Mapeamento Objeto-Relacional

- Paradigma orientação a objetos
  - baseia-se em princípios como o acoplamento, coesão e encapsulamento
  - percorrem-se os objetos através das suas relações (referências)

- Paradigma bases de dados relacionais
  - baseado em princípios matemáticos, nomeadamente a teoria de conjuntos
  - duplica-se informação para poder fazer o *join* de registos (linhas) das tabelas

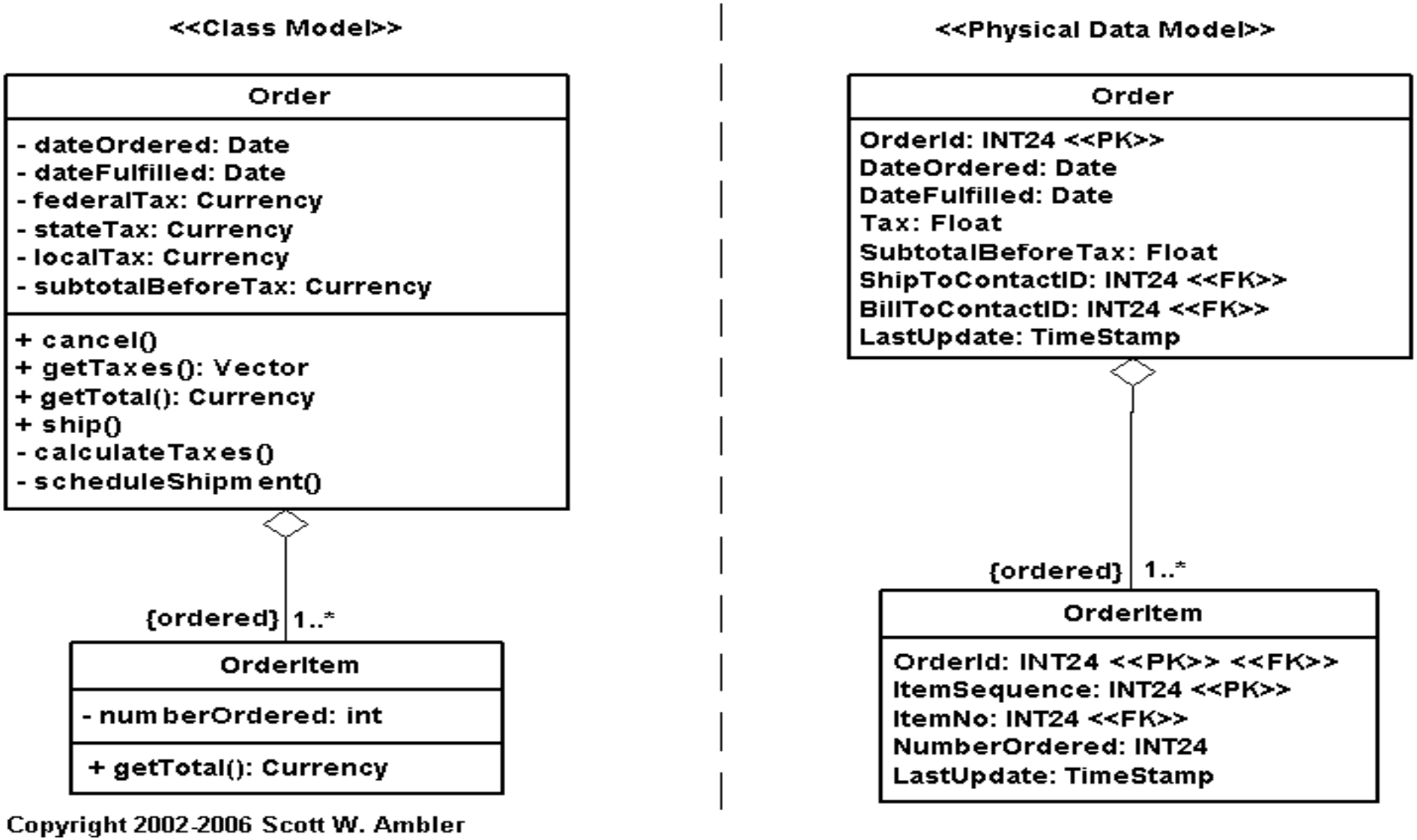
- As classes implementam comportamento e dados, ao passo que as tabelas de BD implementam apenas dados

