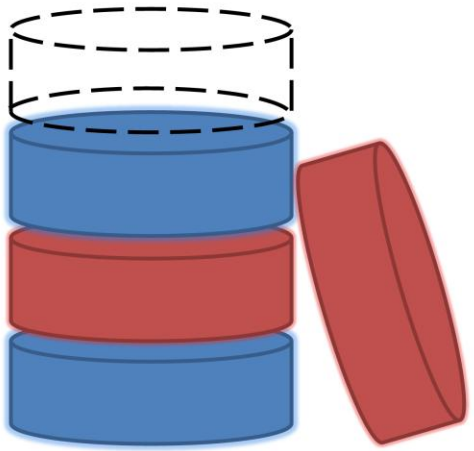
Forme normale

continuare



Dependențe multivaloare

course	teacher	book
alg101	Green	Alg Basics
alg101	Green	Alg Theory
alg101	Brown	Alg Basics
alg101	Brown	Alg Theory
logic203	Green	Logic B.
logic203	Green	Logic F.
logic203	Green	Logic intro.

relația e în BCNF

Dependențe multivaloare

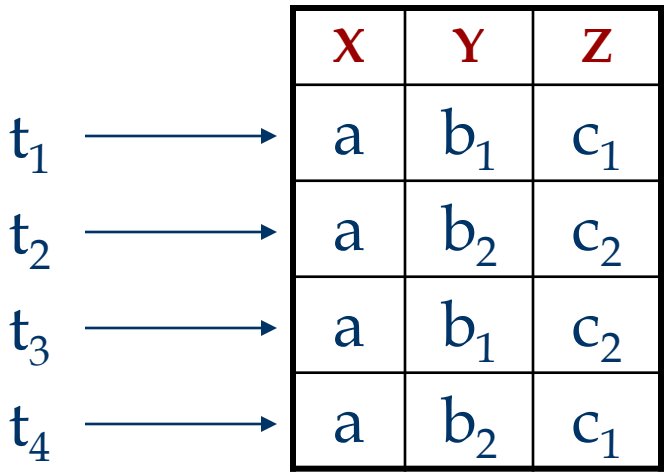
■ Fie α, β două submulțimi de atribute din R. Dependența multivaloare $\alpha \rightarrow\rightarrow \beta$ este respectată de R dacă, pentru fiecare instanță validă r a lui R, fiecare valoare α este asociată cu o mulțime de valori pentru β și această mulțime valori este independentă de valorile altor atribute.

■ Formal: dacă $\alpha \rightarrow\rightarrow \beta$ e respectată de R şi γ=R-$\alpha\beta$, următoarea afirmație e adevărată pentru orice instanță validă r a lui R:

$$t_1, t_2 \in r \text{ şi } \pi_\alpha(t_1) = \pi_\alpha(t_2) \Rightarrow$$
$$\exists t_3 \in r \text{ a.î. } \pi_{\alpha\beta}(t_1) = \pi_{\alpha\beta}(t_3) \text{ şi } \pi_\gamma(t_2) = \pi_\gamma(t_3)$$

Ca şi consecință, pentru t_2 şi t_1 se poate deduce că există şi $t_4 \in r$ a.î. $\pi_{\alpha\beta}(t_2) = \pi_{\alpha\beta}(t_4)$ şi $\pi_\gamma(t_1) = \pi_\gamma(t_4)$

Dependențe multivaloare



	X	Y	Z
t_1	a	b_1	c_1
t_2	a	b_2	c_2
t_3	a	b_1	c_2
t_4	a	b_2	c_1

$\forall t_1, t_2 \in r$ şi $\pi_X(t_1) = \pi_X(t_2) \Rightarrow$
$\exists\, t_3 \in r$ astfel încât
$\pi_{XY}(t_1) = \pi_{XY}(t_3)$,
$\pi_Z(t_2) = \pi_Z(t_3)$

Reguli adiționale:

Complementare: $X \rightarrow\!\!\rightarrow Y \Rightarrow X \rightarrow\!\!\rightarrow R - XY$

Augumentare: $X \rightarrow\!\!\rightarrow Y$, $Z \subseteq W \Rightarrow WX \rightarrow\!\!\rightarrow YZ$

Tranzitivitate: $X \rightarrow\!\!\rightarrow Y$, $Y \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow Z - Y$

Replicare: $X \rightarrow Y \Rightarrow X \rightarrow\!\!\rightarrow Y$

Fuzionare: $X \rightarrow\!\!\rightarrow Y$, $W \cap Y = \varnothing$, $W \rightarrow Z$, $Z \subseteq Y \Rightarrow X \rightarrow Z$

A patra formă normală (4NF)

Definiție. Fie R o schemă relațională şi F o mulțime de dependențe funcționale şi multivaloare pe R.

Spunem că R este în a patra forma normală NF4 dacă este în 3NF și pentru orice dependență multivaloare $X \twoheadrightarrow Y$:

- $Y \subseteq X$ sau
- $XY = R$ sau
- X e super-cheie

A patra formă normală (4NF)

course	teacher	book
alg101	Green	Alg Basics
alg101	Green	Alg Theory
alg101	Brown	Alg Basics
alg101	Brown	Alg Theory
logic203	Green	Logic B.
logic203	Green	Logic F.
logic203	Green	Logic intro.

course→→teacher

Relatia se poate descompune in:
(Course,Teacher)
si (Course,Book)

course	teacher
alg101	Green
alg101	Brown
logic203	Green

course	book
alg101	Alg Basics
alg101	Alg Theory
logic203	Logic B.
logic203	Logic F.
logic203	Logic intro.

Dependența *Join*

- Spunem ca R satisface dependența *join* $\otimes\{R_1,\ldots,R_n\}$ dacă $R_1, R_2,\ldots,R_n$ este o descompunere cu joncțiuni fără pierderi a lui R.

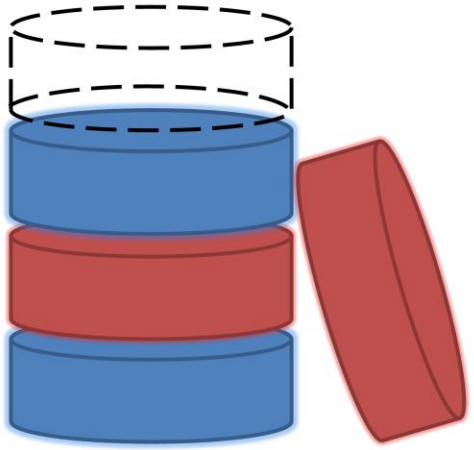
O dependență multivaloare $X \rightarrow\rightarrow Y$ poate fi exprimată ca o dependență join: $\otimes\{XY, X(R-Y)\}$.

A cincea formă normală (5NF)

O relație R este în NF5 dacă şi numai dacă pentru orice dependență *join* a lui R:

- R_i=R pentru un *i* oarecare, sau
- dependența este implicată de o mulțime de dependențe functionale din R în care partea stângă e o cheie pentru R

Proiectarea bazelor de date



5

Proiectarea bazelor de date

- Proiectare conceptuală (ex. diagrama de clase)
  - Identificarea entităților şi 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 şi a problemelor conexe
- Proiectare fizică şi eficientizare
  - Indexare
  - De-normalizare!

Diagrama de clase UML - Clase

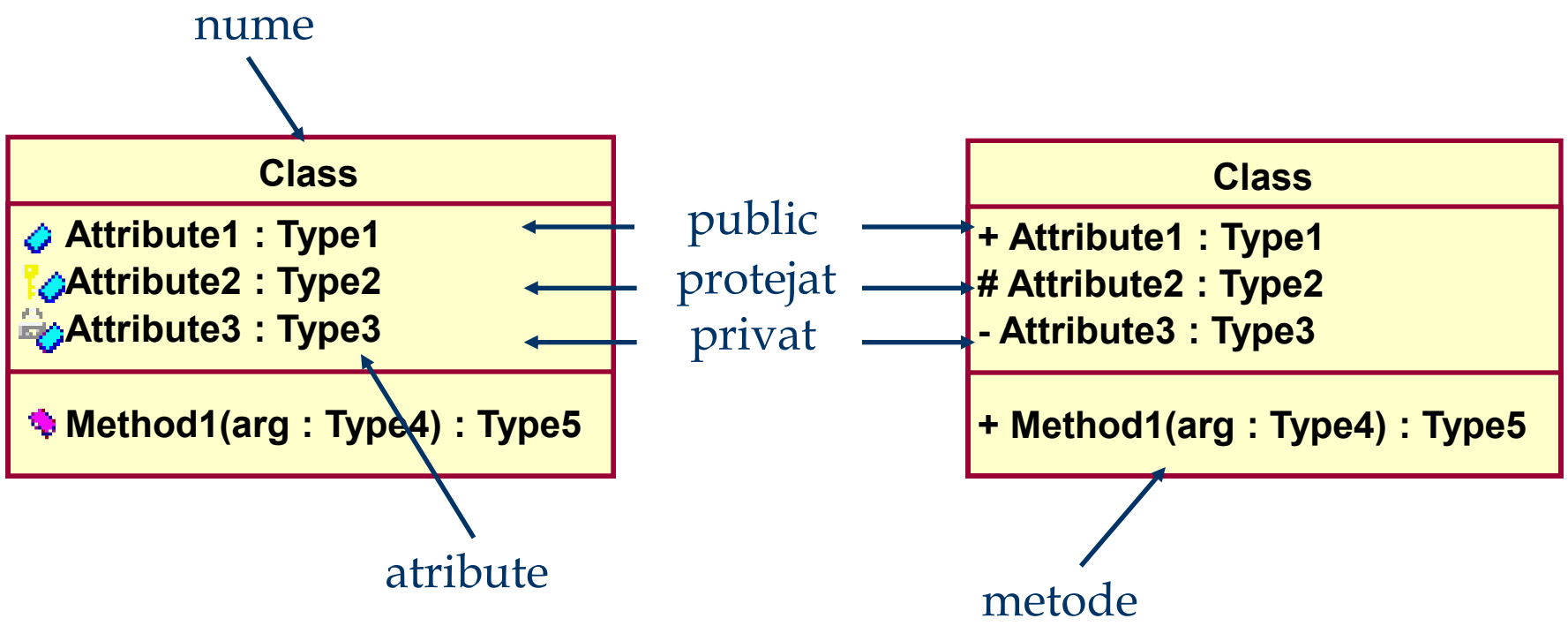
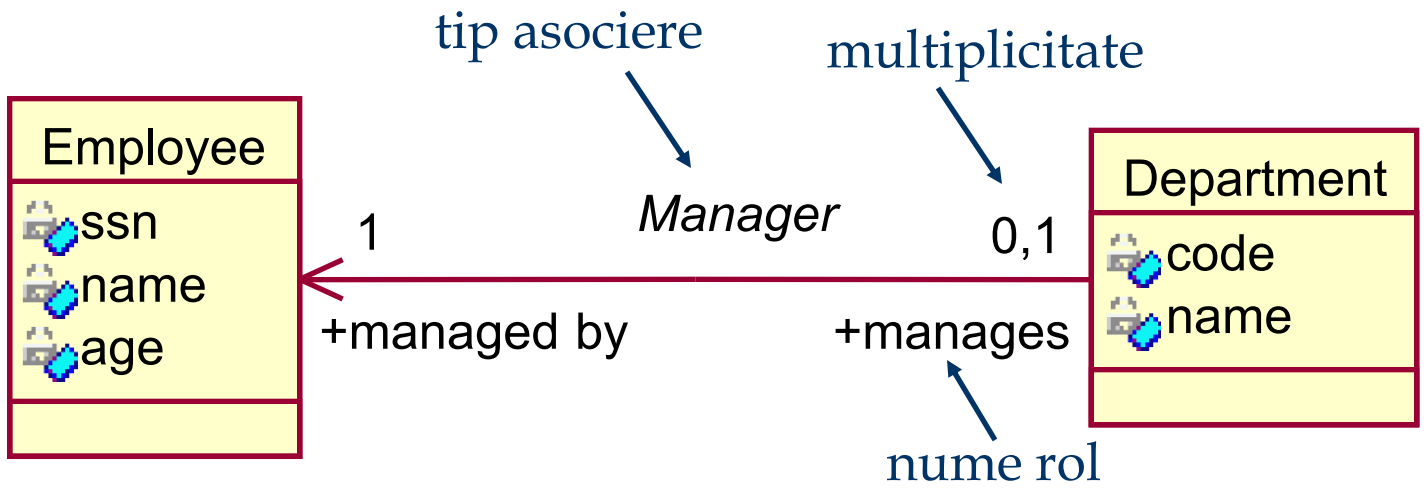


Diagrama de clase UML - Asocieri



- Multiplicități:
  - valori: 4,5
  - intervale: 1..10
  - nedefinit: *
- Navigabilitatea asocierii:
  - un sens
    - bidirectional

Diagrama de clase UML - Asocieri

- Citirea numelor de rol

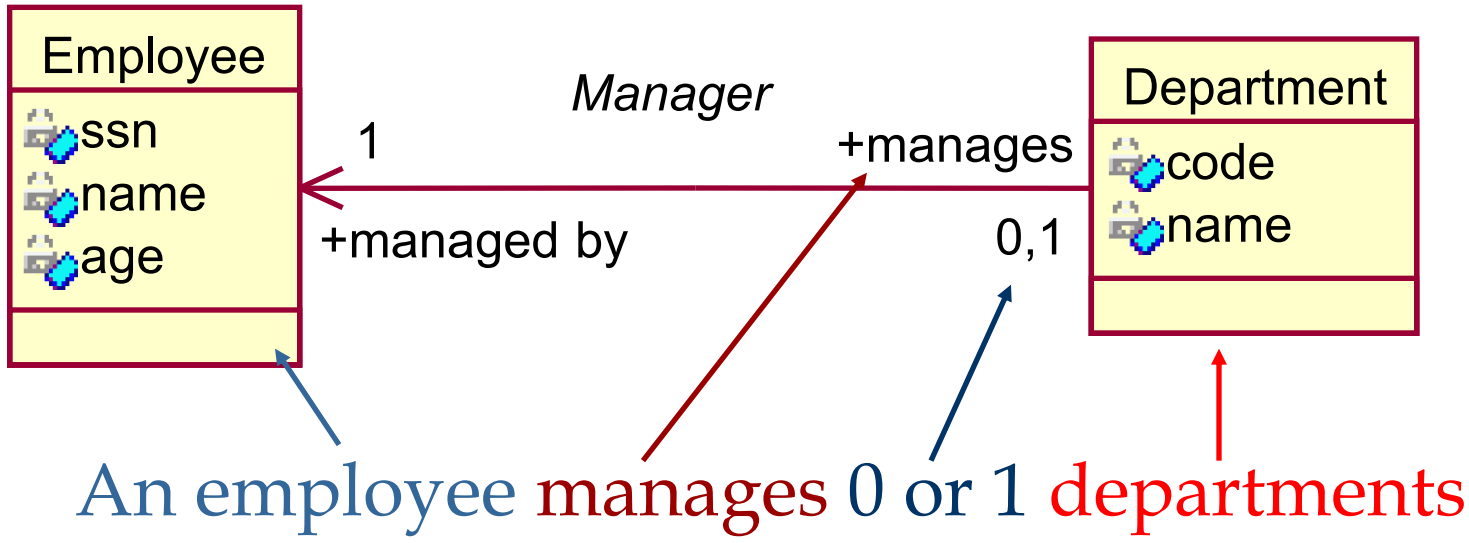


Diagrama de clase UML - Asocieri

- Agregare
  - asociere parte-intreg
- Compunere
  - "weak entities"
- Clasa asociere
- Asociere reflexiva

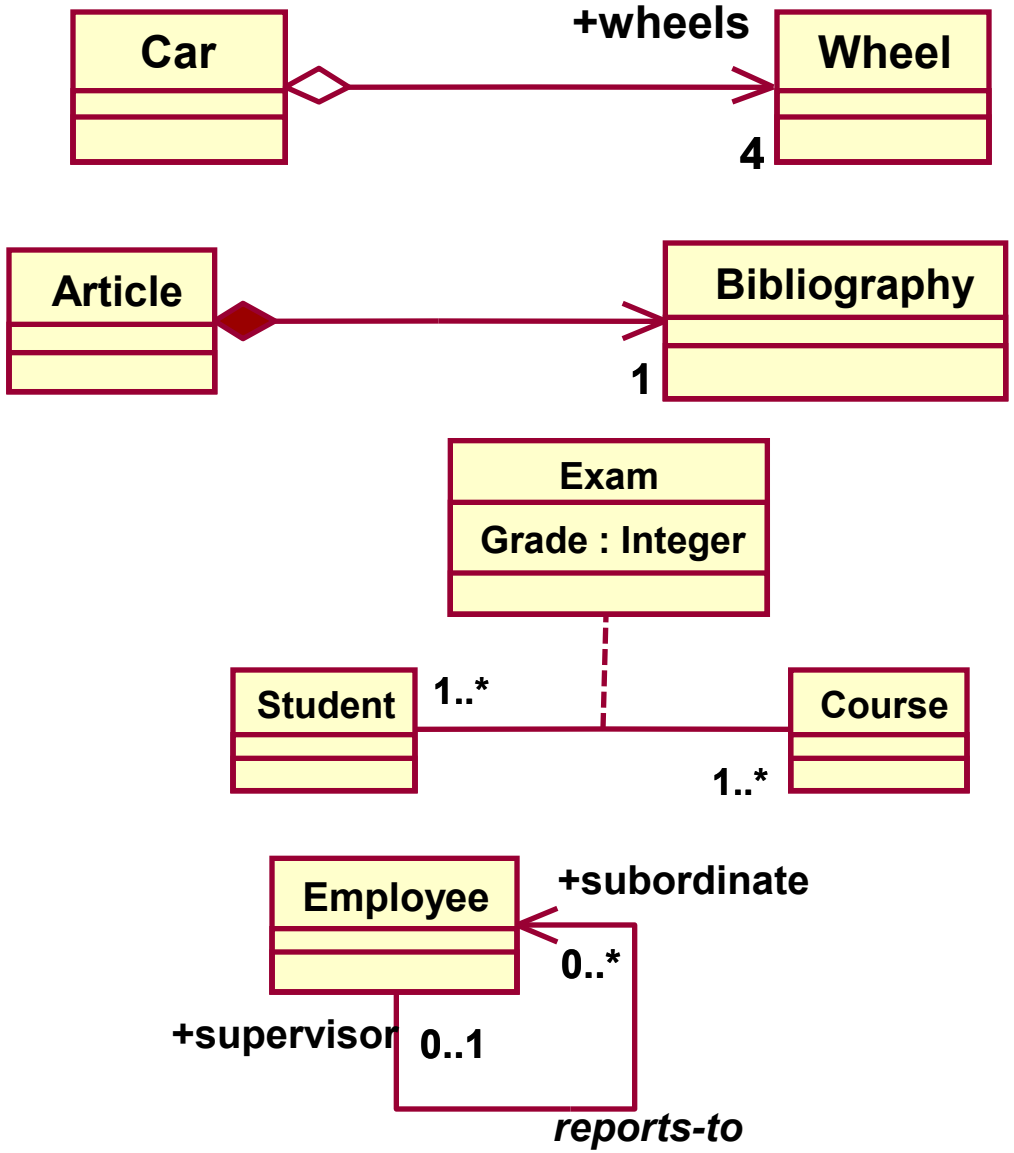
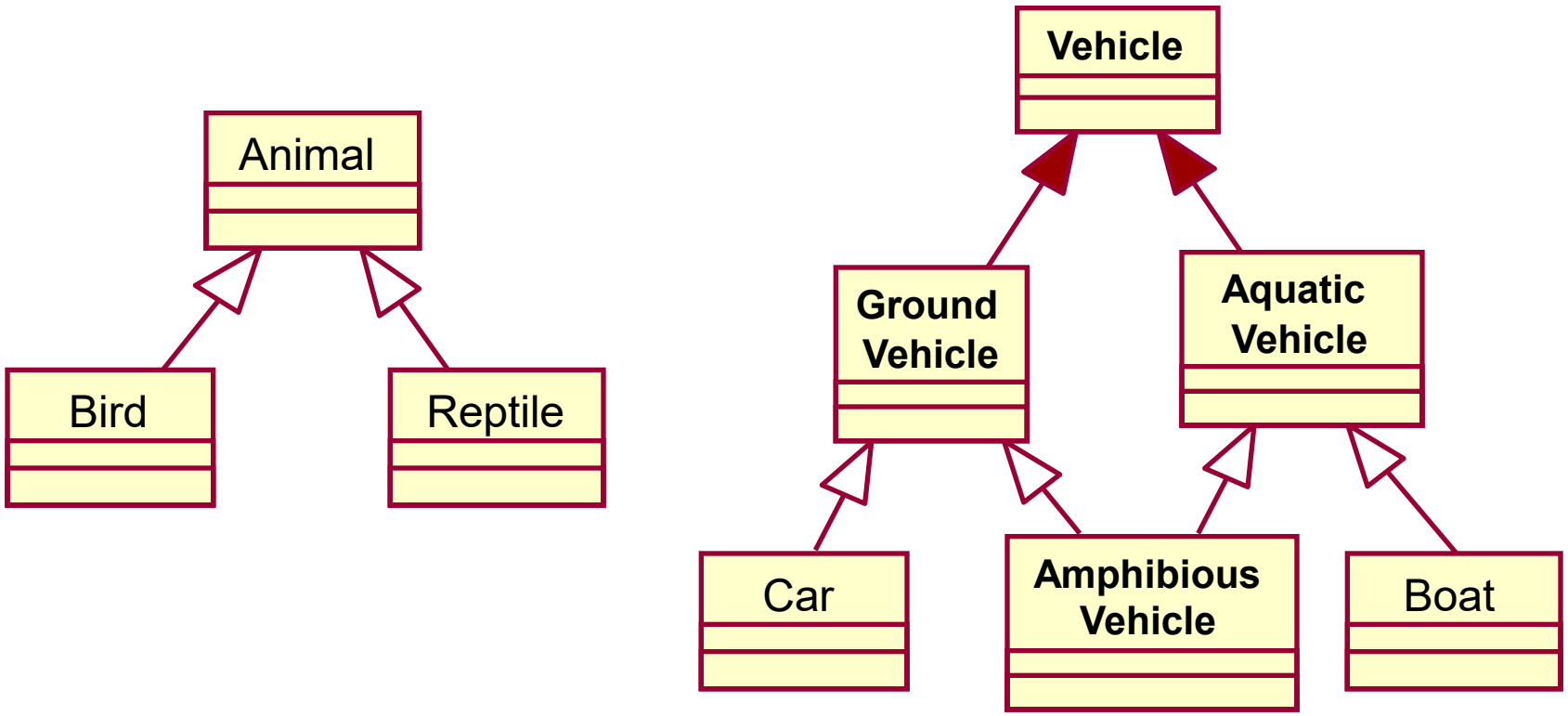


