Baze de date

Curs 1

Organizare

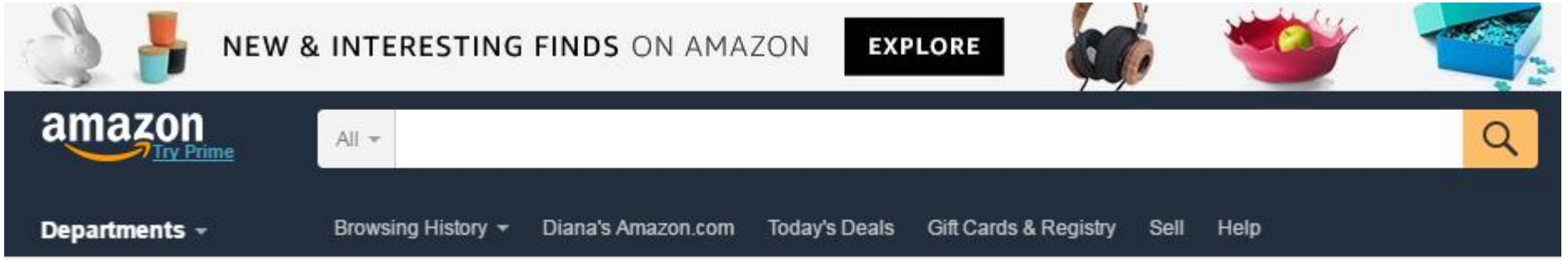
- Nota finală:
  - 50% medie laborator (L)
  - 50% examen online în ultimul curs (E)

- Pentru promovare:
  - $L \geq 5$
  - $E \geq 5$

Platforme si tehnologii folosite la curs

- Materialele de la curs și laborator le găsiți pe site: www.cs.ubbcluj.ro/~dianat/
- E-mail: dianat[at]cs.ubbcluj.ro
- Zoom – pentru întâlnire live
  - Voi incerca sa pastrez acelasi link pentru intalnirile viitoare
  - Daca se schimba informatiile pentru urmatoarea intalnire le voi afisa pe site si pe facebook.
- Moodle – pentru examen
  - Aveti conturi?

Unde întâlnim baze de date?



De ce folosim baze de date?

- Explozie informațională fără precedent, seturi de date complexe, de dimensiuni foarte mari

- O organizație trebuie să își gestioneze datele eficient, să poată obține informații corecte și în timp util pentru o anumită întrebare

- Nevoia de sisteme de gestiune a datelor puternice și flexibile – simplificarea gestiunii datelor și a extragerii de informații utile în timp optim

De ce folosim baze de date?

- Componentele unei aplicații:
  - Date (memorate în fișiere sau baze de date)
  - Algoritm de gestiune
  - Interfața cu utilizatorul

Caracteristicile fișierelor

- există diverse formate de memorare a datelor
- există **redundanță** (unele date se memorează în mai multe fișiere) ⇒ **inconsistență**
- descrierea în program a operațiilor de citire/scriere ⇒ se ia în considerare o structura a înregistrărilor ⇒ greutăți la dezvoltarea unui program
- este dificil să se obțină datele care îndeplinesc anumite condiții
- complexitatea actualizarilor datelor (ștergeri de înregistrări, modificarea unor valori din înregistrări)
- verificarea prin program a unor restricții de integritate (corectitudine)
- lipsa procedurilor de securitate și integritate (de exemplu accesul la notele unui student trebuie restricționat)
- nu se poate controla accesul concurent la date

Ce sunt bazele de date?

- O baza de date este o colectie de date de dimensiuni mari, stocata in scopul analizei ulterioare.
- O baza de date modeleaza aspect ale lumii reale sau conceptuale folosind un **model de date.**

- Proiectarea bazei de date:
  - Descrierea structurilor de date folosite pentru modelarea datelor

- Analiza datelor:
  - Cum obținem informațiile dorite formulând interogări pe datele din baza de date

Scopul bazelor de date

- Bazele de date sunt utile în **stocarea** şi **gestionarea** datelor într-un mod eficient.
- Domenii în care se folosesc baze de date:
  - Bănci (conturi, carduri, etc.)
  - Gestionarea datelor clienților unei companii de asigurări
  - Platforme de E-learning
  - E-commerce Websites (Amazon, Emag, etc)
  - Rețele sociale (Facebook, Instagram, etc)
  - …
- Baze de date necomputerizate:
  - Carte de telefon
  - Dicționar

Model de date

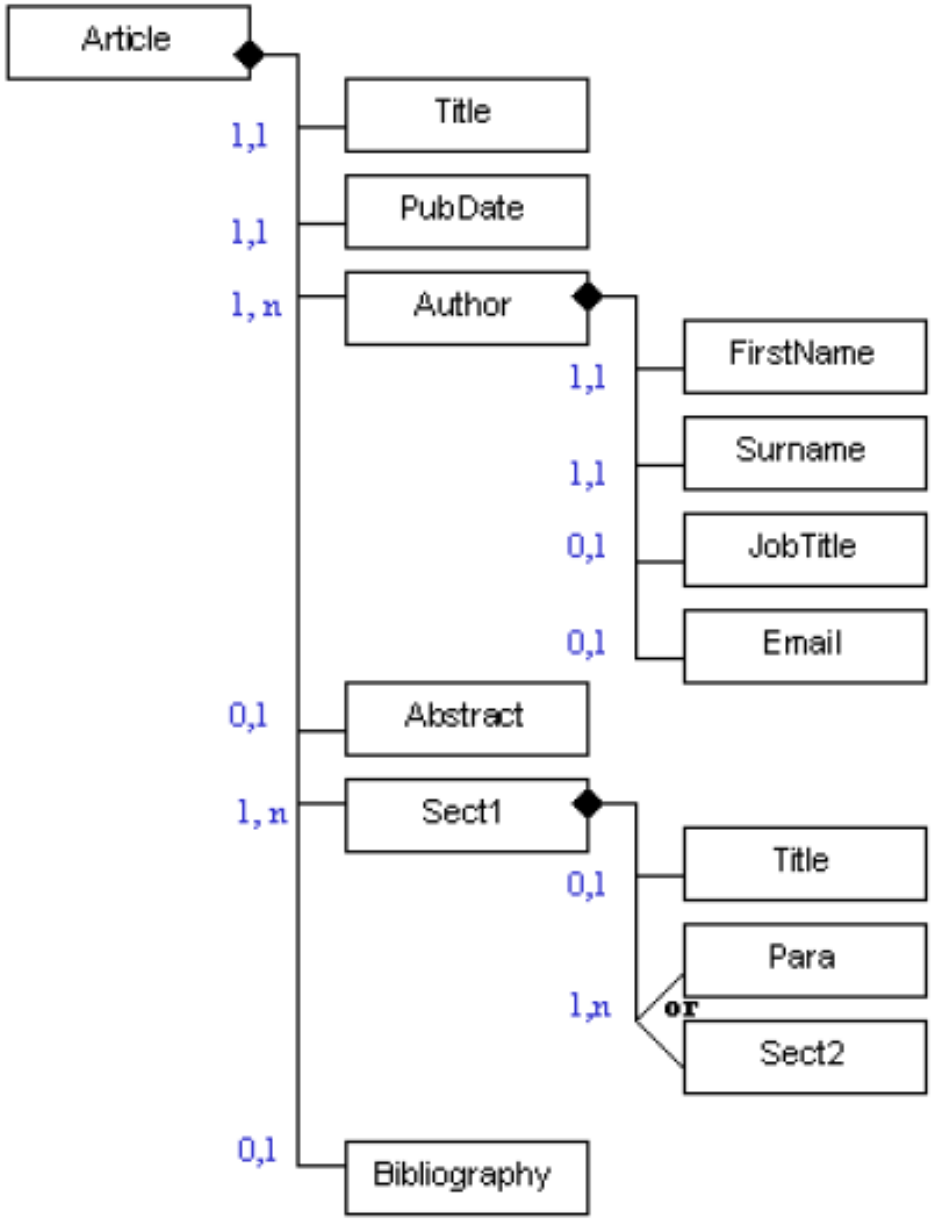
- Modelul de date:
  - Determină conceptele și regulile folosite pentru modelarea datelor.
  - Aceste concepte permit descrierea structurii datelor, precizarea restricțiilor de consistență (corectitudine) și a relațiilor între date.

- Structura/Schema bazei de date:
  - În cadrul unui model, pentru a descrie o anumită colecție de date se folosesc anumite structuri de date, care constituie schema bazei de date.
  - Datele din colecție respectă schema și pot să fie considerate instanțe ales schemei.

Modele de descriere a datelor

- Entitate-relație
- **Relațional**
- Rețea
- Ierarhic
- Orientat obiect
- noSQL
- Semistructurat (XML), RDF

Exemplu
- Modelul Ierarhic



Structură/Schemă vs. Date

- ***Structura bazelor de date***
  - Se modifică rar
  - Denumită şi *metadata*

- ***Starea bazei de date***
  - Se modifică frecvent;
  - Sistemel de gestiune a bazelor de date asigură că starea fiecărei baze de date este una *validă*.

- ***Instanţa bazei de date*** se referă în general la combinaţia dintre structură şi stare

Sisteme de gestiune a bazelor de date (SGBD)

- În timp ce o bază de date reprezintă o colecție structurată de date, aplicațiile care utilizează informațiile conținute în bazele de date sunt de obicei create pe baza unor instrumente furnizate de un **mediu de dezvoltare a bazelor de date.**
- Aceste medii de dezvoltare diferă de mediile tradiționale de programare

→ oferă instrumente specifice pentru a gestiona cu ușurință datele

→ utilizatorul nu se preocupă de detaliile de nivel coborât asociate în mod normal programării tradiționale (cum ar fi gestionarea memoriei etc.).

Sisteme de gestiune a bazelor de date (SGBD)

- Un SGBD este o colecţie integrată de **instrumente** pentru
  - crearea unei baze de date şi specificarea structurii acesteia;
  - interogarea şi modificarea eficientă a datelor;
  - securizarea datelor;
  - controlul accesului la date de către *mai mulţi* utilizatori la *un moment dat*;

- Exemple de SGBD:
  - Pentru modelul relațional: **MS SQL Server**, MySQL, Oracle, etc.
  - Pentru alte modele relaționale: ierarhic – IBM, orientat obiect – ObjectStore, etc.

Când este util să folosim baze de date?

Atunci când nevoile aplicației noastre implică:

- Persistență
- Cantitate mare de date
- Date structurate
- Acces distribuit si concurrent la date
- Integritate
- Securitate
- Partajarea datelor cu alte aplicații

Când NU utilizăm baze de date?

- Investiția inițială e prea mare
- Prea mult efort
- Dezvoltăm o aplicație foarte simplă, bine-definită și care nu presupune modificări ulterioare
- Nu este necesar accesul mai multor utilizatori la date.

Alternativa: fișiere text.

Modelul entitate-relație

- este un model **semantic**, mai abstract, de nivel înalt, care ușurează sarcina de a realiza o bună descriere inițială a datelor

- un design într-un astfel de model e transformat ulterior în termenii modelului de date al sistemului care gestionează baza de date (transformarea ER –relațional)

Modelul entitate-relație

- Concepte: entitatea, atributul și relația
- **Entitatea** – o dată care reprezintă un obiect din lumea reală; un tip de entitate are un nume și o listă de atribute (proprietăți)
- Mulțimea entităților cu aceeași structură (de exemplu, mulțimea studenților) sunt instanțe ale unui **tip de entitate** (clasă/ schemă a entității)
- În precizarea tipului de entitate, fiecare **atribut** are: nume, domeniu pentru valorile posibile și eventuale condiții pentru a verifica dacă valoarea este corectă
- La un tip de entitate se poate defini o **cheie** (care este o restricție): o mulțime de atribute care iau valori distincte/unice în instanțele tipului de entitate

Modelul entitate-relație

- **relația** –este o dată și precizează o legătură (asociere) între două sau mai multe entități; la această asociere se pot folosi și atribute suplimentare
- toate relațiile cu aceeași structură (legături între entități de aceleași tipuri) trebuie descrise printr-un **tip de relație** sau schemă a relației
- un tip de relație are un nume, tipurile de entitate folosite în asociere și posibile atribute suplimentare
- **Schema modelului** este formată dintr-o mulțime de tipuri de entitate și tipuri de relație

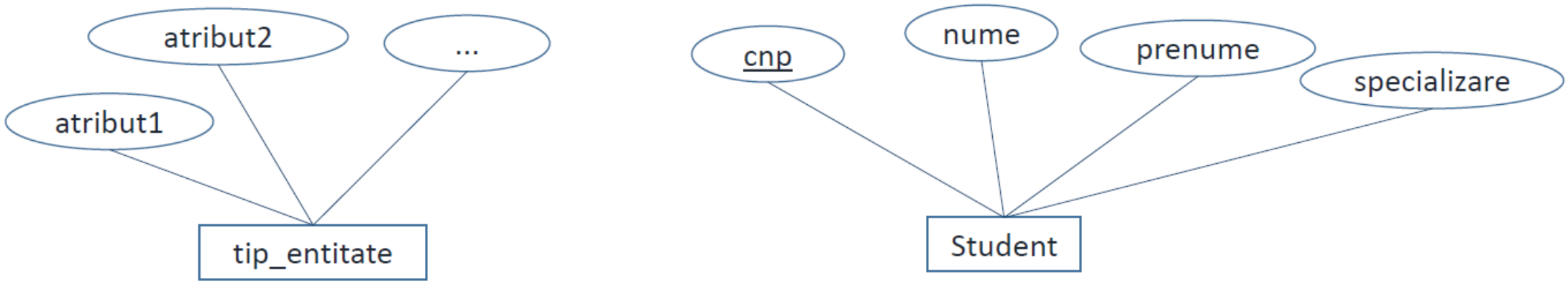
Modelul entitate-relație

Pentru **relațiile binare** (între tipurile de entitate T1 și T2) se pot defini următoarele **tipuri particulare de relație** (**restricții** pentru baza de date)**:**

- **1:1**: dacă o entitate de tipul T1 se asociază cu cel mult o entitate de tipul T2, iar o entitate de tipul T2 se asociază cu cel mult o entitate de tipul T1
  - Ex. asocierea dintre student si cont la facultate
- **1:M**: dacă o entitate de tipul T1 se asociază cu oricâte entități de tipul T2, iar o entitate de tipul T2 se asociază cu cel mult o entitate de tipul T1
  - Ex. asocierea dintre grupă și studenți – pentru a preciza componența grupelor
- **M:N:** dacă o entitate de tipul T1 se asociază cu oricâte entități de tipul T2, iar o entitate de tipul T2 se asociază cu oricâte entități de tipul T1
  - Ex. asocierea dintre discipline și studenți – pentru a preciza contractele de studiu