- Ponto de partida para o mapeamento: os atributos das classes

Atributos e colunas

- Um atributo de uma classe corresponde a zero, uma ou mais colunas da tabela da BD
  - nem todos os atributos são persistentes, alguns podem ser usados para cálculos temporários
    - Exemplo: na BD grava-se a data de nascimento, que é um valor fixo; a idade pode ser um atributo mas é calculada, porque depende da data em que é consultada
- Situação mais simples: um atributo corresponde a uma coluna de uma tabela
  - ainda mais simples: são do mesmo tipo (String vs CHAR, int vs INTEGER, float vs FLOAT, etc.)
- A figura da página seguinte exemplifica uma situação deste tipo. Mostra parte de um *schema* simples de um sistema de gestão de encomendas
  - Notar: o número campos de imposto é diferente; se calhar devem ser somados para obter o imposto total
  - Há campos de tipo diferente (subBeforeTax); pode haver necessidade de conversão

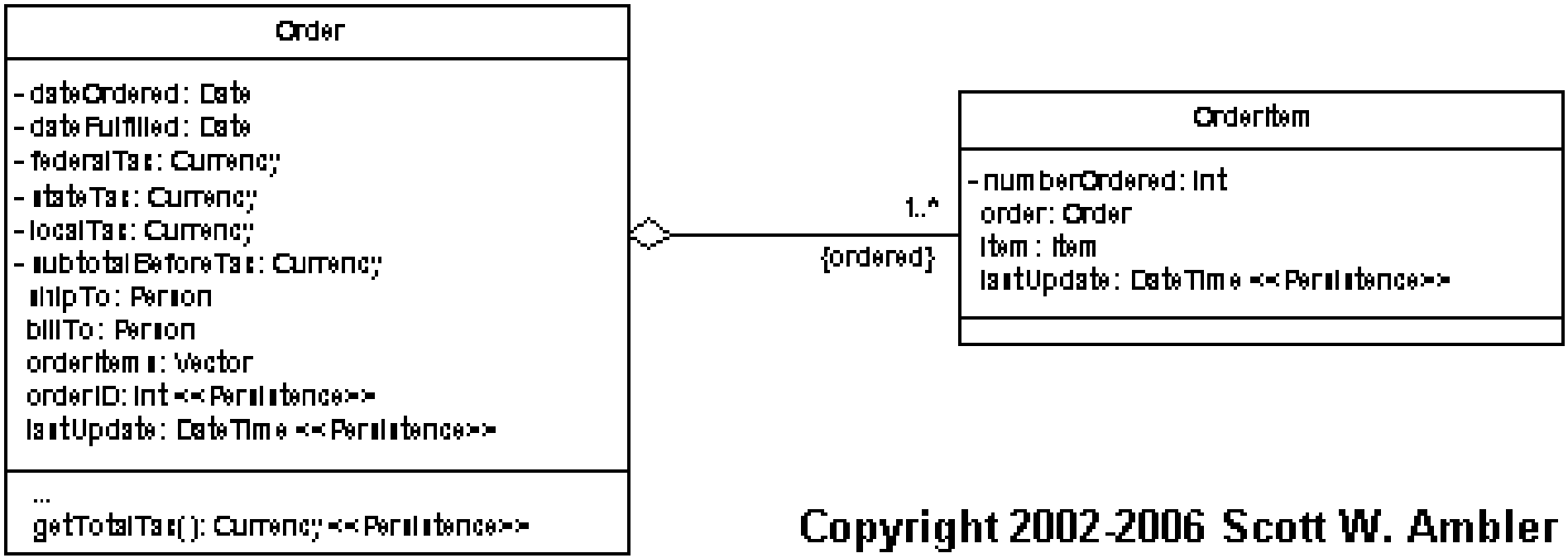
Diagrama de classes / modelo de dados físico



http://www.agiledata.org/essays/mappingObjects.html

Shadow information

- *Shadow information* é informação que tem que ser introduzida nas classes, para além da necessária para o processamento normal, para suporte à persistência
- Podem ser necessários atributos de suporte à implementação das relações entre as classes; p.ex. orderItems
- O método getTotalTax() é necessário (daí se falar por vezes em mapeamento de propriedades em vez de atributos)

