança
6. *Late Binding* (ligação tardia)

# Modelo de Dados OO - Conceitos

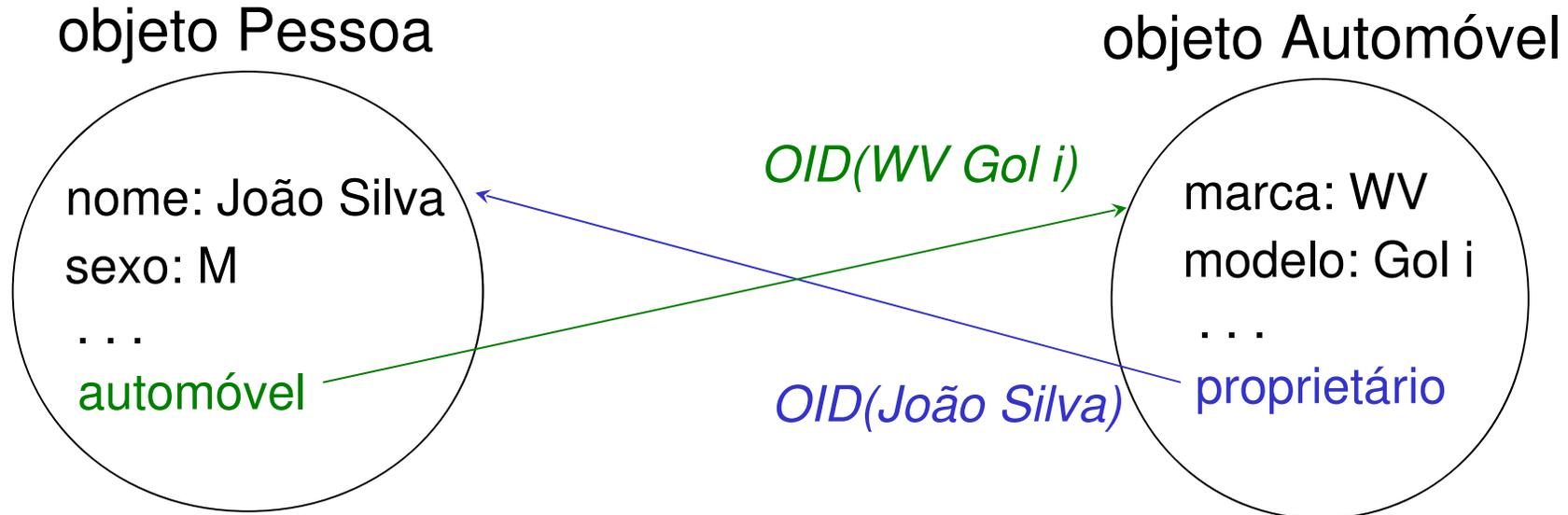
- 1. Identidade de objeto (OID)**
2. Métodos
3. Classes
4. Estruturas complexas
5. Herança
6. *Late Binding* (ligação tardia)

# Identidade do Objeto (OID)

- Identificador **único** de cada objeto
  - gerado pelo SGBDOO e imutável
  - visível ou não para o usuário
- Diferenças com relação a BDR
  - chave primária é passível de alteração
    - consistência de unicidade
    - consistência de integridade referencial
  - chave primária em alguns casos é um atributo artificial e visível ao usuário
    - atributo adicional sem muita semântica