Diagrama de clase UML - Mostenire

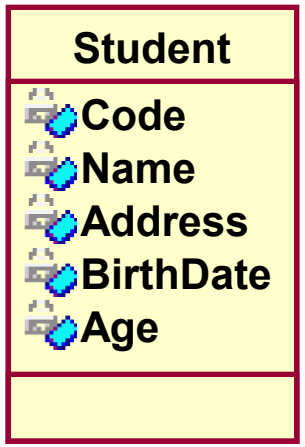


Modelul conceptual ⇒ bază de date relațională

- 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 moştenirii
  - Denormalizarea datelor – anumite date se regăsesc în mai multe tabele

Transformarea claselor în tabele

- Numele tabelei 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



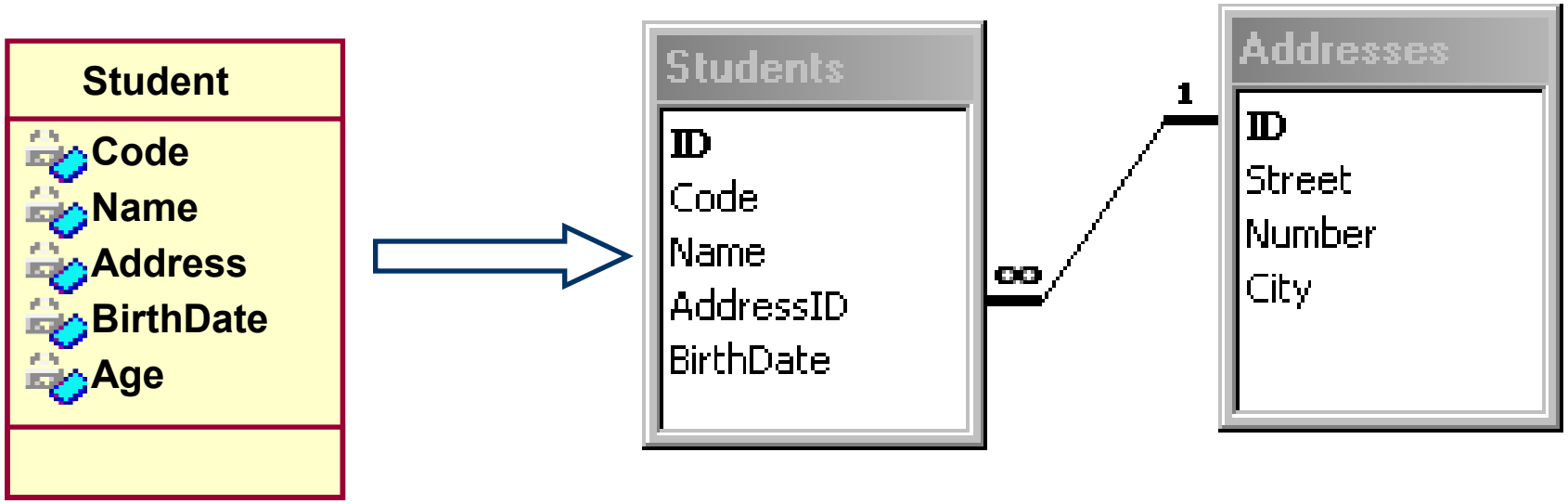
Students (Code, Name, BirthDate)

Addresses (Street, Number, City)

Transformarea claselor în tabele

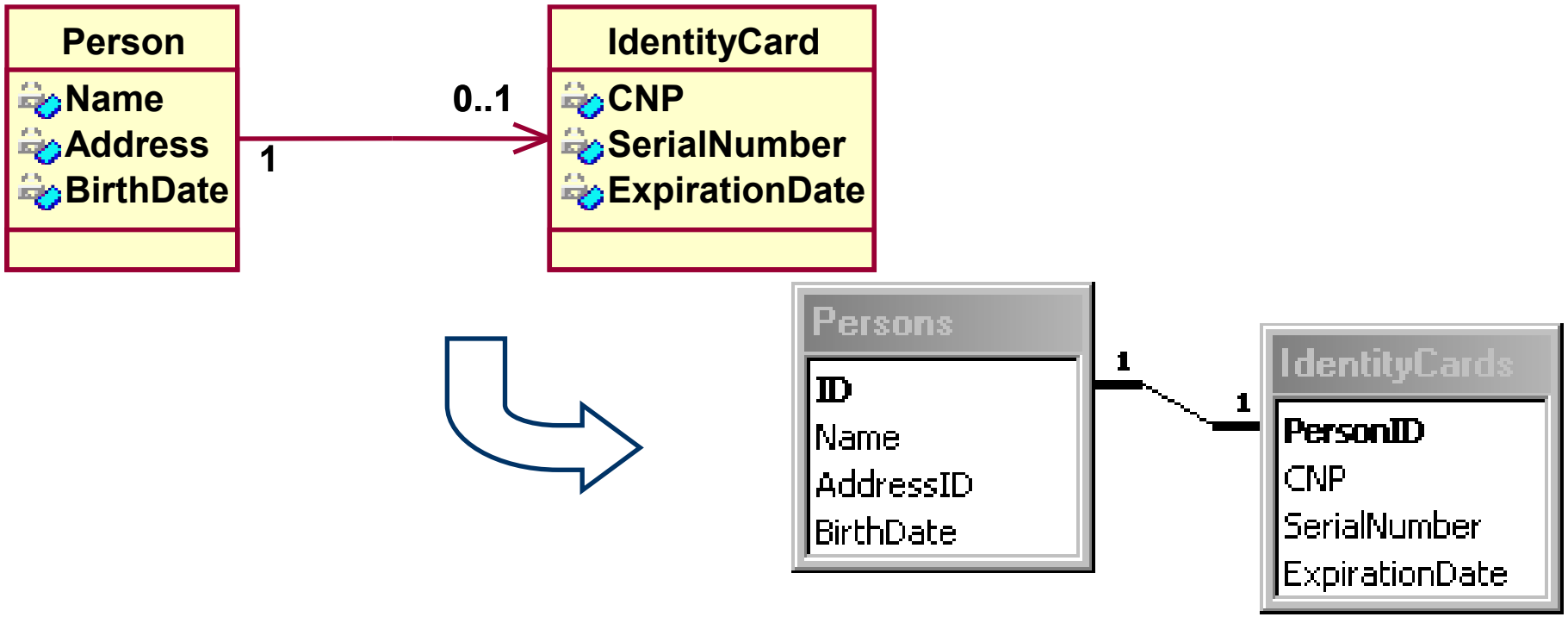
- Chei surogat – chei care nu sunt obținute din domeniul problemei modelate
- Conceptul de cheie nu este definit în cadrul claselor UML
- O *bună practică*: utilizarea (atunci când este posibil) a cheilor de tip întreg generate automat de SGBD:
  - uşor de întreținut (responsabilitatea sistemului)
  - eficient (interogări rapide)
  - simplifică definire 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)



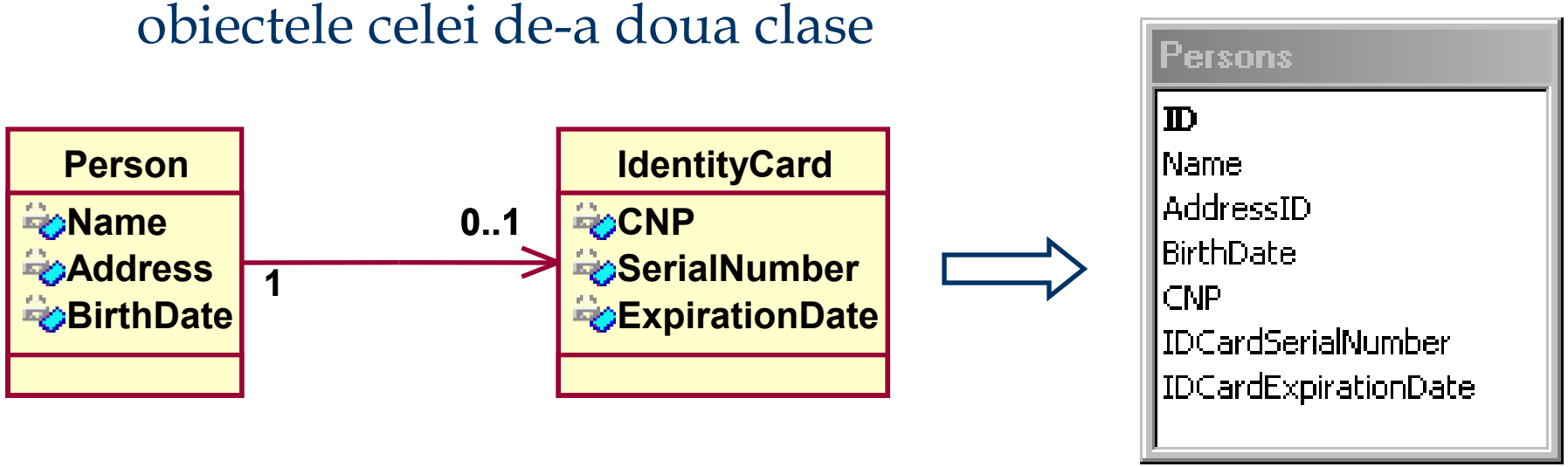
Transformarea asocierilor simple

- 1 : 0,1
  - se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
  - cheia tabelei 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")



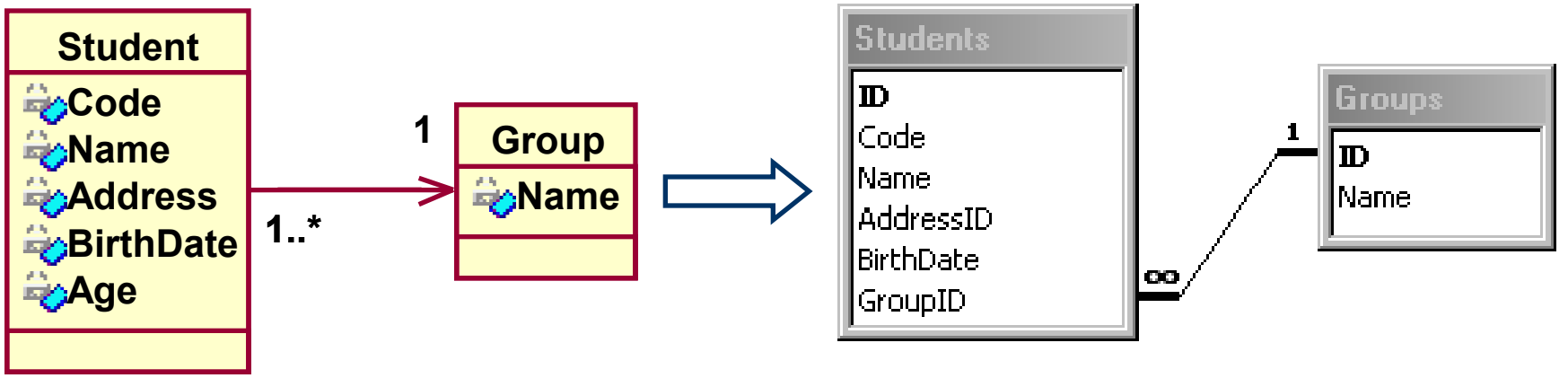
Transformarea asocierilor simple (cont)

- 1 : 1
  - se crează o singură tabelă ce conține atributele ambelor clase asociate
  - aceasta variantă de transformare se aplică şi asocierilor *"1 : 0,1"* atunci când este vorba de un număr relativ mic de cazuri in care obiectele primei clase nu sunt legate de obiectele celei de-a doua clase



Transformarea asocierilor simple (cont)

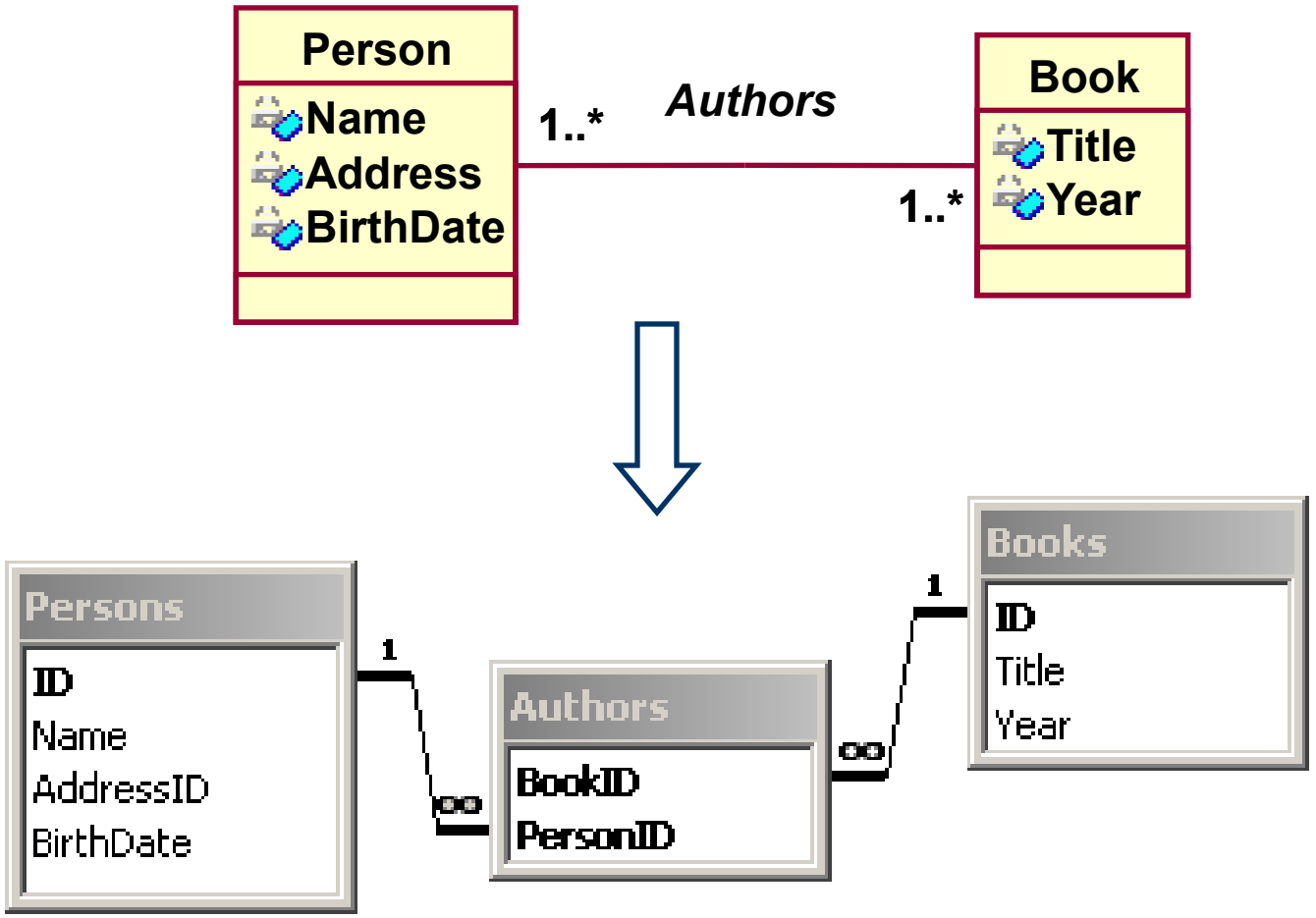
- 1 : 1..*
  - se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
  - cheia tabelei corespunzătoare multiplicității " 1" este cheia străină în cea de-a doua tabelă, corespunzătoare multiplicității "1..*"



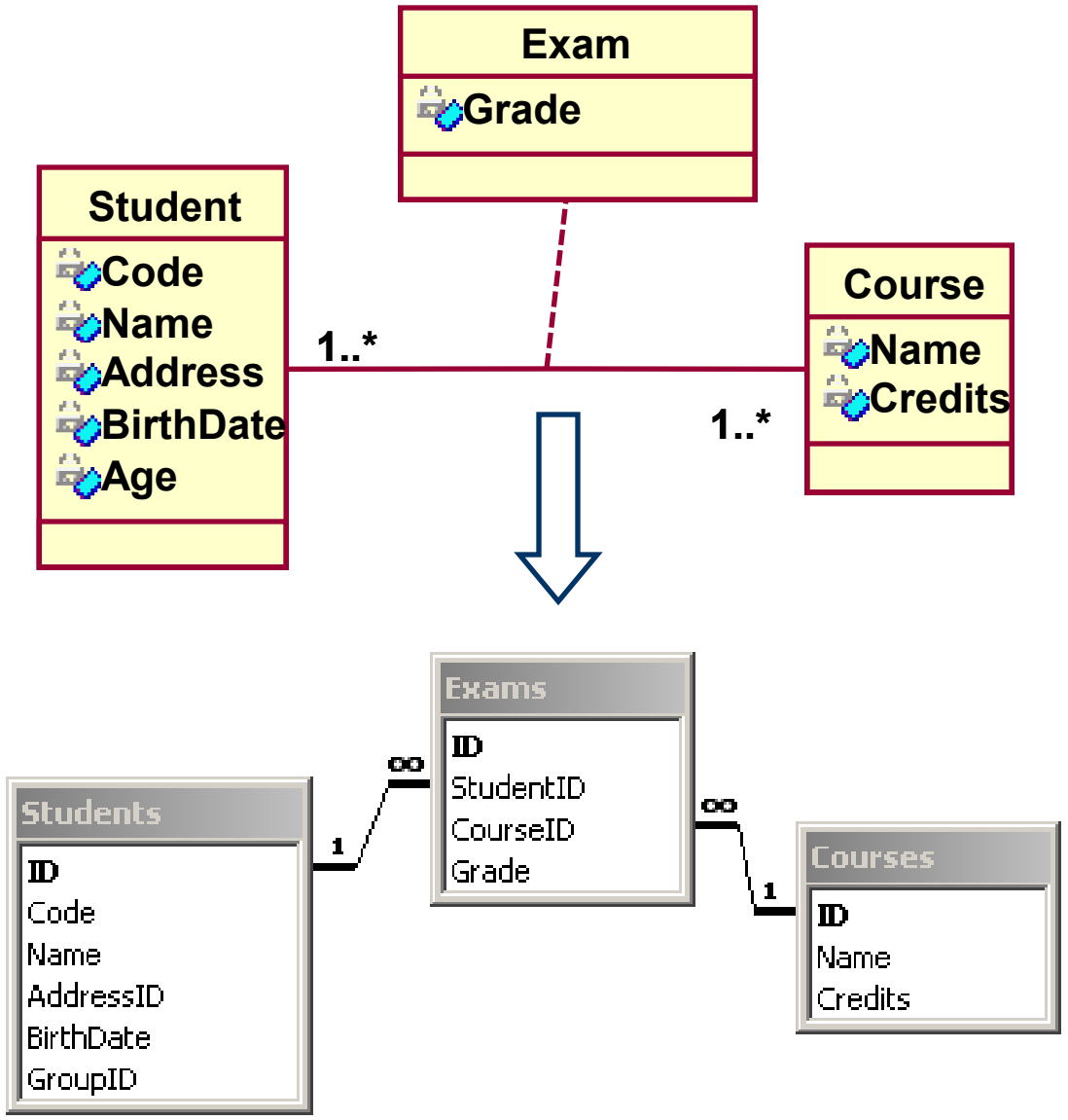
Transformarea asocierilor simple (cont)

- 1..* : 1..*
  - se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
  - se crează o tabelă adiţională numită tabelă de intersecţie (*cross table*)
  - cheile primare corespunzătoare tabelelor iniţiale sunt definite ca şi chei străine în tabela de intersecţie
  - cheia primară a tabelei de intersecţie este, de obicei, compusă din cele două chei străine spre celelate tabele. Sunt cazuri în care se utilizează şi aici cheie surogat.
  - dacă asocierea conţine o clasă asociere, toate atributele acestei clase vor fi inserate în tabela de intersecţie
  - uzual, numele tabelei de intersecţie este o combinaţie a numelor tabelelor iniţiale dar acest lucru nu este necesar.

Transformarea asocierilor simple (cont)



Transformarea asocierilor simple (cont)



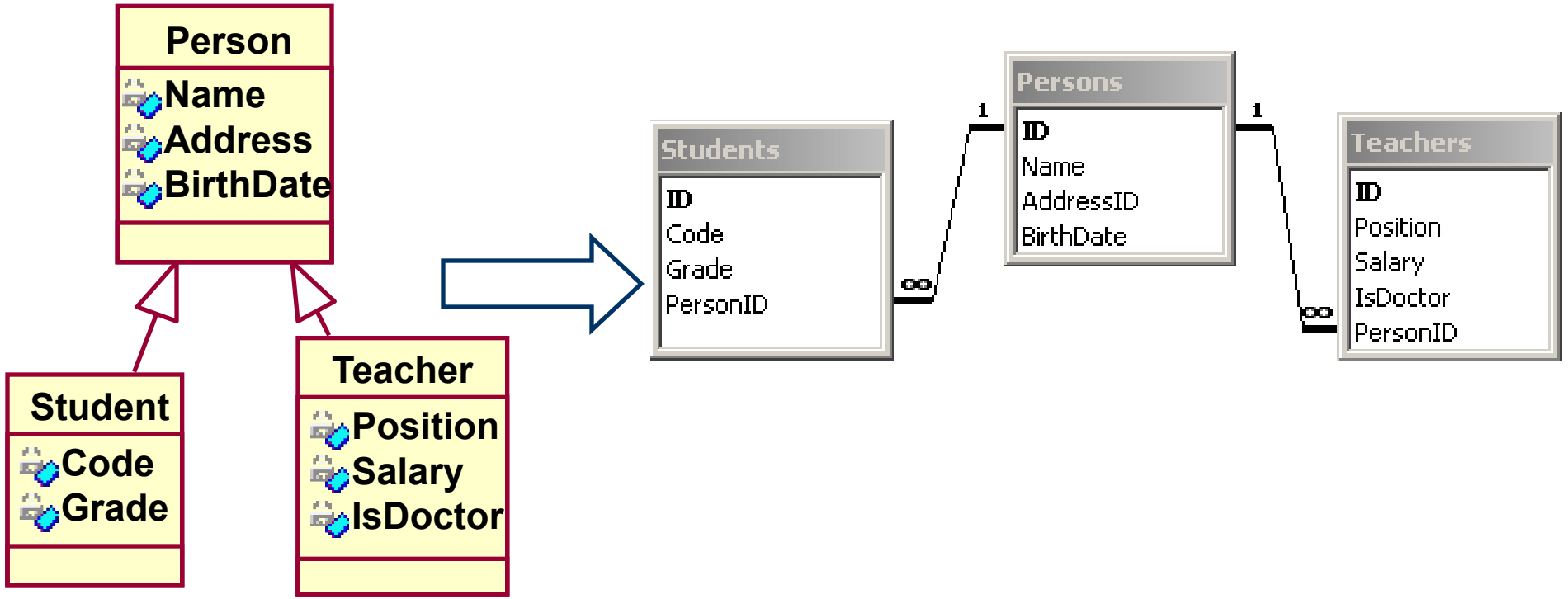
Transformarea moştenirii

Metoda 1

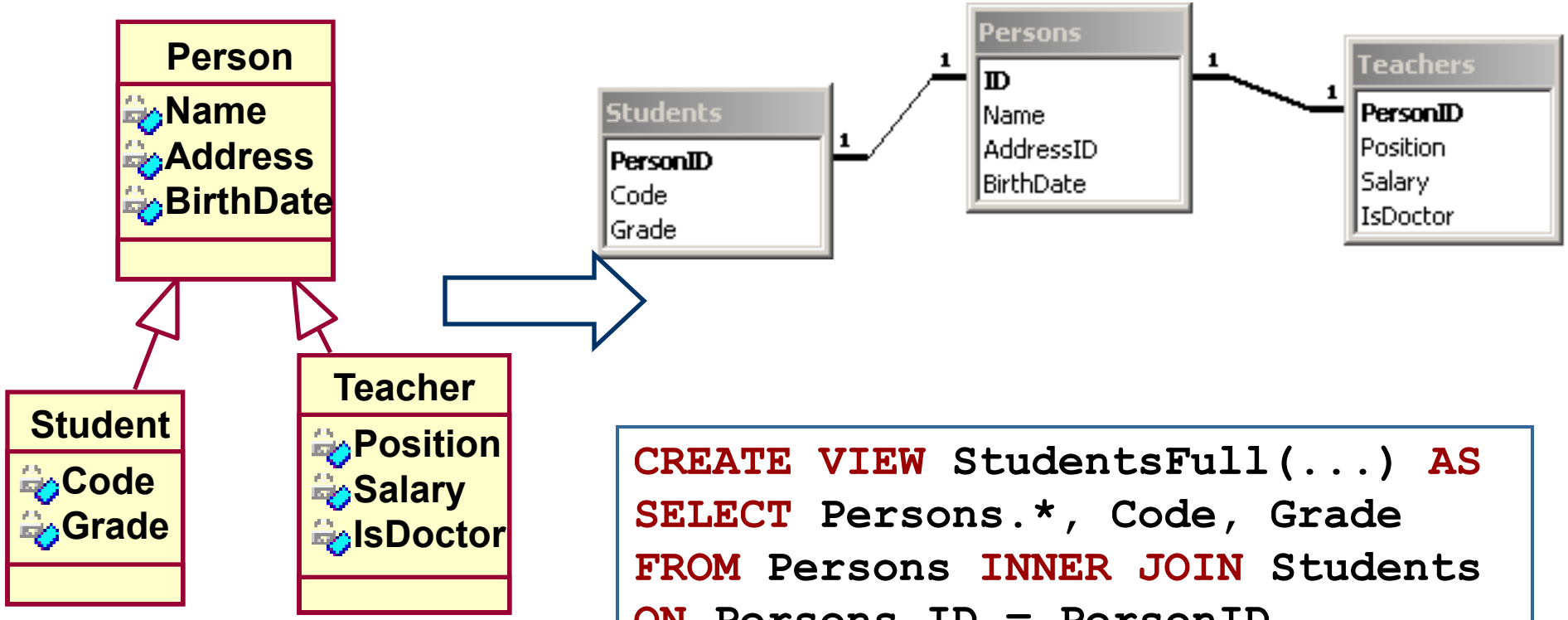
Presupune crearea câte unui tabel corespunzător fiecărei clase şi a câte unui *view* pentru fiecare pereche super-clasă/subclasă

- 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 moştenirii



Transformarea moştenirii




```
CREATE VIEW StudentsFull(...) AS
SELECT Persons.*, Code, Grade
FROM Persons INNER JOIN Students
ON Persons.ID = PersonID
```

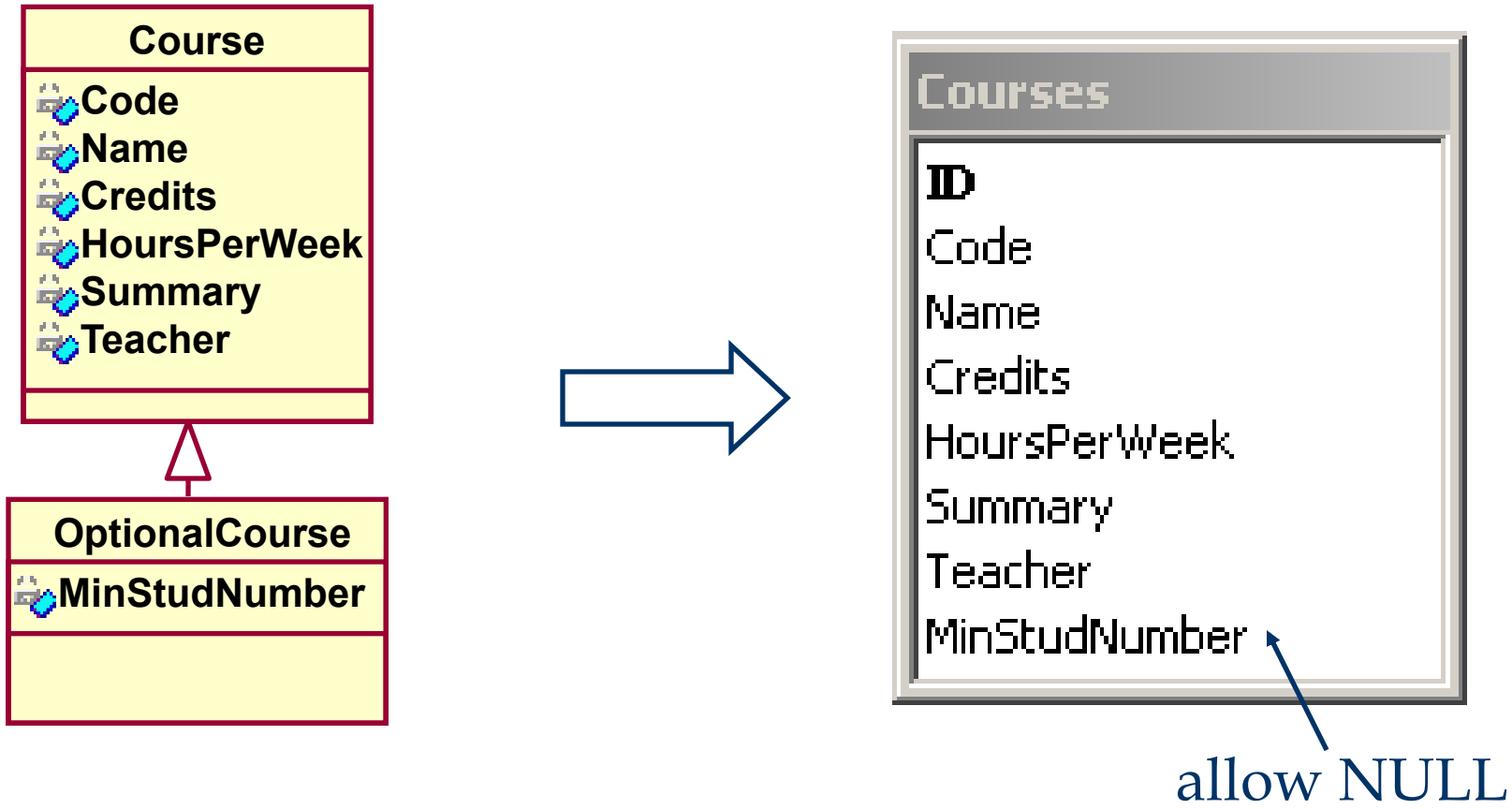
Transformarea moştenirii

Metoda 2

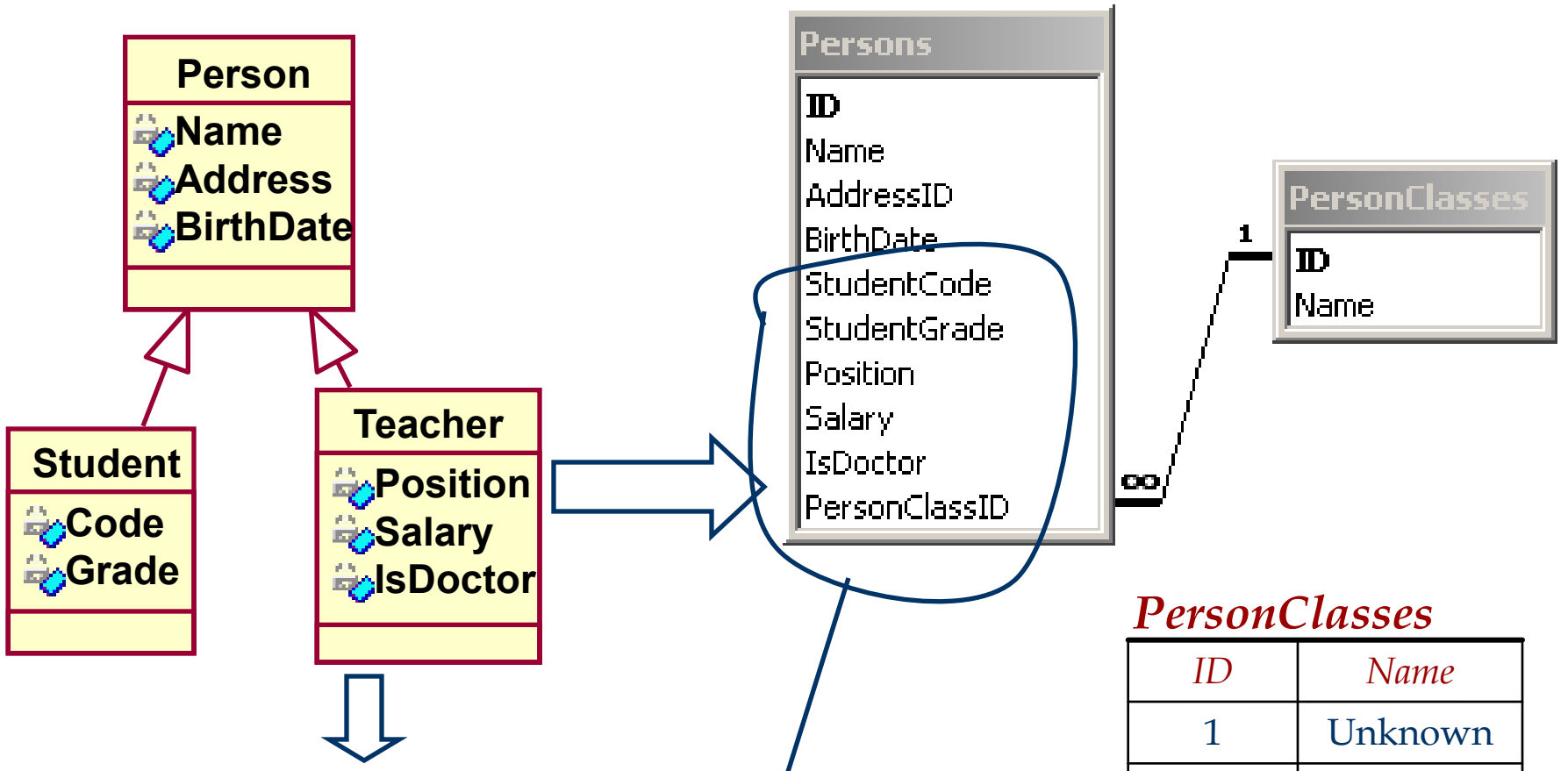
Se crează o singură tabelă (corespunzătoare superclasei) şi se de-normalizează toate atributele subclaselor acesteia.

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

Transformarea moştenirii



Transformarea moştenirii




```
CREATE VIEW Students(...) AS
SELECT ID,Name,AddressID,
BirthDate,Code,Grade FROM Persons
WHERE PersonClassID = 2
```

PersonClasses

ID	*Name*
1	Unknown
2	Student
3	Teacher

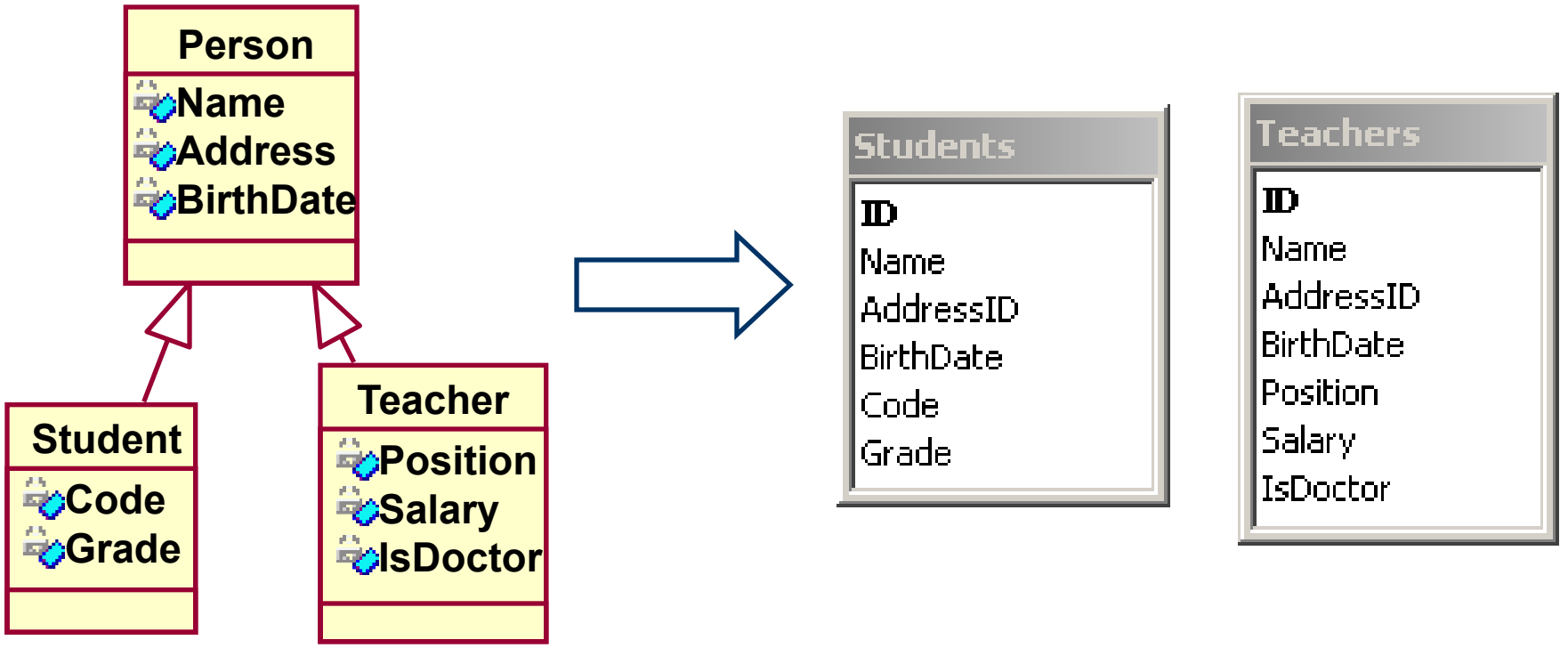
Transformarea moştenirii

Metoda 3

Presupune crearea câte unui tabel corespunzător fiecărei sub-clase şi 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 affecteaza toate tabelele definite!

Transformarea moştenirii



Transformarea moştenirii

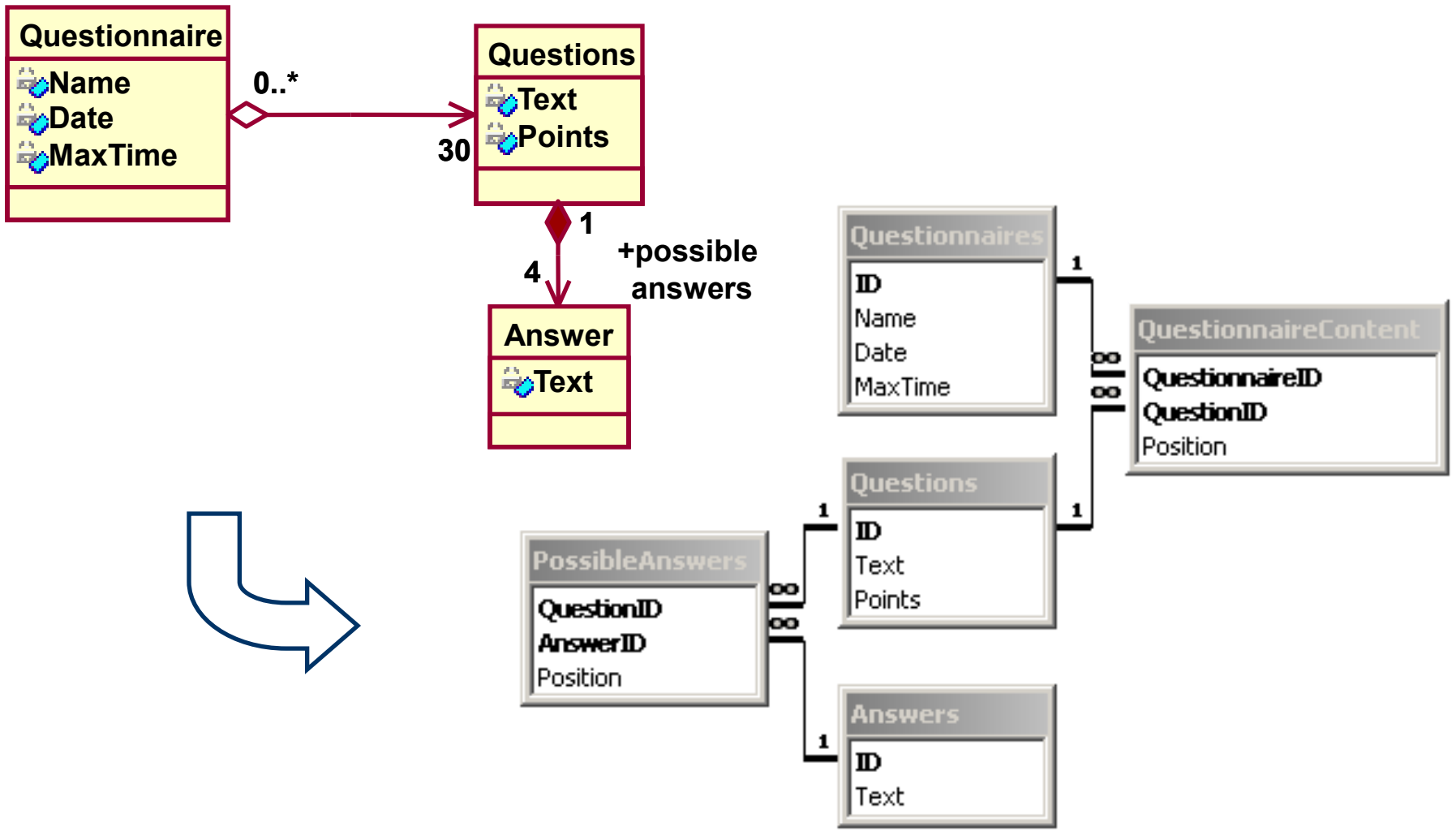
Care este metoda potrivită?

- 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 atribute (comparativ cu subclasele sale) atunci metoda potrivită este Metoda 3.
- Dacă subclasele au instanțe puține atunci cea mai bună este utilizarea Metoda 2.

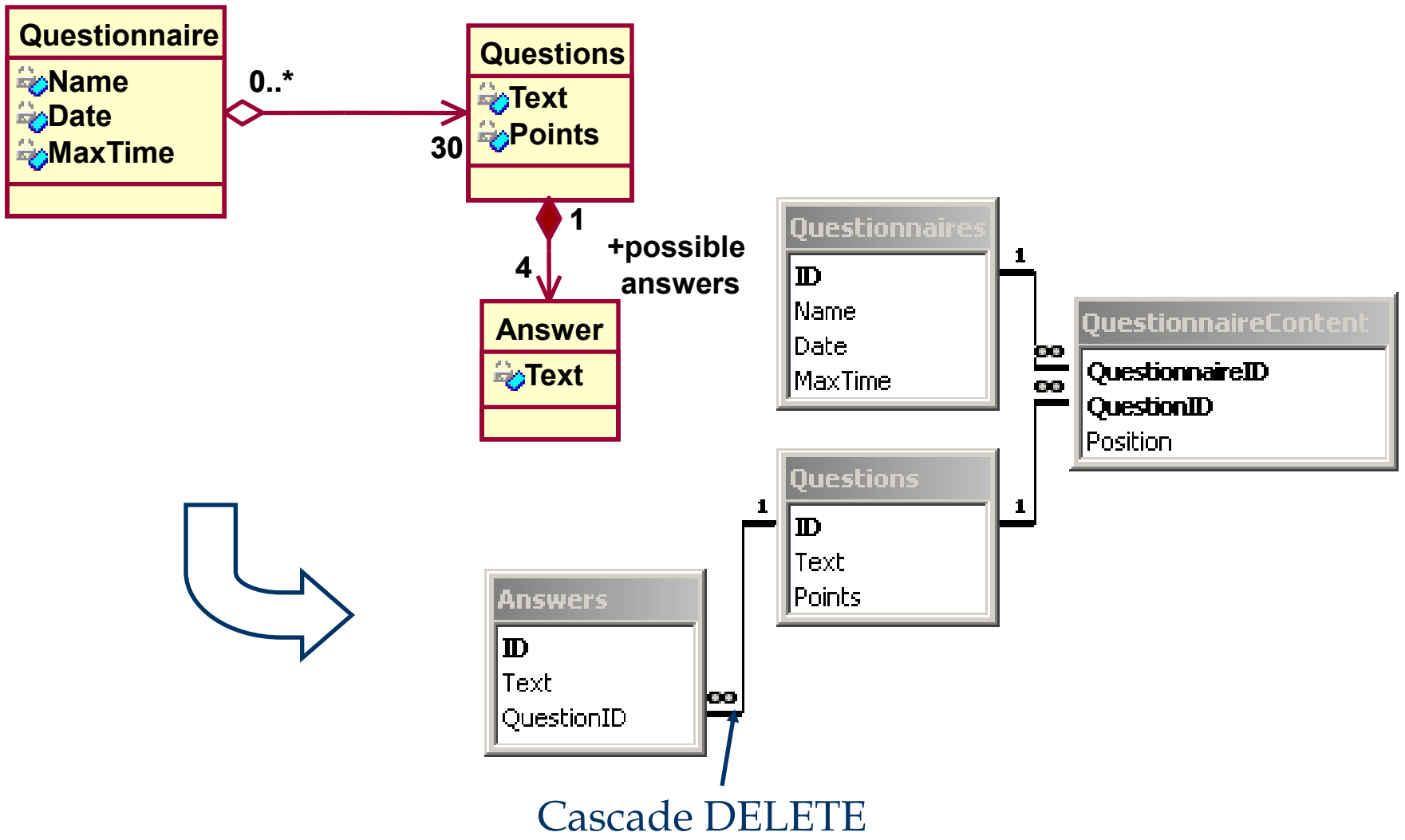
Transformarea agregării/compunerii

- Agregarea şi 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 "ştergerii în cascadă" (în cazul agregării acest lucru nu este necesar)

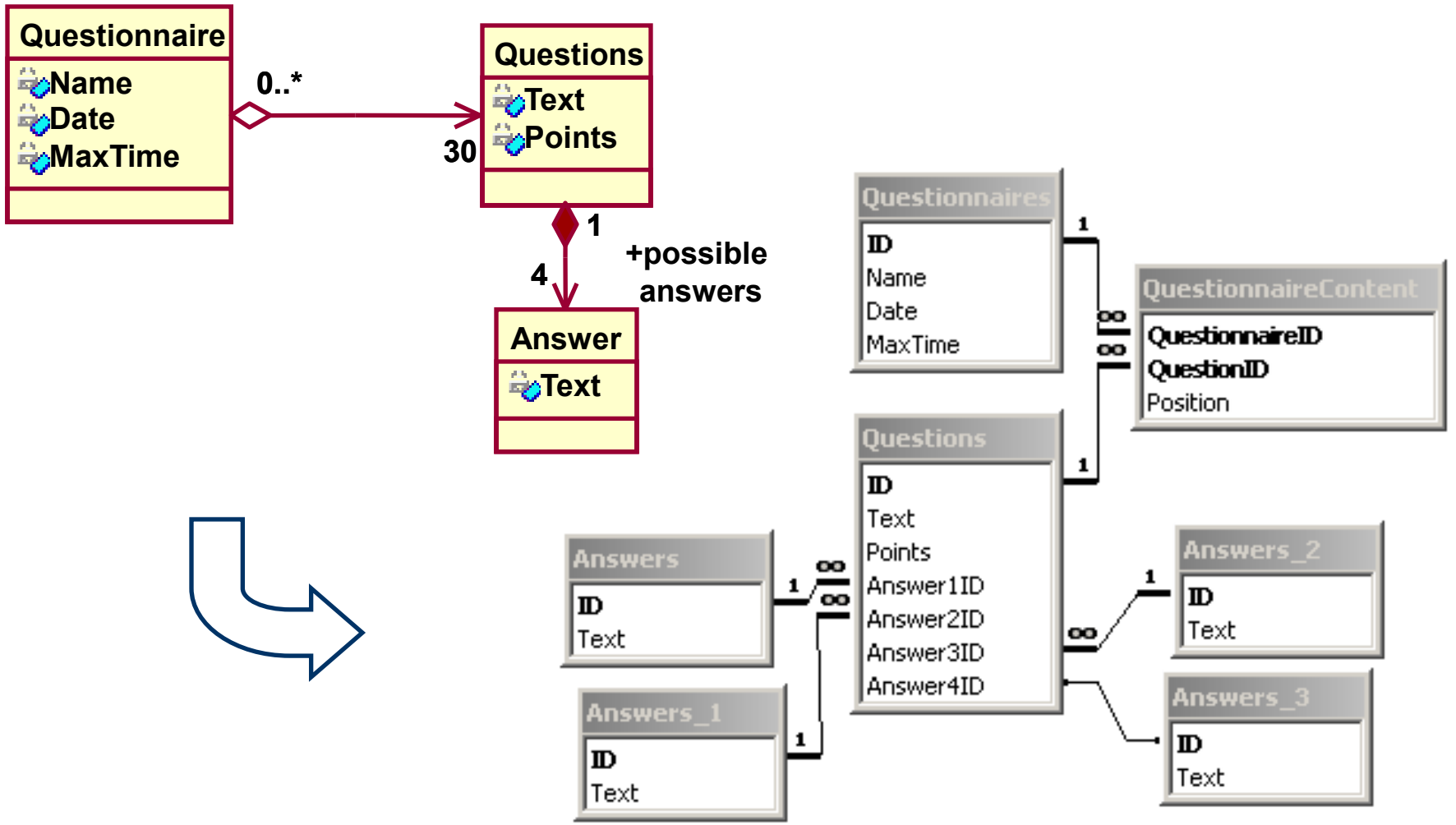
Transformarea agregării/compunerii



Transformarea agregării/compunerii

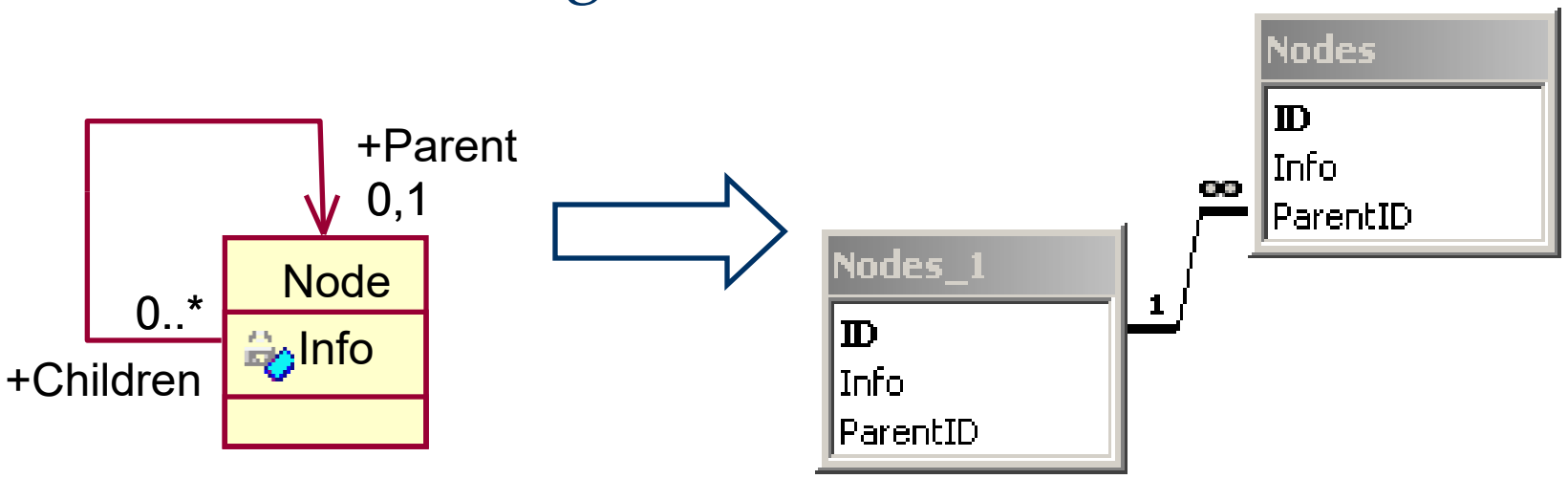


Transformarea agregării/compunerii



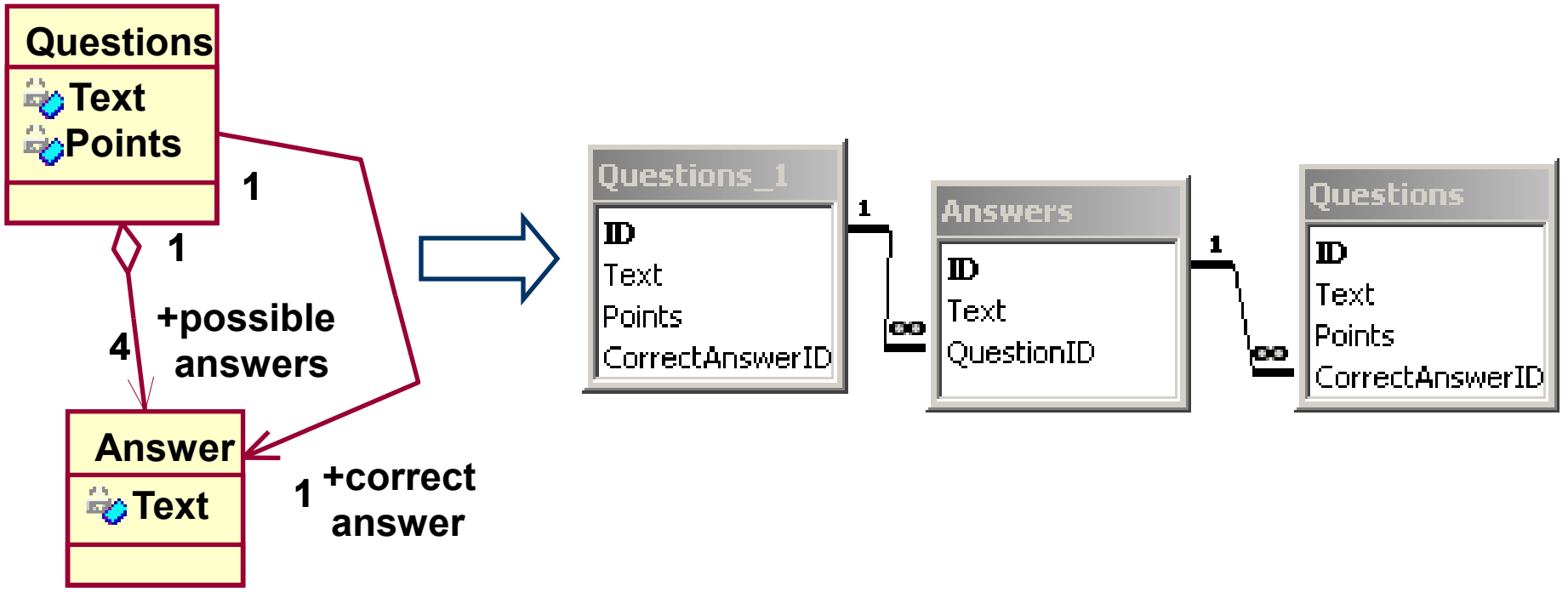
Transformarea auto-asocierilor

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



Transformarea auto-asocierilor

- "Ştergerea în cascadă" generează o problemă similară şi în cazul a două tabele ce se referă reciproc



Generarea automata a bazelor de date

- CASE tool: instrument de modelare vizuală
- Automatizează anumiți paşi privind translatarea diagramelor de clase în tabele relaționale.
  - Este necesară şi 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)