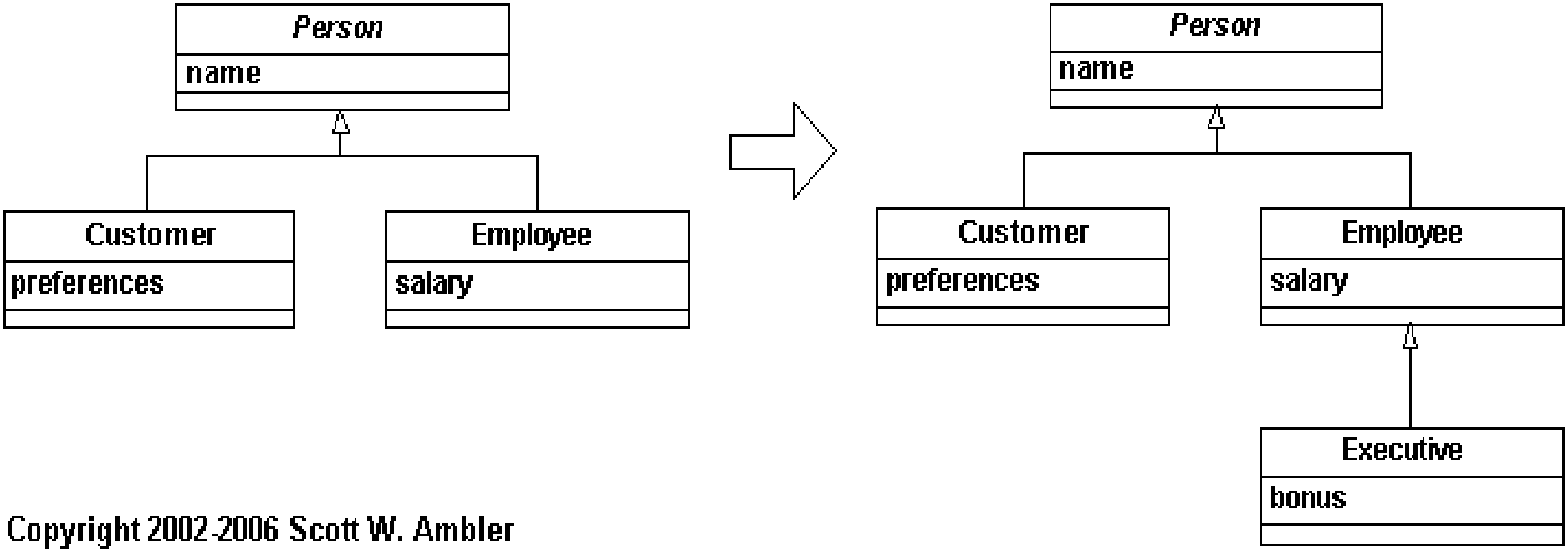


Exemplo de correspondência entre propriedades e colunas

Propriedade	Coluna
Order.orderId	Order.OrderId
Order.dateOrdered	Order.DateOrdered
Order.dateFulfilled	Order.DateFulfilled
Order.getTotalTax()	Order.Tax
Order.subtotalBeforeTax	Order.SubtotalBeforeTax
Order.shipTo.personId	Order.ShipToContactId
Order.billTo.personId	Order.BillToContactId
Order.lastUpdate	Order.LastUpdate
OrderItem.ordered	OrderItem.OrderId
Order.orderItems.position(orderItem)	Order.ItemSequence
OrderItem.item.number	OrderItem.ItemNo
OrderItem.numberOrdered	OrderItem.NumberOrdered
OrderItem.lastUpdate	OrderItem.LastUpdate

