



```
SELECT curs  
FROM BAZE_DE_DATE
```

```
WHERE INITCAP(capitol) = 'Proiectarea Bazelor De Date Relaționale'  
AND topic = ' Modelul Entitate/Asociere'
```

Mihaela Elena Breabăn

© FII 2024-2025

Baze de date – cuprinsul cursului

▶ Concepte din Baze de date (C1)

▶ Algebra relațională (C2-C3)

▶ Dependențe funcționale și multivaluate (C4-C5)

▶ Normalizare (C6-C7)

▶ **Proiectarea conceptuală: Modelarea Entitate/Asociere (C9)**

▶ **Proiectarea fizică (C10-C11)**

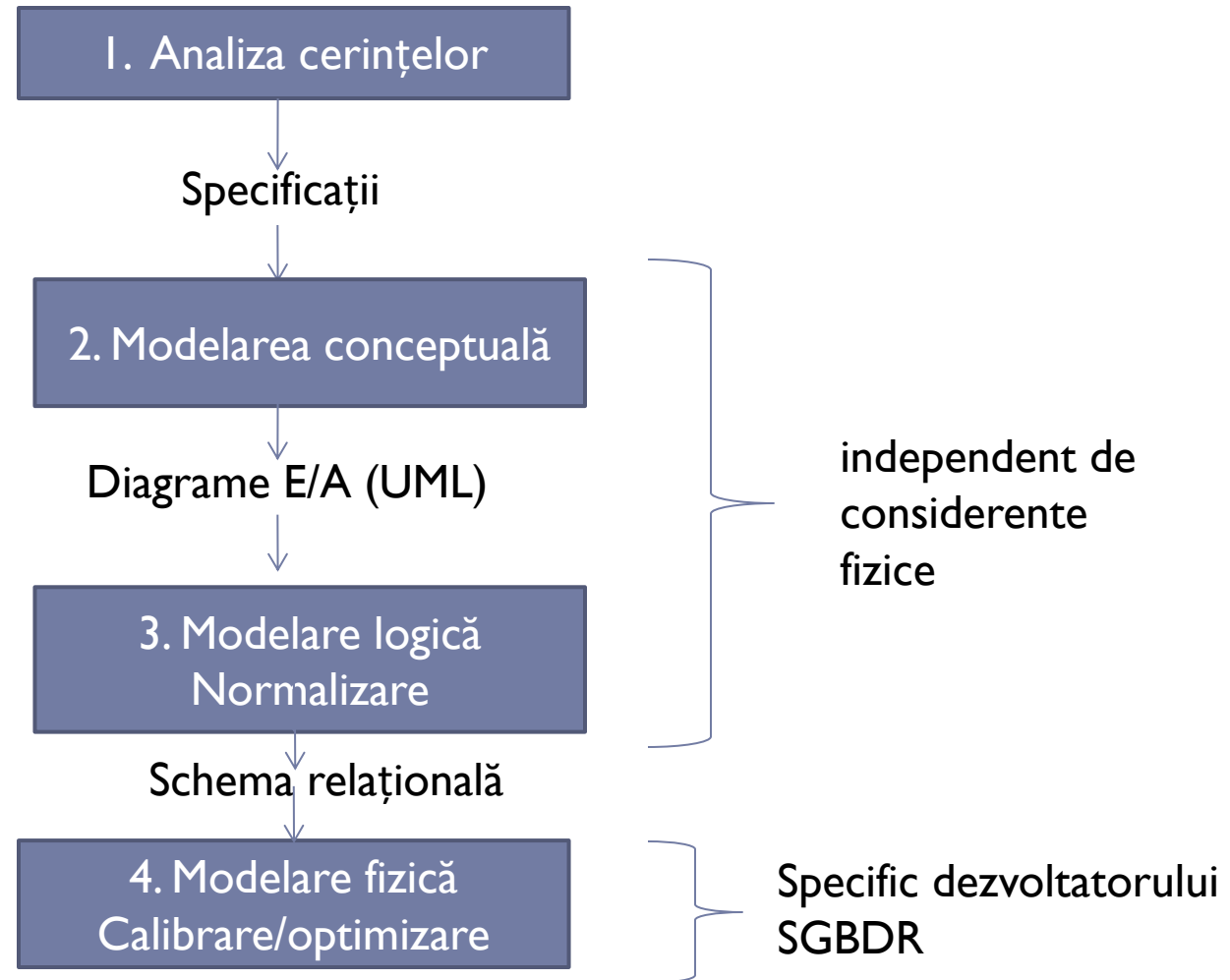
▶ **Indexarea (C12-C13)**

▶ **Procesarea interogărilor (C14)**

**Proiectarea
și optimizarea
Bazelor de date
Relaționale**

Proiectarea Bazelor de date Relaționale

Metodologie



Planul prelegerii

- ▶ Problematika proiectării schemei
- ▶ Proiectarea E/A în notația lui Chen
 - ▶ Concepte E/A
 - ▶ Modelarea constrângerilor
 - ▶ Capcane de conectare
- ▶ Proiectarea E/A în UML
- ▶ Din E/A în schemă relațională

Proiectarea schemei bazei de date

- ▶ Pentru o bază de date putem propune mai multe scheme
 - ▶ Unele sunt (mult) mai bune decât altele
 - ▶ Redundanță?
 - ▶ Eficiență?
 - ▶ Consistență?
- ▶ **Cum generăm scheme bune?**
- ▶ Două abordări:
 - ▶ Descompunere - normalizare (Codd, '70-'74)
 - ▶ Modelarea E/A (Chen,'76)
- ▶ De obicei sunt aplicate împreună, în doi pași: se începe cu proiectarea E/A și se continuă cu normalizarea

Concepte E/A clasice (Chen 1976)

▶ Entitate

- ▶ Date ce pot fi modelate ca obiecte având existență independentă
- ▶ *Mulțime-entitate* sau *tip-entitate* - corespunde unui grup de obiecte de același tip, deci unei mulțimi omogene de entități; este caracterizată de un nume și o listă de proprietăți
- ▶ O *instanță – entitate* este un obiect din grup și trebuie să fie **unic identificabilă în cadrul mulțimii**

▶ Asociere (Relationship)

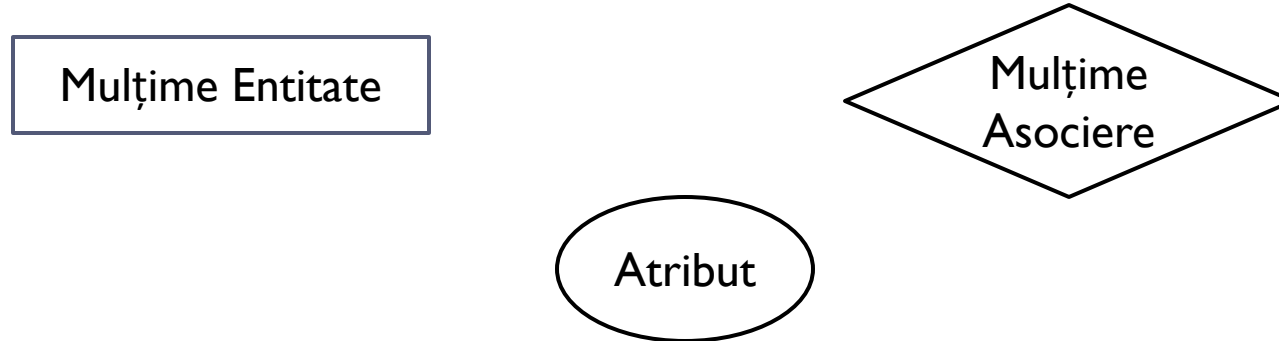
- ▶ Conexiune/asociere între două sau mai multe entități de tip diferit sau de același tip
- ▶ *Mulțime-asociere* sau *tip-asociere* – corespunde unei mulțimi omogene de asocieri, modelează interacțiuni între mulțimi-entitate
- ▶ O *instanță-asociere* este identificabilă prin instanțele-entitate participante
- ▶ Gradul asocierii = numărul de mulțimi-entitate participante
 - ▶ unare/recursive, binare, ternare...

▶ Atribut

- ▶ Pentru entități reprezintă proprietăți ce descriu obiectele/instanțele entitate
- ▶ Pentru asocieri
 - ▶ Atribute ale entităților implicate
 - ▶ Atribute specifice asocierii ce reprezintă informații noi (atribute proprii)

Diagrame E/A

- ▶ **Reprezentare grafică a conceptelor E/A**
 - ▶ Există mai multe standarde grafice, aici varianta Chen

