

Proiectarea bazelor de date

Proiectarea bazelor de date

- **Proiectare conceptuală** (ex. diagrama de clase)
 - Identificarea entităților și a relațiilor dintre ele
- **Proiectarea logică**
 - Transformarea modelului conceptual într-o structură de baze de date (relațională sau nu)
- **Rafinarea bazei de date (normalizare)**
 - Eliminarea redundanțelor și a problemelor conexe
- **Proiectare fizică și eficientizare**
 - Indexare
 - De-normalizare!

Diagrama de clase UML - Clase

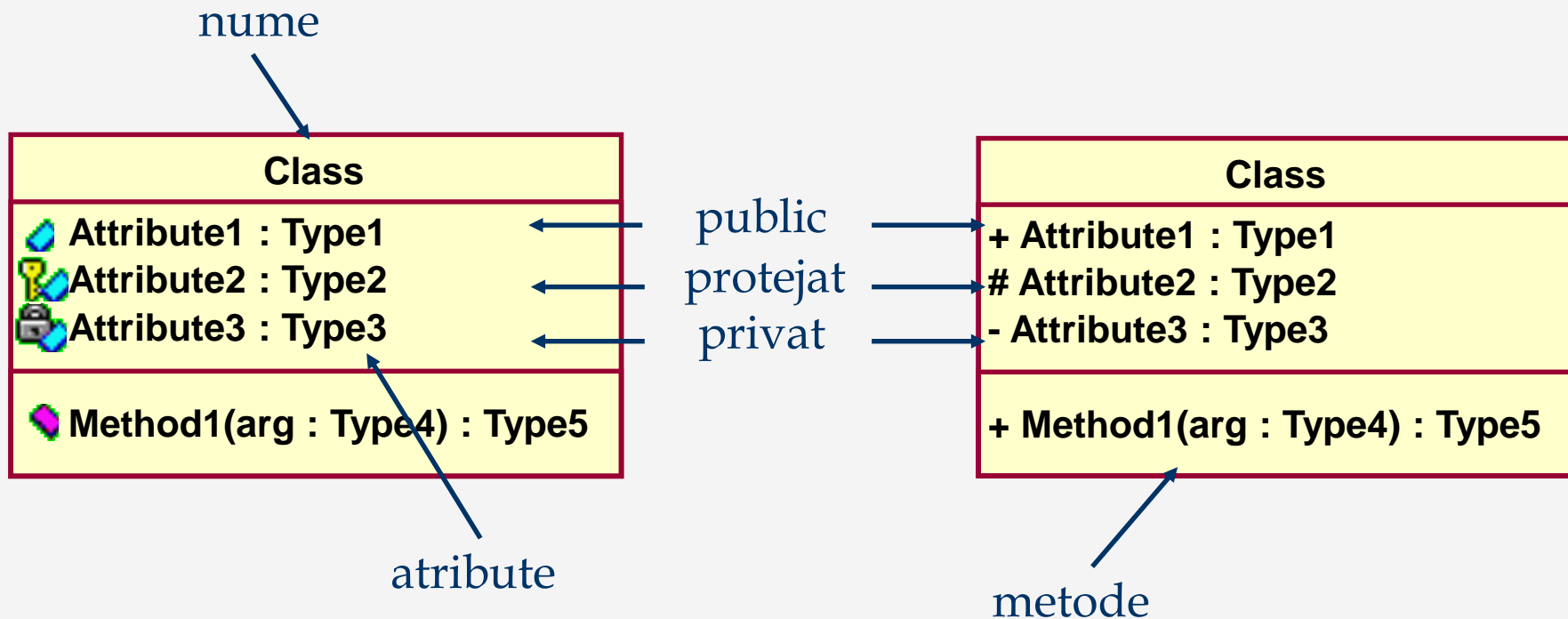
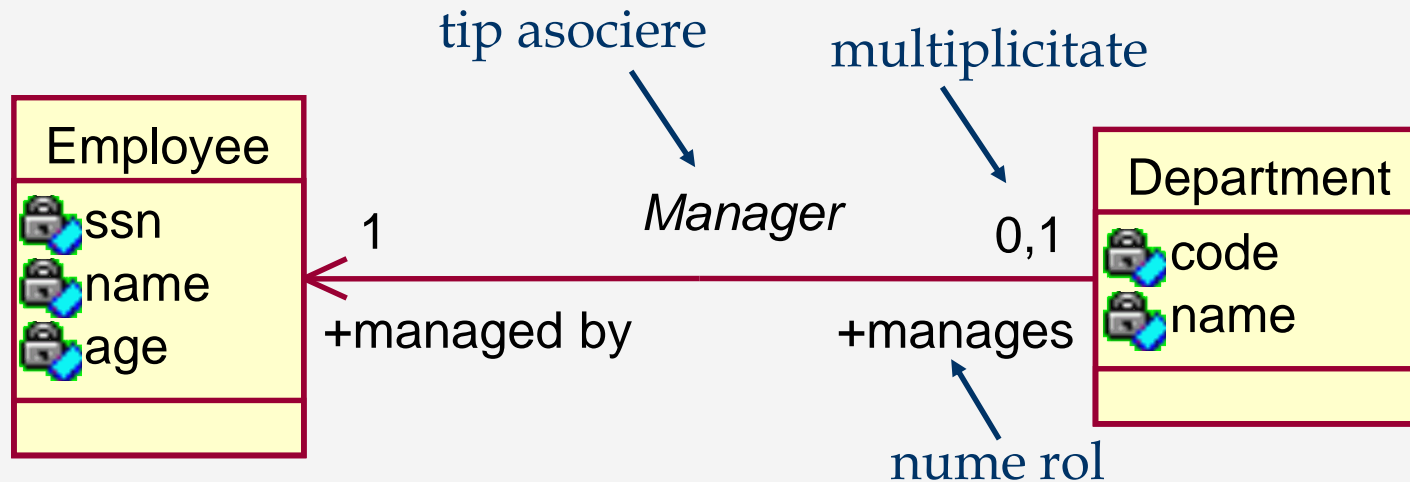


Diagrama de clase UML - Asocieri



■ Multiplicități:

- valori: 4,5
- intervale: 1..10
- nedefinit: *

■ Navigabilitatea asocierii:

- un sens
- bidirectional

Diagrama de clase UML - Asocieri

■ Citirea numelor de rol

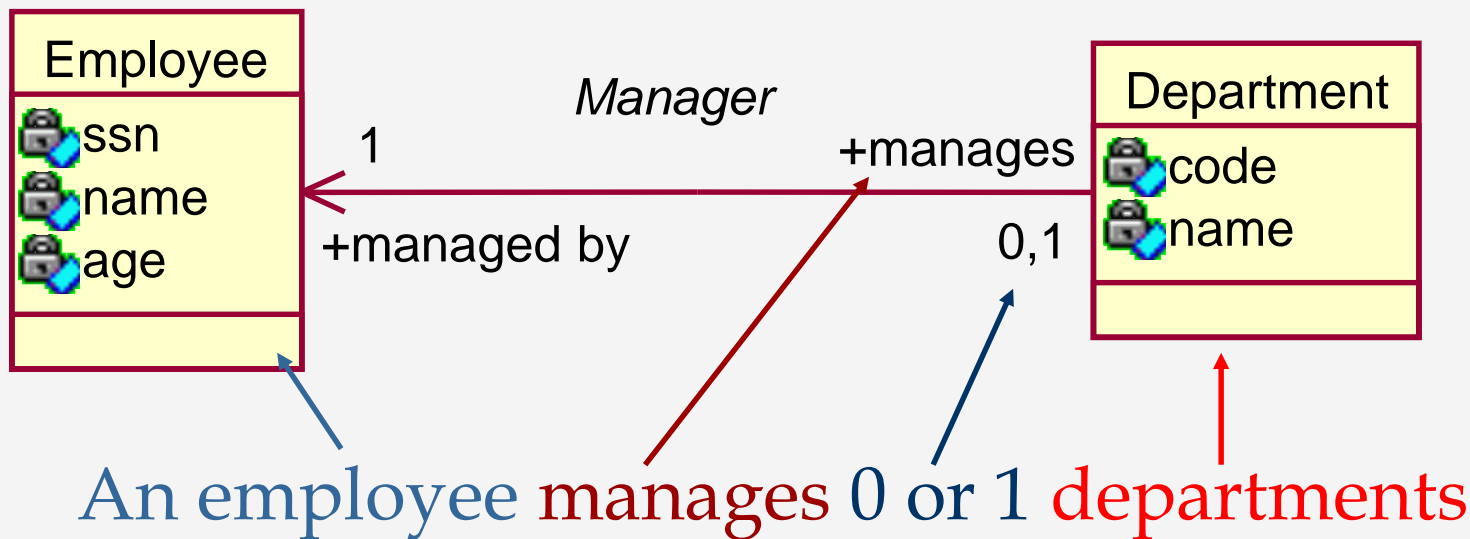


Diagrama de clase UML - Asocieri

- Agregare
 - asociere parte-intreg
- Compunere
 - “weak entities”
- Clasa asociere
- Asociere reflexiva