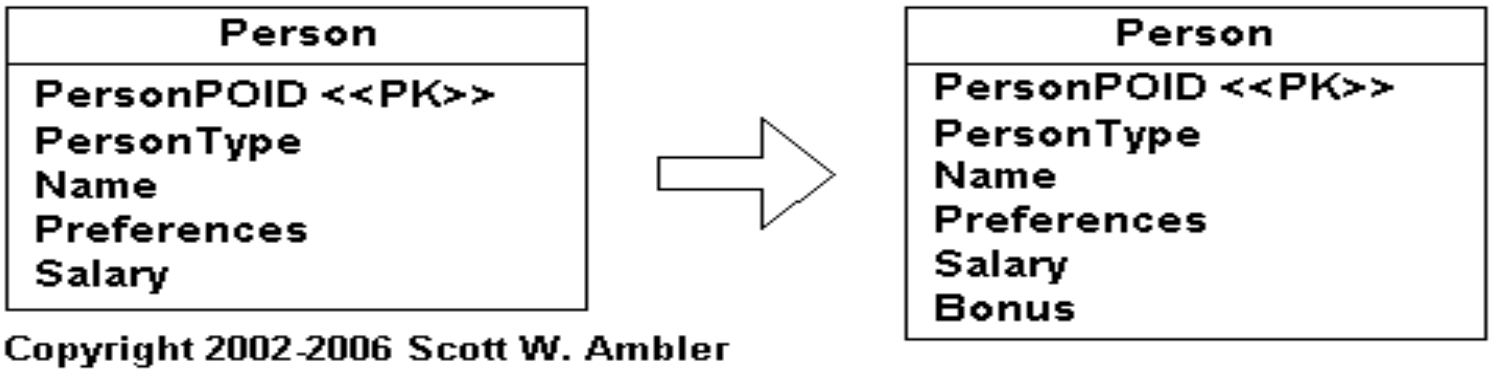
Herança

- As BD relacionais não suportam herança diretamente
- Vejamos várias soluções possíveis com o exemplo seguinte, que inclui duas versões de uma hierarquia de classes

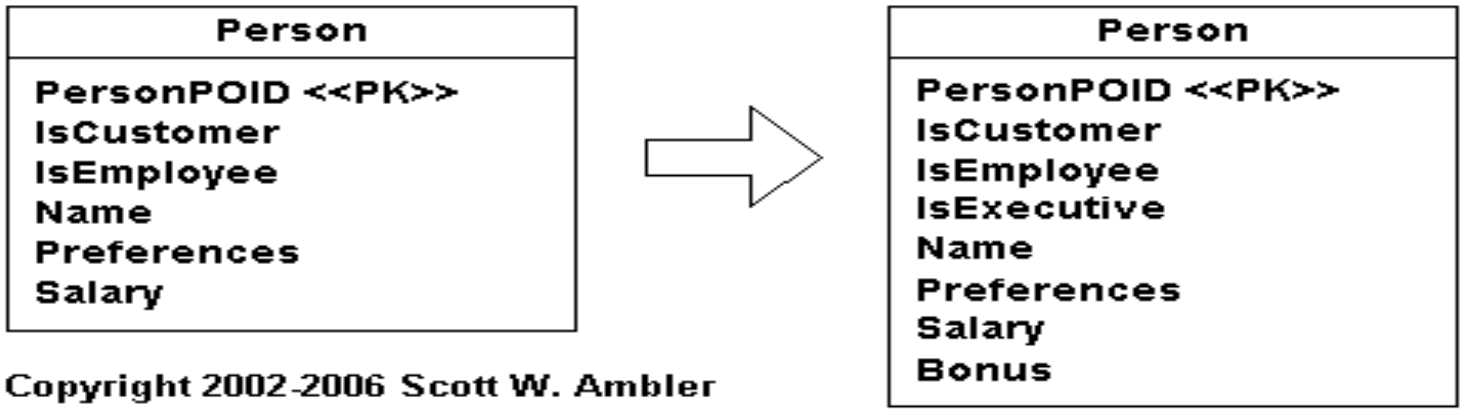


Herança (cont.)