# Relacionamentos entre Objetos

- Referências a OIDs

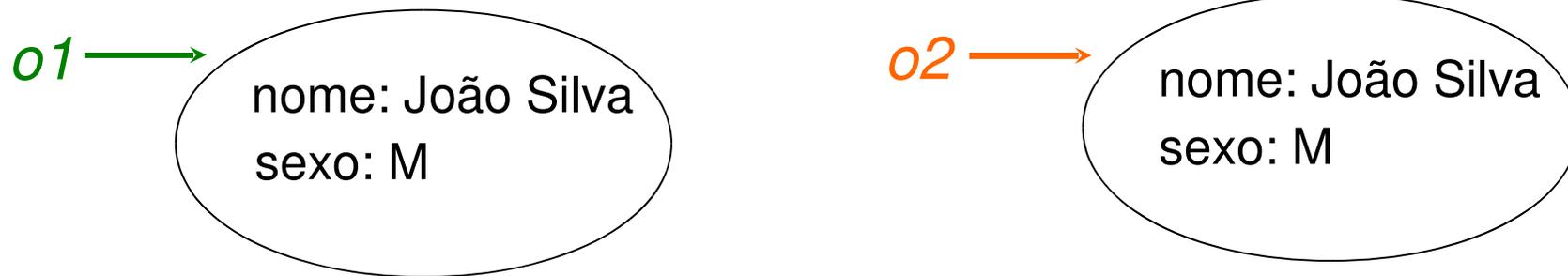


- Vantagem

- tipo do dado para referências é uniforme (OID)
  - evita consistência de tipo de dado

# Igualdade de Identidade

- Introduz dois tipos de comparação
  - igualdade de identidade ( $=$ ) (*para OIDs*)
  - igualdade de valor ( $= =$ ) (*para atributos*)



- $o1 = = o2$  (*verdadeiro!*)
  - $o1 = o2$  (?)
- Observação
  - OID não dispensa (não substitui) a definição de um identificador visível para o usuário

# Modelo de Dados OO - Conceitos

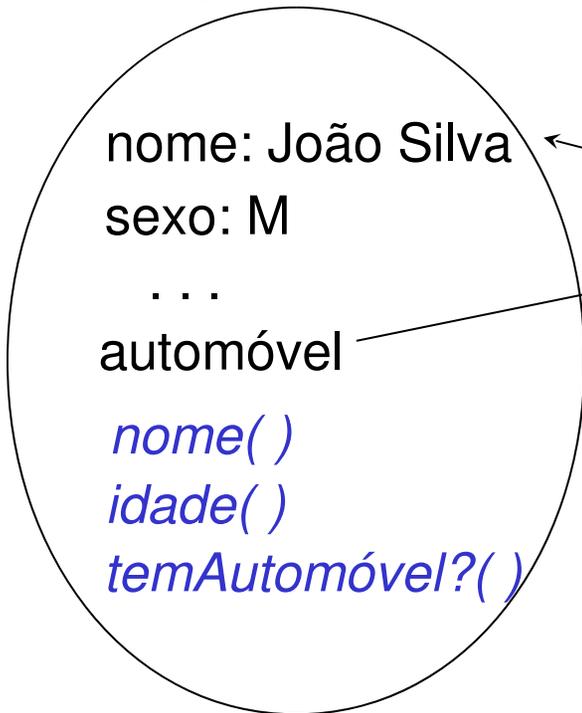
1. Identidade de objeto (OID)
2. **Métodos**
3. Classes
4. Estruturas complexas
5. Herança
6. *Late Binding* (ligação tardia)