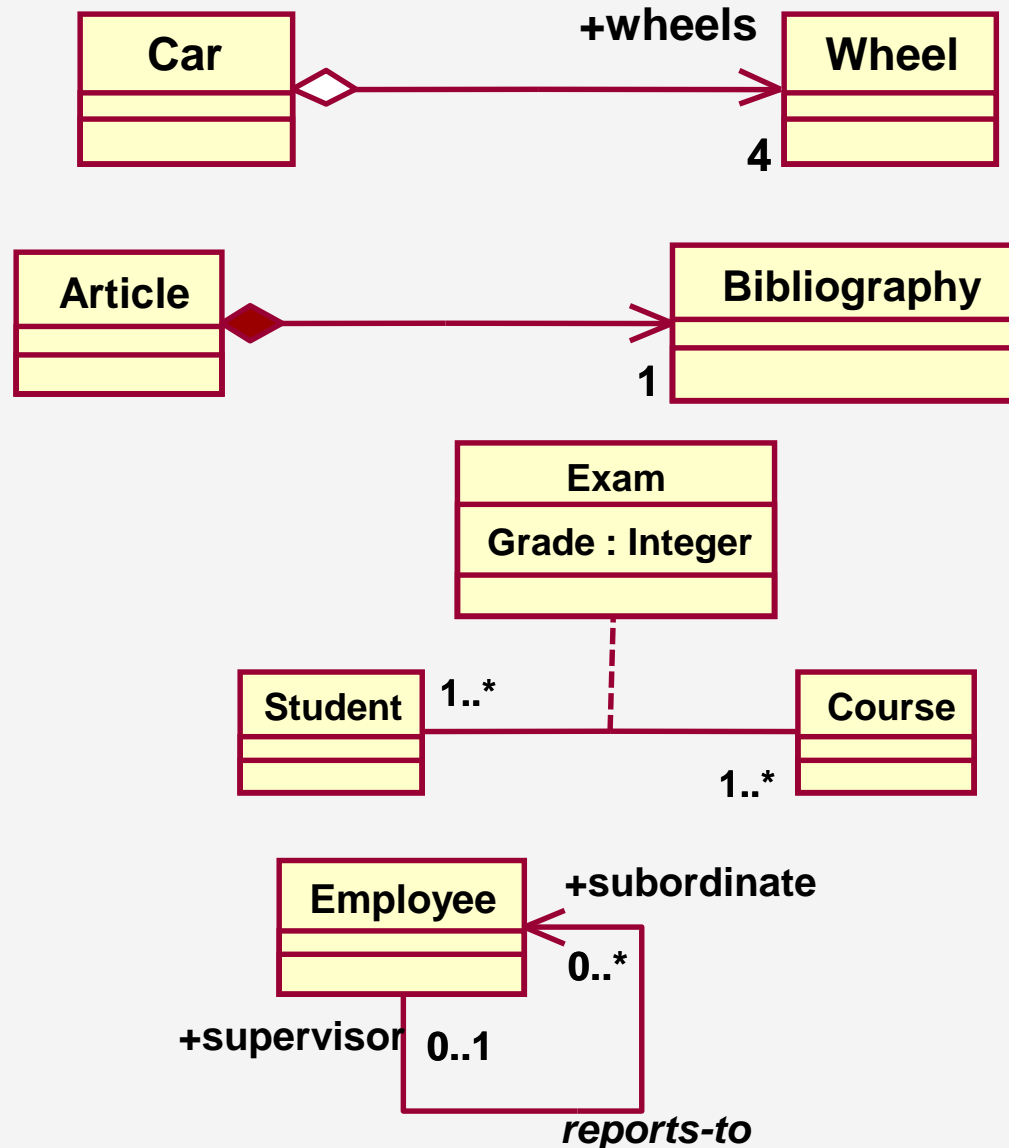
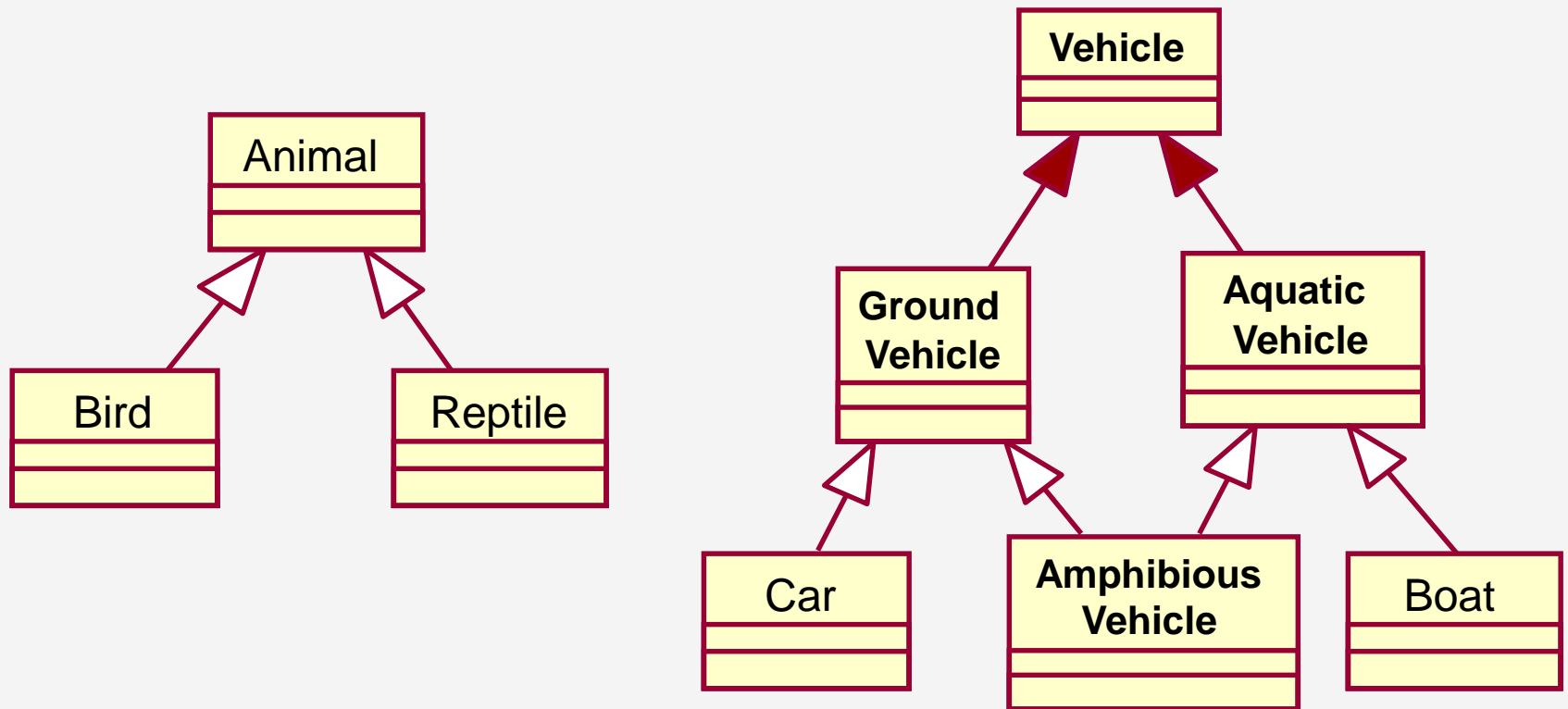


Diagrama de clase UML - Mostenire

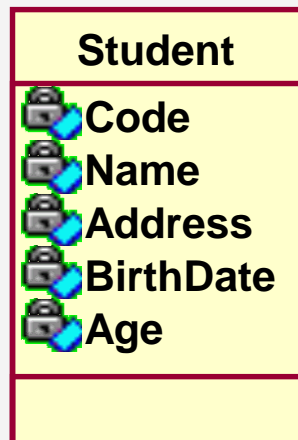


Modelul conceptual \Rightarrow bază de date relațională

- Transformare 1:1 a claselor în tabele:
 - Prea multe tabele – pot rezulta mai multe tabele decât este necesar
 - Prea multe op. join – consecință imediată a faptului că se obțin prea multe tabele
 - Tabele lipsă – asocierile m:n între clase implică utilizarea unei tabele speciale (*cross table*)
 - Tratarea necorespunzătoare a moștenirii
 - Denormalizarea datelor – anumite date se regăsesc în mai multe tabele

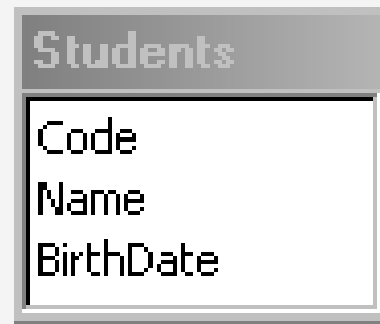
Transformarea claselor în tabele

- Numele tabelului reprezintă pluralul numelui clasei
- Toate atributele simple sunt transformate în câmpuri
- Atributele compuse devin tabele de sine stătătoare
- Atributele derivate nu vor avea nici un corespondent în tabelă



Students (Code, Name, BirthDate)

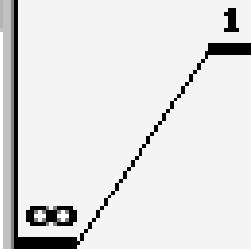
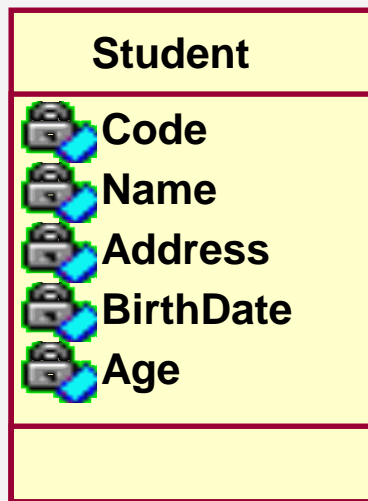
Addresses (Street, Number, City)



Transformarea claselor în tabele

- Chei surogat – chei care nu sunt obținute din domeniul problemei modelate
- Conceptul de cheie nu este definit în cadrul claselor UML
- O *bună practică*: utilizarea (atunci când este posibil) a cheilor de tip întreg generate automat de SGBD:
 - ușor de întreținut (responsabilitatea sistemului)
 - eficient (interogări rapide)
 - simplifică definirea cheilor străine
- Disciplină de proiectare a BD:
 - toate cheile surogat vor fi numite **ID**
 - toate cheile străine se numesc **<NumeTabel>ID**

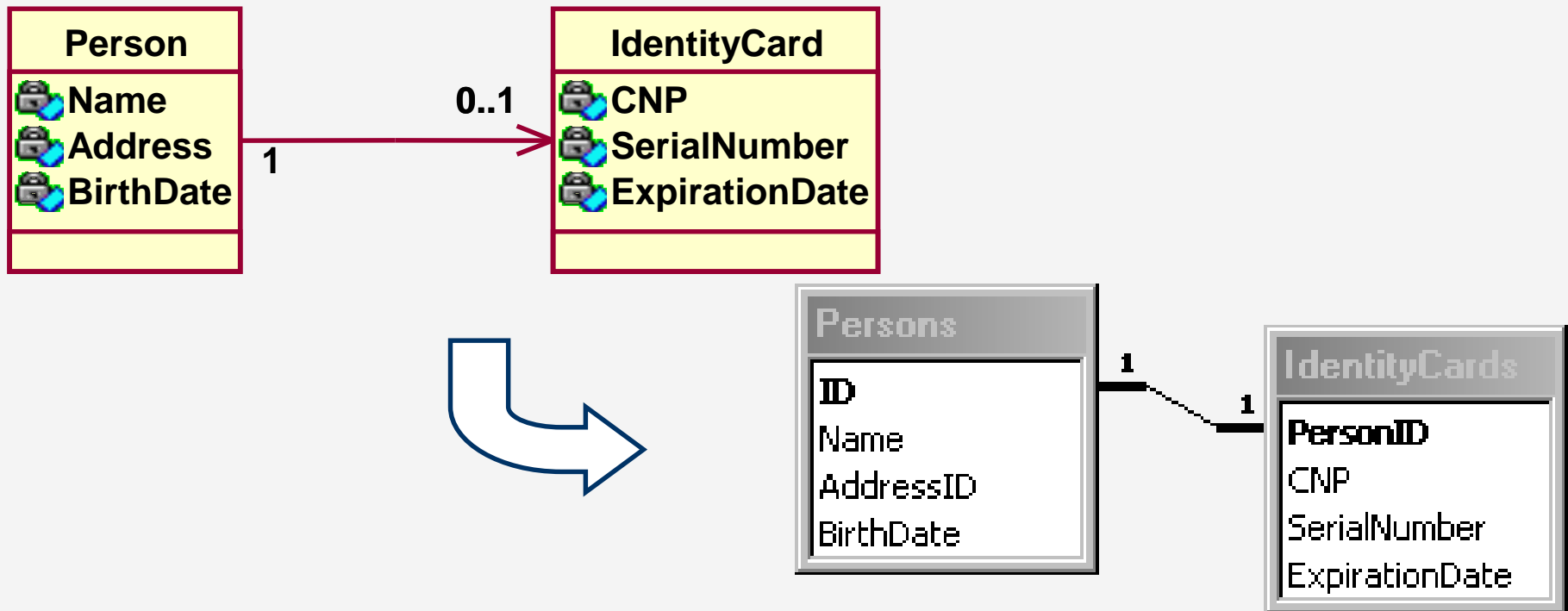
Transformarea claselor în tabele (cont)



Transformarea asocierilor simple

■ 1 : 0,1

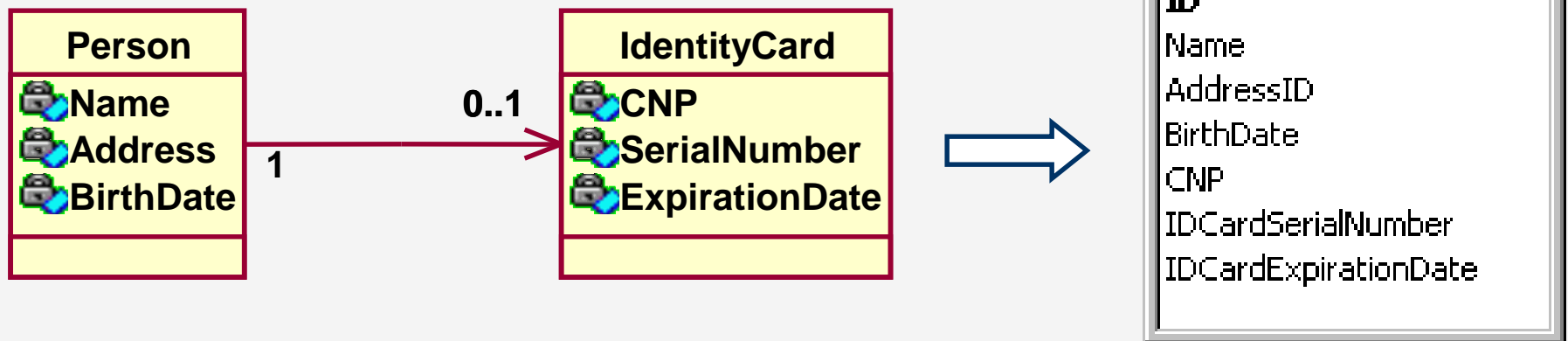
- se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
- cheia tablei corespunzătoare multiplicității “0, 1” este cheia străină în cea de-a doua tabelă
- o singură cheie va fi generată automat (de obicei cea corespunzătoare multiplicității “1”)



Transformarea asocierilor simple (cont)

■ 1 : 1

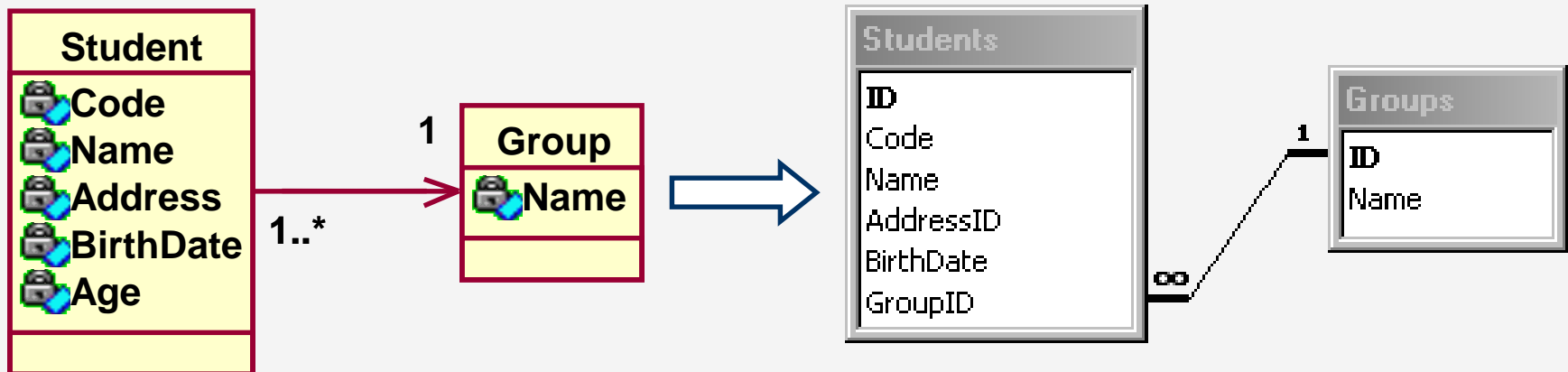
- se crează o singură tabelă ce conține attributele ambelor clase asociate
- aceasta variantă de transformare se aplică și asocierilor “1 : 0,1” atunci când este vorba de un număr relativ mic de cazuri în care obiectele primei clase nu sunt legate de obiectele celei de-a doua clase



Transformarea asocierilor simple (cont)

■ 1 : 1..*

- se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
- cheia tabelului corespunzătoare multiplicității “ 1 ” este cheia străină în cea de-a doua tabelă, corespunzătoare multiplicității “1..*”

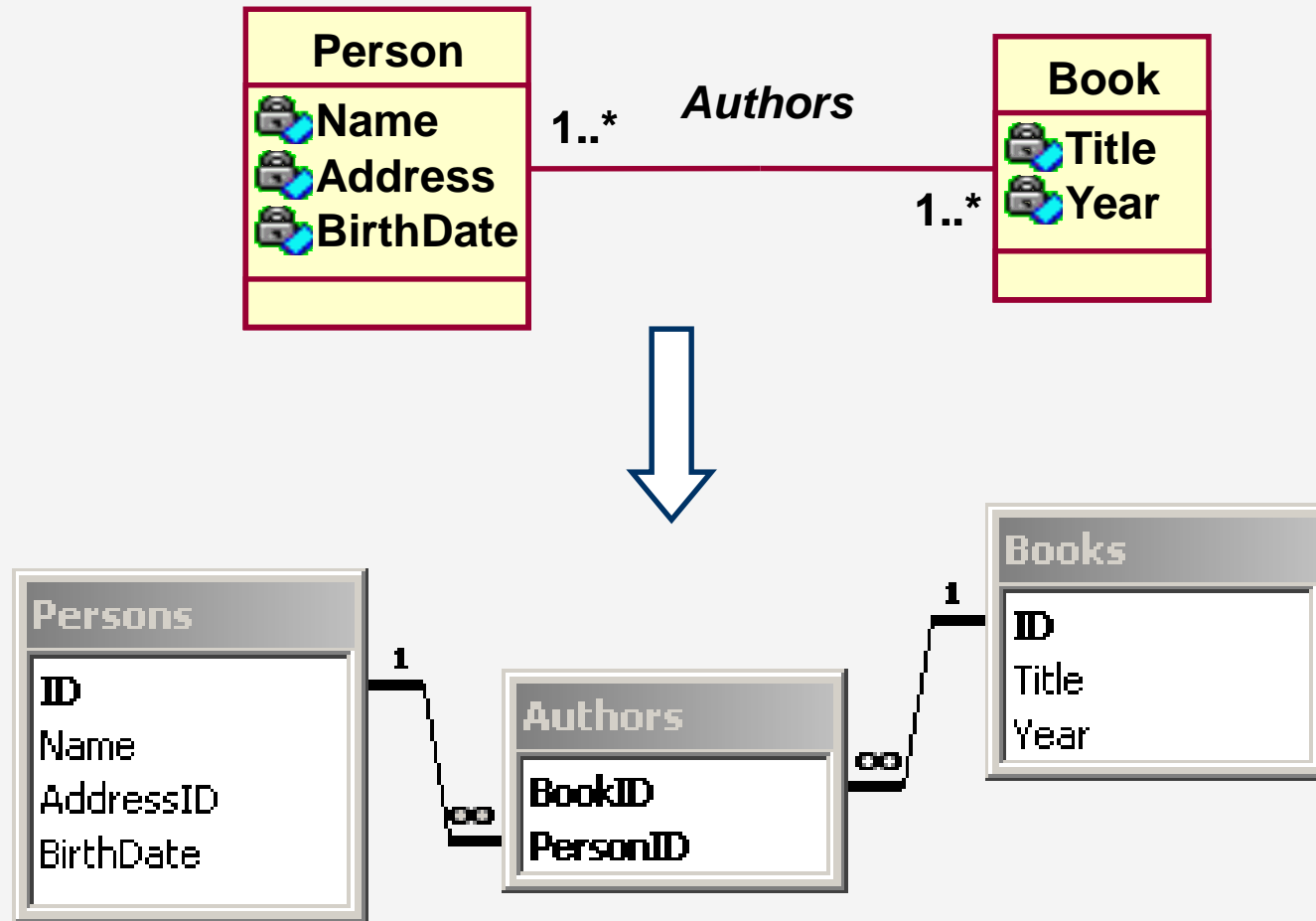


Transformarea asocierilor simple (cont)

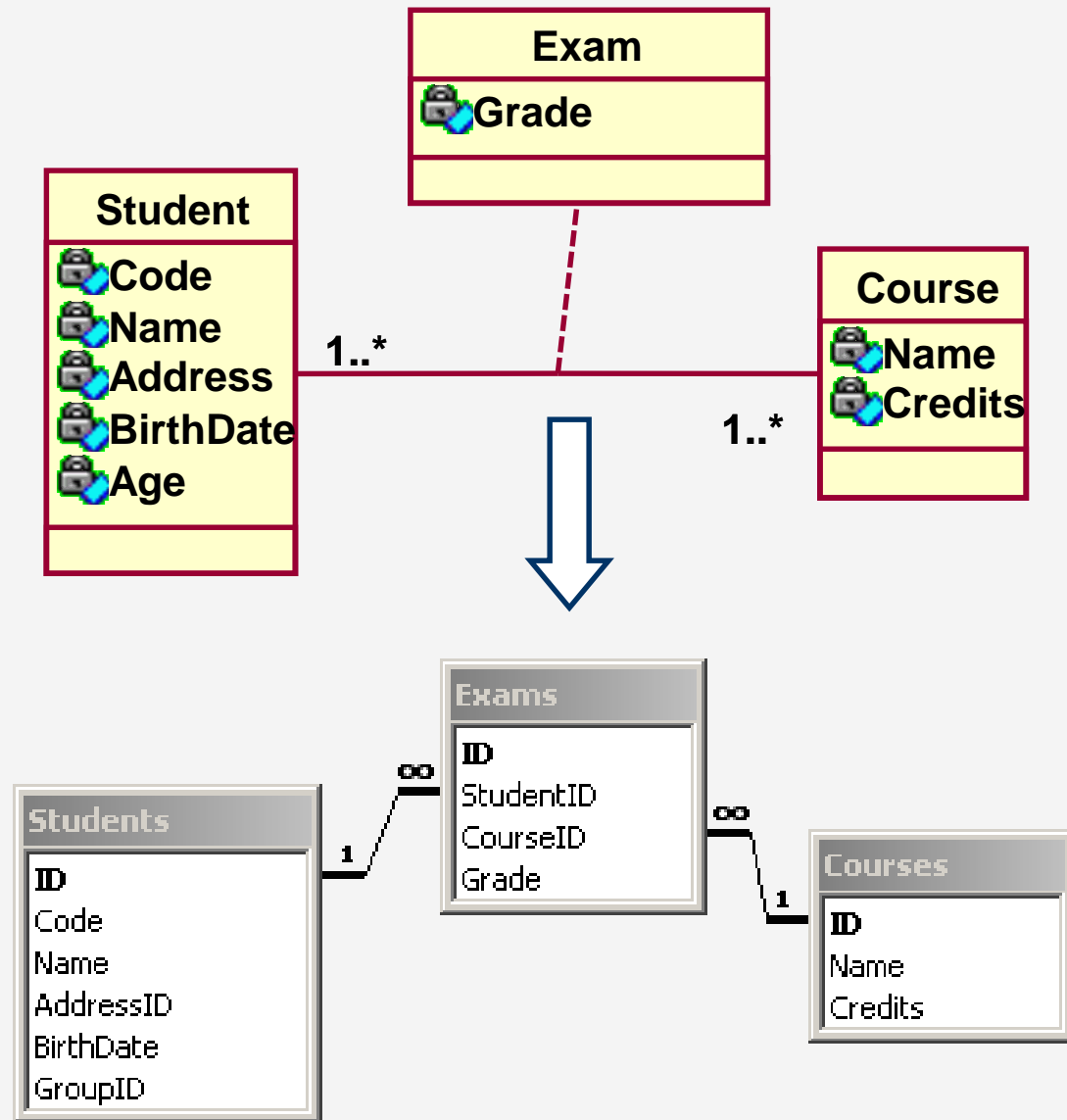
■ 1..* : 1..*

- se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
- se crează o tabelă adițională numită tabelă de intersecție (*cross table*)
- cheile primare corespunzătoare tabelelor inițiale sunt definite ca și chei străine în tabela de intersecție
- cheia primară a tabelii de intersecție este, de obicei, compusă din cele două chei străine spre celelalte tabele. Sunt cazuri în care se utilizează și aici cheie surogat.
- dacă asocierea conține o clasă asociere, toate atributele acestei clase vor fi inserate în tabela de intersecție
- uzual, numele tabelii de intersecție este o combinație a numelor tabelilor inițiale dar acest lucru nu este necesar.

Transformarea asocierilor simple (cont)



Transformarea asocierilor simple (cont)



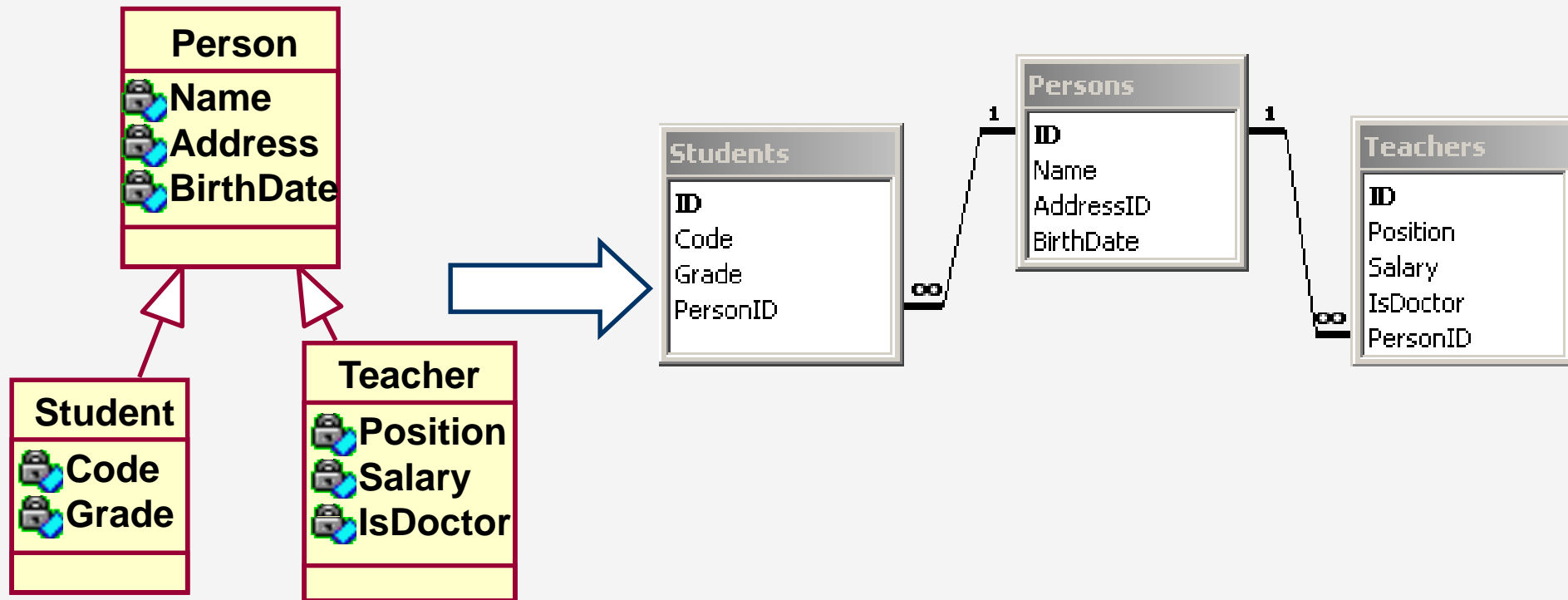
Transformarea moștenirii

Metoda 1

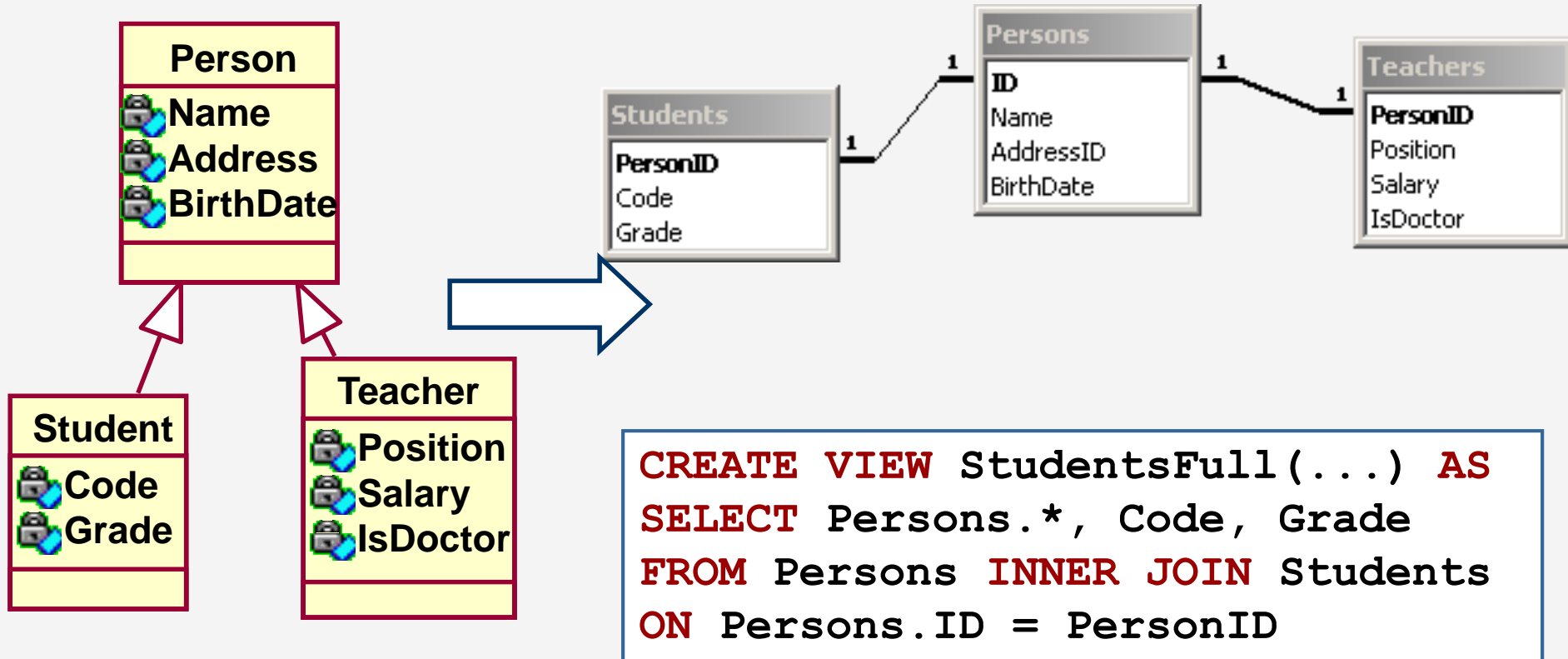
Presupune crearea câte unui tabel corespunzător fiecărei clase și a câte unui *view* pentru fiecare pereche super-clasă/subclasă

- Flexibilitate – permite adăugarea viitoarelor subclase fără impact asupra tabelelor/*view*-urilor deja existente
- Implică crearea celor mai multe tabele/*view*-uri
- Posibile probleme de performanță deoarece fiecare access va implica execuția unui *join*

Transformarea moștenirii



Transformarea moștenirii



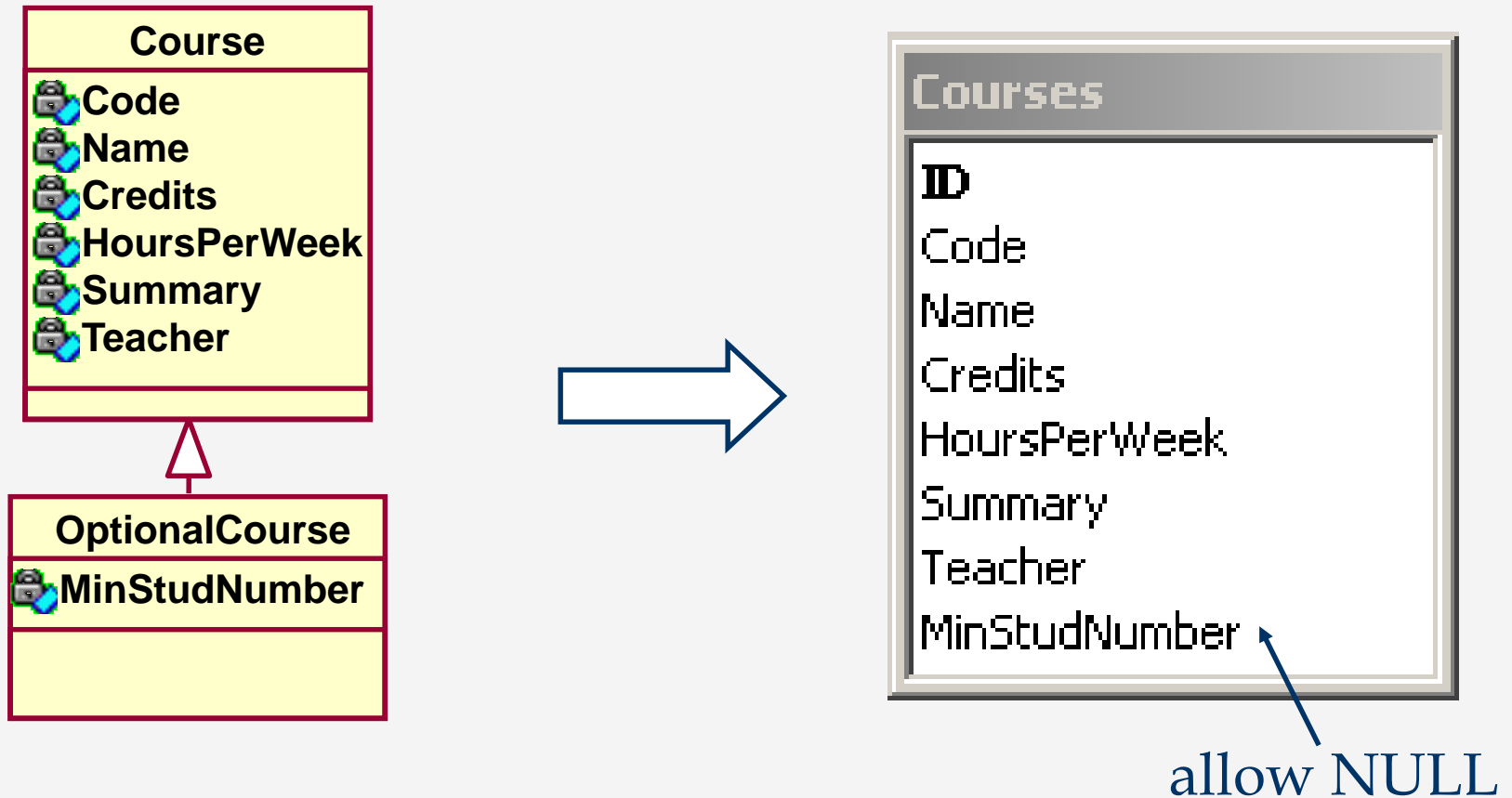
Transformarea moștenirii

Metoda 2

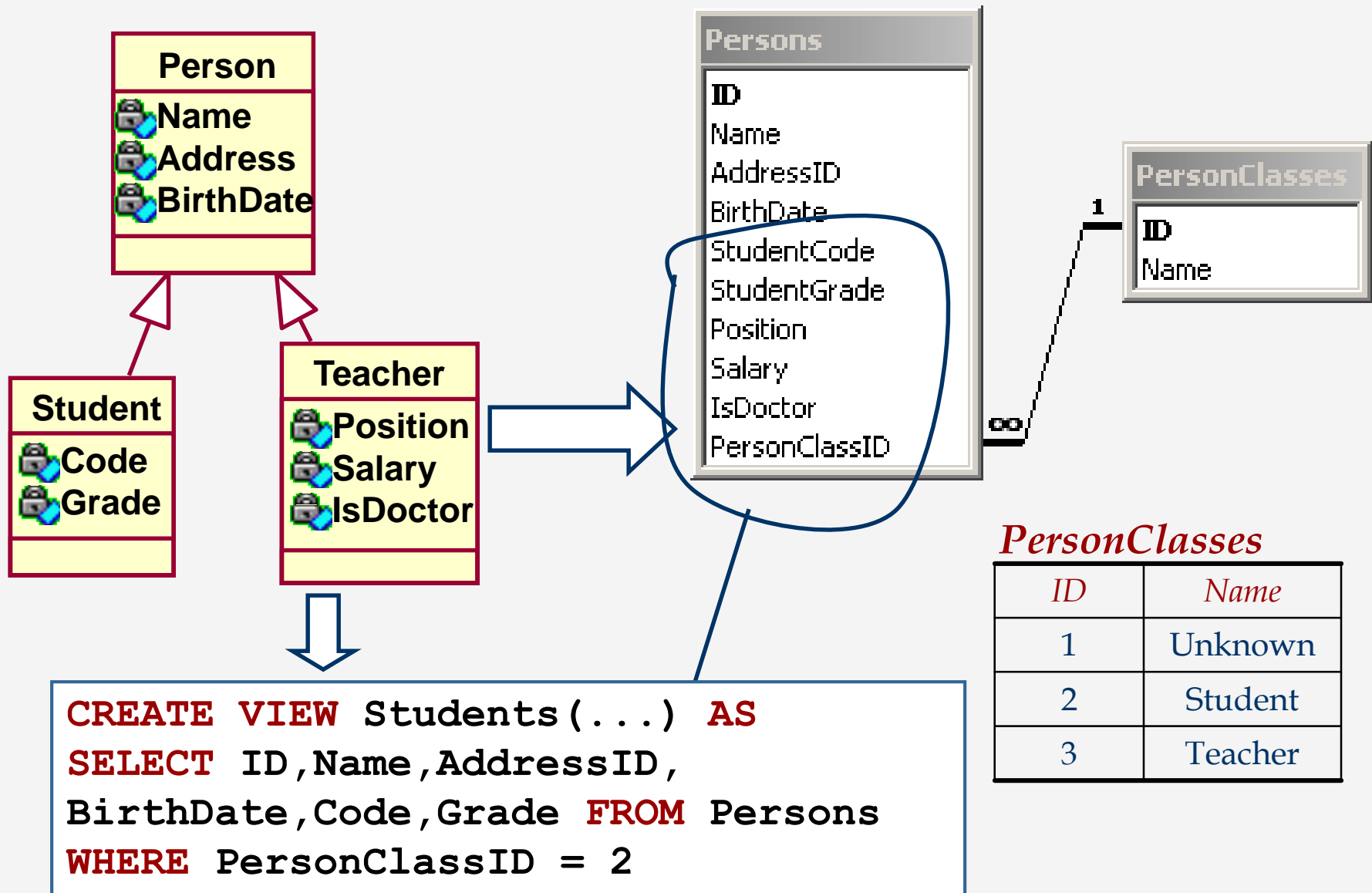
Se crează o singură tabelă (corespunzătoare superclasei) și se de-normalizează toate attributele subclaselor acesteia.

- Implică crearea celor mai puține tabele/*view*-uri - opțional, se poate defini o tabelă de subclase și *view*-uri corespunzătoare fiecărei subclase.
- Se obține, de obicei, cea mai mare performanță
- Adăugarea unei noi subclase implică modificări structurale
- Creștere “artificială” a spațiului utilizat

Transformarea moștenirii



Transformarea moștenirii



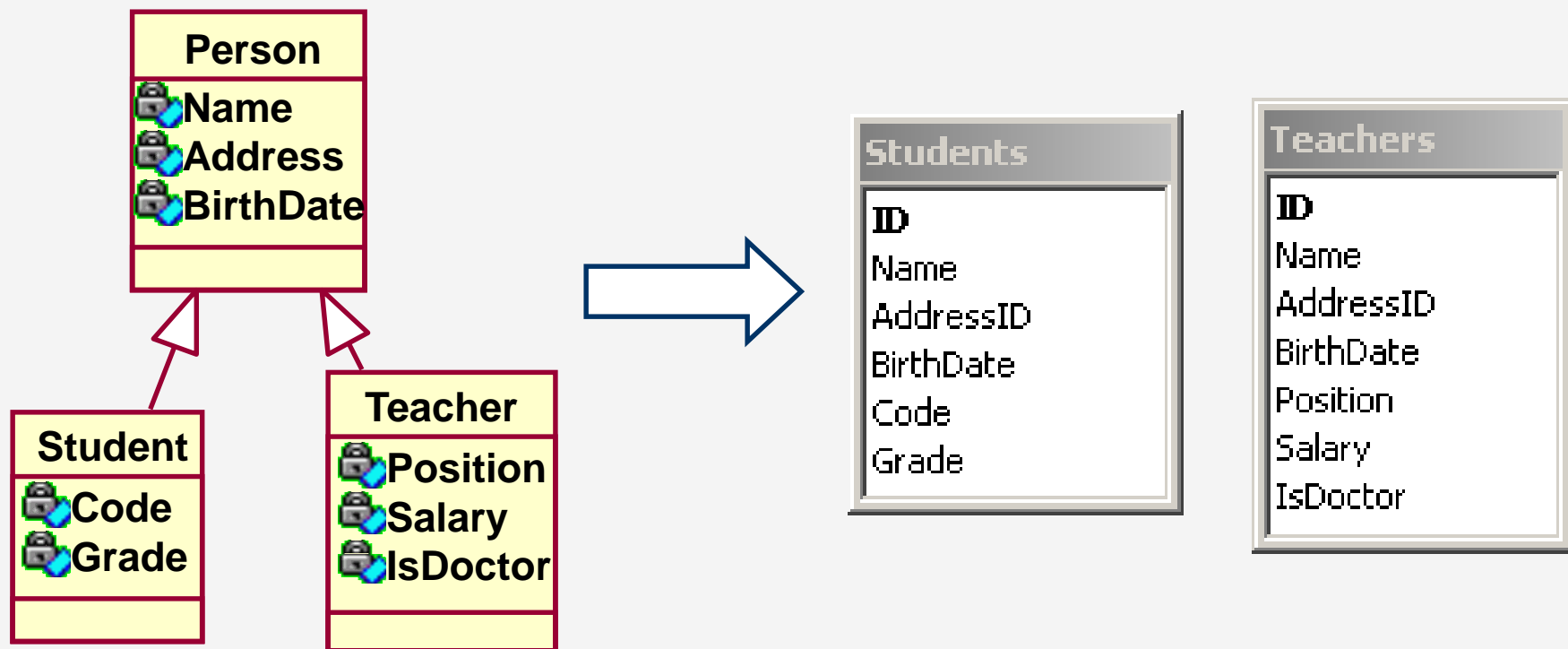
Transformarea moștenirii

Metoda 3

Presupune crearea câte unui tabel corespunzător fiecărei sub-clase și de-normalizarea atributelor super-clasei în fiecare dintre tabelele create

- Performanța obținută este satisfăcătoare
- Adăugarea unei noi subclase **nu** implică modificări structurale
- Posibilele modificări structurale la nivelul superclasei afectează toate tabelele definite!

Transformarea moștenirii



Transformarea moștenirii

Care este metoda potrivită?

- Dacă numărul înregistrărilor stocate în tabele este redus (deci performanța nu reprezintă o problemă), atunci poate fi selectată cea mai flexibilă metodă -

Metoda 1

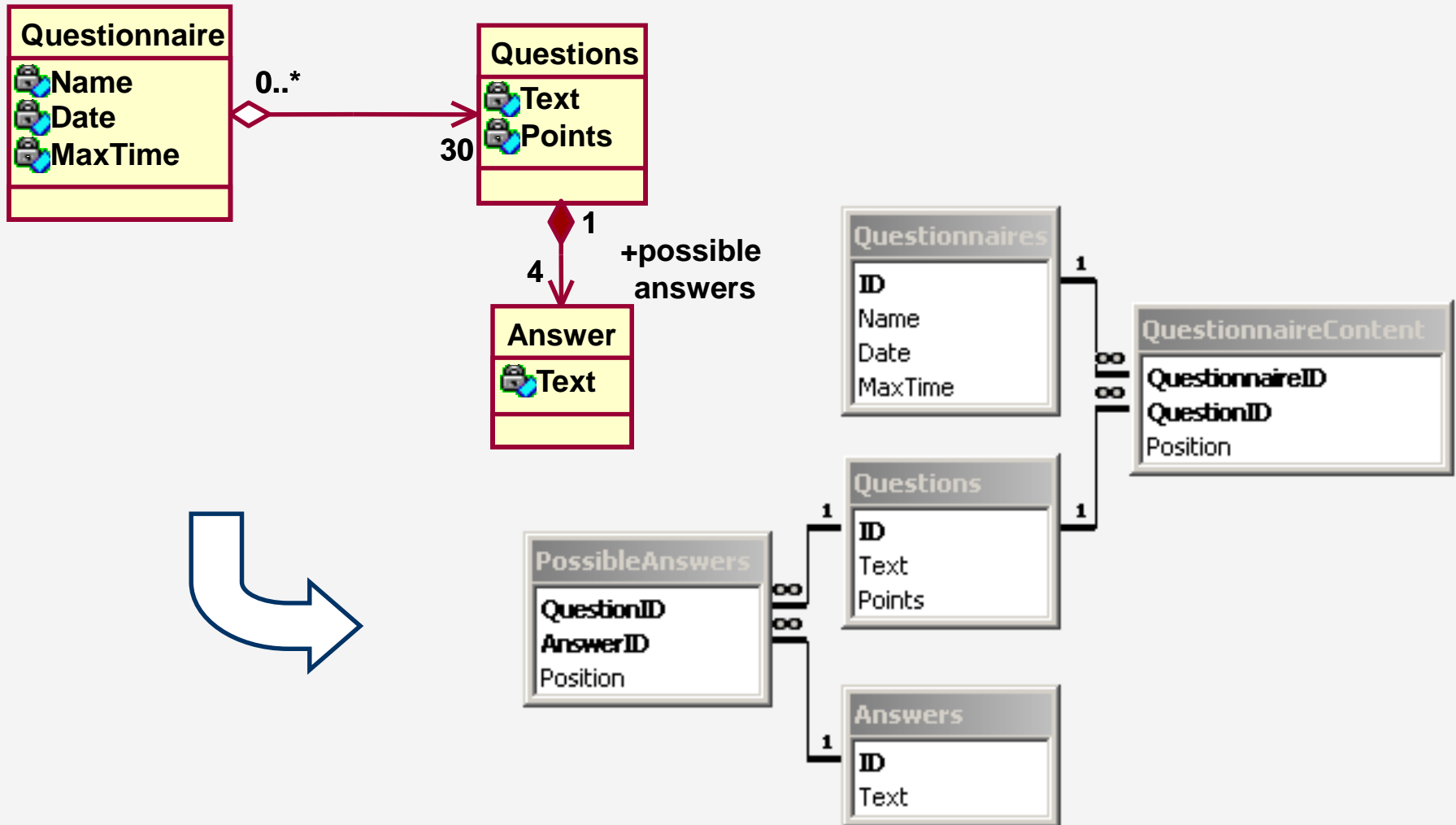
- Dacă superclasa are un număr restrâns de attribute (comparativ cu subclasele sale) atunci metoda potrivită este **Metoda 3**.

- Dacă subclasele au instanțe puține atunci cea mai bună este utilizarea **Metoda 2**.

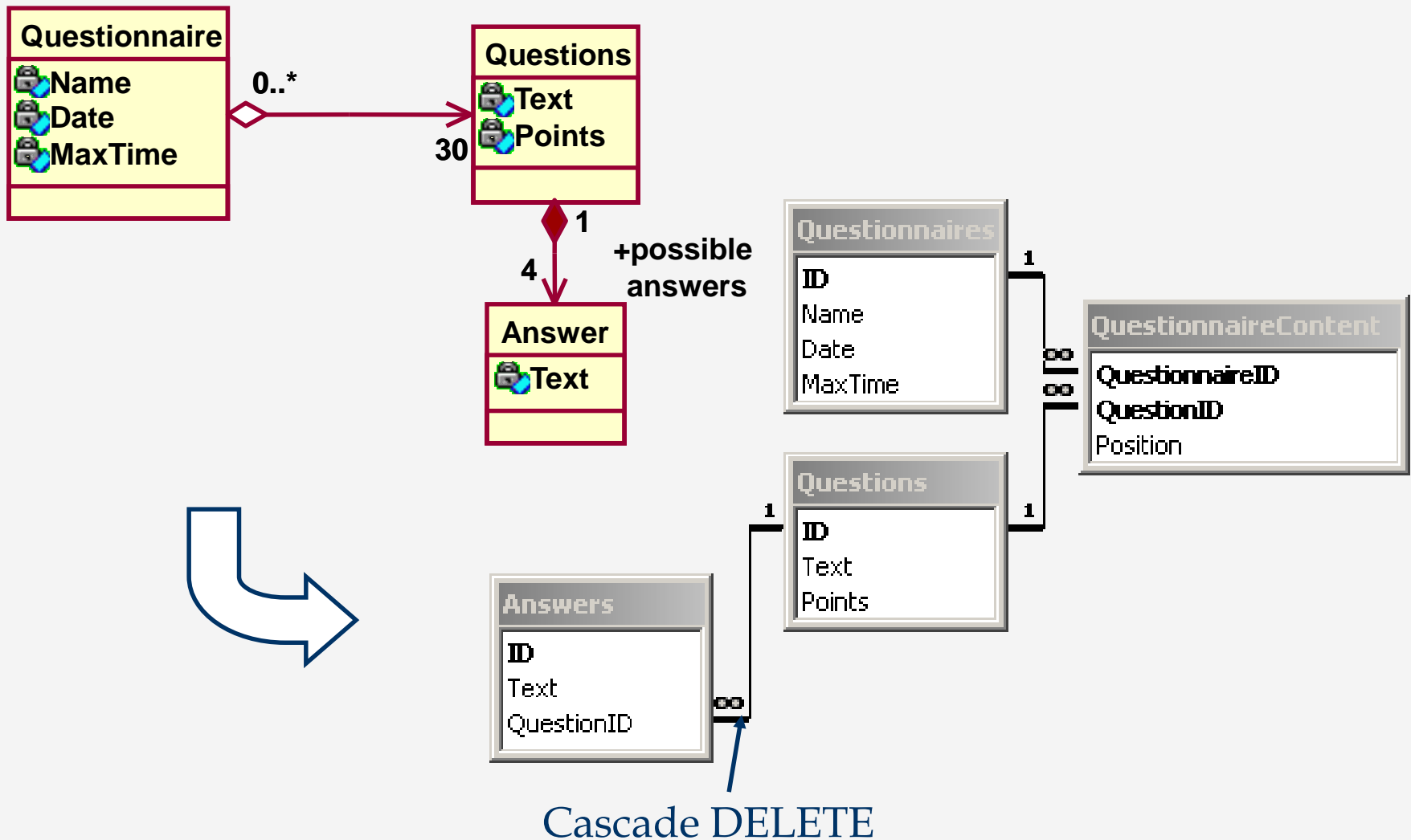
Transformarea agregării/compunerii

- Agregarea și compunerea sunt modelate în mod asemănător modelării asocierilor
- În cazul relațiilor de compunere de obicei se utilizează o singură tabelă (*cross-tables*) - deoarece compunerea implică mai multe relații 1:1
- Numărul fix de “părți” într-un “întreg” presupune introducerea unui număr egal de chei străine în tabela “întreg”
- În cazul implementării compunerii în tabele separate este necesară setarea “ștergerii în cascadă” (în cazul agregării acest lucru nu este necesar)

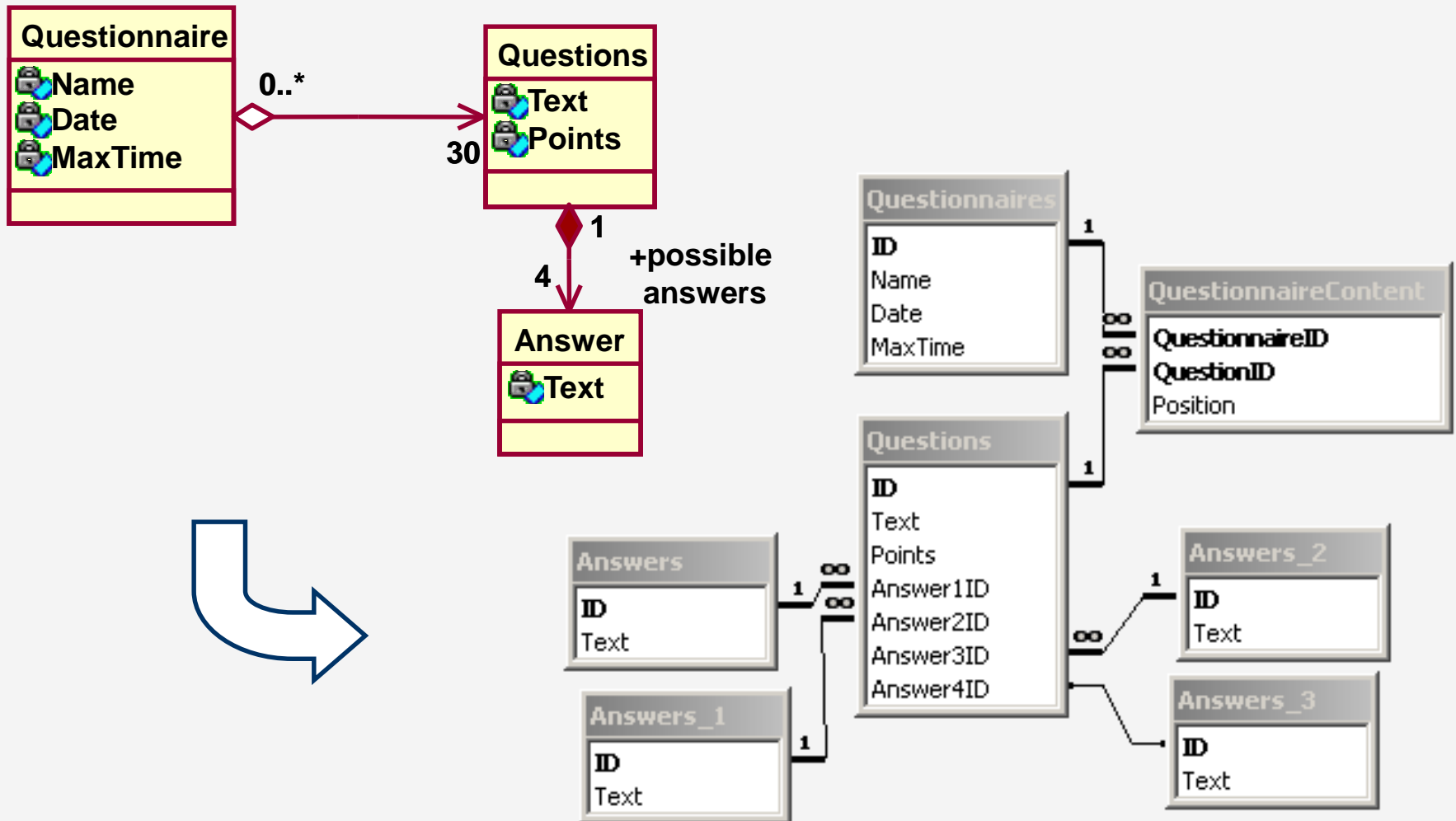
Transformarea agregării/compunerii



Transformarea agregării/compunerii

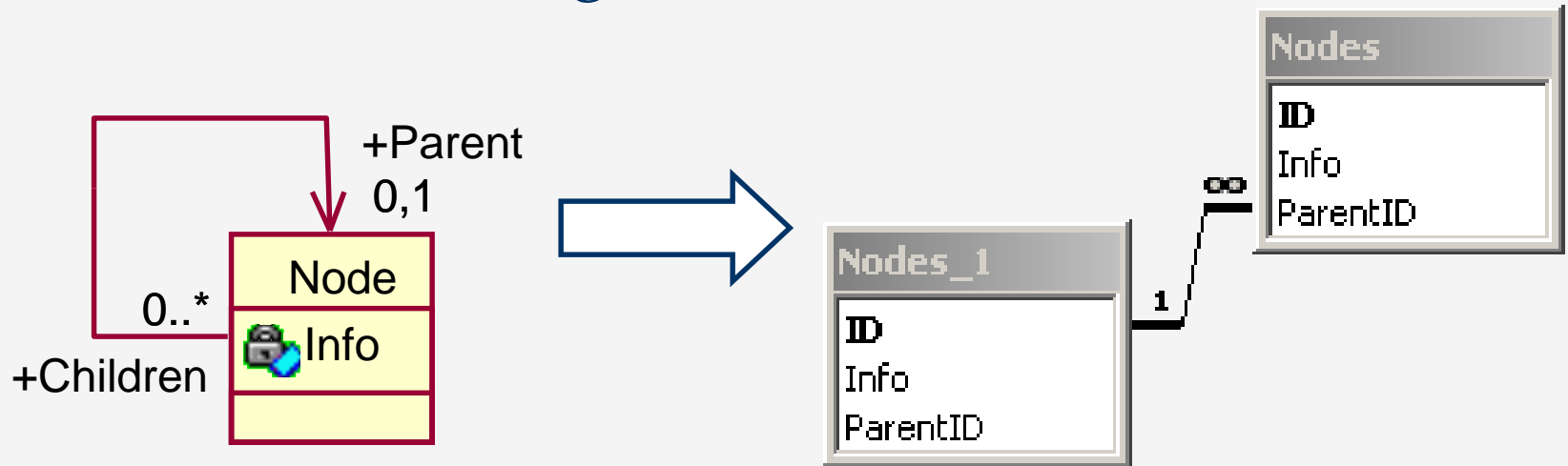


Transformarea agregării/compunerii



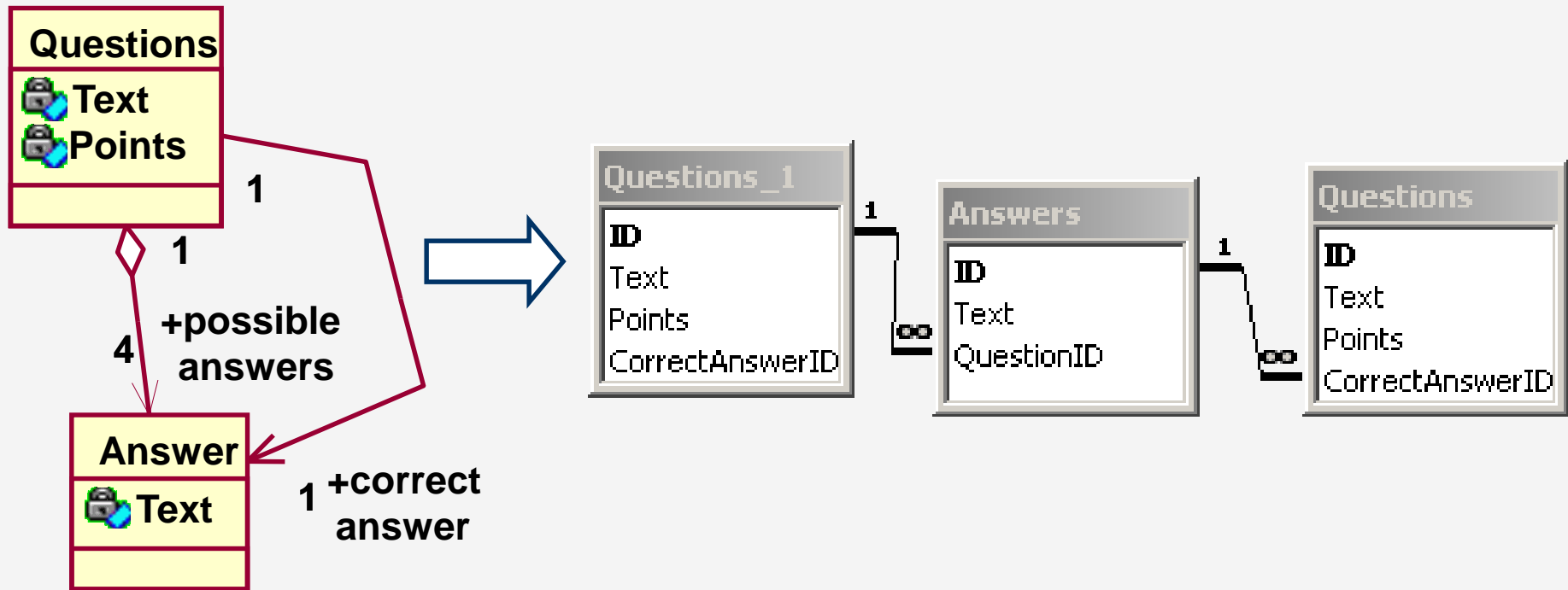
Transformarea auto-asocierilor

- Se introduce o cheie străină ce pointează spre aceeași (numit *relație recursivă*)
- Dacă este setată proprietatea ștergerii în cascadă există 2 înregistrări care se referă reciproc, ștergerea uneia dintre ele va genera o eroare



Transformarea auto-asocierilor

- “Ștergerea în cascadă” generează o problemă similară și în cazul a două tabele ce se referă reciproc



Generarea automata a bazelor de date

- CASE tool: instrument de modelare vizuală
- Automatizează anumiți pași privind translatarea diagramelor de clase în tabele relaționale.
 - Este necesară și intervenția manuală
- Object-Relational Mapping (ORM)
 - biblioteci/componente ce generează comenzi SQL de creare a tabelelor si manipulare a datelor
 - Hibernate (Java),
 - Entity Framework, NHibernate (C#),
 - Django ORM, SQLAlchemy (Python)