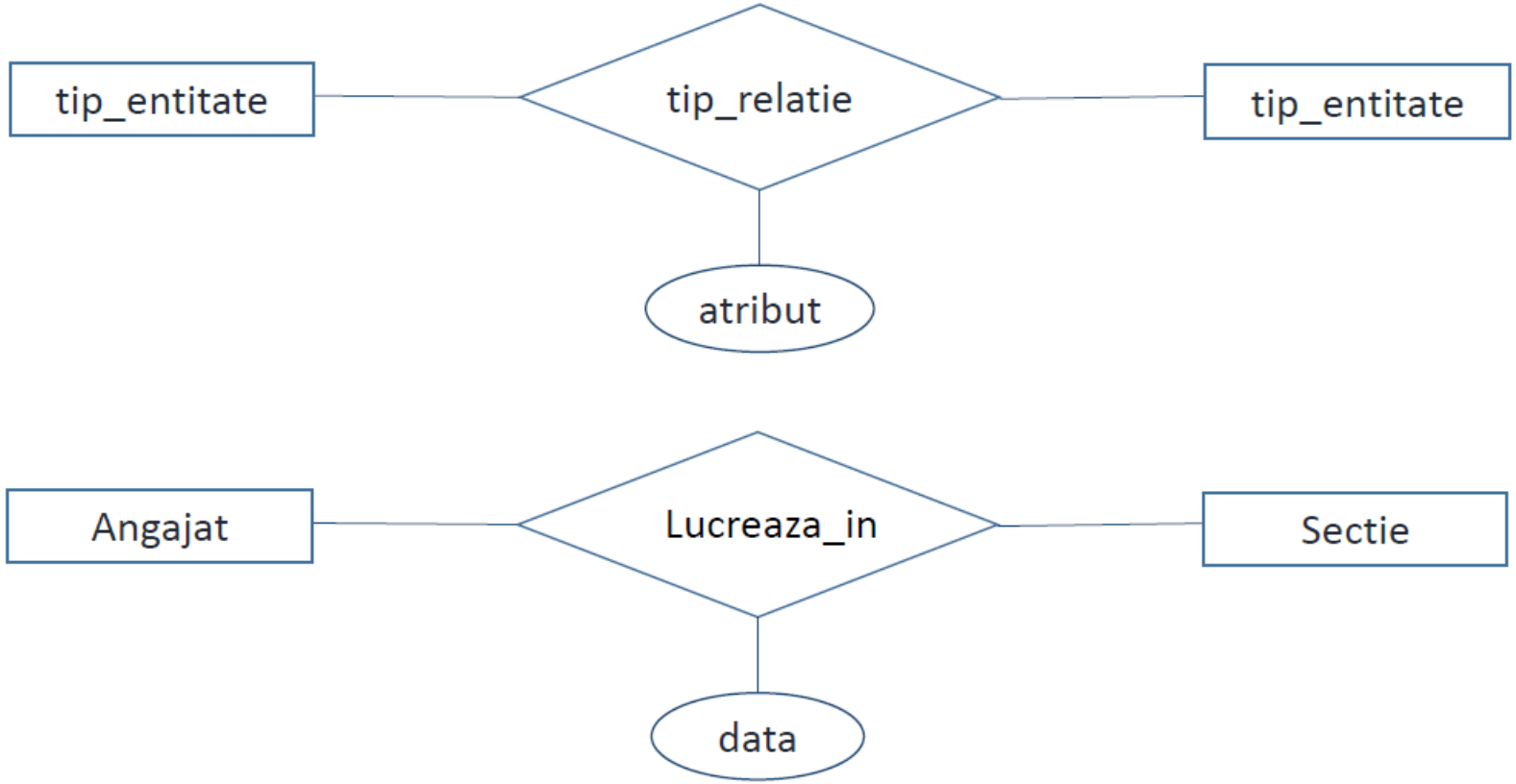
Modelul entitate-relație

- Entități și atributele asociate:



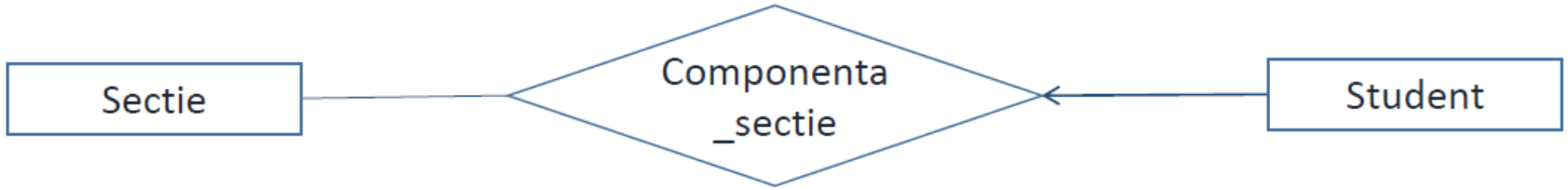
Modelul entitate-relație

- Relații și atributele asociate:

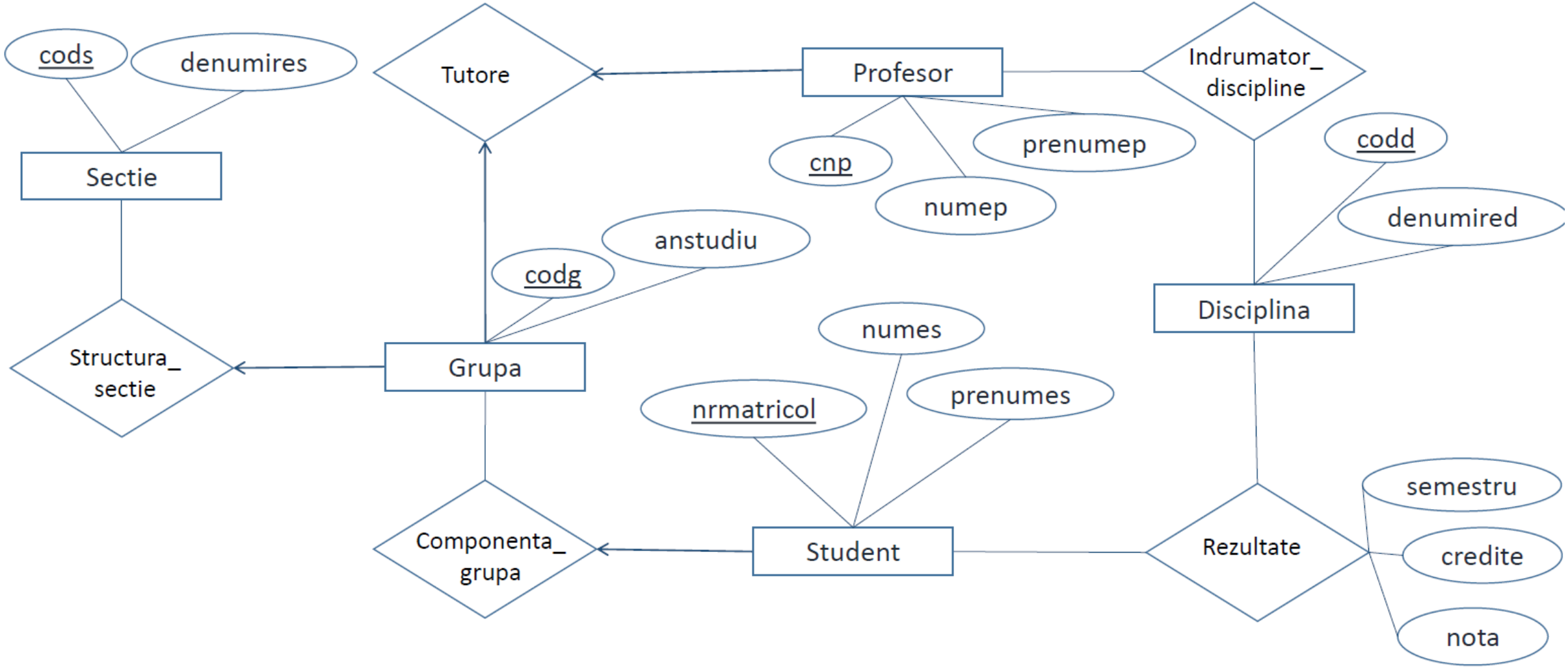


Modelul entitate-relație

- Convenție reprezentare tip de relație 1:M:



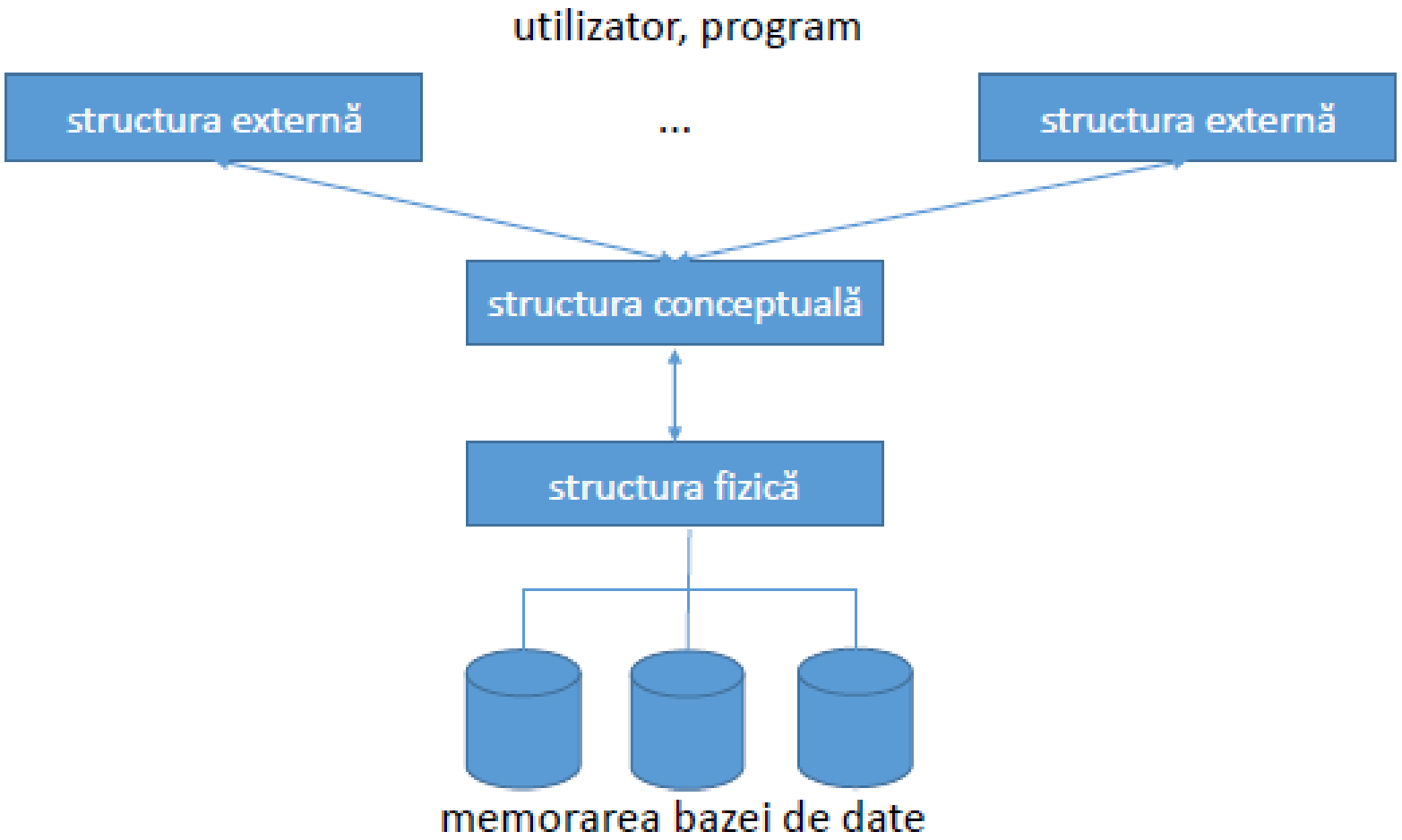
Modelul entitate-relație Exemplu



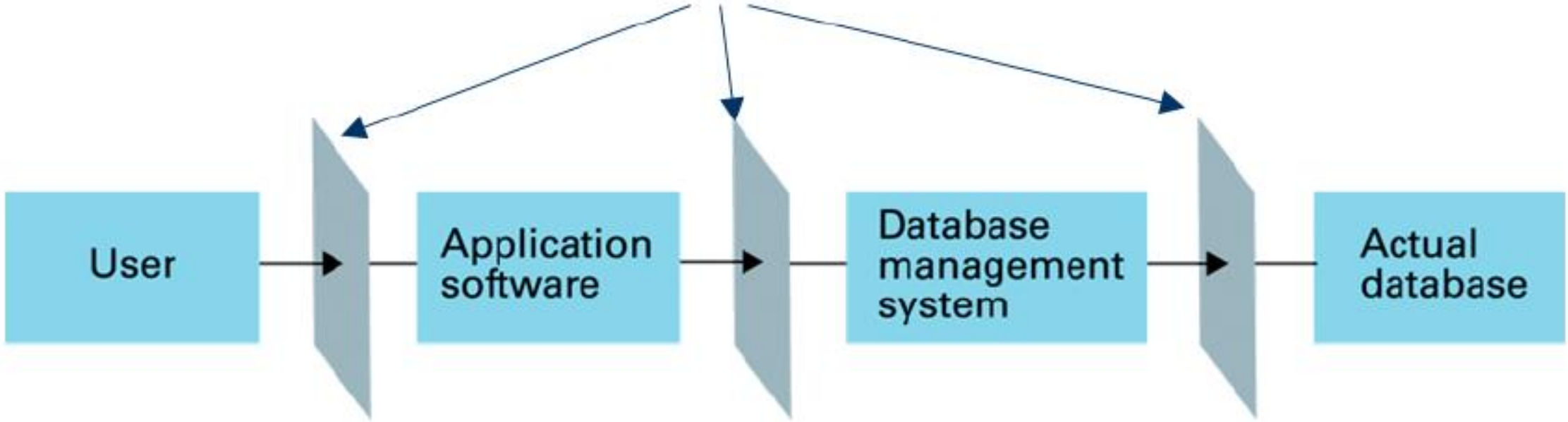
Arhitectura ANSI-SPARC

- O arhitectură pentru un sistem de baze de date organizat pe trei niveluri:
  - **Structura conceptuală (schema bazei de date)**: descrie toate structurile de date si restricțiile folosite în baza de date
  - **Structuri externe**: descrierea structurilor de date folosite de un utilizator/program particular; această descriere se face într-un anumit model de organizare
  - **Structura fizică (structura internă)**: descrierea structurilor de memorare a bazei de date (fișiere de date, indecși, etc)

Arhitectura ANSI-SPARC



Niveluri de abstractizare



Modelul Relațional

- Avantaje:
  - Flexibil, poate fi folosit universal
  - Multiple sisteme de gestiune (gratis sau la diverse prețuri)
  - Cel mai răspândit

- Utilizează o structură de date simplă: ***Tabela***
  - simplu de înțeles
  - utilă în modelarea multor situaţii/entităţi din lumea reală
  - conduc la interogări de o complexitate redusă

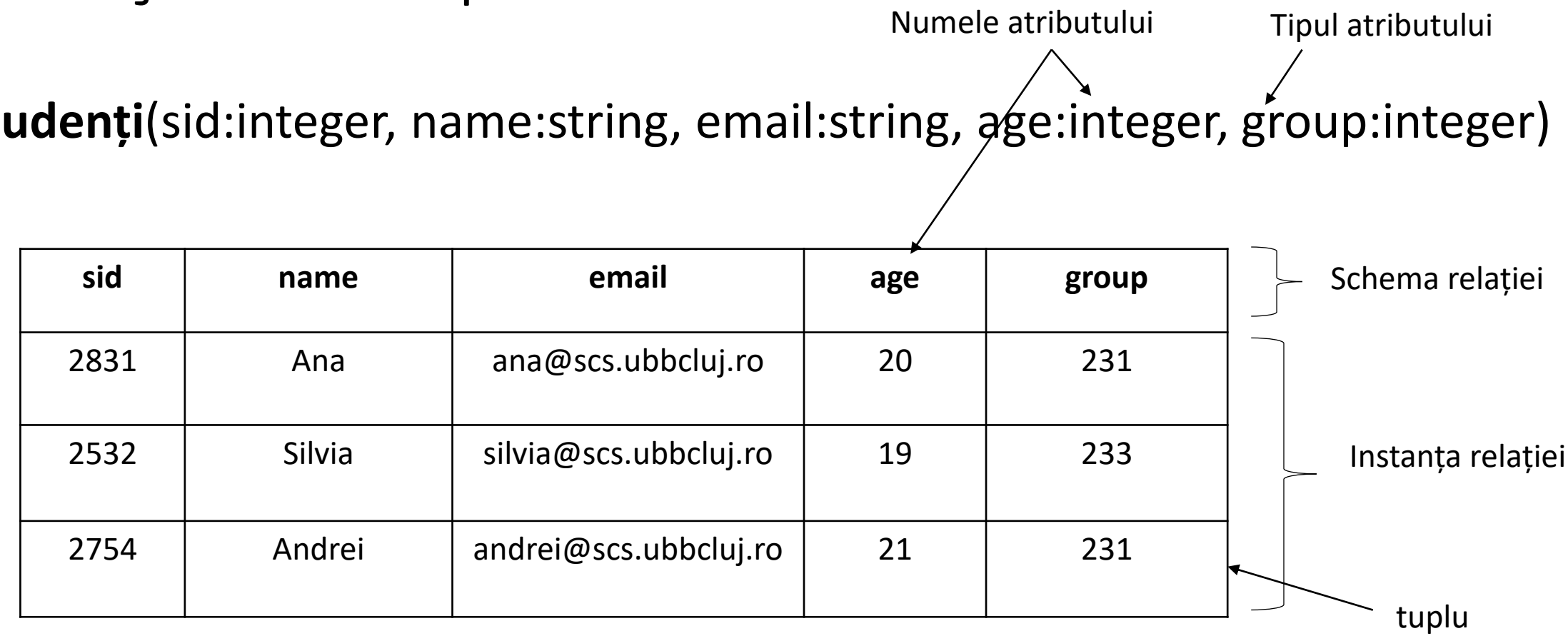
- Utilizează matematica în descrierea/reprezentarea înregistrărilor şi a colecţiilor de înregistrări: ***Relația***
  - pot fi modelate formal
  - permit utilizarea de limbaje de interogare formale
  - au proprietăţi ce pot fi modelate şi demonstrate matematic