- Solução 1: toda a hierarquia é representada por uma tabela
  - Um campo inteiro (PersonType, neste caso) indica o tipo de objeto

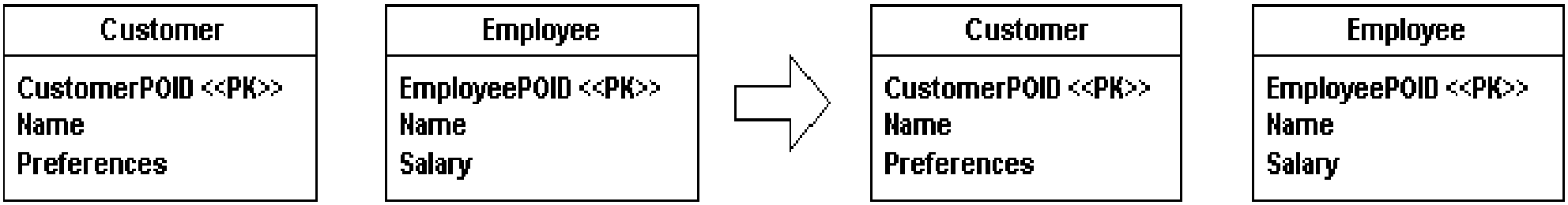


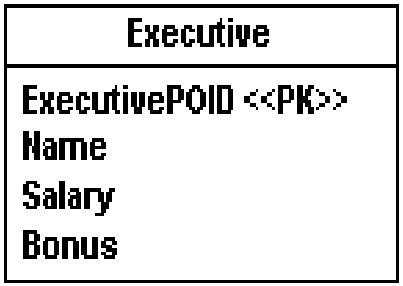