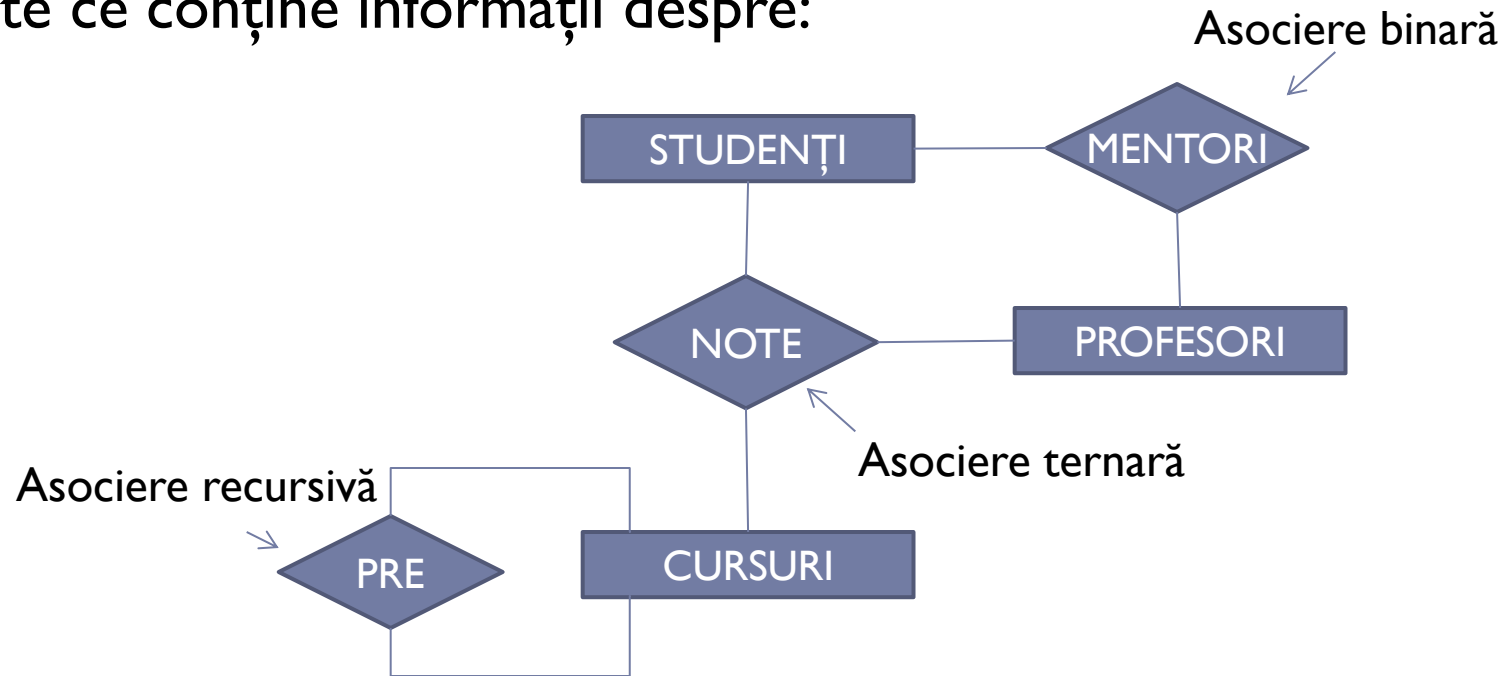


- ▶ **Un graf**
 - ▶ Mulțimile-entitate, mulțimile-asociere și attributele sunt noduri
 - ▶ Există muchii doar între
 - ▶ noduri-entitate și noduri-asociere
 - ▶ noduri-entitate și noduri-atribute
 - ▶ noduri-asociere și noduri-atribute

Exemplu

▶ O bază de date ce conține informații despre:

- ▶ Studenți
- ▶ Profesori
- ▶ Cursuri
- ▶ Note
- ▶ Mentori



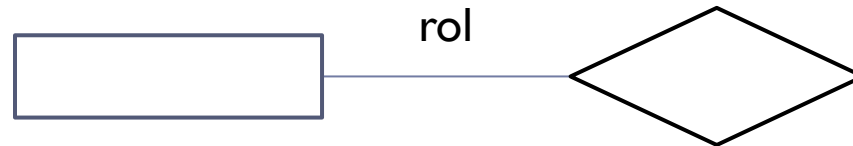
▶ Cerințe:

- ▶ Putem determina notele obținute la cursuri pentru orice student, precum și profesorii care au pus notele
- ▶ Putem determina mentorul (profesorul îndrumător al) oricărui student
- ▶ Putem identifica condițiile necesare pentru a studia un curs (fiecare curs ar putea necesita cunoașterea informațiilor din alte cursuri)

Alte concepte E/A

▶ Rol

- ▶ Explică semnificația entităților în asocieri



▶ Cheie primară

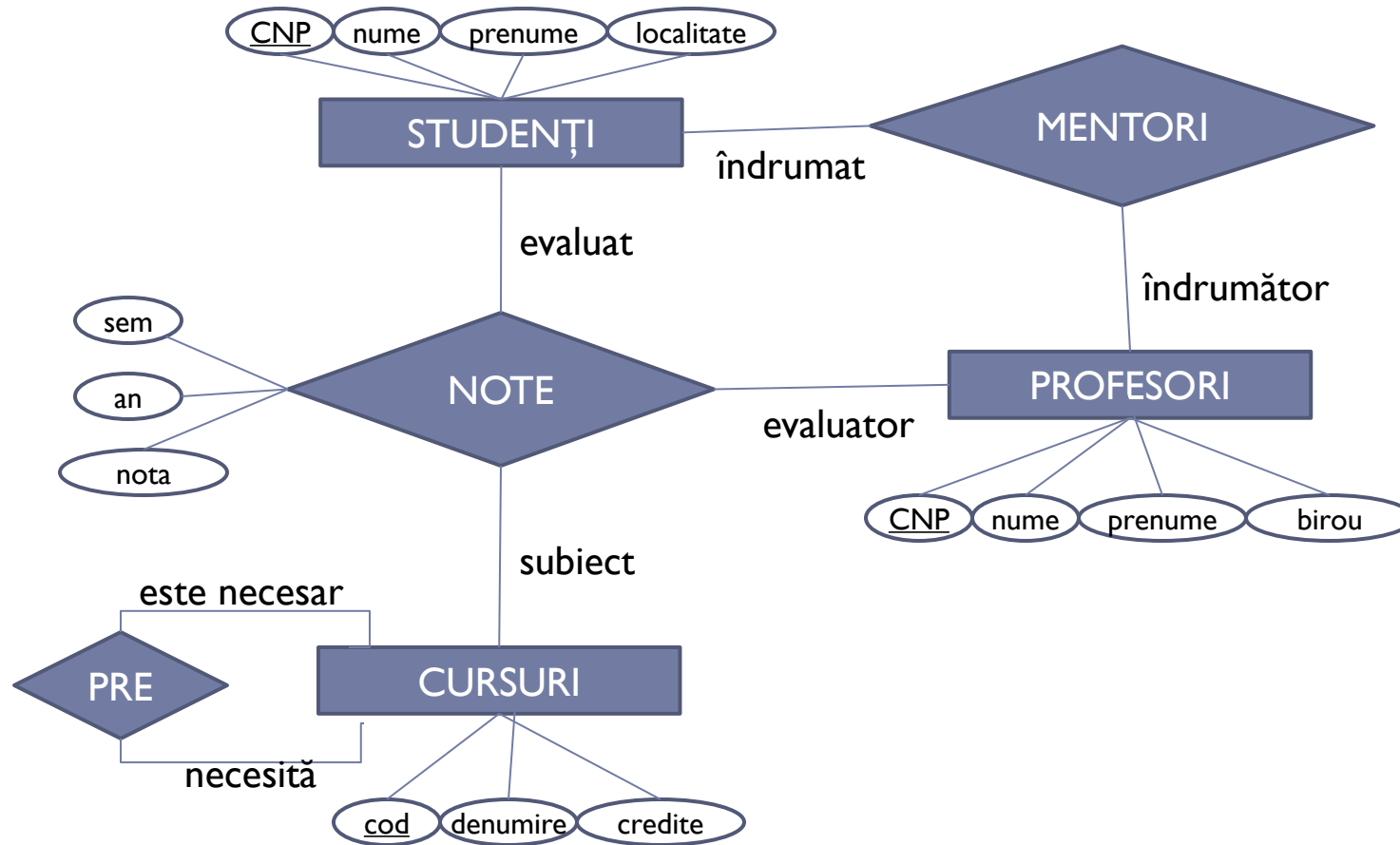
- ▶ Un atribut sau o submulțime minimală de atribute ce identifică unic o instanță-entitate sau o instanță-asociere
- ▶ **Obligatorie pentru entități**, pentru a indica care instanțe participă în asocieri

Cheie primară

▶ Cheie străină **pentru o asociere**

- ▶ Un atribut sau o mulțime de atribute care constituie cheia primară pentru entitățile implicate

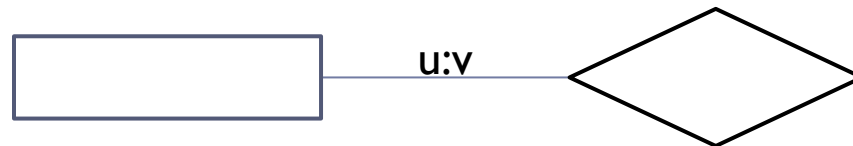
Exemplu



Care sunt cheile străine pentru cele trei mulțimi de asocieri?