Relaţia –definiţie formală

- O **relaţie** sau **structura unei relaţii R** este o listă de **nume de atribute**[A_1, A_2, …, A_n].
- **Domeniu =** mulţime de valori scalare (tipuri atomice -întreg, text, dată, etc)
- D_i= Dom(A_i) –domeniul lui A_i, i=1..n
- **Instanţa unei relaţii (**[R]) e o submulţime a D_1 x D_2 x… x D_n
- **Grad** (**aritate**) = numărul tuturor atributelor din structura unei relaţii
- **Tuplu** = un element al instanţei unei relaţii, o înregistrare. Toate tuplurile unei relaţii sunt distincte!
- **Cardinalitate** = numărul tupluri unei relaţii

Relaţia - Exemplu

- **Studenți**(sid:integer, name:string, email:string, age:integer, group:integer)



cardinalitate = 3
grad = 5

Baze de date relaţionale

- **Bază de date** = o colecție de relații având nume distincte
- **Structura/Schema** unei baze de date este mulţimea structurilor relaţiilor acesteia
- **Instanţa**(**starea**) unei baze de date este mulţimea instanţelor relaţiilor acesteia

Relaţia

- Valorile unui atribut sunt **atomice** (nu se pot descompune în valori pentru alte atribute) și **scalare** (nu se pot memora tablouri)
- Liniile din tabel **nu sunt ordonate:**

sid	name	email	age	group
2831	Ana	ana@scs.ubbcluj.ro	20	231
2532	Silvia	silvia@scs.ubbcluj.ro	19	233

sid	name	email	age	group
2532	Silvia	silvia@scs.ubbcluj.ro	19	233
2831	Ana	ana@scs.ubbcluj.ro	20	231

- Liniile din tabel **sunt distincte** – o relație este definită ca fiind o mulțime de tupluri distincte(în practică, sistemele comerciale permit tabelelor să conțină duplicate)

Relaţia - Contraexemplu

- Exemplu de relație care nu e bine proiectată:

name	study_programme	study_year	group
Ana Popa	IR	1	231
Silvia Manda	IR	1	231
Andrei Chira	IR	2	233
Ioana Demian	IR	3	233

- Ce probleme observați?

Relaţia - Contraexemplu

- Exemplu de relație care nu e bine proiectată:

name	study_programme	study_year	group
Ana Popa	IR	1	231
Silvia Manda	IR	1	231
Andrei Chira	IR	2	233
Ioana Demian	IR	3	233

- Ce probleme observați?
  - Asocierea grupei, a secției si a anului de studiu se păstrează de mai multe ori
  - Ultimii doi studenți sunt în aceeași grupă, dar în ani diferiți, ceea ce este o eroare (inconsistență)
- Soluție: **normalizarea relațiilor**

Constrângeri de integritate (CI)

- Sunt condiţii ce trebuie să fie îndeplinite de către *orice* instanţă a unei baze de date
- Specificate la momentul definirii structurii relaţiei
- Verificate la modificarea conţinutului relaţiei

- O instanţă a unei relaţii că este *legală* dacă satisface toate CI specificate
- SGBD nu va permite instanţe *ilegale*

- Exemple de constrângeri: constrângeri de **domeniu**, constrângeri de **unicitate**, constrângeri de **integritate referențială**

Constrângeri de integritate -exemple

- ***Students****(sid:integer, name:string, email:string, age:integer, group:integer)*
  - Constrângere de domeniu: *group:integer*
  - Constrângere de interval: 18 ≤ *age* ≤ 70

- ***TestResults****(sid:string, TotalQuestions:integer, NotAnswered:integer, CorrectAnswers:integer, WrongAnswers:integer)*
  - *TotalQuestions= NotAnswered + CorrectAnswers + WrongAnswers* – **nu e o CI!**

Chei Primare / Primary Key

- O mulţime de atribute reprezintă o **cheie** a unei relaţii dacă:

1. Nu există două tuple care au aceleaşi valori pentru toate atributele
   → **unicitate**
2. Acest lucru nu este adevărat pentru nici o submuţime a cheii
   → **minimalitate**
3. Este **definită** pentru toate tuplurile (nu apar valori nule)

- Dacă a 2-a afirmaţie este falsă → **super cheie**
- Daca există > 1 cheie pentru o relaţie → **chei candidat**
- Una dintre cheile candidat este selectată ca și **cheie primară**

Chei Primare / Primary Key

- Exemplu:

Carti (autor, titlu, editura, an_aparitie)

- Ce ați alege ca si cheie primară?

Chei Primare / Primary Key

- Exemplu:

Carti (autor, titlu, editura, an_aparitie)

- Ce ați alege ca si cheie primară?
  - Se poate alege grupul de atribute {autor, titlu, editura, an_aparitie}
  - În astfel de situații este utilă includerea unui atribut suplimentar, care sa aibă valori distincte (valorile se pot genera automat la adăugarea de înregistrări)

Carti (cid, autor, titlu, editura, an_aparitie)

- Ce ați alege ca si cheie primară acum?

Chei Primare / Primary Key

- Exemplu:

Carti (autor, titlu, editura, an_aparitie)

- Ce ați alege ca si cheie primară?
  - Se poate alege grupul de atribute {autor, titlu, editura, an_aparitie}
  - În astfel de situații este utilă includerea unui atribut suplimentar, care sa aibă valori distincte (valorile se pot genera automat la adăugarea de înregistrări)

Carti (cid, autor, titlu, editura, an_aparitie)

- Ce ați alege ca si cheie primară acum?
  - Atenție, mulțimea {cid, titlu} nu e o cheie, deoarece conține cheia {cid}, deci nu e minimală, dar e o supercheie

Chei Primare / Primary Key

- Exemplu:

  Orar(zi, ora, sala, profesor, formatie, disciplina)

- Ce ați alege ca si cheie primară?

Chei Primare / Primary Key

- Exemplu:

  Orar(zi, ora, sala, profesor, formatie, disciplina)

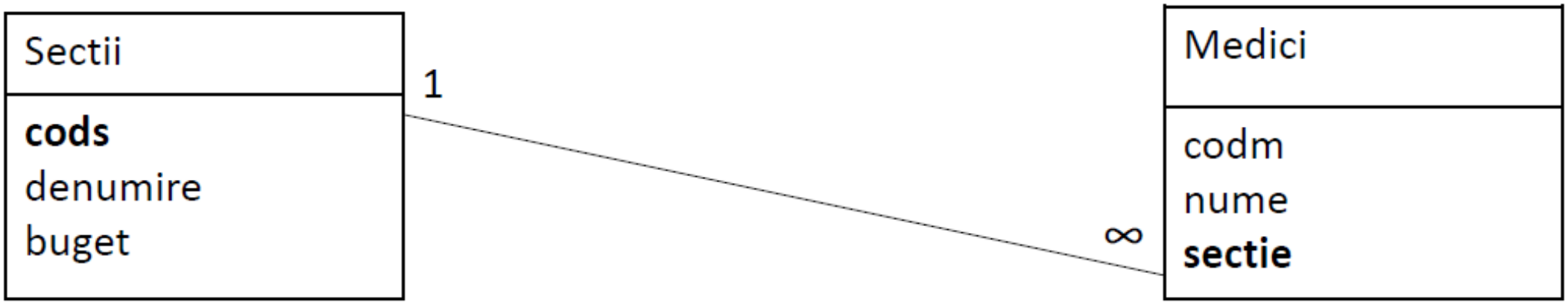
- Ce ați alege ca si cheie primară?
  - Se pot alege ca și chei următoarele mulțimi de atribute:
    - **{zi, ora, sala}**
    - **{zi, ora, profesor}**
    - **{zi, ora, formatie}**

Chei străine (externe) / Foreign Key

- O **cheie străină**(**externă**) este o mulţime de câmpuri a unei relaţii utilizate pentru a *referi* un tuplu al unei alterelaţii (un fel de *pointer logic*).
- Aceasta trebuie să corespundă cheii primare din a doua relaţie.
- Cheia externă trebuie să aibă același număr de coloane cu cheia primară spre care face referire, iar tipurile de date ale coloanelor trebuie să fie compatibile (numele coloanelor in schimb poate să difere)

Chei străine (externe) / Foreign Key

- Prin cheie externă se pot memora **legături 1:M** între entități: la o secție corespund oricâți medici, iar unui medic îi este asociată cel mult o secție



Chei străine (externe) / Foreign Key

- Cheia externă se poate folosi și pentru a memora **legături M:N** între entități

- Exemplu: ***Students*** și ***Courses*** – la un curs sunt înscriși mai mulți studenți, iar un student are asociate mai multe cursuri; varianta de memorare cuprinde o relație intermediară

  Students*(sid:integer, name:string, email:string, age:integer, group:integer)*

  Courses*(cid:integer, title: string, ects: integer)*

  Enrolled *(sid: string, cid: string, grade: double)*

  → *sid* este cheie externă referind ***Students***

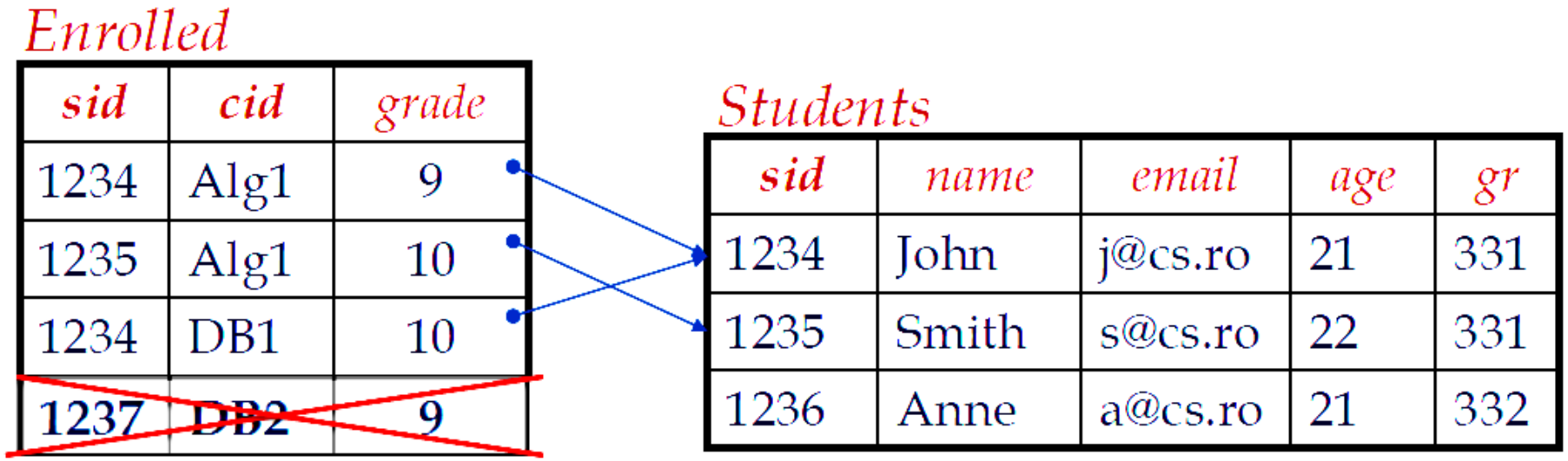
  → *cid* este cheie externă referind ***Courses***

- putem avea studenți care nu s-au înscris încă la nici un curs (*sid* nu apare in Enrolled)

Integritate referenţială

- **Integritate referenţială** = nu sunt permise valori pentru cheia străină care nu se regăsesc în tabela referită.

- Fie ***Students*** şi ***Enrolled***; *sid* în ***Enrolled*** este o cheie străină ce referă înregistrări din ***Students***.
- Dacă încercăm să adăugăm în ***Enrolled*** un tuplu cu un id de student inexistent, acesta va fi respins de SGBD.



Enrolled

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
~~1237~~	~~DB2~~	~~9~~

Students

sid	*name*	*email*	*age*	*gr*
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Integritate referenţială

- Dacă o înregistrare din ***Students*** este ştearsă dar ea este referită din ***Enrolled*** există mai multe soluții***:***
  - se şterg toate înregistrările ce o refera din ***Enrolled***.
  - nu se permite ştergerea înregistrării din ***Students***
  - sid din ***Enrolled*** va avea asignată o valoare implicită.
  - sid din ***Enrolled*** va avea asignată valoarea *null*.

- Observație. Același lucru se poate face și în cazul modificării chei primare a unei înregistrări care e referită, deși în practică cheile primare nu își modifică în general valoarea.

Cum apar constrângerile de integritate?

- CI se bazează pe semantica entităţilor din lumea reală / conceptuală modelate.

- Putem verifica dacă o CI este încălcată de instanţa unei tabele, însă NU vom putea deduce dacă o CI este adevărată doar consultând o singură instanţă.
  - O CI se referă la *toate instanţele* posibile ale unei tabele

- Cheile primare şi externe sunt cele mai comune CI;

Proiectarea bazelor de date

- Proiectare conceptuală (ex. diagrama de clase)
  - Identificarea entităţilor şi a relaţiilor dintre ele
- Proiectarea logică
  - Transformarea modelului conceptual într-o structură de baze de date (relaţională sau nu)
- Rafinarea bazei de date (normalizare)
  - Eliminarea redundanţelor şi a problemelor conexe
- Proiectare fizică şi eficientizare
  - Indexare
  - De-normalizare!