- Solução preferível, com campos boolean, se um objeto puder assumir várias combinações (p.ex. um funcionário também pode ser cliente)



Herança (cont. 2)

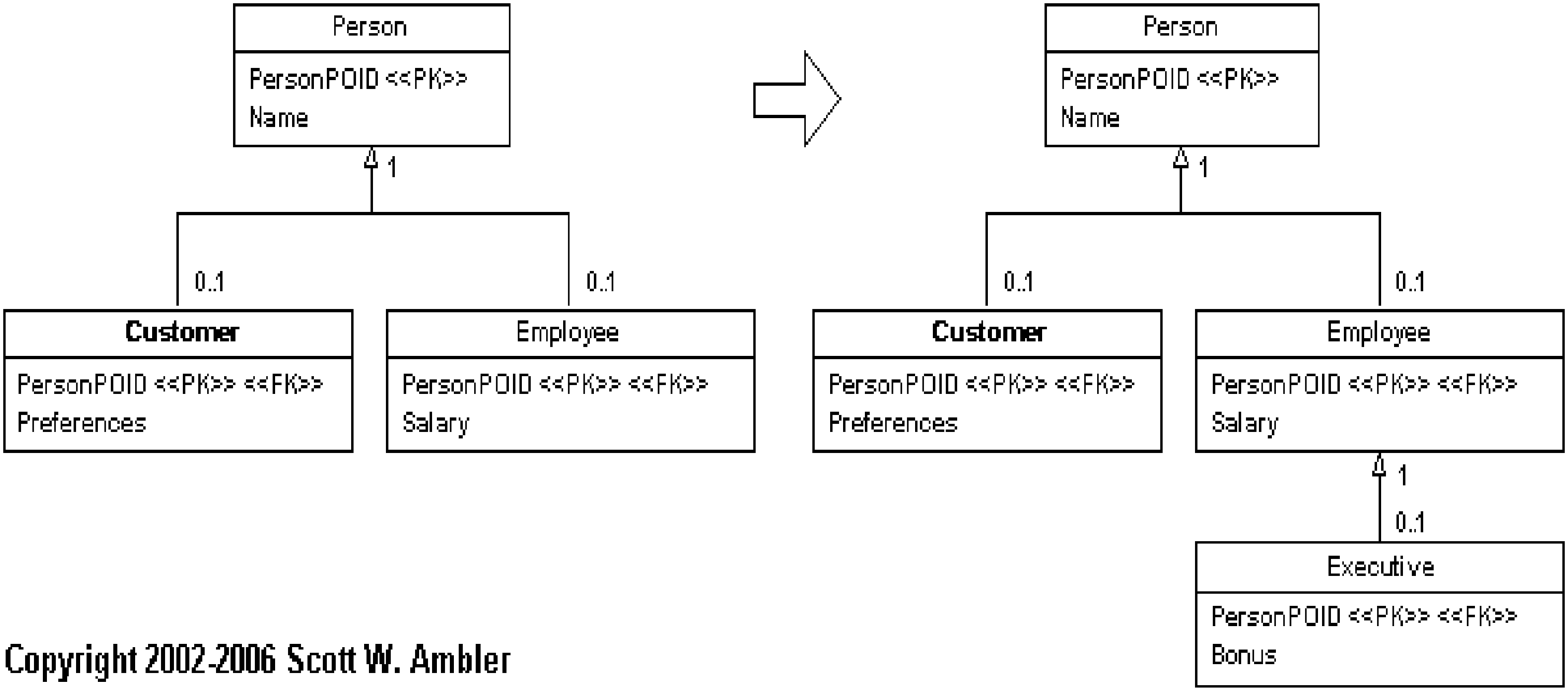
- Solução 2: cada classe concreta é representada pela sua tabela