# Métodos

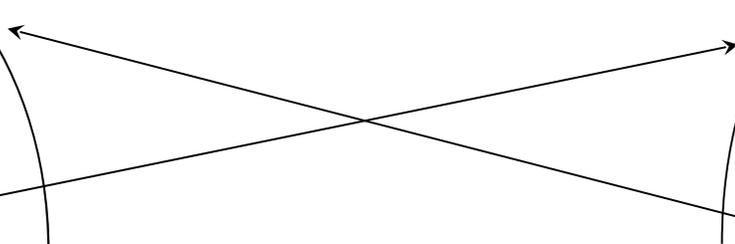
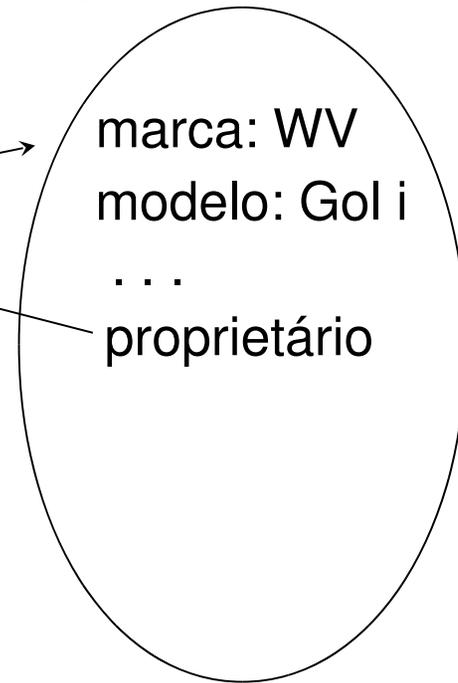
- Operações associadas a um objeto
  - comportamento de um objeto é mantido no BD
  - BDR mantém apenas a estrutura dos dados
- Composição
  - assinatura (*interface pública do objeto*)
  - implementação (*LPOO utilizada pelo BDOO*)
- Vantagem: encapsulamento de comportamento
  - simplifica o código das aplicações
  - cada aplicação acessa uma interface particular
    - autorizações de acesso e/ou visões podem ser aplicadas a nível de métodos
  - métodos podem servir para programação de RIs
    - BDOOs não possuem, em geral, linguagens sofisticadas para RIs, como *checks* e *triggers*

# Métodos

objeto Pessoa



objeto Automóvel

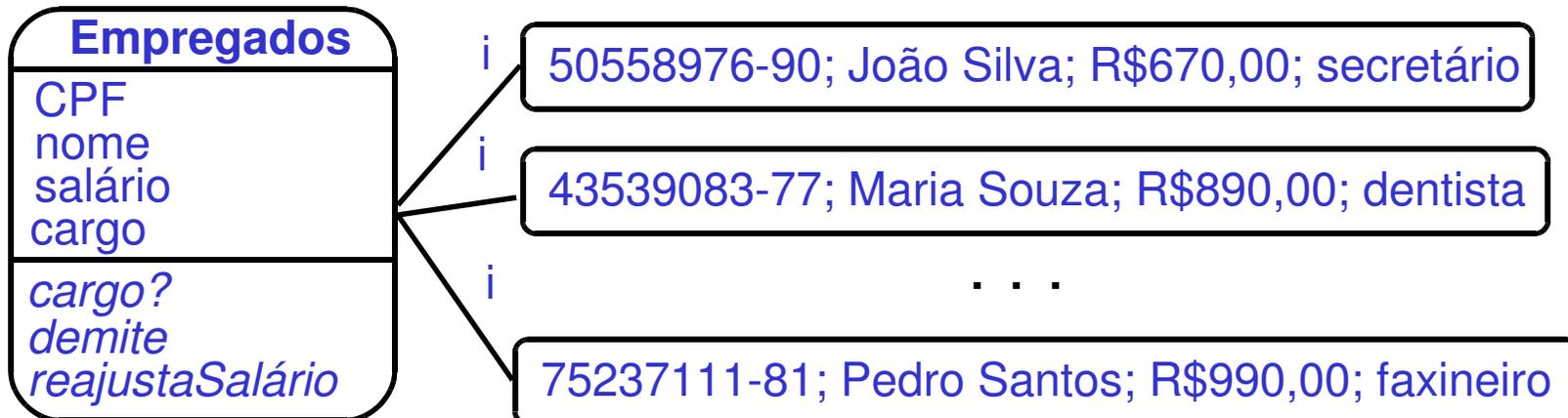


# Modelo de Dados OO - Conceitos

1. Identidade de objeto (OID)
2. Métodos
3. **Classes**
4. Estruturas complexas
5. Herança
6. *Late Binding* (ligação tardia)

# Classe

- Conjunto de objetos (instâncias) com a mesma estrutura e comportamento
- Base para a formulação de operações
  - função idêntica a de uma tabela em um BDR
- Vantagem: reusabilidade
  - instâncias de uma classe compartilham a mesma estrutura e implementação de métodos



# Tipo X Classe

- Ambos os conceitos podem ocorrer em SGBDOOs
- Principal consenso
  - Tipo
    - definição de uma estrutura e/ou assinatura de métodos
    - não possui uma extensão (instâncias)
    - utilizado na definição de uma ou mais classes
  - Classe
    - definição de um conjunto de instâncias (extensão)
    - base para consultas ao BD
    - pode ou não ser definida a partir de um tipo
    - implementa os métodos do tipo, caso ela tenha sido definida a partir de um tipo

# Tipo X Classe - Exemplos

```
type Pessoa (  
    nome string; sexo char;  
    método idade() retorna inteiro;  
    ...)
```

```
classe Empregado type Pessoa (  
    método idade() retorna inteiro  
    begin . . . end;  
    ...)
```

```
classe Estudante type Pessoa (. . .)
```

```
classe Automóvel ( marca: string; modelo: string; ...)
```

# Modelo de Dados OO - Conceitos

1. Identidade de objeto (OID)
2. Métodos
3. Classes
4. **Estruturas complexas**
5. Herança
6. *Late Binding* (ligação tardia)

# Estruturas Complexas

- Atributos com domínios **não-atômicos**
  - característica não suportada por BDRs
- Tipos de domínios
  - **primitivos** (atômicos)
    - inteiros, cadeias de caracteres, datas, ...
  - **referência** (OIDs)
    - nomes de classes (determinam relacionamentos)
  - **construídos a partir de construtores de tipos**
    - definição de domínios complexos pelo usuário
- Vantagem
  - **flexibilidade** na definição de objetos complexos

# Construtores de Tipos

- **Tupla** (*tuple*)
  - domínio é um registro
- **Conjunto/Coleção** (*set / bag*)
  - domínio é um grupo de dados
- **Lista** (*list*)
  - domínio é um grupo ordenado de dados
- **Exemplos de domínios complexos**
  - conjunto de inteiros
  - tuplas de listas de *strings*
  - listas de conjuntos de tuplas
  - ...

# Exemplo de Classe

Classe Empregados (

CPF: *integer*,

nome: *string*,

endereço: **TUPLE** ( rua: *string*,

número: *integer*,

cidade: *Cidades* ),

especializações: **LIST**(*string*), (*por ordem de experiência*)

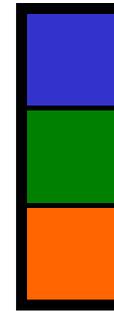
cargo: *string*;

departamento: *Departamentos*,

salário: *real*,

atividades: **SET** (**TUPLE** (projeto: *Projetos*,

tarefa: *string*) ) );



primitivos

referência

construtores de tipos

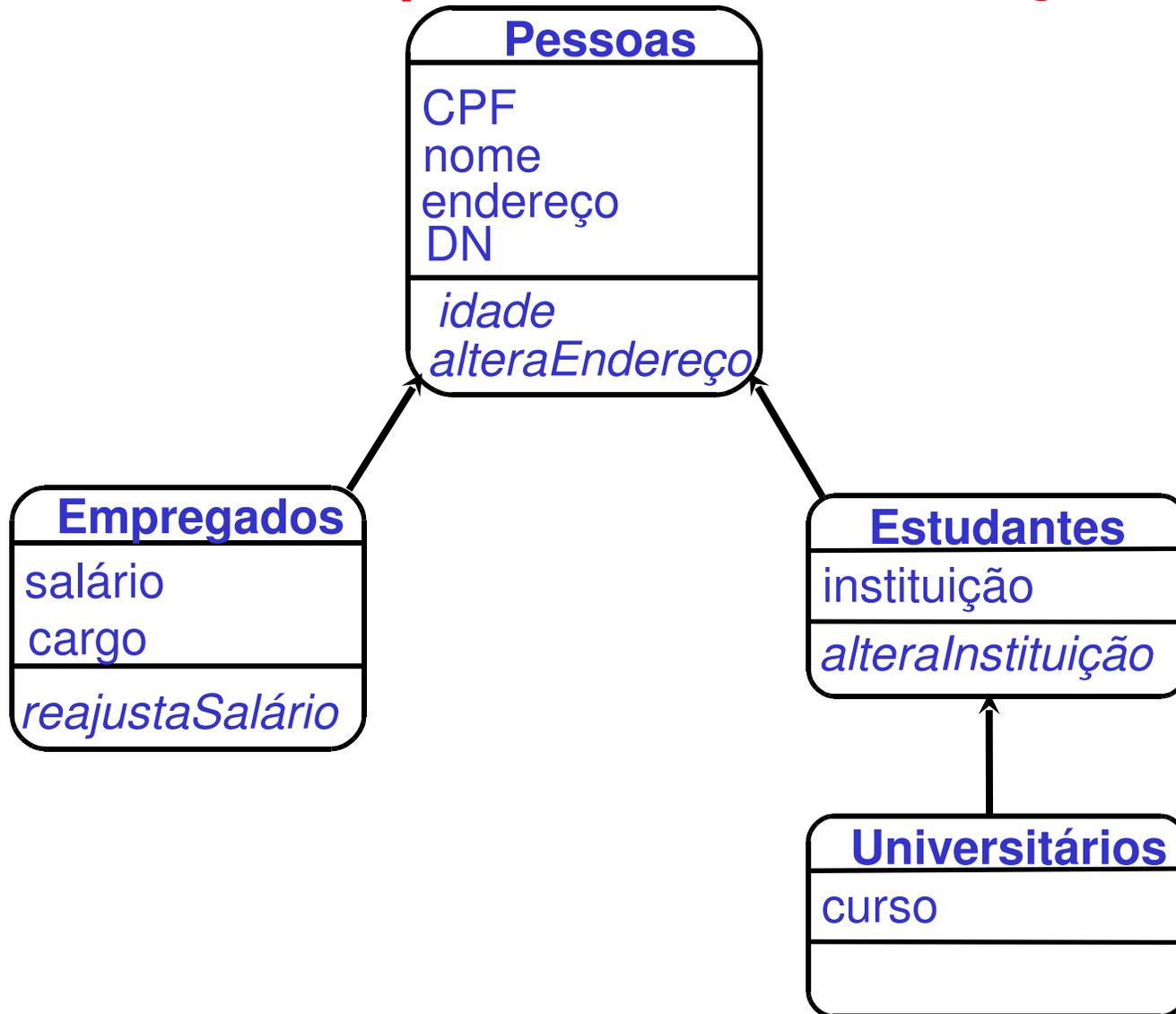
# Modelo de Dados OO - Conceitos

1. Identidade de objeto (OID)
2. Métodos
3. Classes
4. Estruturas complexas
5. **Herança**
6. *Late Binding* (ligação tardia)

# Herança

- Suporte à representação de relacionamentos com semântica de generalização e especialização
  - especialização
    - classe (subclasse) herda propriedades de outra classe (superclasse) e define novas propriedades
    - subclasse: categorização da superclasse
  - generalização (*É-UM*)
    - propriedades comuns de classes (subclasses) podem ser definidas uma única vez em uma superclasse
- Vantagem: reusabilidade

# Hierarquia de Herança



# Questões sobre Herança

- Redefinição de propriedades (*Overriding*)
  - **preocupação**: consultas válidas na superclasse
  - alternativas
    - (i) redefinição não é permitida (*herança estrita*); ou
    - (ii) atributos: domínios mais restritos
      - exemplo: A: real (superclasse) → A: inteiro (subclasse)
    - e métodos: domínios mais restritos para os parâmetros e para o tipo do resultado
- **Metaclasses**
  - superclasses sem extensão (*“classes abstratas”*)
  - definem atributos e implementam métodos que são herdados para classes concretas

# Questões sobre Herança

- Herança múltipla

- subclasse herda propriedades de mais de uma superclasse (conflitos podem ocorrer!)

- algumas alternativas

- (i) herança múltipla não é permitida

- (ii) herança de subclasse *default* (a primeira na lista, ...)

- (iii) herança da propriedade com domínio mais restrito (no caso de origem comum)

- (iv) herança da propriedade é definida pelo usuário

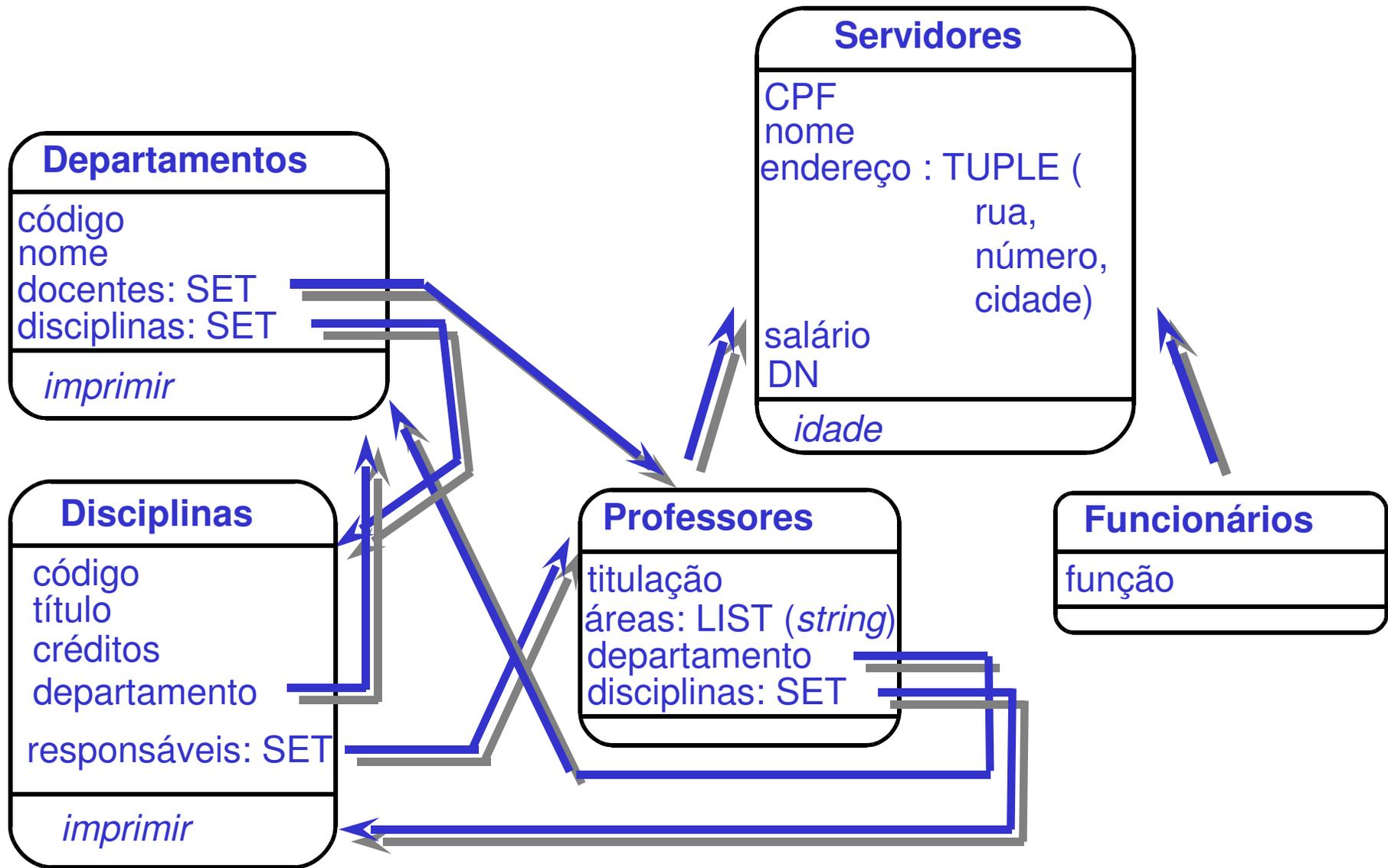
# Modelo de Dados OO - Conceitos

1. Identidade de objeto (OID)
2. Métodos
3. Classes
4. Estruturas complexas
5. Herança
6. ***Late Binding*** (ligação tardia)

# *Late Binding*

- Característica herdada de LPOO
- *Overloading* (sobrecarga)
  - uso de um mesmo nome para mais de um método
  - exemplo: *imprimir()* em Pessoas e Imagens
- *Late Binding* (vinculação tardia)
  - escolha do código do método a ser executado em tempo de execução (*depende da classe do objeto e dos parâmetros*)
- LP procedurais: *early binding*
  - vinculação entre código e nome de procedimento definido em tempo de compilação
  - não é possível um nome ligado a mais de um código

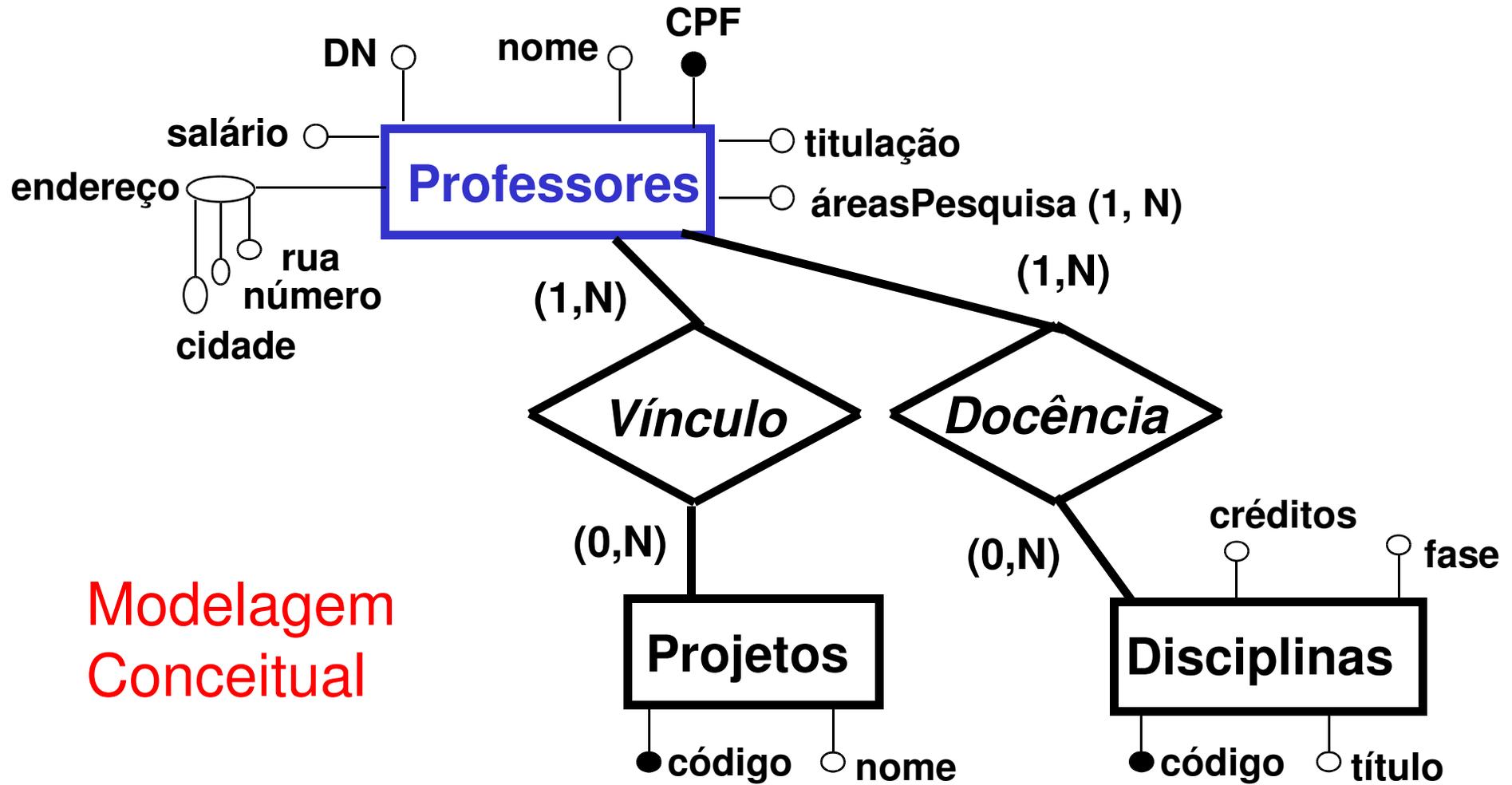
# Exemplo de Esquema OO



# Modelo de Dados OO

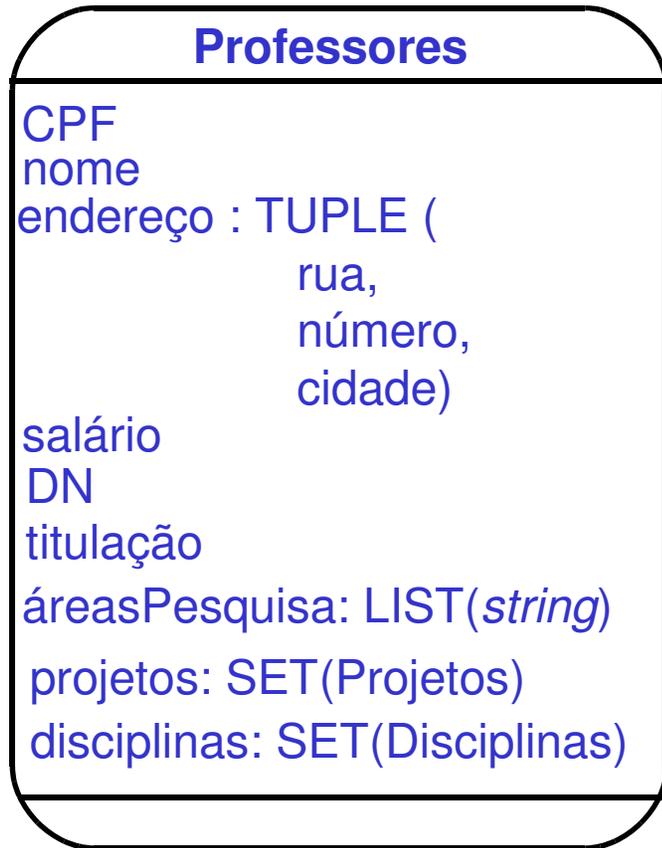
- Comparação com o modelo relacional
  - maior nível de abstração
    - modelo de objetos tem maior afinidade semântica com modelos conceituais de dados
      - representação mais natural de uma entidade do mundo real
  - mais adequado a representação de dados complexos de aplicações não-convencionais
    - aplicações CAD, sistemas de informação geográfica, ...
  - modelo mais complexo
    - maior número de conceitos

# Exemplo



Modelagem  
Conceitual

# Exemplo



Modelagem  
Lógica OO

# Exemplo

Professores

<u>CPF</u>	nome	rua	número	cidade	salário	titulação	DN
------------	------	-----	--------	--------	---------	-----------	----

ÁreasPesquisa

<u>CPF</u>	<u>Área</u>	ordem
------------	-------------	-------

Docência

<u>CPF</u>	<u>Codd</u>
------------	-------------

Pesquisa

<u>CPF</u>	<u>Codp</u>
------------	-------------

Modelagem  
Lógica Relacional