Diagrama de clase UML - Clase

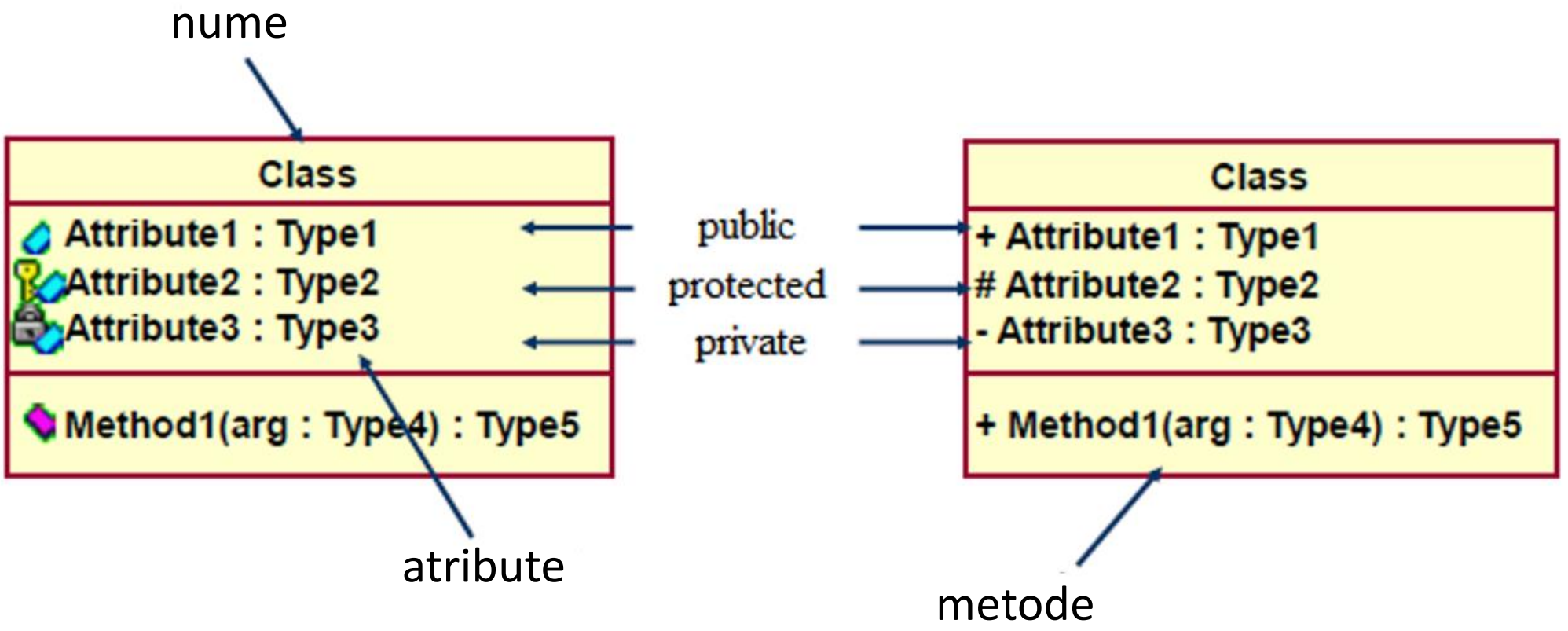
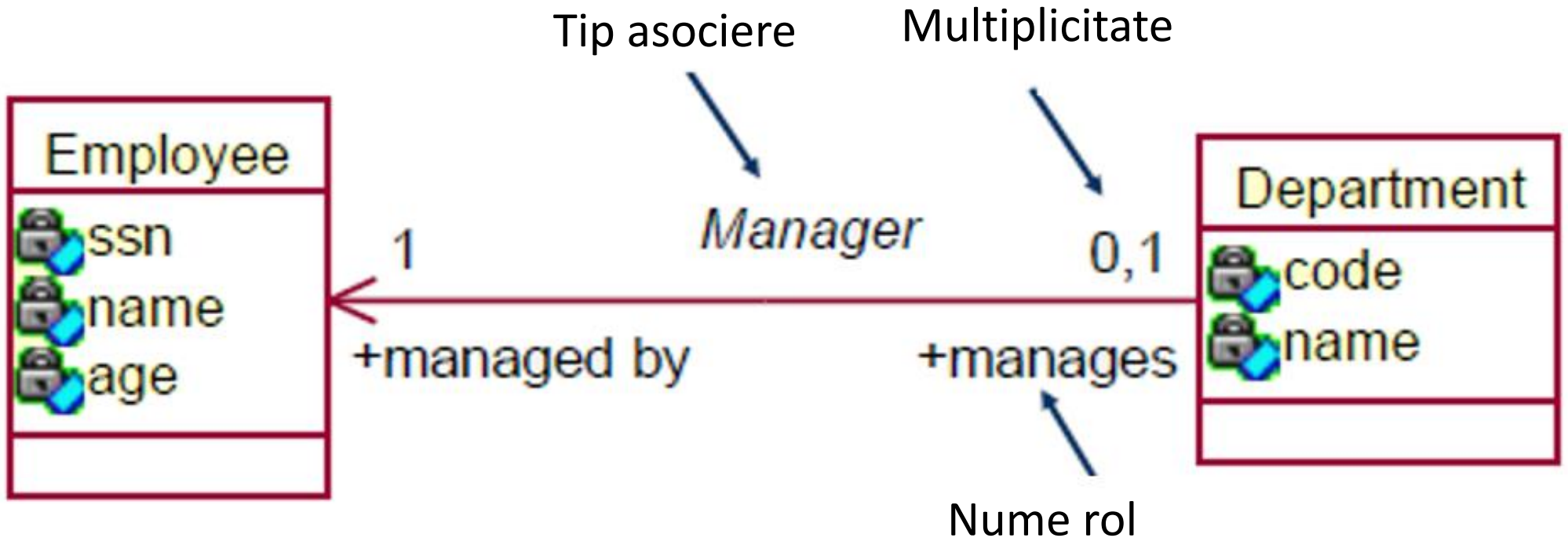


Diagrama de clase UML -Asocieri



Multiplicități:

- Valori: 4,5
- Intervale: 1..10
- Nedefinit: *

An employee manages 0 or 1 departments

Diagrama de clase UML – Clasa Asociere

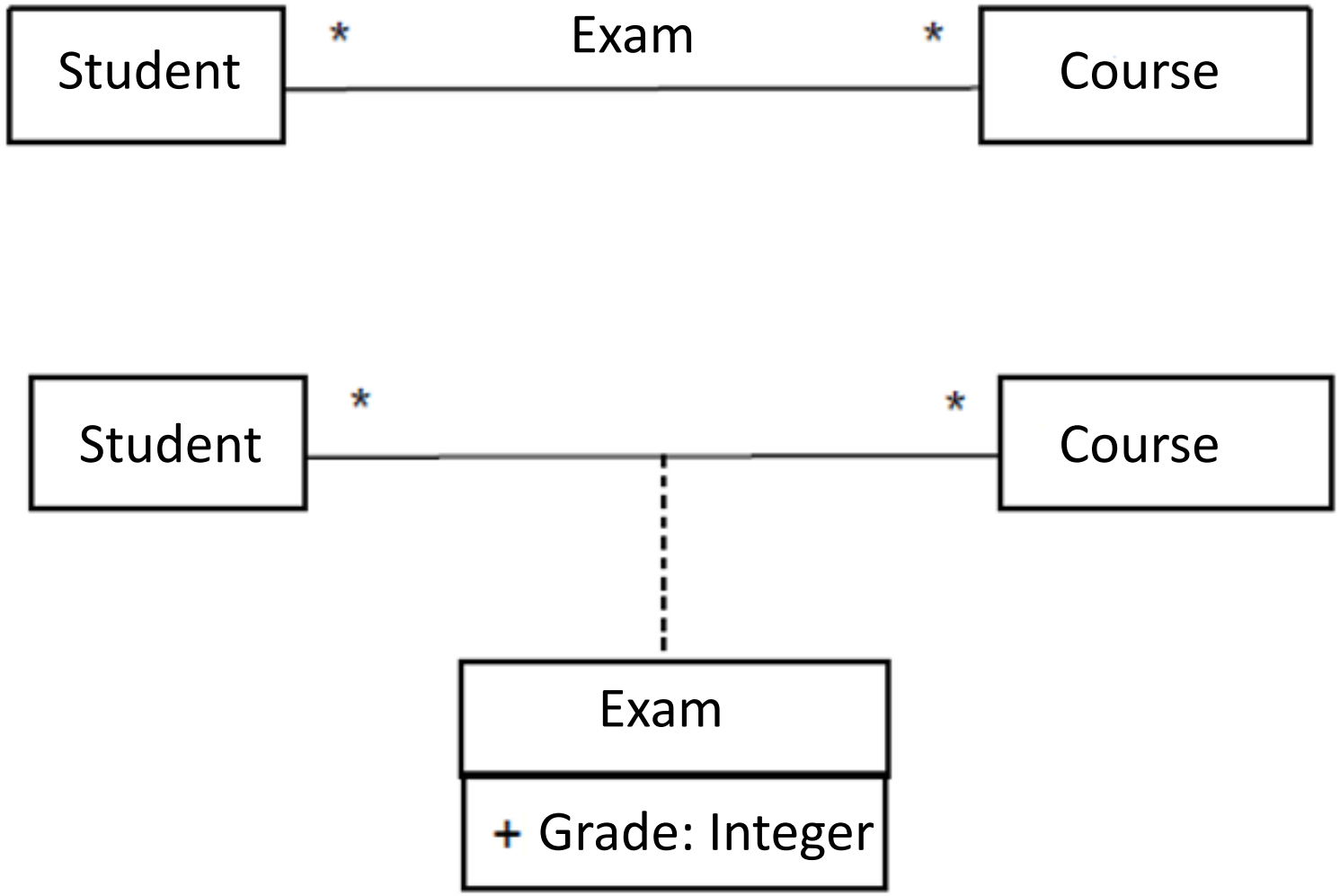


Diagrama de clase UML -Asocieri

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

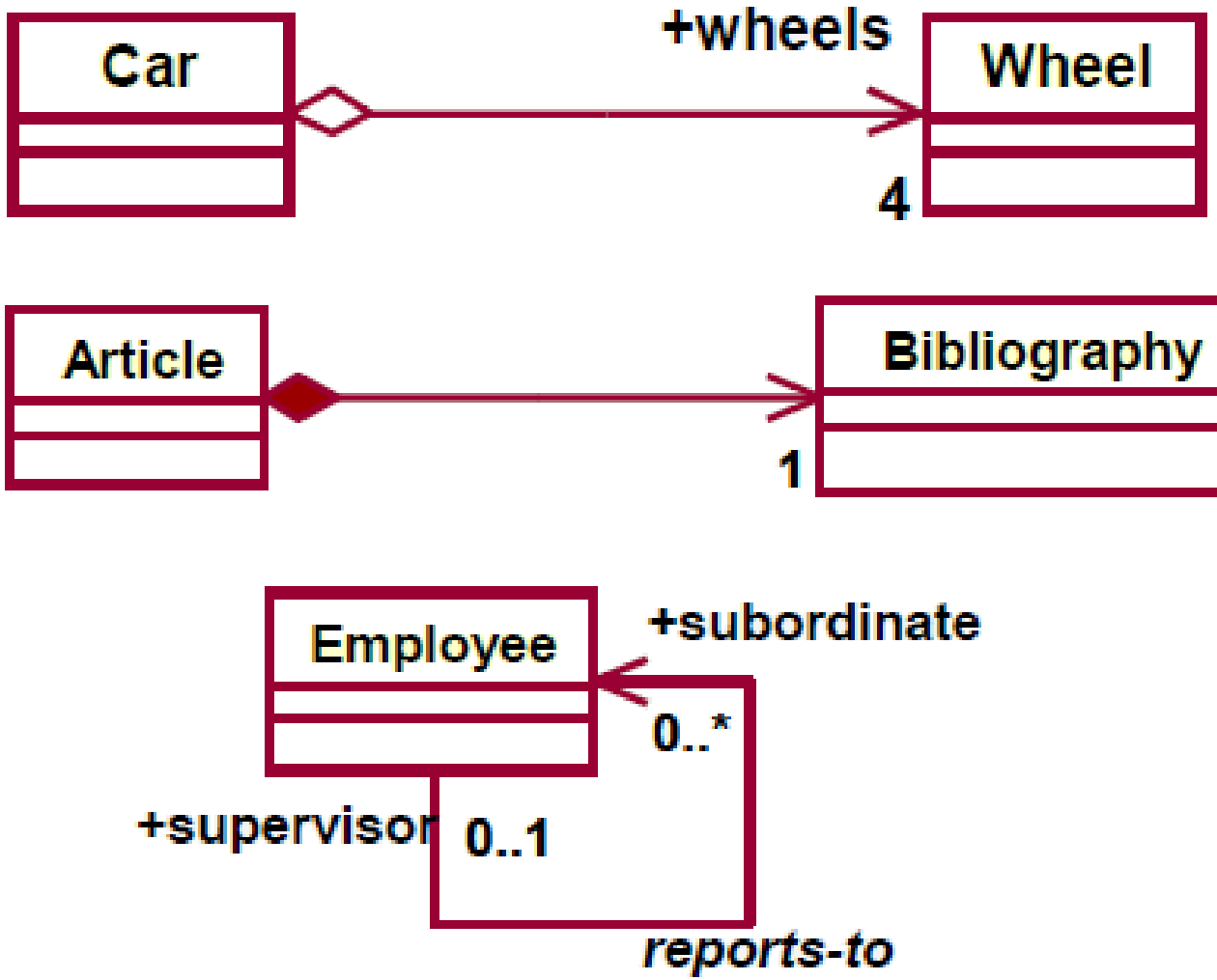
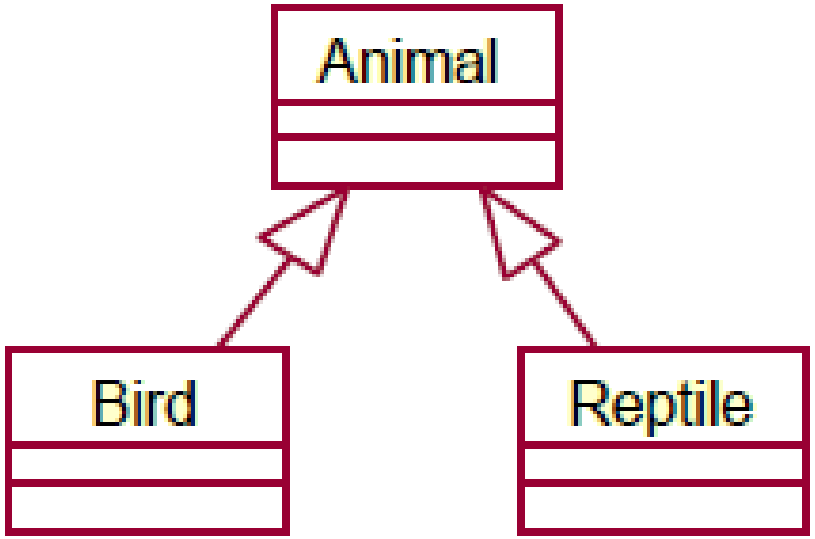
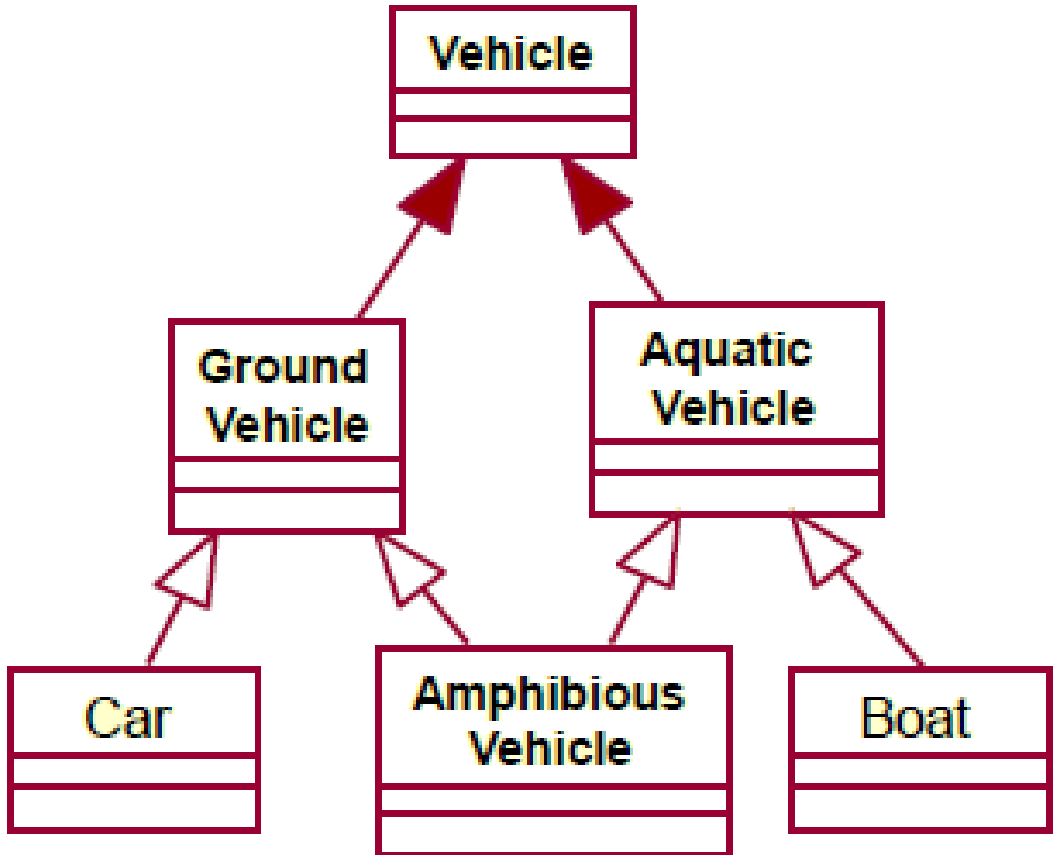


Diagrama de clase UML –Moștenire



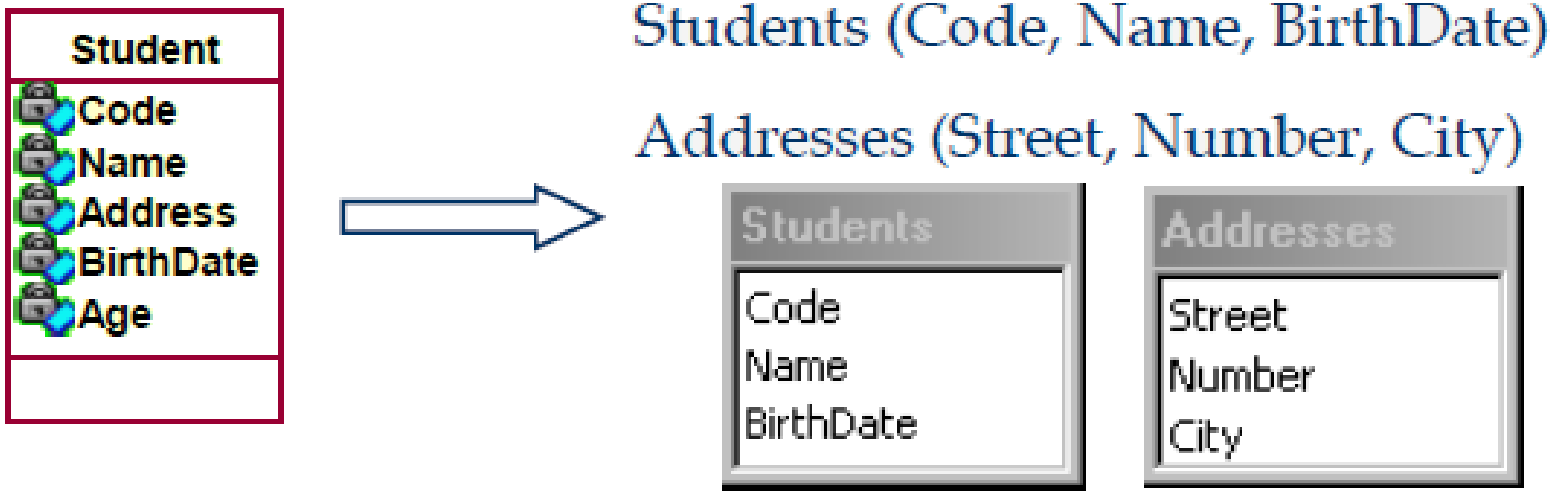


Transformare Model conceptual în bază de date relaţională

- Probleme care apar la transformare 1:1 a claselor în tabele:
  - Prea multe tabele – pot rezulta mai multe tabele decât este necesar
  - Prea multe op. *Join* – consecinţă imediată a faptului că se obţin 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

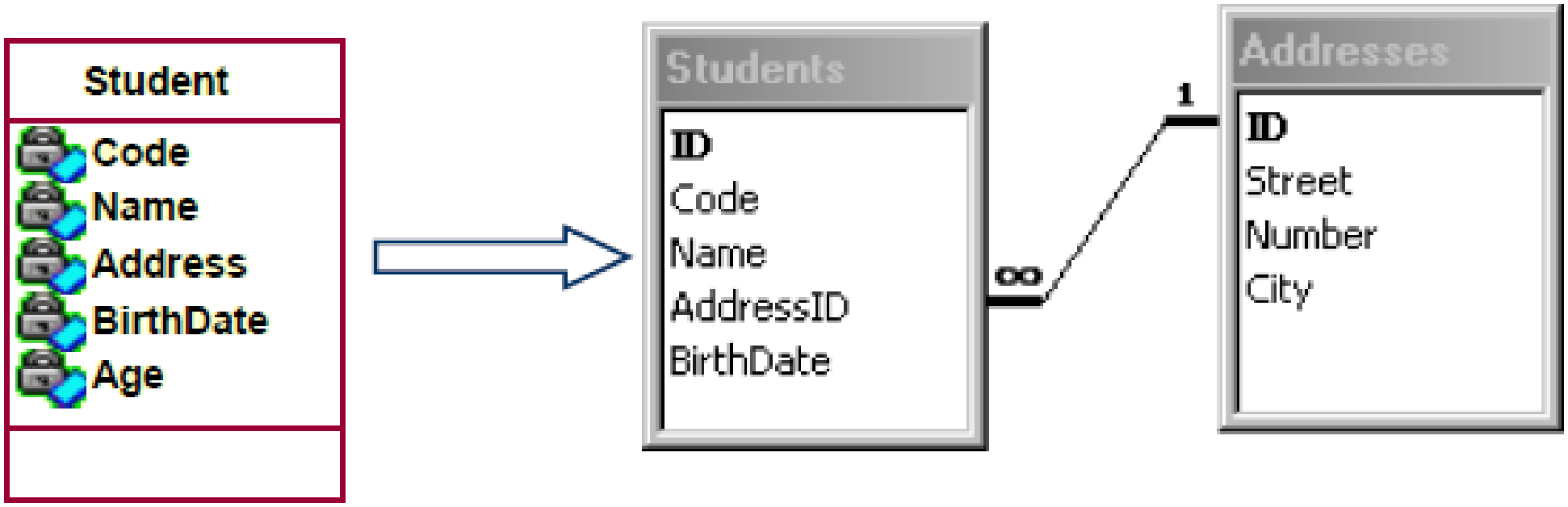
- Numele tabelei reprezintă pluralul numelui clasei
- Toate atributele simple sunt transformate în câmpuri
- Atributele compuse devin tabele de sine stătătoare
- Atributele derivate nu vor avea nici un corespondent în tabelă



Transformarea claselor în tabele

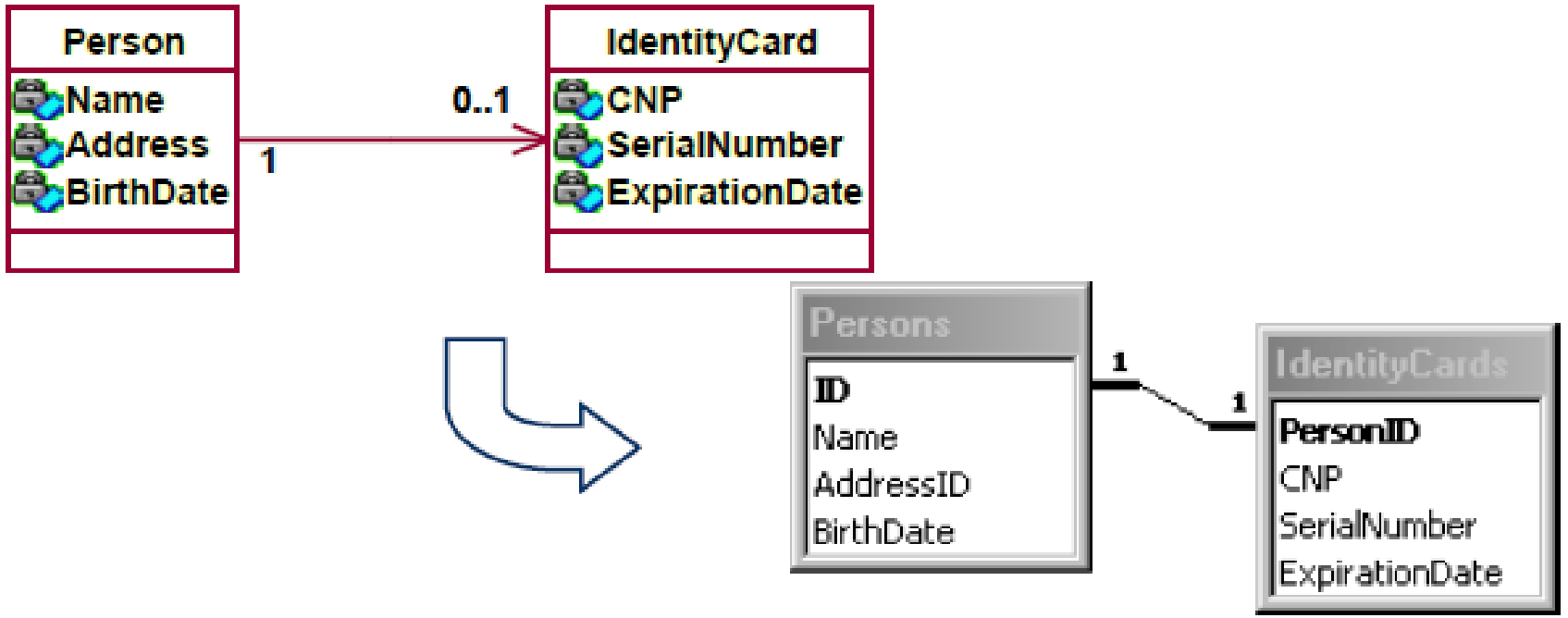
- Chei surogat – chei care nu sunt obţinute din domeniul problemei modelate
- Conceptul de cheie nu este definit în cadrul claselor UML
- O *bună practică*: utilizarea (atunci când este posibil) a cheilor de tip întreg generate automat de SGBD:
  - uşor de întreţinut (responsabilitatea sistemului)
 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

Transformarea claselor în tabele



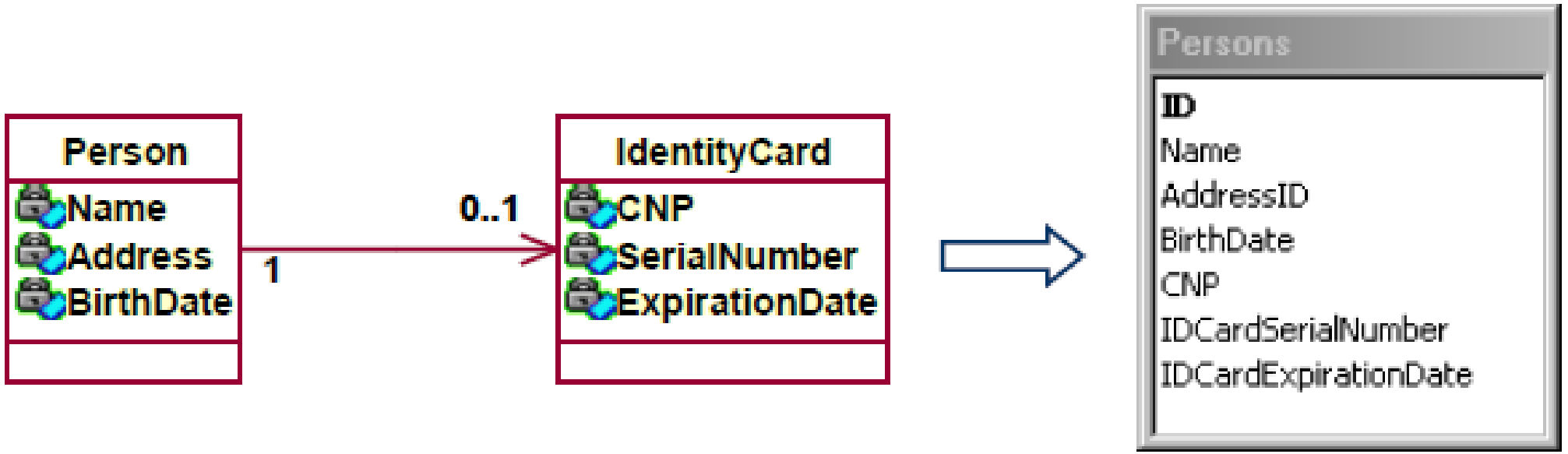
Transformarea asocierilor simple

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



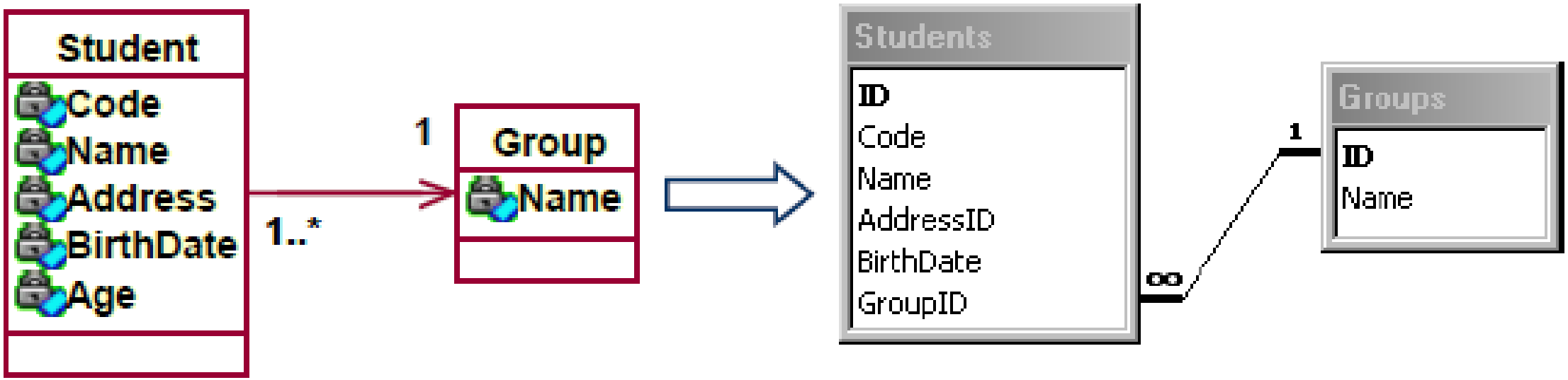
Transformarea asocierilor simple

- 1 : 1
  - se crează o singură tabelă ce conţine atributele ambelor clase asociate
  - această 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

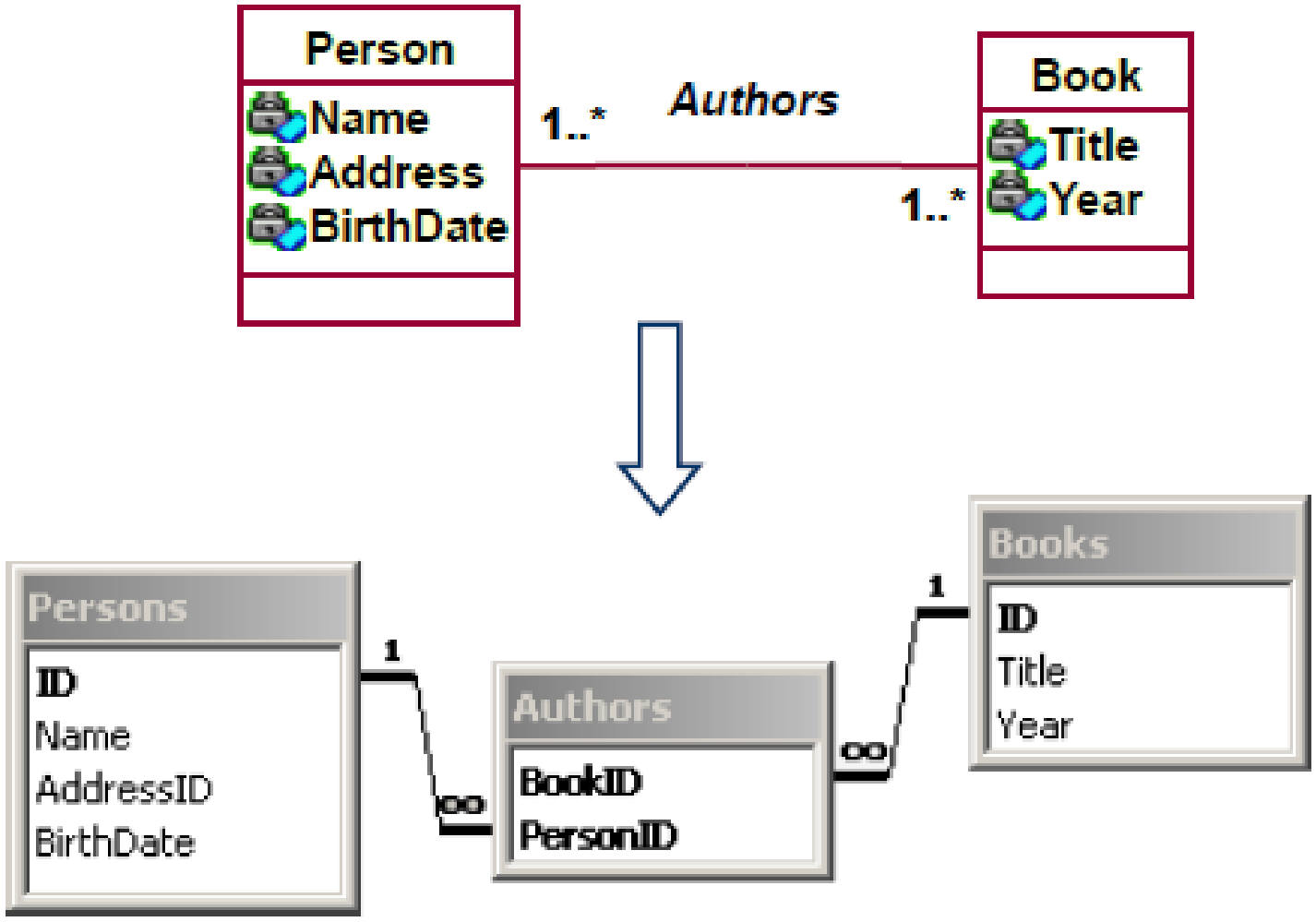
- 1 : 1..*
  - se crează câte o tabelă corespunzătoare fiecărei clase implicate în asociere
  - cheia tabelei corespunzătoare multiplicităţii “ 1” este cheia străină în cea de-a doua tabelă, corespunzătoare multiplicităţii “1..*”



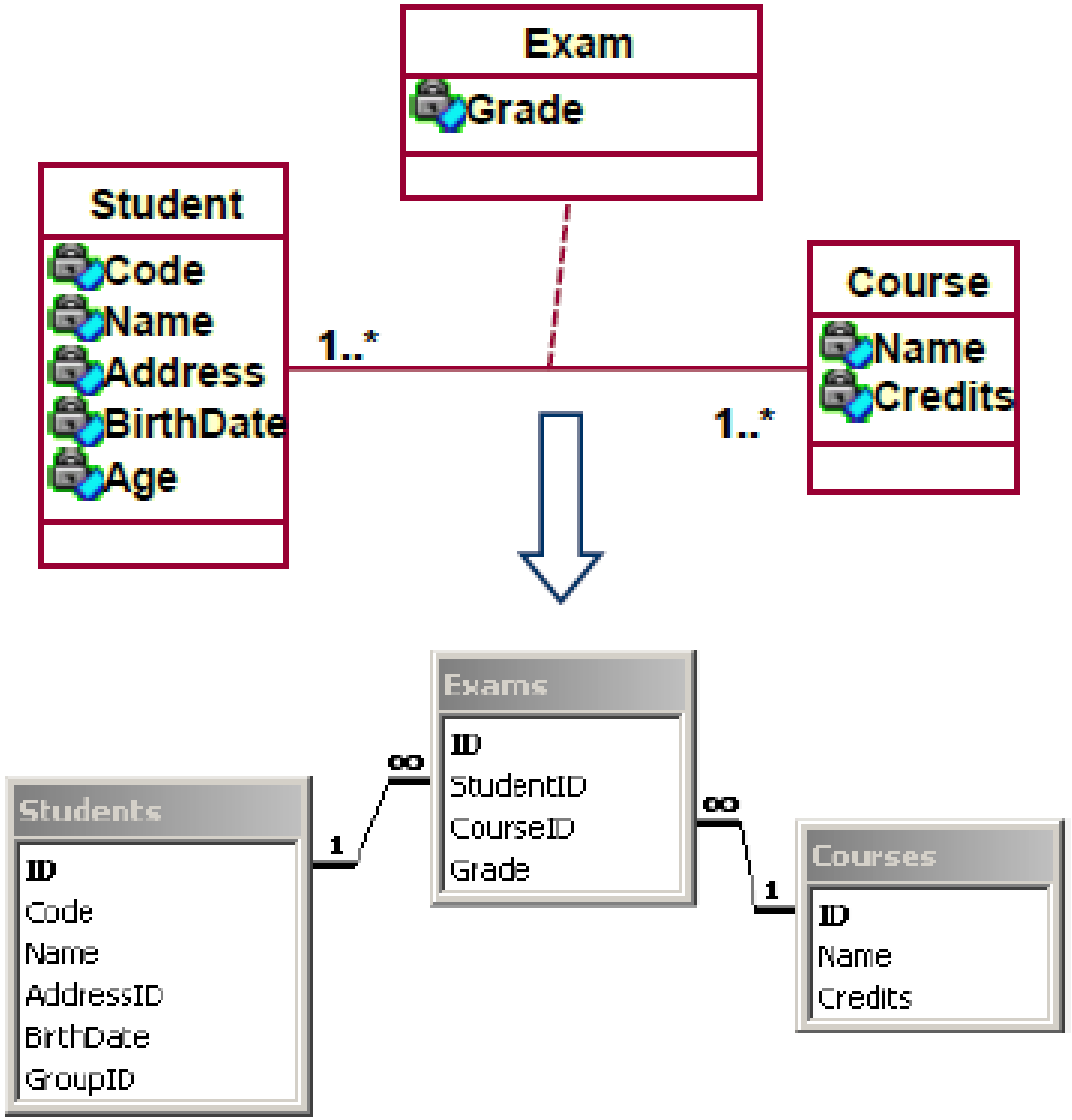
Transformarea asocierilor simple

- 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



Transformarea asocierilor simple

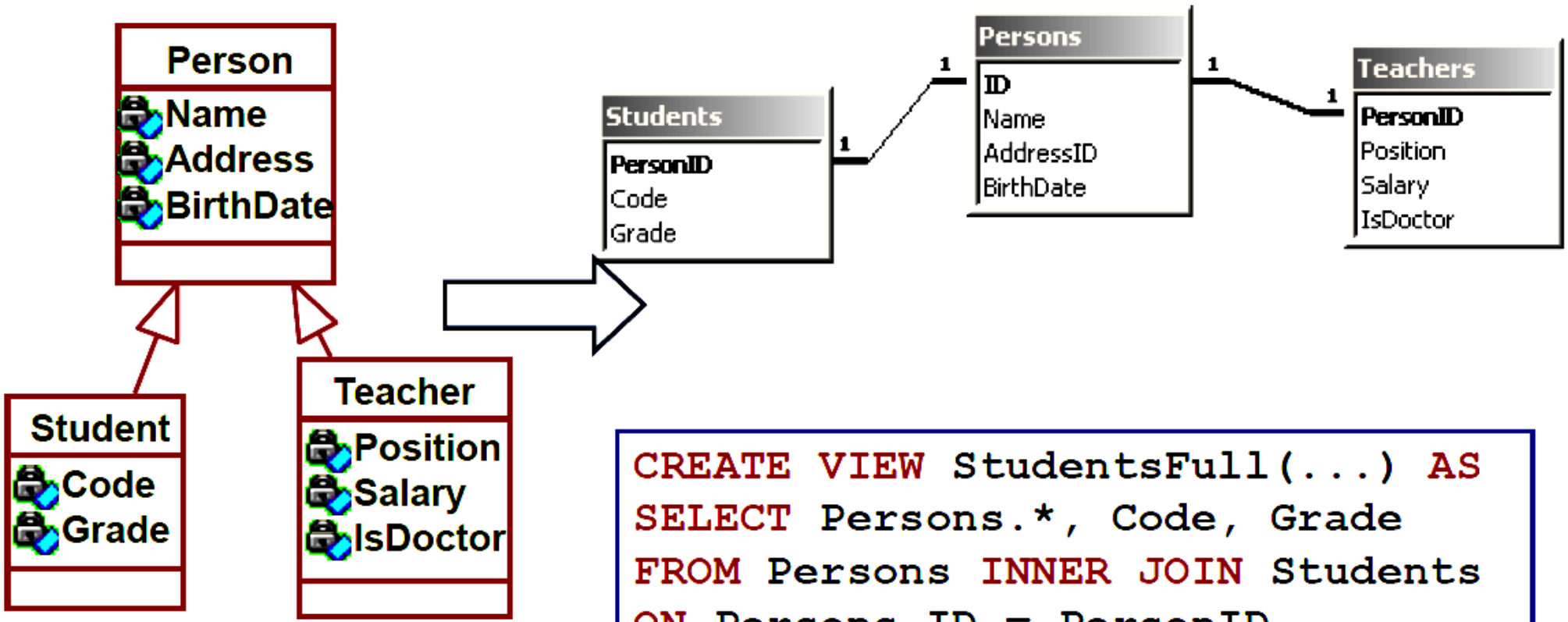


Transformarea moştenirii

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 performanţă deoarece fiecare access va implica execuţia unui *join*

Transformarea moştenirii



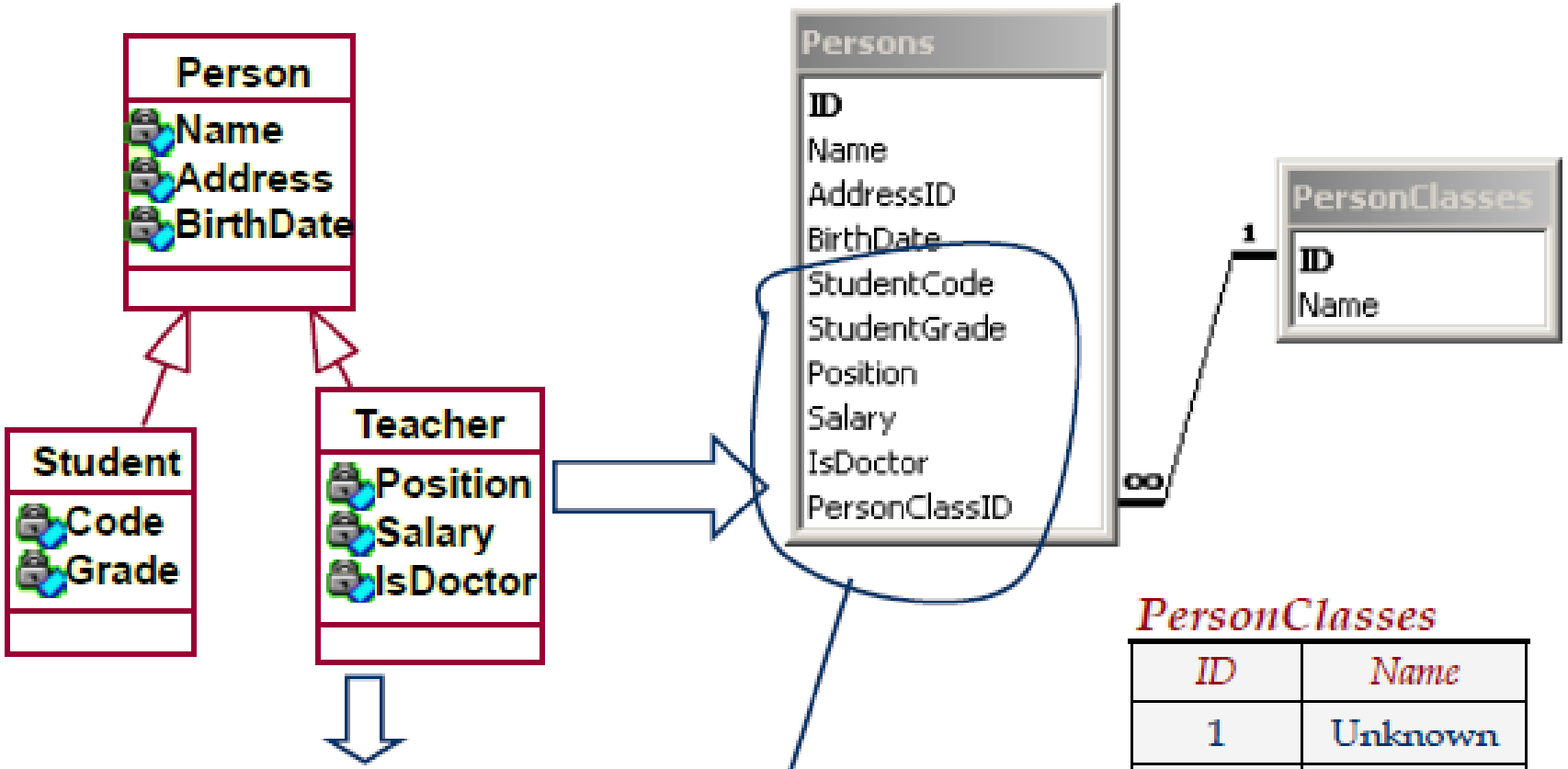

```
CREATE VIEW StudentsFull(...) AS
SELECT Persons.*, Code, Grade
FROM Persons INNER JOIN Students
ON Persons.ID = PersonID
```

Transformarea moştenirii

Metoda 2

- Se crează o singură tabelă (corespunzătoare superclasei) şi se de-normalizează toate atributele subclaselor acesteia.
- Implică crearea celor mai puţine tabele/*view*-uri -opţional, se poate defini o tabelă de subclase şi *view*-uri corespunzătoare fiecărei subclase.
- Se obţine, de obicei, cea mai mare performanţă
- Adăugarea unei noi subclase implică modificări structurale
- Creştere “artificială” a spaţiului utilizat

Transformarea moştenirii

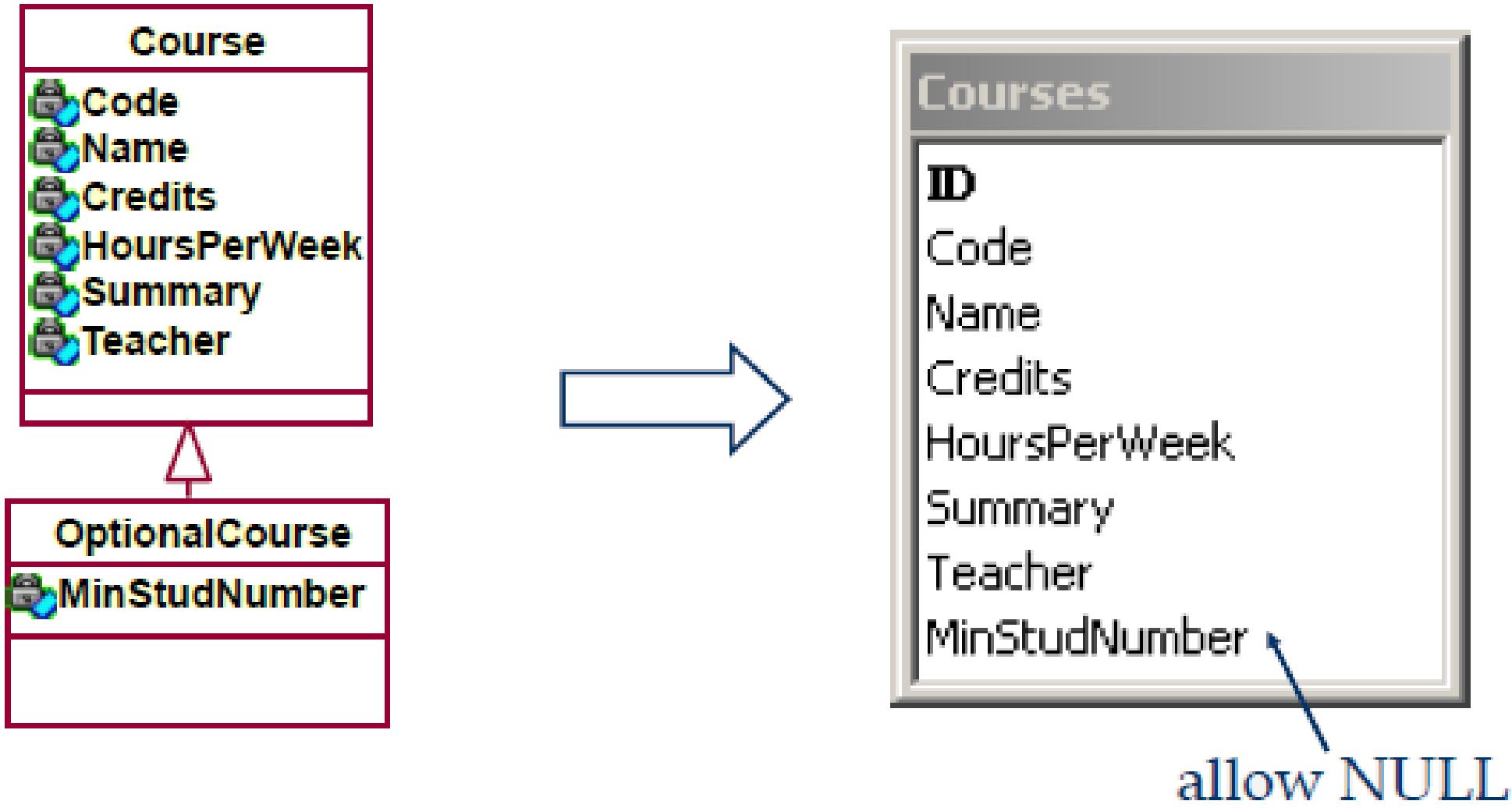



```
CREATE VIEW Students(...) AS
SELECT ID,Name,AddressID,
BirthDate,Code,Grade FROM Persons
WHERE PersonClassID = 2
```