Herança (cont. 3)

- Solução 3: cada classe (mesmo abstrata) é representada pela sua tabela
  - na tabela Person pode ser incluída uma coluna de tipo, ou colunas booleanas, para indicar qual o subtipo (classe concreta) do objeto



Comparação de soluções (herança)

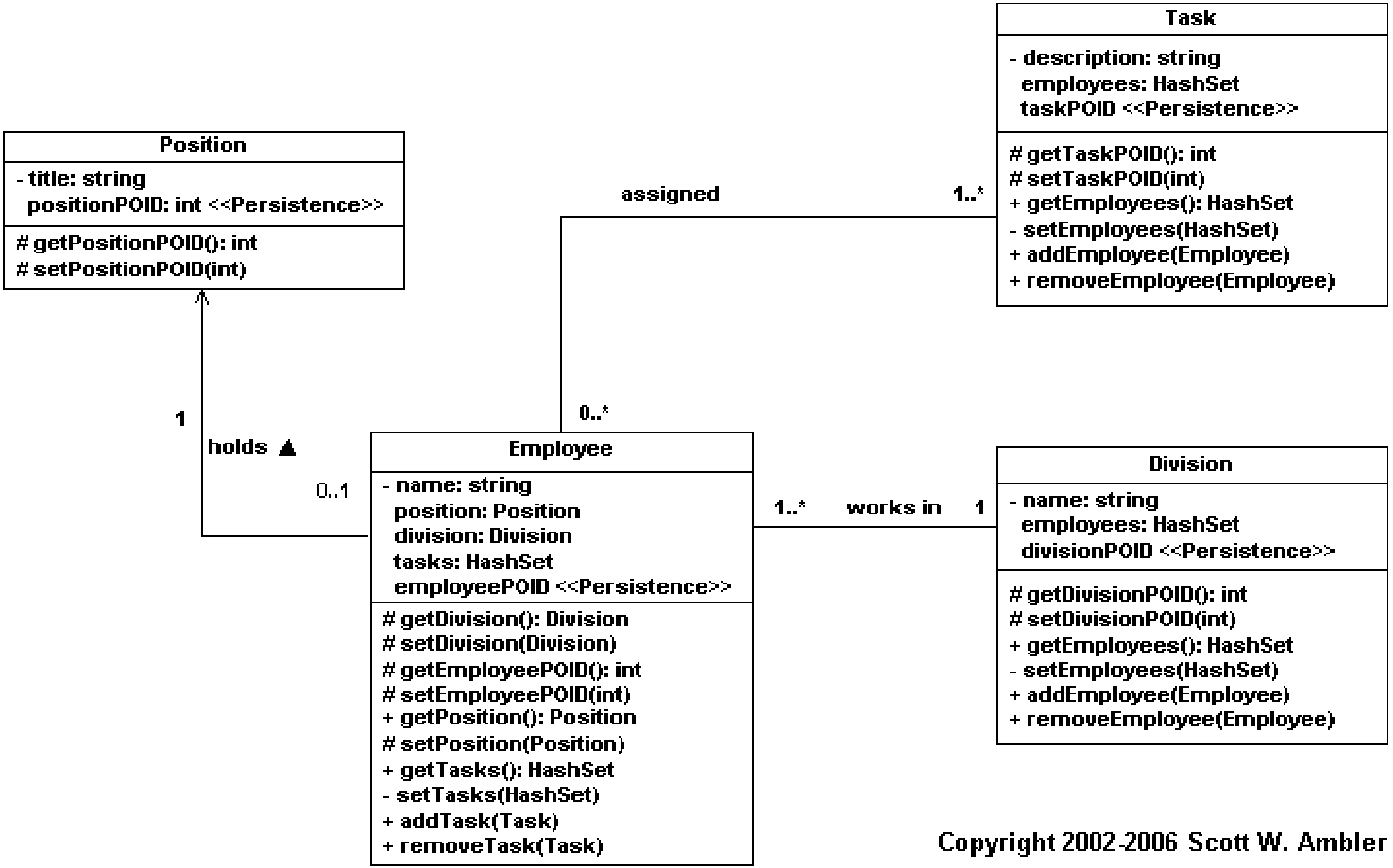
Fatores	Uma tabela	Uma tabela por classe concreta	Uma tabela por classe
Relatórios ad-hoc	Simples	Médio	Médio/difícil
Facilidade de implementação	Simples	Médio	Difícil
Facilidade de acesso aos dados	Simples	Simples	Médio/simples
Acoplamento	Muito elevado	Elevado	Baixo
Velocidade de acesso aos dados	Rápido	Rápido	Médio/rápido
Suporte a polimorfismo	Médio	Baixo	Alto

Relações

- Tipos de relações entre objetos
  - Multiplicidade
    - Um para um (1:1)
    - Um para muitos (1:n)
    - Muitos para muitos (m:n)
  - Direcionalidade
    - Unidirecionais
      - Um objeto conhece outro, mas este não conhece o primeiro
    - Bidirecionais
      - Os dois objetos conhecem-se mutuamente
  - Natureza
    - Herança
    - Associação, agregação, composição
      - Em termos de mapeamento são tratadas da mesma forma

- Em BD relacionais as relações (referências para outros objetos) são implementadas com chaves estrangeiras