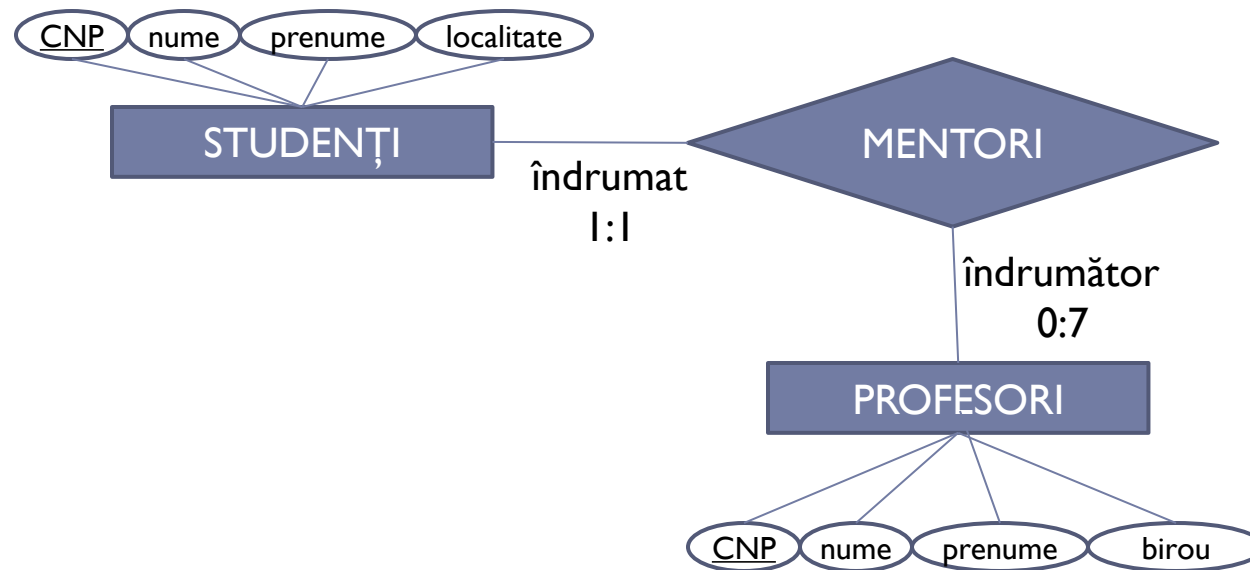
Constrângeri de conectivitate/participare

- ▶ Modelul E/A permite declararea de constrângeri asupra numărului de instanțe-asociere în care o instanță-entitate participă
- ▶ Fie R o mulțime-asociere între n mulțimi-entitate $E_i, i \in 1..n$. Baza de date satisface constrângerea (E_i, u, v, R) dacă fiecare instanță-entitate din E_i participă în cel puțin u și cel mult v instanțe-asociere din R .

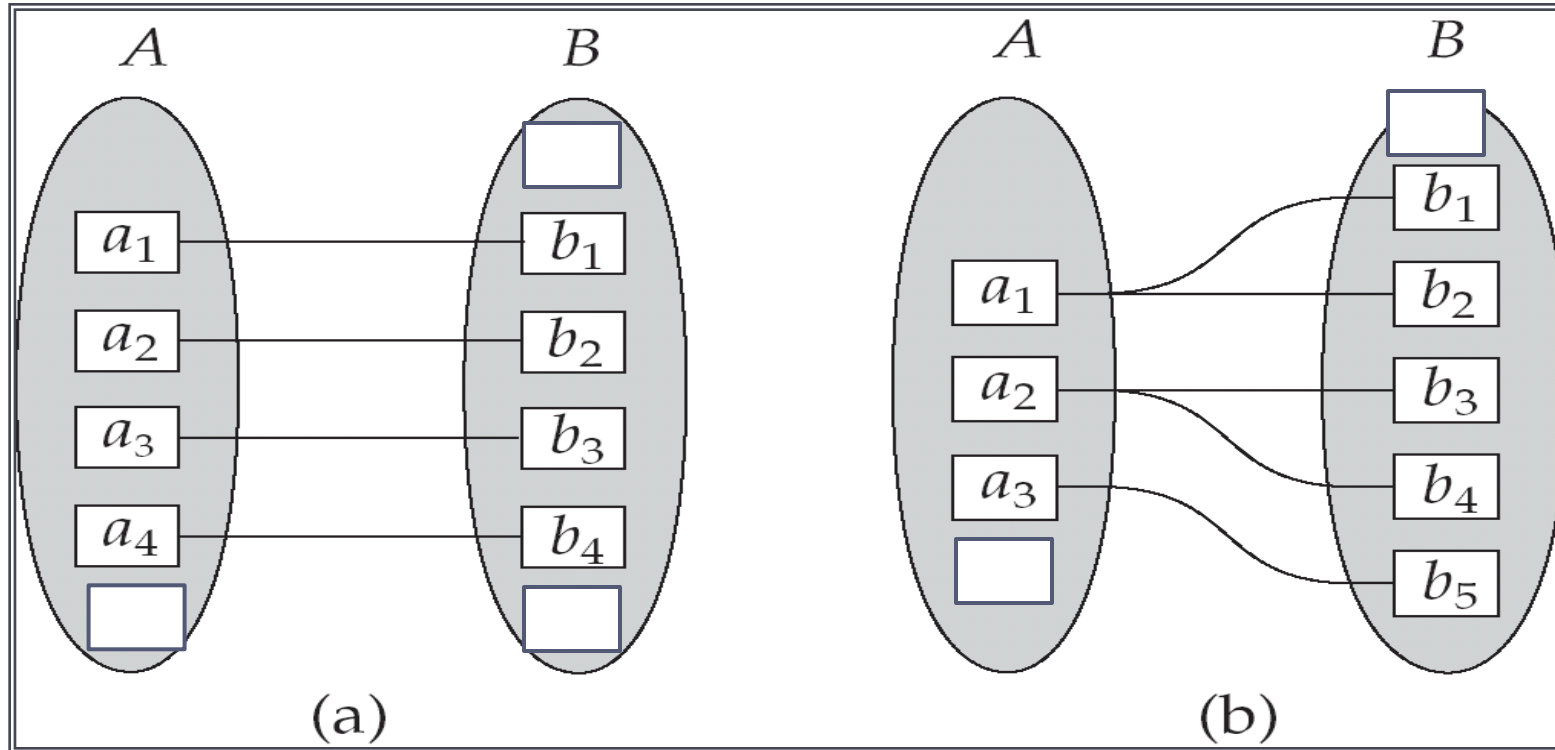


Exemplu

- ▶ (Studenti, 1, 1 Mentori)
- ▶ (Profesori, 0, 7, Mentori)
- ▶ Fiecare student are un singur profesor drept mentor iar un profesor poate fi mentor pentru cel mult 7 studenti



Constrângeri de conectivitate pentru asocieri binare (1)



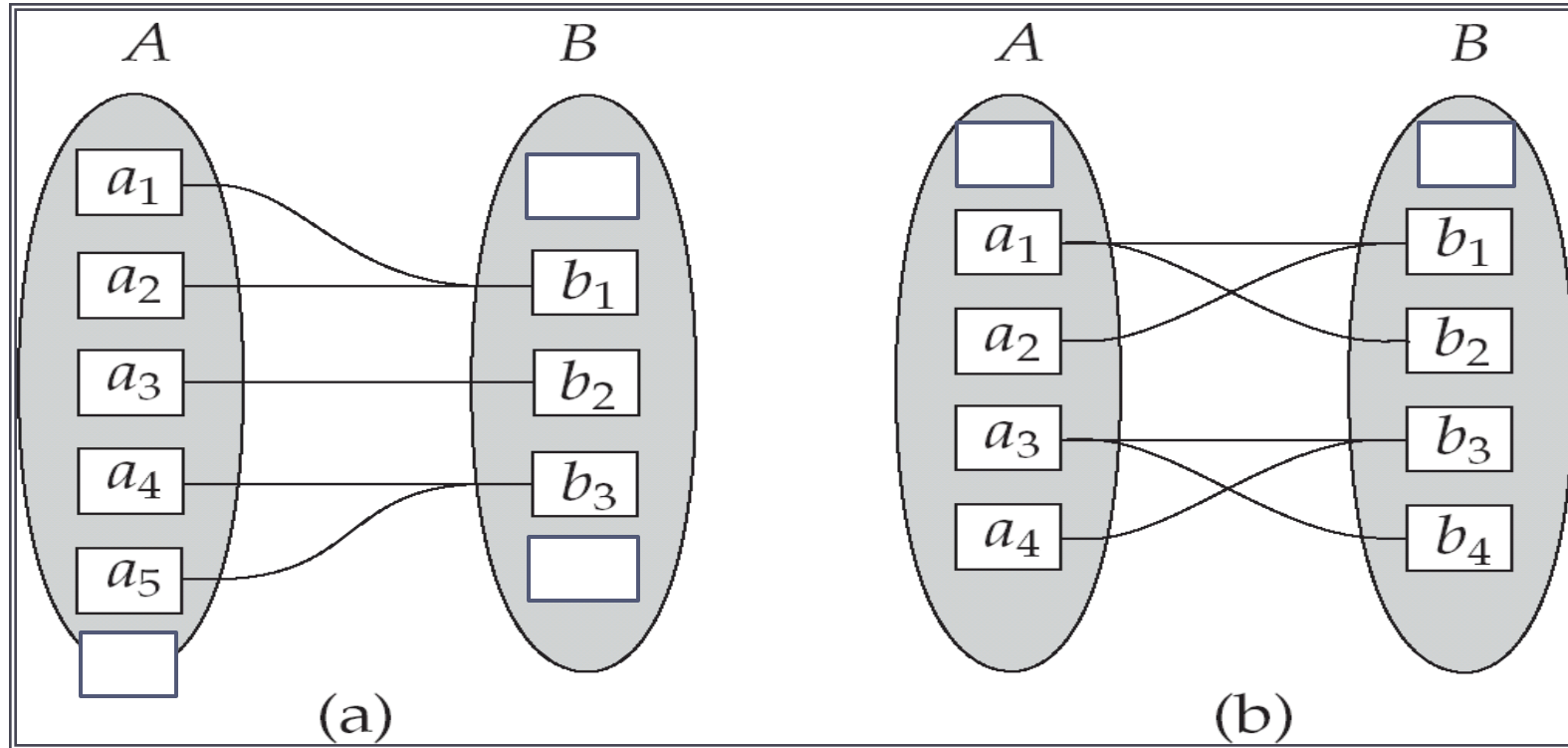
a) Asociere unu la unu (incompletă)
(A,0,1,R) (B,0,1,R)



b) Asociere unu la mulți
(A,0,n,R) (B,0,1,R), $n > 1$



Constrângeri de conectivitate pentru asocieri binare (2)



a) Asociere mulți la unu
(A,0,1,R) (B,0,n,R)

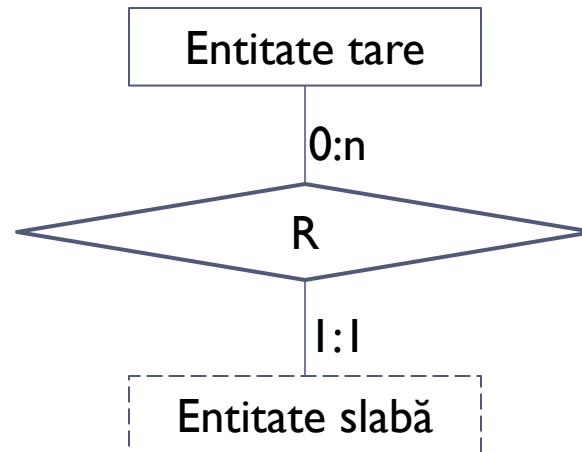


b) Asociere mulți la mulți
(A,0,m,R) (B,0,n,R), $m,n > 1$



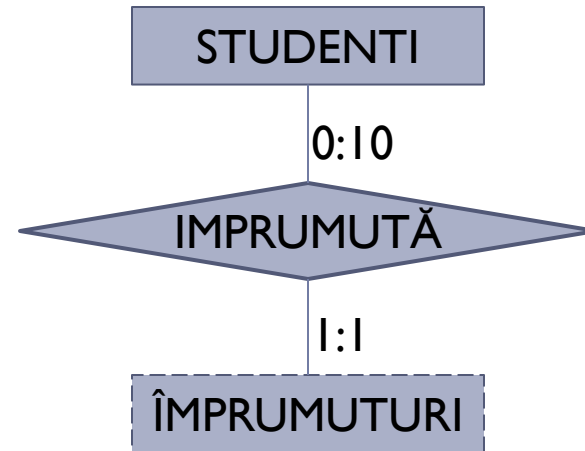
Entitate slabă

- ▶ O mulțime-entitate este slabă dacă existența instanțelor sale depinde de existența instanțelor altei mulțimi-entitate (dependență existențială)

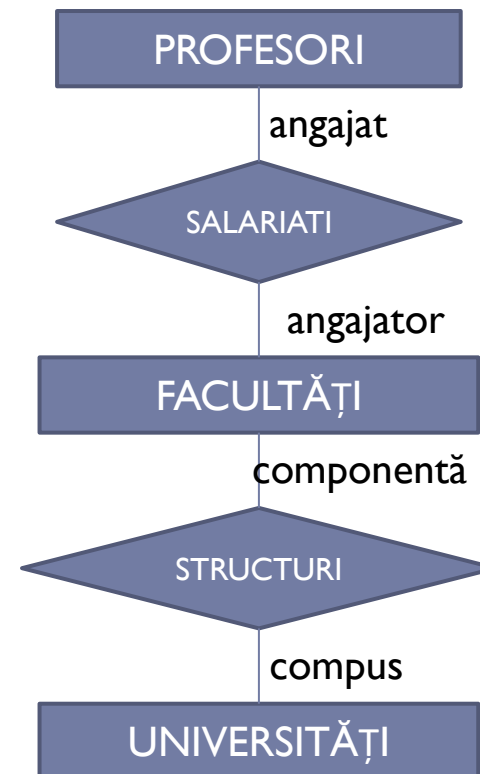
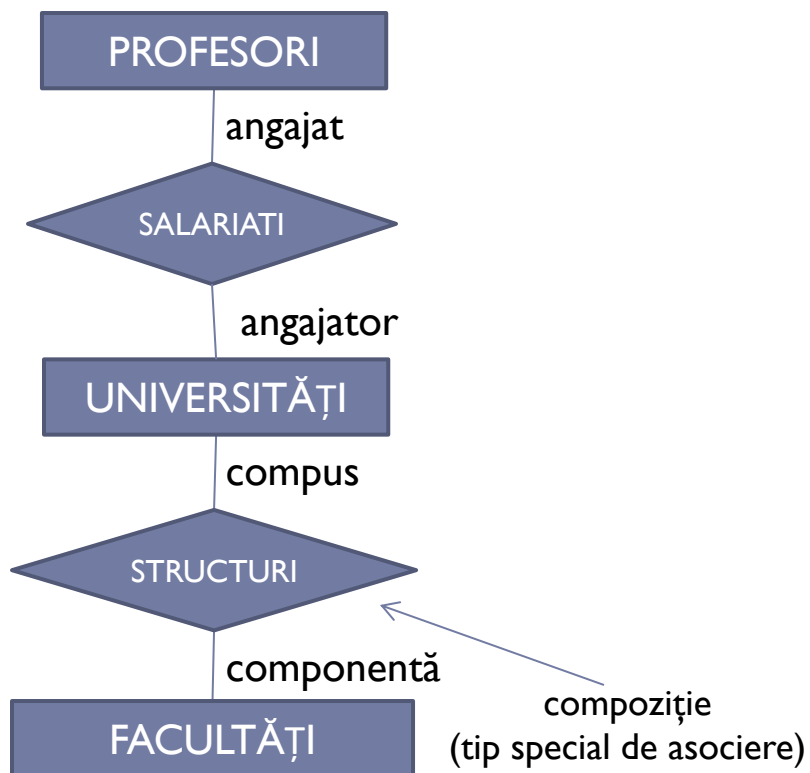
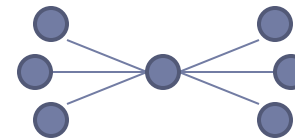


- ▶ Nu are cheie
- ▶ Satisface constrângerea de conectivitate (Entitate_slaba, 1, 1, R), deci participă într-o asociere de tip unu la mulți relativ la entitatea tare

Exemplu



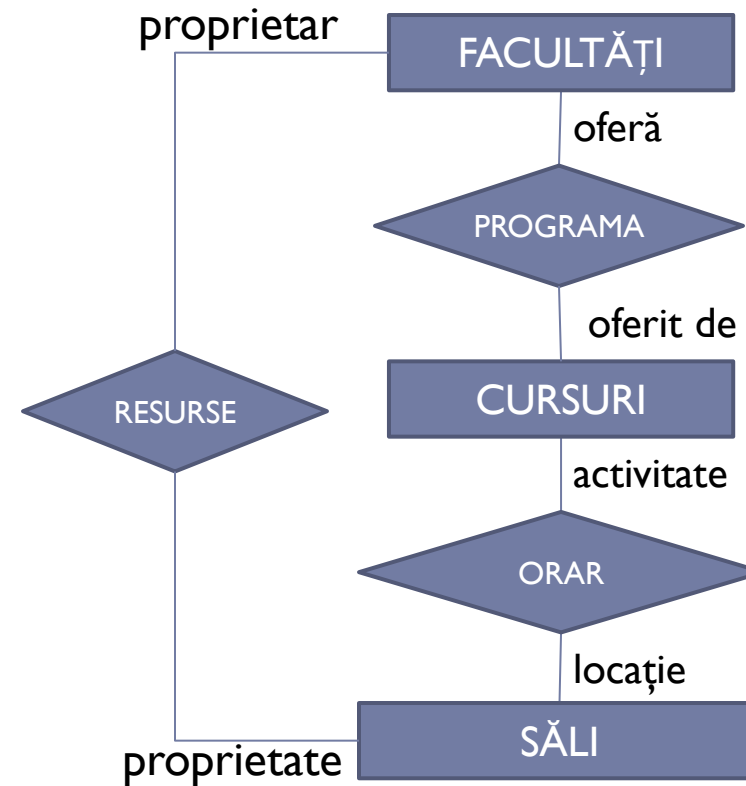
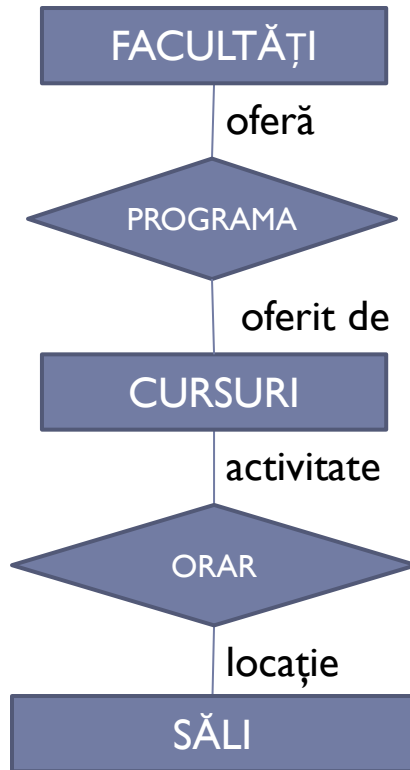
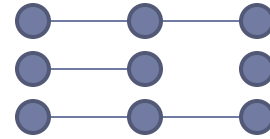
Capcane de conectare (Fan traps)



Problema:
La ce facultate aparține profesorul X?

Soluția:
Model restructurat

Capcane de conectare (Chasm traps)



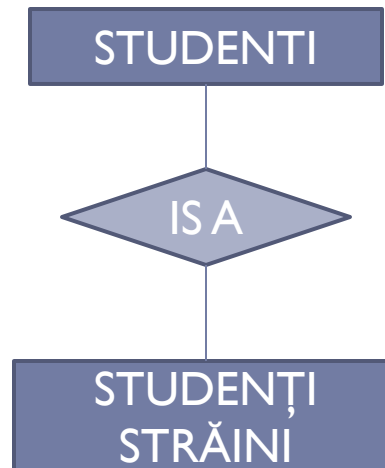
Problema:
Care sunt toate sălile ce aparțin unei facultăți?

Soluția:
Noi asocieri

Modelul E/A extins

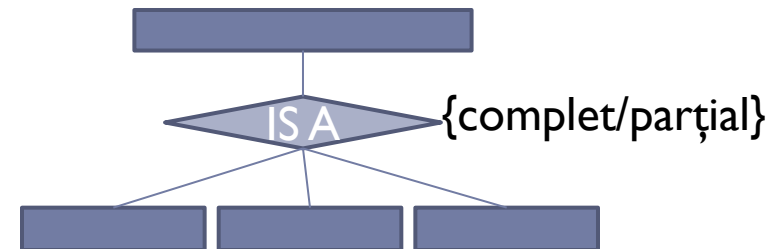
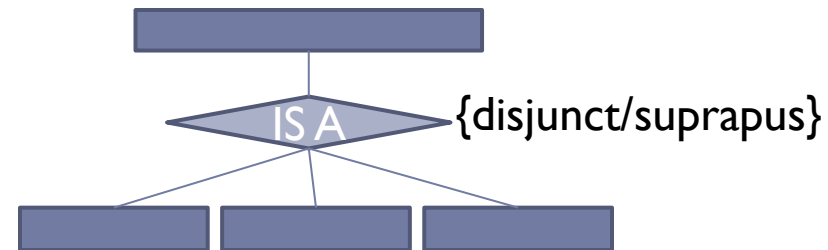
Specializare

- ▶ **Subgrupuri distinctive de instanțe-entitate**
 - ▶ Au în plus anumite atribute sau/și participă în asocieri la care nu participă toate instanțele-entitate
 - ▶ Corespund unei mulțimi de entități specializate care se află într-o asociere de tip IS-A cu mulțimea de entități de bază



Constrângeri specifice specializării

- ▶ Instanțele specializării moștenesc toate atributele și asocierile mulțimii de entități de bază, inclusiv cheia
- ▶ Clasificare
 - ▶ ○ instanță a unei mulțimi-entitate poate aparține la una sau la mai multe specializări
 - ▶ Specializări disjuncte (exclusive)
 - ▶ Specializări cu suprapunere
 - ▶ ○ instanță a unei mulțimi-entitate trebuie sau nu să aparțină la cel puțin o specializare
 - ▶ Complet
 - ▶ Incomplet (parțial)



Modelare UML

▶ Unified Modeling Language

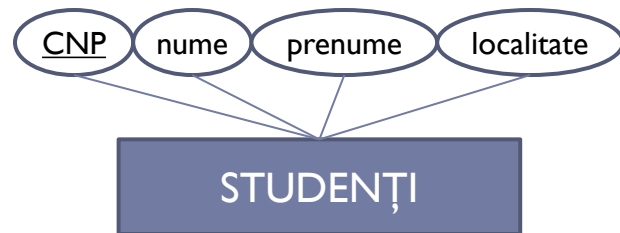
- ▶ Utilizat în ingineria software
- ▶ Bazat pe concepte orientate obiect
- ▶ Instrument de comunicare cu clientul în termenii utilizați în companie
- ▶ Un limbaj bogat, utilizăm un set restrâns de elemente (diagrama de clase) pentru a modela o bază de date.

Mapare E/A – UML

E/R	UML
Mulțime-entitate cu attribute	Clasă
Mulțime-asociere fără attribute proprii	Asociere
Mulțime-asociere cu attribute proprii	Clasă de asociere
Specializare	Subclasă
	Compoziție și agregare

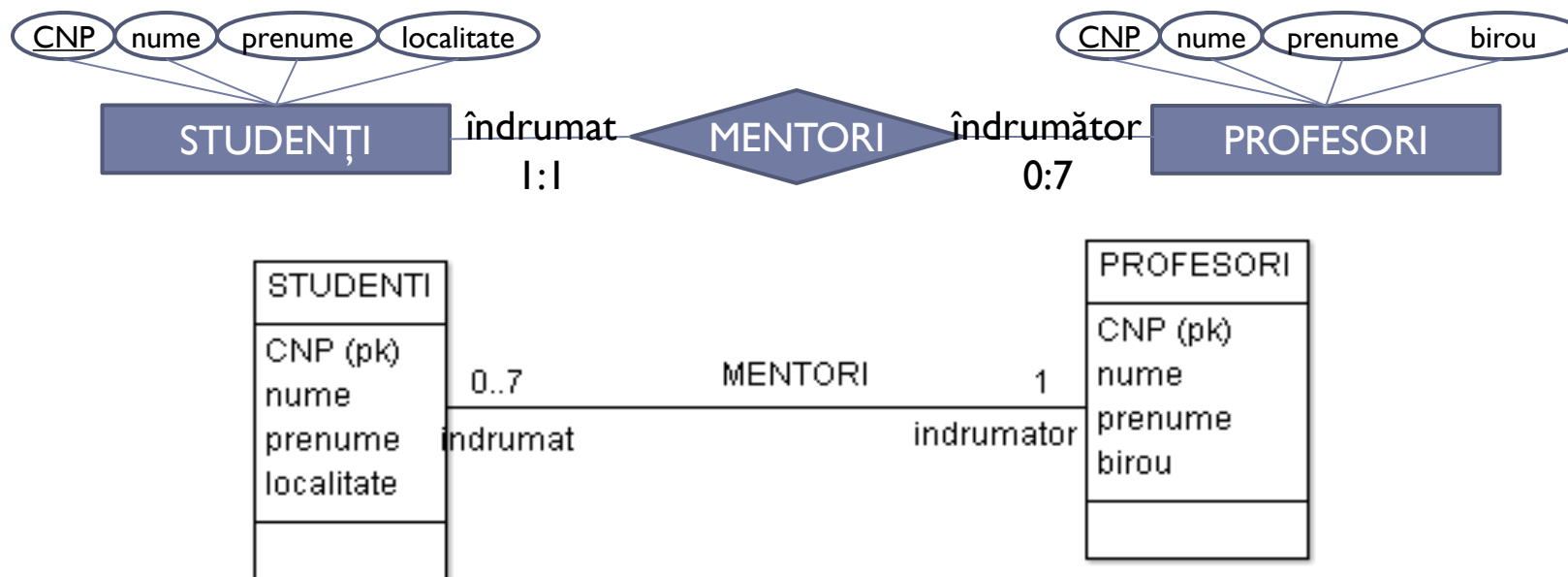
Clase

- ▶ Componente: nume, attribute, metode
- ▶ BD: nume, attribute (cheia primară)



Asocieri

- ▶ Exprimă asocierea dintre obiectele aparținând la două clase
- ▶ BD: asocierea dintre instanțele a două mulțimi-entitate



- ▶ Obs: constrângerile de conectivitate se specifică invers decât în diagramele E/A

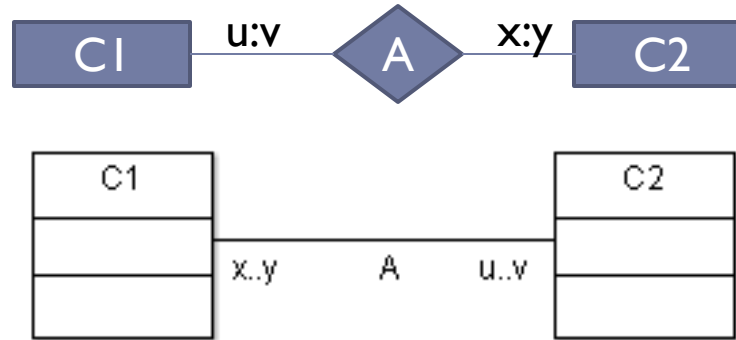
Asocieri

Constrângeri de conectivitate/multiplicitate

- ▶ Restricții

- ▶ (C1,u,v,A)

- ▶ (C2,x,y,A)



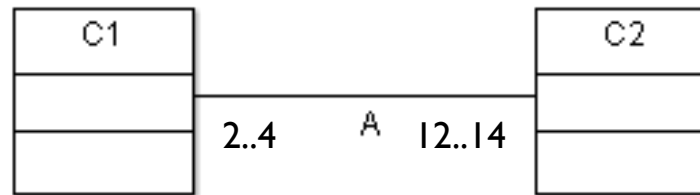
- ▶ Fiecare obiect (instanță-entitate) din C1 este asociat cu cel puțin u și cel mult v obiecte (instanțe-entitate) din C2
- ▶ Fiecare obiect (instanță-entitate) din C2 este asociat cu cel puțin x și cel mult y obiecte (instanțe-entitate) din C1

x..y	u..v	Tip asociere
0..1	0..1	unu la unu incompletă
1..1 (1)	1..1 (1)	unu la unu completă
0..1	0..* (*)	unu la multi incompletă
...

???

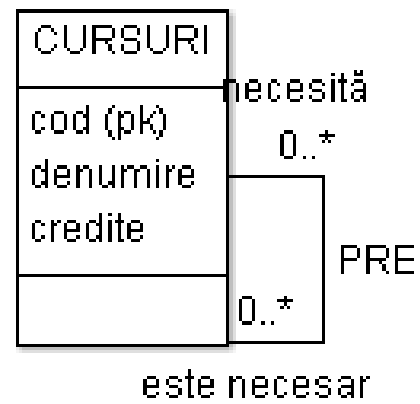
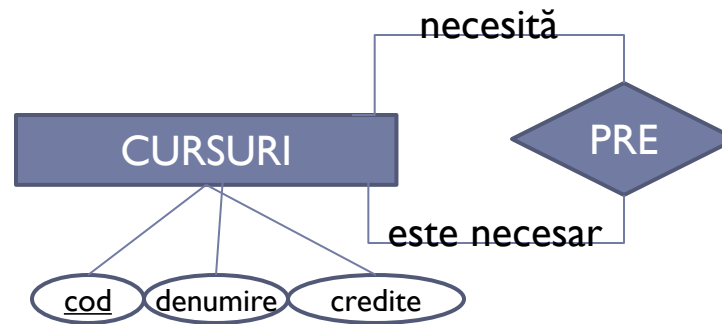
- ▶ Modelați asocierea dintre STUDENȚI și UNIVERSITĂȚI. Un student poate studia la cel mult 2 universități și e necesar să studieze la cel puțin una. O universitate primește cel mult 10.000 studenți.

- ▶ Fie asocierea

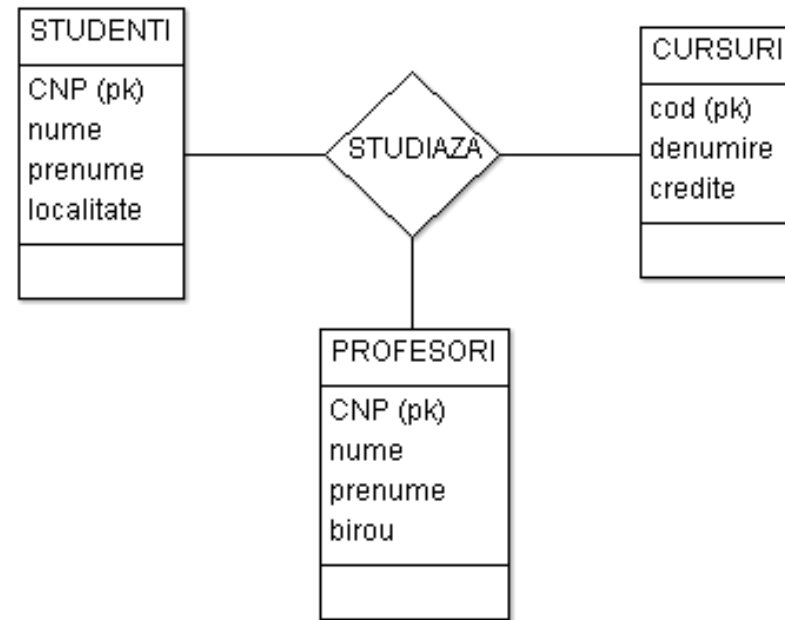


Care e numărul minim de instanțe pentru mulțimea-entitate C1 și pentru C2?

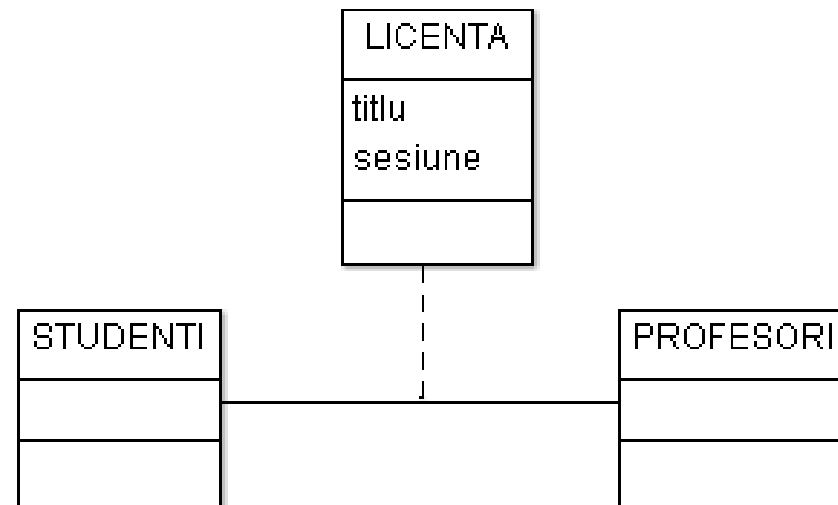
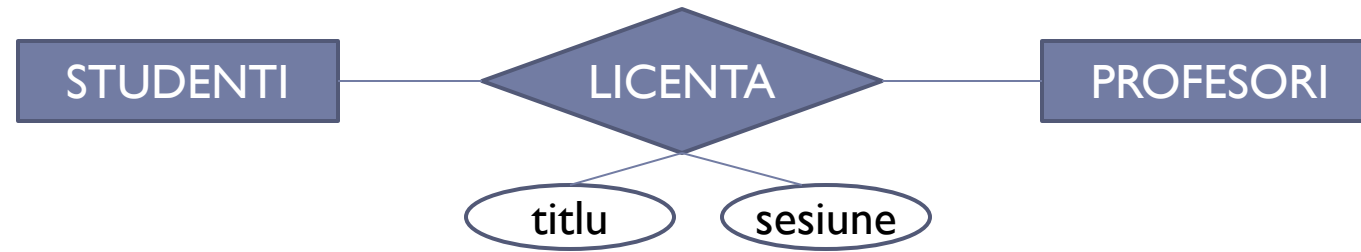
Asocieri recursive



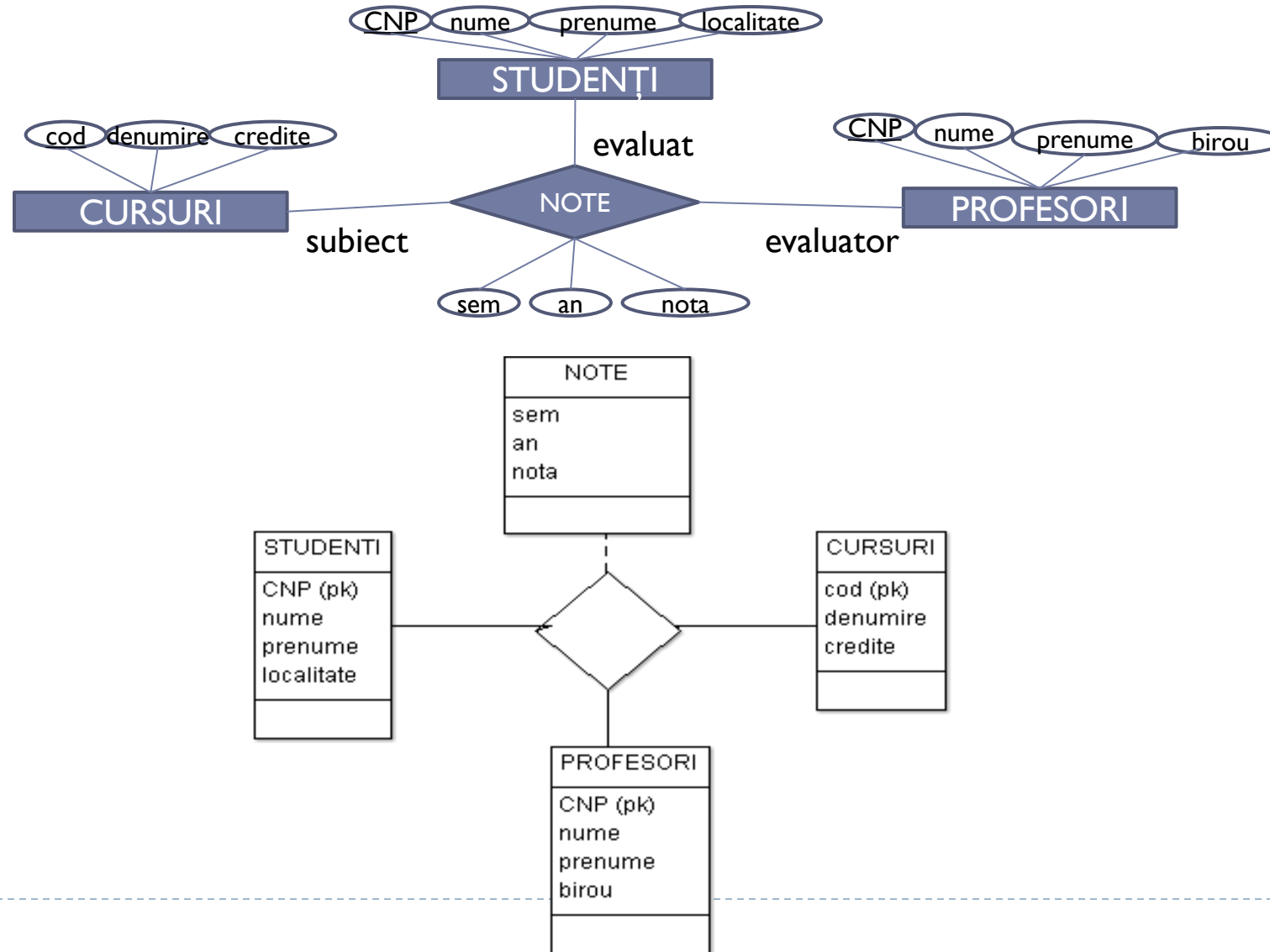
Asocieri n-are



Clase de asociere

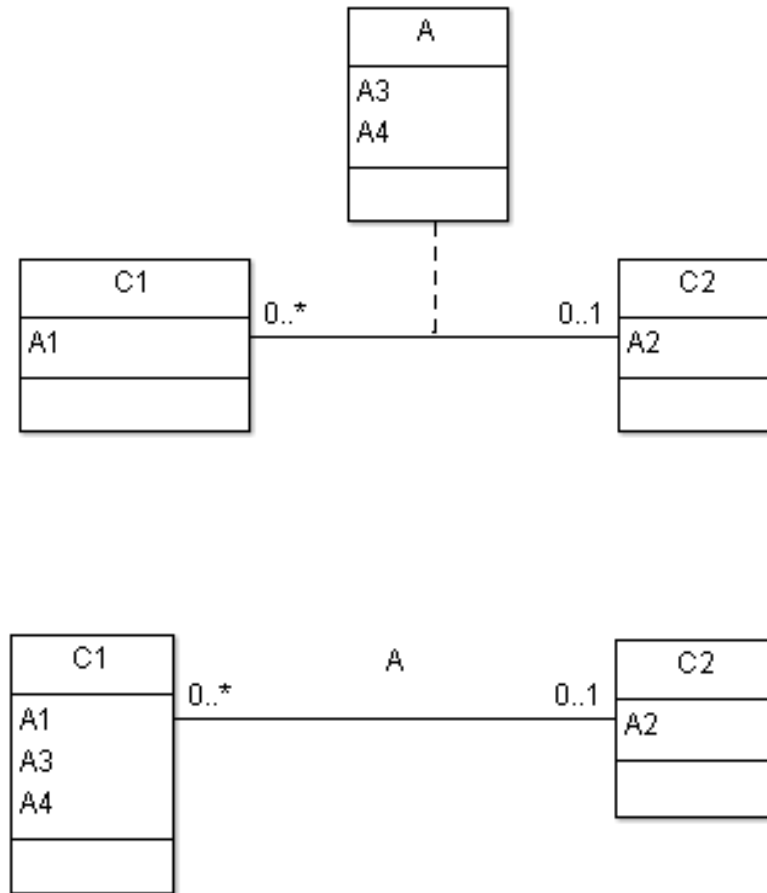


Clase de asociere

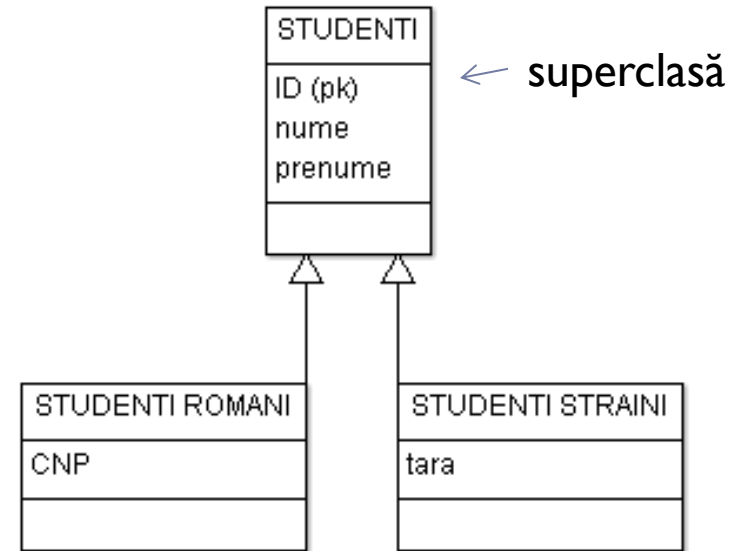
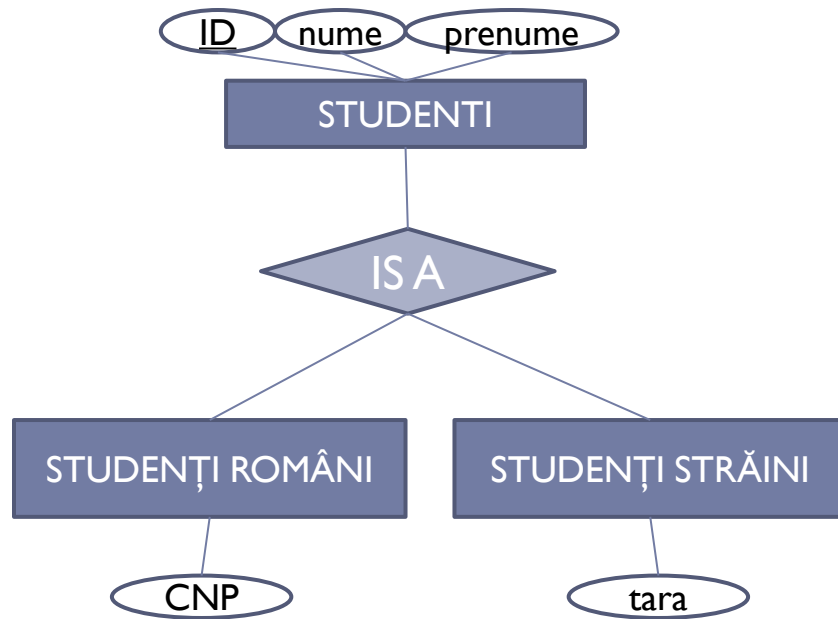


Eliminarea claselor de asociere

- ▶ Atunci când avem multiplicitate 0..1 sau 1..1

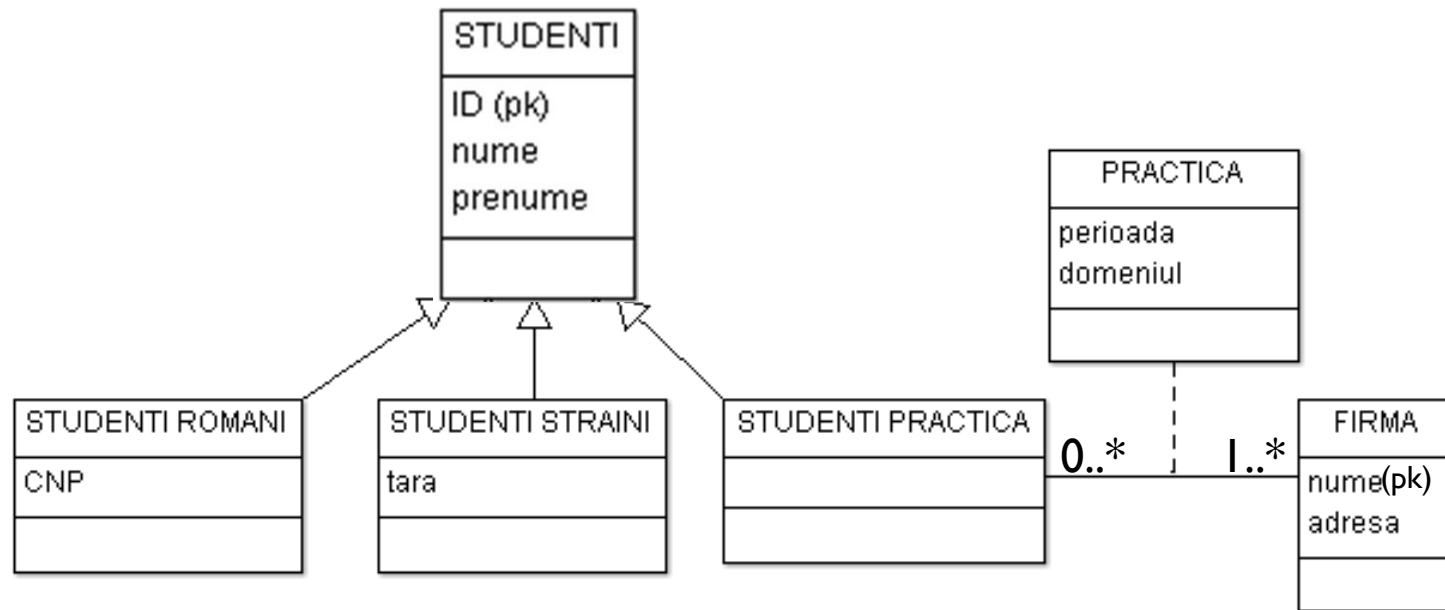


Subclasă (1)



(Specializare completă, disjunctă)

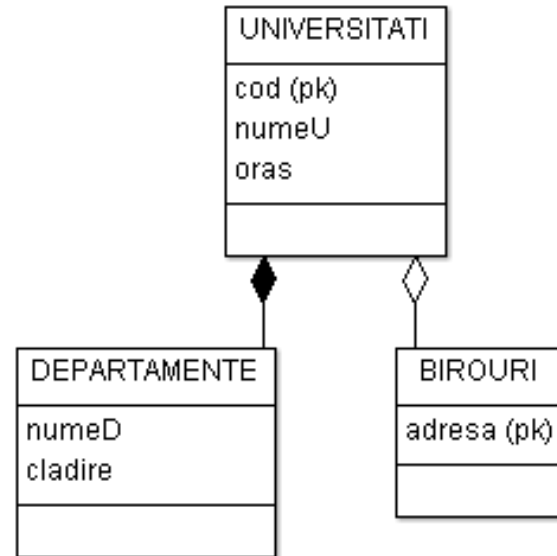
Subclasă (2)



(Specializare completă, cu suprapunere)

Compoziție și agregare

- ▶ Obiecte dintr-o clasă aparțin obiectelor din altă clasă
- ▶ Tipuri speciale de asociere

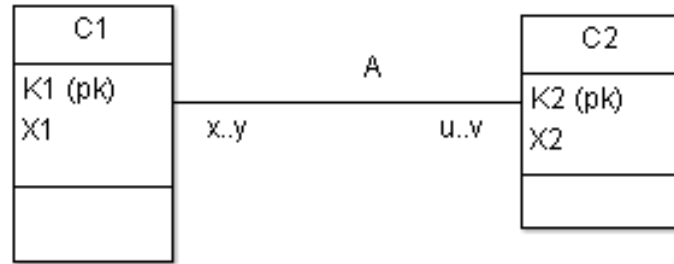


- ▶ **Compoziția:** **toate** obiectele unei clase *părți* aparțin obiectelor dintr-o clasă *compusă*; clasei *părți* îi corespunde de obicei o entitate slabă (multiplicitate 1..1; fără cheie primară);
- ▶ **Agregarea:** **unele** obiecte din clasa *părți* aparțin obiectelor din clasa *compusă* (multiplicitate 0..1)

Mapare E/A, UML -> schema BD relațională

E/A	UML	Schema relațională
Mulțime-entitate cu attribute	Clasă	Relație cu cheie primară
Mulțime-asociere fără attribute proprii	Asociere	Relație cu chei străine
Mulțime-asociere cu attribute proprii	Clasă de asociere	Relație cu chei străine și alte attribute
Specializare	Subclasă	Relație cu cheie primară (cea a superclasei) și attribute particulare/specializate
	Compoziție și agregare	Relație cu cheie străină și attribute particulare

Mulțimi-entitate/clase și asocieri



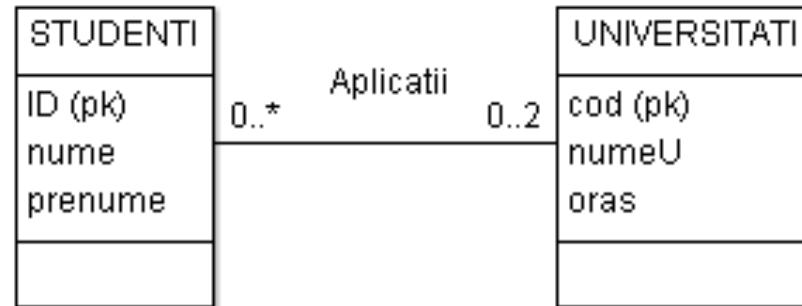
{C1(K1, X1), C2(K2, X2), A(K1, K2)}

► Cheia primară pentru asociere depinde de multiplicitate

x..y	u..v	Cheia primară pt A	Observații
0..1 1..1	*	K2	Nu e necesară relația A {C1(<u>K1</u> , X1), C2(<u>K2</u> , X2, K1)}
*	0..1 1..1	K1	Nu e necesară relația A {C1(<u>K1</u> , X1, K2), C2(<u>K2</u> , X2)}
*	*	de obicei (K1, K2)	

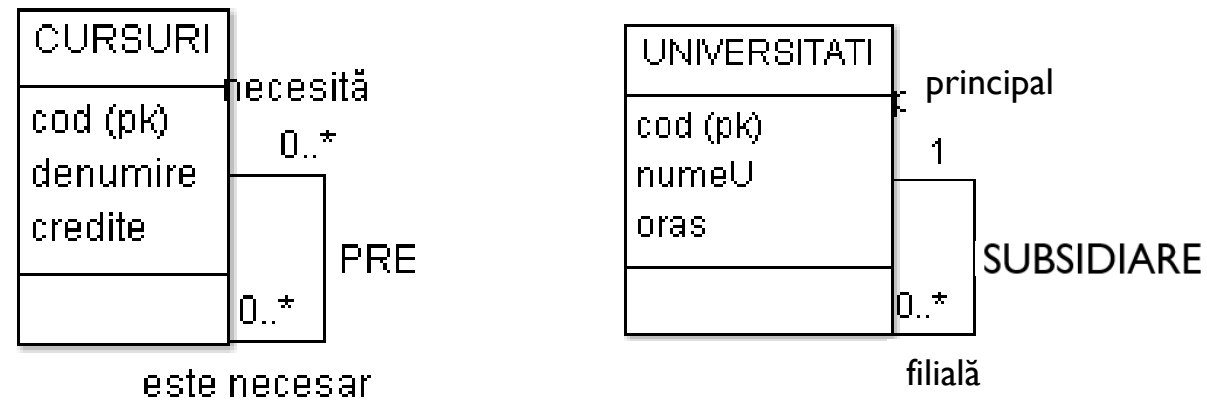
???

► Fie diagrama



► Mai este posibilă renunțarea la relația corespunzătoare asocierii?

Asocieri recursive



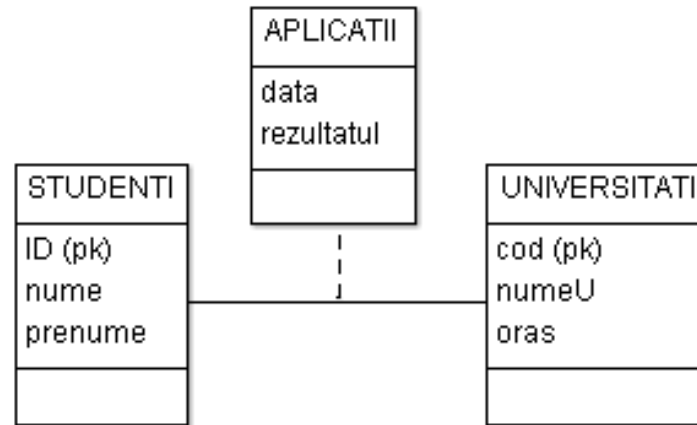
{CURSURI (cod, denumire, credite)
PRE (cod1, cod2)}

{UNIVERSITATI (cod, numeU, oras)
SUBSIDIARE (codFiliala, codParinte)}

sau

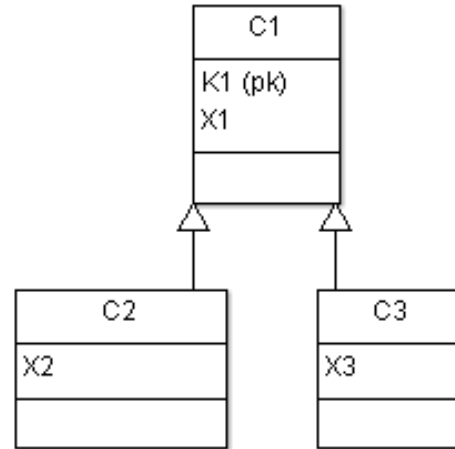
UNIVERSITATI (codFiliala, codPrincipal, numeU, oras)

Clase de asociere



{STUDENTI (ID, nume, prenume)
UNIVERSITATI (cod, numeU, oras)
APLICATII (ID, cod, data, rezultatul)}

Specializare / Subclase



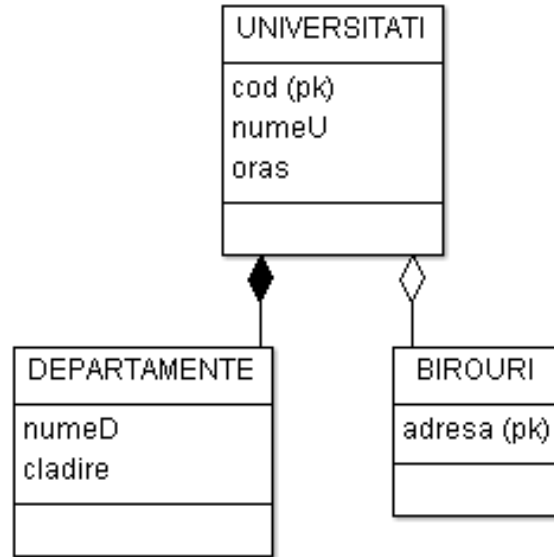
► Posibilități

- Relații subclasă ce conțin cheia superclasei și attributele specializate
 $C1(\underline{K1}, X1), C2(\underline{K1}, X2), C3(\underline{K1}, X3)$
- Relații subclasă ce conțin attributele superclasei (inclusiv atributul cheie) și attributele specializate; superclasa conține doar tuple nespecializate
 $C1(\underline{K1}, X1), C2(\underline{K2}, X1, X2), C3(\underline{K2}, X1, X3)$
- O singură relație ce conține attributele din superclasă și subclasă
 $C(\underline{K1}, X1, X2, X3)$

???

-
- Fie superclasa S cu un număr de subclase. Considerăm că relația de specializare este incompletă și cu suprapunere. Dacă n_1 , n_2 și n_3 reprezintă numărul total de tuple necesare fiecărei scheme de decodificare din cele 3 date anterior (în ordinea dată), care este relația dintre cele 3 valori?
- $n_1 < n_2 < n_3$
 - $n_1 \leq n_2 \leq n_3$
 - $n_3 < n_2 < n_1$
 - $n_3 \leq n_2 \leq n_1$

Compoziție și agregare



{ UNIVERSITATI(cod, numeU, oras)
DEPARTAMENTE(codU, numeD, cladire)
BIROURI (codU, adresa)}

← NU acceptă NULL
← acceptă NULL

Modelare EA/UML

Sumar

▶ PROS

- ▶ Tehnică populară de modelare conceptuală
- ▶ Construcții expresive, descriu punctul de vedere personal asupra aplicației
- ▶ Permite exprimarea unor tipuri de constrângeri (chei primare, străine, multiplicitate, exclusivitate...)

▶ CONS

- ▶ Tehnică subiectivă (entitate sau atribut, entitate sau asociere, subclasare sau nu, compoziție sau nu)
- ▶ Nu permite modelarea tuturor dependentelor
- ▶ Necesită utilizarea ulterioară a normalizării

Bibliografie

- ▶ Capitolele 11 și 12 in Thomas Connolly, Carolyn Begg: *Database Systems: A Practical Approach to Design, Implementation and Management*, (5th edition) Addison Wesley, 2009

- ▶ Instrumente:
 - ▶ <https://creately.com> (diagrame EA, diagrame UML de clasă)
 - ▶ <http://diagramo.com/> (diagrame EA)
 - ▶ <https://argouml-tigris-org.github.io/tigris/argouml/>