PersonClasses

ID	*Name*
1	Unknown
2	Student
3	Teacher

Transformarea moştenirii

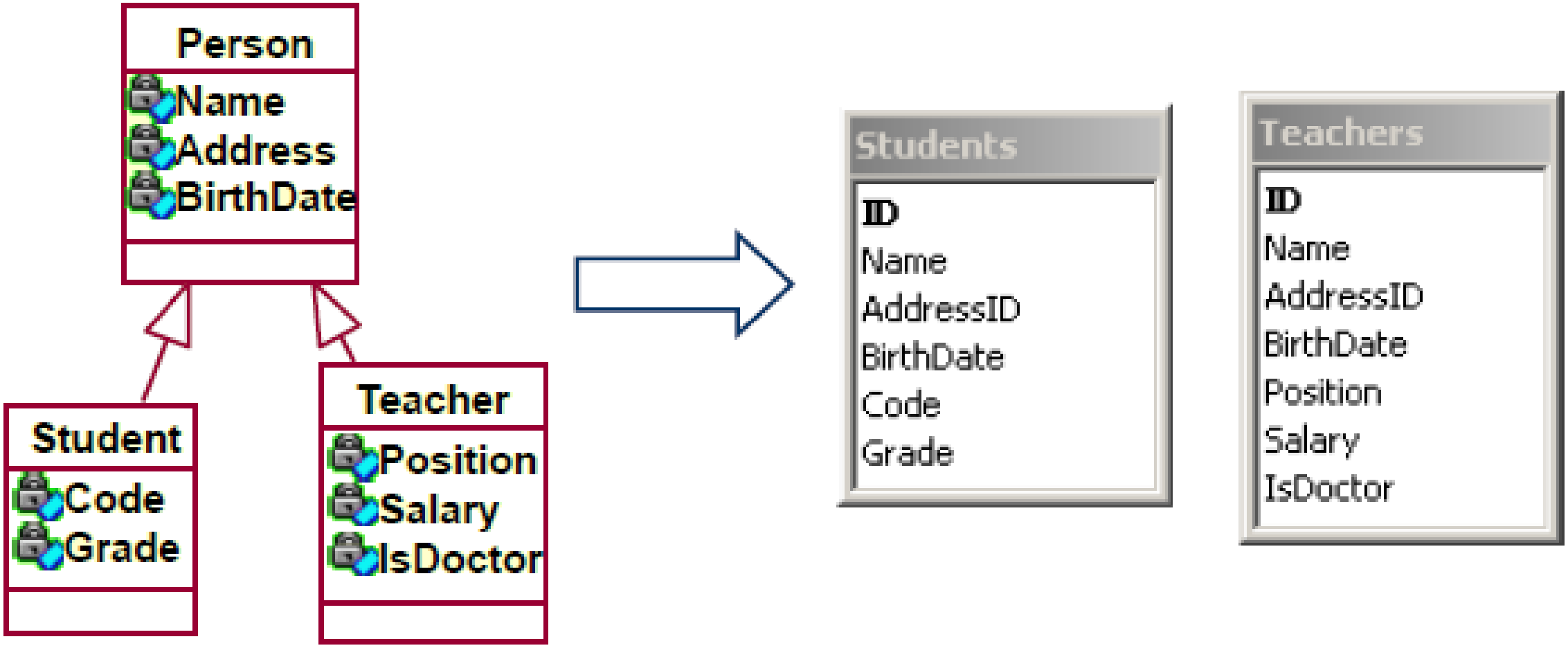


Transformarea moştenirii

Metoda 3

- Presupune crearea câte unui tabel corespunzător fiecărei sub-clase şi de-normalizarea atributelor super-clasei în fiecare dintre tabelele create
- Performanţa obţinută este satisfăcătoare
- Adăugarea unei noi subclase **nu** implică modificări structurale
- Posibilele modificări structurale la nivelul superclasei afecteaza toate tabelele definite!

Transformarea moştenirii



Transformarea moştenirii

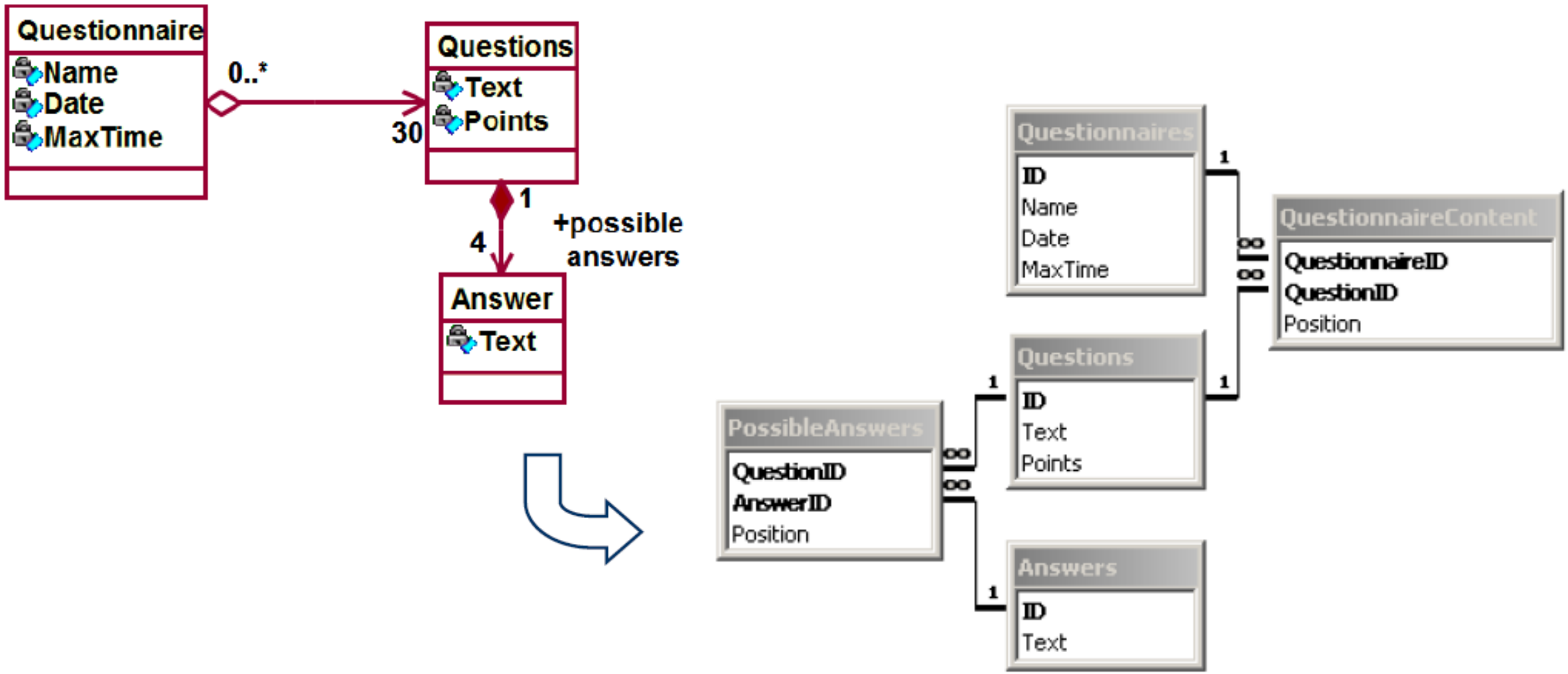
Care este metoda potrivită?

- Dacă numărul înregistrărilor stocate în tabele este redus (deci performanţa 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 instanţe puţine atunci cea mai bună este utilizarea Metoda 2.

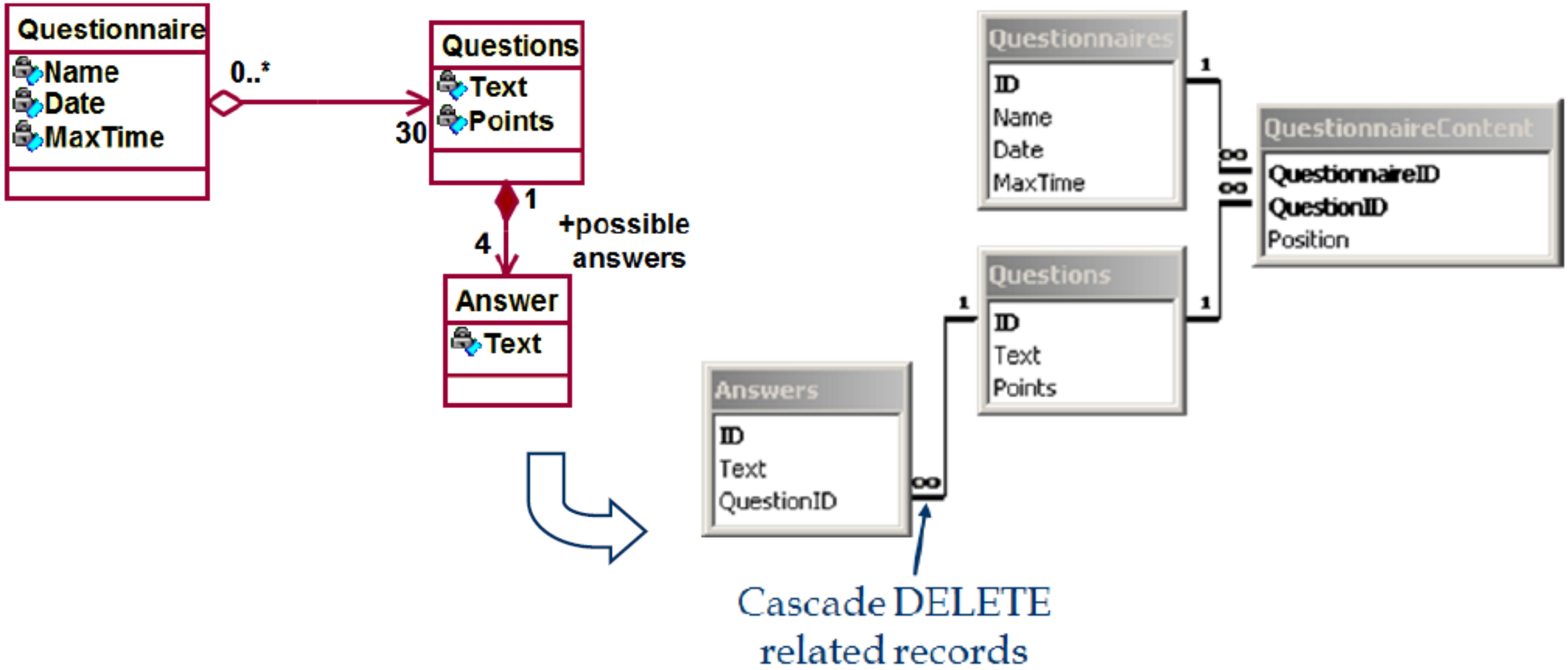
Transformarea agregării/compunerii

- Agregarea şi compunerea sunt modelate în mod asemănător modelării asocierilor
- În cazul relaţiilor de compunere de obicei se utilizează o singură tabelă (*cross-tables*) -deoarece compunerea implică mai multe relaţii *1:1*
- Numărul fix de “părţi” 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)

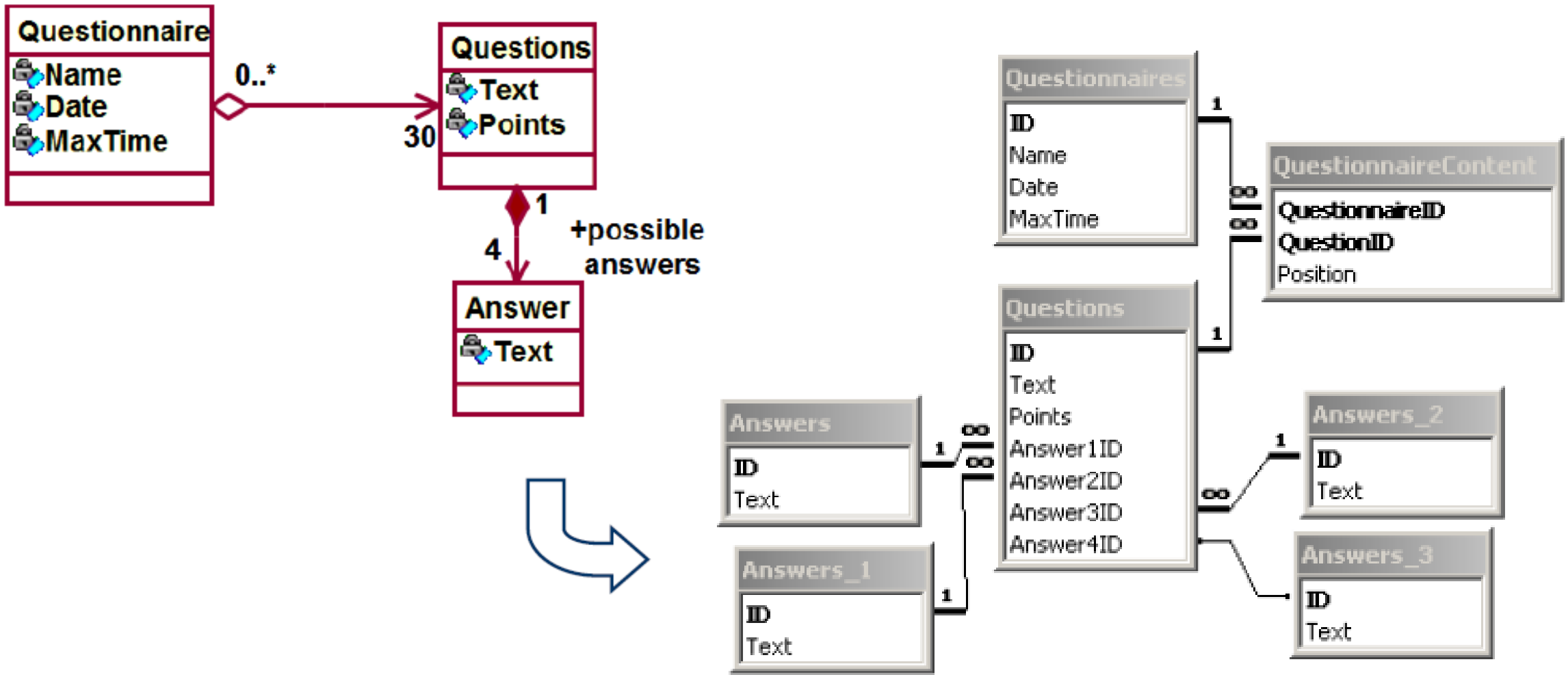
Transformarea agregării/compunerii



Transformarea agregării/compunerii

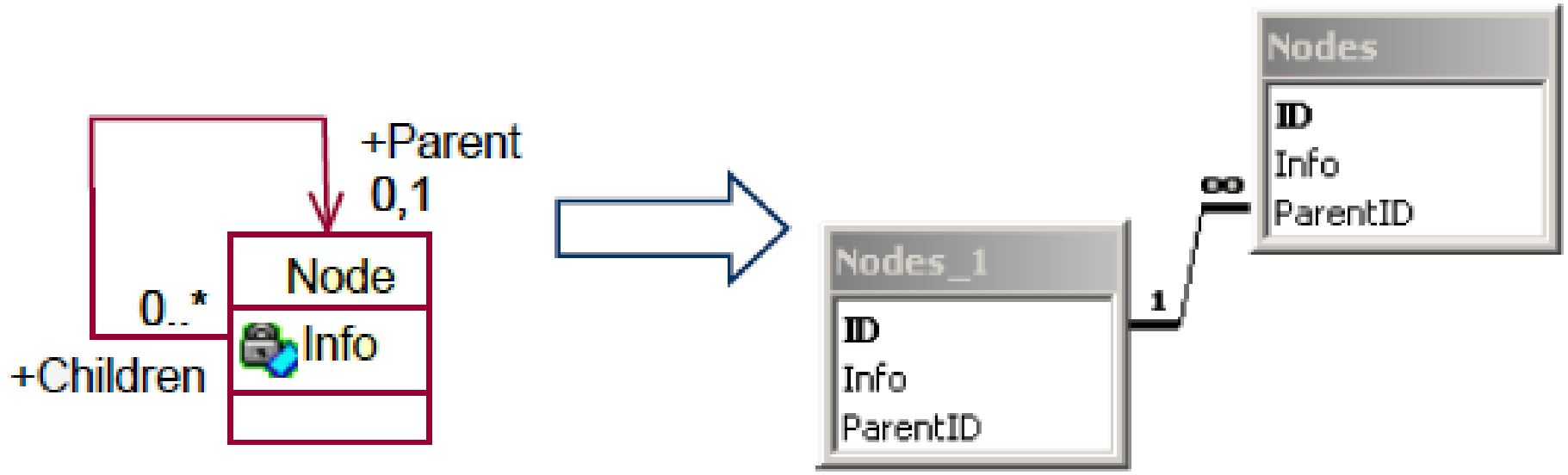


Transformarea agregării/compunerii



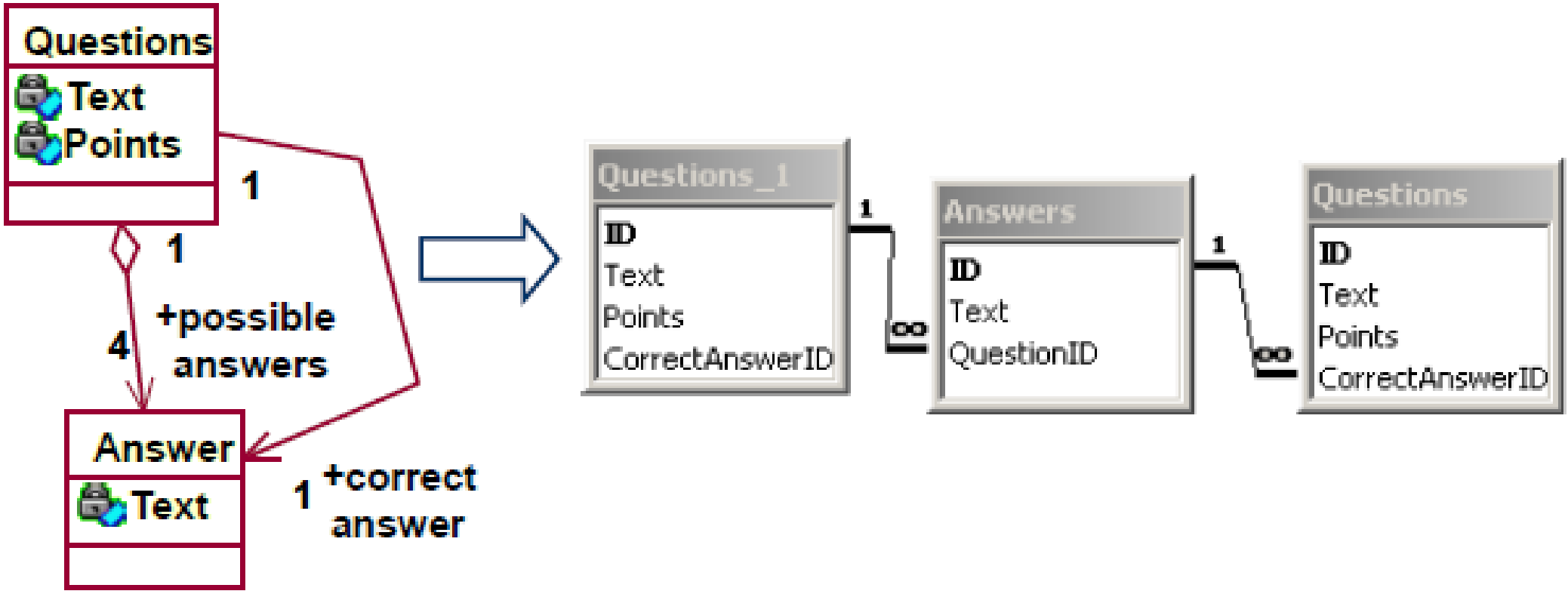
Transformarea auto-asocierilor

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



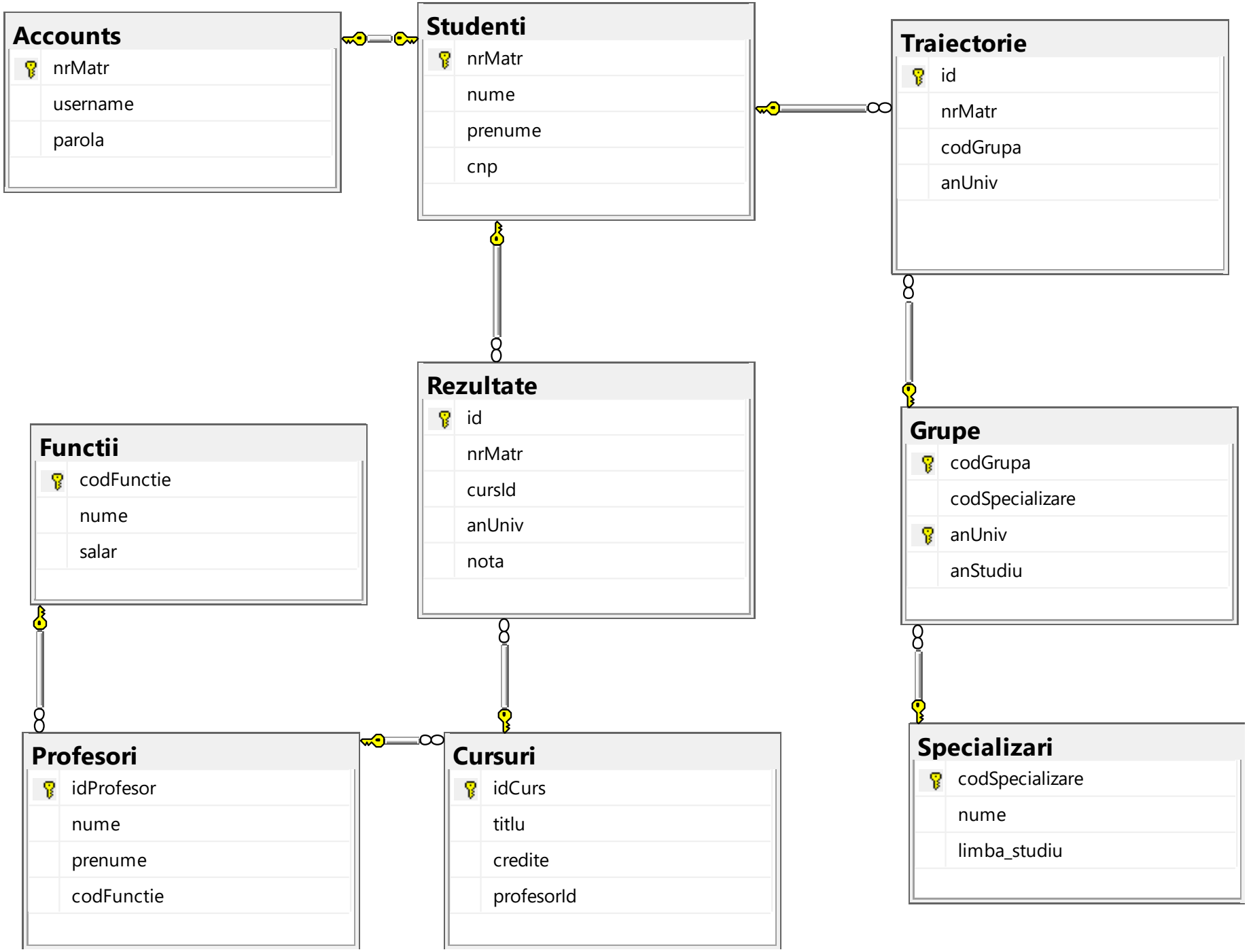
Transformarea auto-asocierilor

- ON DELETE CASCADE (ștergerea în cascadă) generează o problemă similară şi în cazul a două tabele ce se referă reciproc



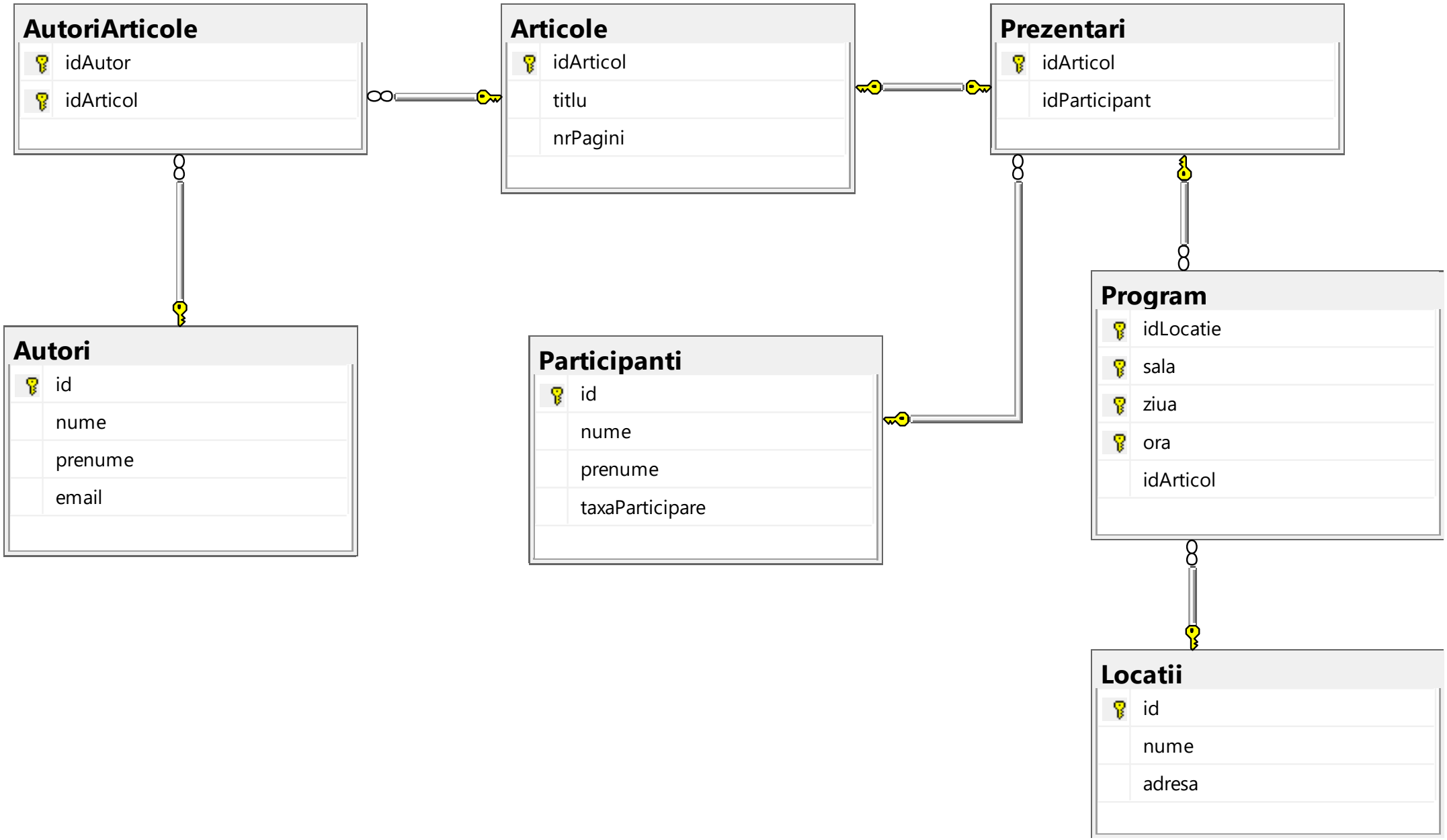
Facultate – Exercițiu de modelare

- Vrem să ținem evidența unor informații dintr-o facultate pentru a gestiona următoarele:
- Studenți: nume, prenume, cnp, număr matricol și an de studiu
- Conturile studenților (username, parola)
- Grupe de studiu (pentru mai mulți ani universitari): specializări, grupe și apartenența studenților la aceste formații
- Numerotarea grupelor pentru un an universitar la o specializare se păstrează (ex. 2018-2019: 711,712 pt anul I la IR, in 2019-2020 acei studenți trec în grupele 721, 722, iar 711 si 712 se vor folosi pentru noii studenți de anul 1)
- Dacă un student nu trece anul va trebui să repete anul și va fi înscris într.-o altă grupă
- Disciplinele și rezultatele studenților (disciplina, nota și numărul de credite, anul universitar și semestrul)
- Un student poate da examenul în mai mulți ani universitari (dacă e restanțier)
- Profesorii si funcțiile lor (asistent, lector, etc.), fiecare funcție are un salar fix


Accounts
nrMatr
username
parola
Studenti
nrMatr
nume
prenume
cnp
Traiectorie
id
nrMatr
codGrupa
anUniv
Rezultate
id
nrMatr
cursId
anUniv
nota
Functii
codFunctie
nume
salar
Grupe
codGrupa
codSpecializare
anUniv
anStudiu
Profesori
idProfesor
nume
prenume
codFunctie
Cursuri
idCurs
titlu
credite
profesorId
Specializari
codSpecializare
nume
limba_studiu

Conferință – Exercițiu de modelare

- Vrem să ținem evidența unor informații pentru o conferință pentru a gestiona următoarele:
- Autorii articolelor (nume, prenume, email)
- Articolele care urmează să fie prezentate la conferință (titlu, numar pagini, autori)
- Participanții la conferință – nu toți autorii participă la conferință, și nu toți participanții au în mod obligatoriu un articol la conferință
- Fiecare articol trebuie să fie prezentat de către unul din autori (care este și participant la conferință)
- Programul prezentărilor (ziua, ora, sala, locația)


AutoriArticole
idAutor
idArticol
Articole
idArticol
titlu
nrPagini
Prezentari
idArticol
idParticipant
Autori
id
nume
prenume
email
Participanti
id
nume
prenume
taxaParticipare
Program
idLocatie
sala
ziua
ora
idArticol
Locatii
id
nume
adresa