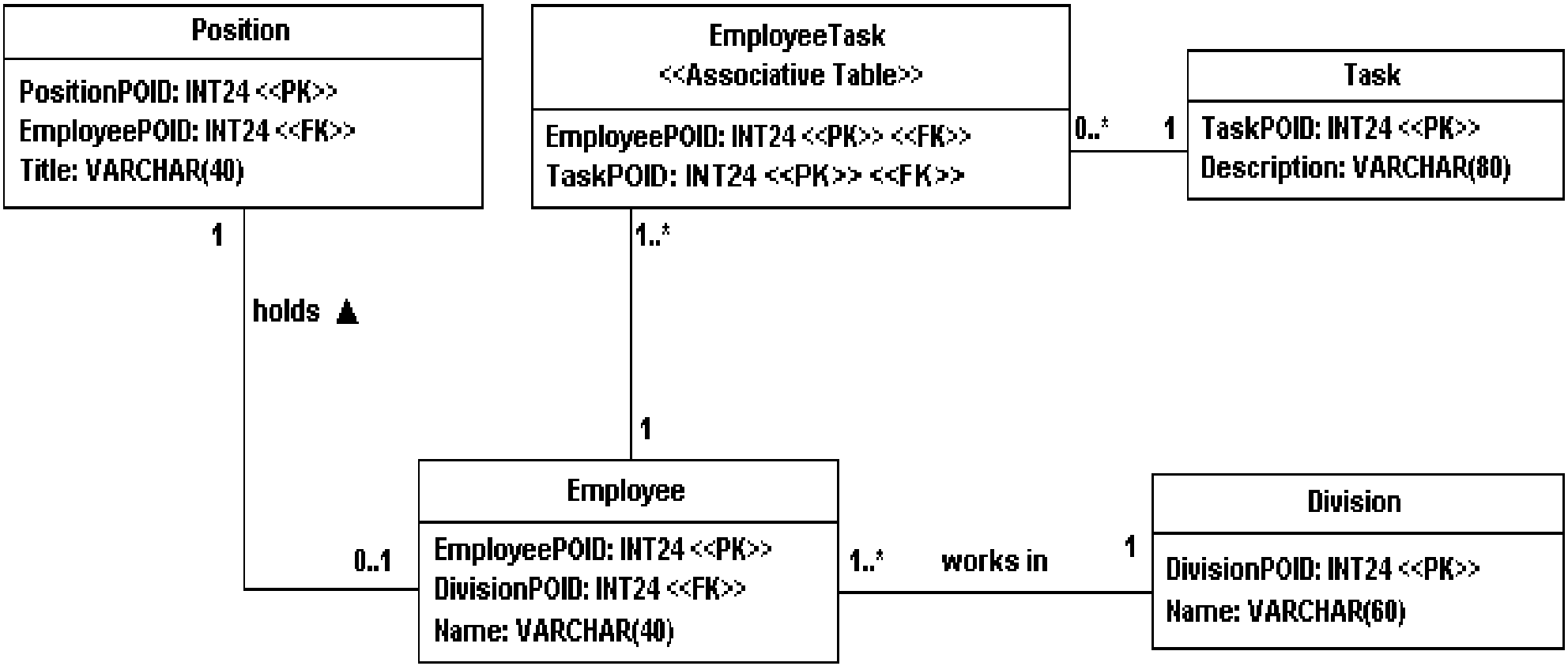
Relações (diagrama de classes)



Implementação das relações pelas classes

- Um para um
  - implementa-se com uma referência para o outro objeto, um método *get* e um método *set*
  - por exemplo, na classe Employee, a relação com a divisão em que trabalha é implementada com o campo division (do tipo Division), e os métodos getDivision() e setDivision(...)
- Um para muitos
  - implementa-se com um atributo cujo tipo é uma coleção, por exemplo ArrayList, HashSet ou HashMap.
  - por exemplo, a classe Division contém um atributo do tipo HashSet chamado employees, e os métodos getEmployees() para aceder ao HashSet, setEmployees(...) para definir o HashSet, addEmployee(...) para juntar um employee ao HashSet e removeEmployee(...) para retirar um employee do HashSet

Relações (tabelas da BD)



Implementação de relações (1:1 e 1:n)

- Relação um para um
  - é implementada com uma das tabelas contendo uma chave estrangeira que referencia a outra tabela
  - a chave estrangeira pode ser implementada em qualquer uma das tabelas
  - em alternativa, pode ser implementada com uma tabela associativa
  - o SELECT terá que se adaptar à solução escolhida, fazendo o *join* das tabelas; exemplo:

    SELECT * FROM Position, Employee

        WHERE Position.EmployeePoID = Employee.EmployeePoID
- Relação um para muitos
  - implementa-se com uma chave estrangeira na tabela "muitos" que referencia a chave da tabela "um"
  - pode também implementar-se com uma tabela associativa, efetivamente implementando uma relação muitos para muitos
  - se a relação tiver atributos, ficam na tabela "muitos", ou na tabela associativa, se existir

Implementação de relações (m:n)

- Implementa-se com uma tabela associativa
  - a tabela associativa usa como chave primária a combinação das chaves primárias das duas tabelas que associa
  - no exemplo, é o caso da tabela EmployeeTask
  - outro exemplo, mais simples: