Capitolul 2

MODELAREA DATELOR

OBIECTIV

- Proiectarea corectă a **structurii** unei baze de date relaţionale este o premisă fundamentală în:
- scrierea programelor de aplicaţie
- functionarea lor fara **anomalile** care pot apare in cazul unei structuri defectuoase.

EXEMPLU: Proiectare greşită

IDP	NUMEP	QTY	IDF	NUMEF	ADRESAF
101	Imprimantă laser	30	20	XY SRL	Str. X, Bucureşti
105	Calculator PC	20	23	Z SRL	Bd. Z, Bucureşti
124	Copiator	10	20	XY SRL	Str. X, Bucureşti

ANOMALII (1)

- **Redundanta:** Redundanta reprezinta stocarea in mod nejustificata a unei aceleiasi informatii de mai multe ori in baza de date.
- Observam ca pentru fiecare produs este stocat numele si adresa furnizorului, desi ele sunt unic determinate de codul acestuia.

ANOMALII (2)

- **Anomalia de stergere:** La stergerea din relatie a ultimului produs al unui furnizor se pierd automat si datele despre acesta.

ANOMALII (3)

- **Anomalia de actualizare:** In cazul actualizarii unei informatii redundante, se poate intampla ca operatia sa modifice unele aparitii ale acesteia iar altele sa ramana cu vechea valoare.

ANOMALII (4)

- **Anomalia de inserare:** Nu putem insera date despre un furnizor (numele si adresa sa) decat daca exista in stoc un produs furnizat de acesta.

SOLUŢIE

- Acest capitol prezinta un model de date folosit in proiectarea conceptuala de nivel inalt numit **modelul entitate asociere** (EA) in varianta clasica (cu unele extensii).
- Intr-un alt capitol vor fi prezentate regulile de transformare din modelul entitate-asociere in modelul relational.

MODELUL ENTITATE-ASOCIERE

- Modelulul entitate-asociere (EA) a dus la gasirea unor cai algoritmizabile de proiectare optima a bazelor de date.
- Modelul entitate-asociere este in acest moment cel mai popular model de comunicare a structurii bazelor de date datorita intuitivitatii şi simplitatii elementelor sale.
- Imbunatatiri sale ulterioare au dus la crearea de variante ale modelului, doua dintre acestea fiind descrise in acest capitol.

EXTENSII ALE MODELULUI EA

- Modelarea cerintelor de secretizare a datelor,
- Documentarea programelor de aplicatie şi usurarea comunicarii proiectant - utilizator,
- Proiectarea bazelor de date complexe pe portiuni (integrarea vederilor).

ETAPELE PROIECTARII (1)

Pentru baza de date:

1. Analiza de sistem → Cerinţe privind baza de date
2. Proiectarea conceptuală → Diagrama EA
3. Transformare în model relaţional → Schema candidat a bazei de date

ETAPELE PROIECTARII (2)

4. Normalizare → Schema conceptuala a bazei de date
5. Implementare specifică SGBD-ului folosit → Schema bazei de date a aplicaţiei

ETAPELE PROIECTARII (3)

Pentru programele de aplicatie:

1. Analiza de sistem → Cerinţe funcţionale ale aplicaţiei
2. Proiectarea funcţională → Diagrame de specificare funcţională

Urmeaza apoi realizarea, testarea, implementarea si instruirea personalului → Programele aplicaţiei

ETAPELE PROIECTARII (1)

Pentru baza de date:

1. Analiza de sistem → Cerinţe privind baza de date
2. Proiectarea conceptuală → Diagrama EA
3. Transformare în model relaţional → Schema candidat a bazei de date

ANALIZA DE SISTEM

- Se realizeaza analiza segmentului din lumea reala care va fi gestionat de aplicatia respectiva.
- Rezulta o specificatie neformalizata a cerintelor constand din doua componente:
  - ***Cerinte privind continutul bazei de date:*** categoriile de date care vor fi stocate şi interdependentele dintre acestea.
  - ***Cerinte privind prelucrarile efectuate de aplicatie:*** prelucrarile efectuate asupra datelor, arborele de meniuri al aplicatiei, machetele formatelor de introducere şi prezentare a datelor şi ale rapoartelor tiparite de aceasta.

ANALIZA DE SISTEM: ACTIVITATI(1)

- Analiza *activitatii* desfasurate la momentul respectiv de beneficiarul aplicatiei sau de o multime reprezentativa de beneficiari
- Analiza continutului de date şi a functionalitatii *aplicatiilor software* - daca exista - care vor fi inlocuite de noua aplicatie (meniuri, machete ecran, machete rapoarte)
- Analiza *formularelor tipizate* şi a altor documente utilizate de beneficiar pentru realizarea activitatii respective.

ANALIZA DE SISTEM: ACTIVITATI(2)

- Identificarea *interdependentelor* dintre datele stocate in baza de date şi a *restrictiilor* privind valorile lor
- Identificarea prelucrarilor care se declanseaza *automat* atat in cazul modificarii bazei de date cat şi la momente prestabilite de timp (de exemplu sfarsit de luna, de an, etc.)
- Identificarea operatiilor care sunt necesare beneficiarului in activitatea curenta dar care in acest moment *nu sunt realizate* prin intermediul aplicatiilor software folosite precum si a operatiilor care *pot fi incluse* in mod natural in noua aplicatie.

ANALIZA DE SISTEM: ACTIVITATI(3)

- Identificarea *bazelor de date existente* care pot fi folosite de noua aplicatie - direct sau printr-un import initial de date - *evitandu-se reintroducerea manuala* a acestora.
- Identificarea modalitatilor de *transfer de date* intre noua aplicatie şi alte aplicatii care ruleaza deja la beneficiar şi care vor fi folosite şi in viitor de catre acesta.
- Identificarea necesitatilor privind datele şi *prelucrarile care pot fi in viitor* necesare beneficiarului, deci a *posibilelor dezvoltari* in timp ale aplicatiei.

ETAPELE PROIECTARII (1)

Pentru baza de date:

1. Analiza de sistem → Cerinţe privind baza de date
2. Proiectarea conceptuală → Diagrama EA
3. Transformare în model relaţional → Schema candidat a bazei de date

PROIECTAREA CONCEPTUALA

- In aceasta etapa, pornind de la rezultatele analizei de sistem, se realizeaza modelarea cerintelor privind datele folosind un model de nivel inalt.
- Cel mai popular model folosit pentru aceasta este modelul entitate-asociere (EA).
- Actualmente exista pe piata numeroase instrumente CASE care folosesc diverse variante ale modelului.

AVANTAJELE EA (1)

- Nu este legat direct de nici unul dintre modelele folosite de sistemele de gestiune a bazelor de date (relational sau orientat obiect) dar exista algoritmi de transformare din model EA in celelalte modele de date.
- Este intuitiv, rezultatul modelarii fiind o diagrama care defineste atat datele stocate in baza de date cat şi interdependentele dintre acestea.

AVANTAJELE EA (2)

- Poate fi usor de inteles de nespecialisti si faciliteaza punerea de acord cu beneficiarul asupra structurii bazei de date a aplicatiei, evitandu-se in acest fel o proiectare neconforma cu realitatea sau cu cerintele
- Proiectarea se poate face pe portiuni, diagramele partiale rezultate putand fi apoi integrate pe baza unor algoritmi şi metode bine puse la punct.

ETAPELE PROIECTARII (1)

Pentru baza de date:

1. Analiza de sistem → Cerinţe privind baza de date
2. Proiectarea conceptuală → Diagrama EA
3. Transformare în model relaţional → Schema candidat a bazei de date

TRANSFORMAREA

- In aceasta etapa entitatile şi asocierile care formeaza diagrama EA se transforma pe baza unor reguli clare in structura relationala a bazei de date.
- Rezulta schema preliminara a acesteia formata din:
  - **tabele** (relatii in terminologia relationala),
  - **coloanele** acestora (atribute ale relatiilor) şi
  - **constrangerile de integritate** care pot fi deduse automat din diagrama incluzand unele interdependente intre date numite şi "dependenţe functionale".
- In cazul variantei specifice uneltelor CASE transformarea se face automat de catre acestea.

ETAPELE PROIECTARII (2)

4. Normalizare → Schema conceptuala a bazei de date
5. Implementare specifică SGBD-ului folosit → Schema bazei de date a aplicaţiei

NORMALIZAREA

- Exista o serie de reguli care descriu ce inseamna o structura corecta. Ele definesc asa numitele ***forme normale***.
- Pe baza structurii bazei de date şi a dependentelor rezultate atat din transformare şi a altor dependente identificate de proiectant in analiza de sistem se poate face o operatie numita ***normalizare:*** se modifica structura bazei de date astfel incat toate tabelele din aceasta sa fie in forma normala dorita.

ETAPELE PROIECTARII (2)

4. Normalizare → Schema conceptuala a bazei de date
5. Implementare specifică SGBD-ului folosit → Schema bazei de date a aplicaţiei

IMPLEMENTARE CU UN SGBD

- In aceasta etapa se realizeaza crearea structurii bazei de date obtinuta in etapa precedenta pe baza facilitatilor oferite de sistemul de gestiune a bazelor de date ales.
- Rezultatul ei este programul de creare scris in limbajul de definitie a datelor acceptat de SGBD-ul utilizat.

EXEMPLU

- Schema conceptala:

Student(<u>CodStud:Numar</u>, Nume:Şir,
DataNasterii:Dată)

- Program de creare (SQL-Oracle):

```
Create table Student(
    CodStud   Number(5) Primary Key,
    Nume      Varchar2(40) Not Null,
    DataNasterii    Date);
```

IMPLEMENTARE CU UN SGBD - continuare

- Programul contine atat definirea tabelelor cat şi definirea celorlalte obiecte ale bazei de date (de exemplu constrangerile de integritate intra-tabela şi inter-tabele): NOT NULL, PRIMARY KEY.
- De asemenea aici se fac şi toate operatiile privind proiectarea la nivel fizic a bazei de date.
- In cazul folosirii de unor unelte CASE programul de creare poate fi generat şi executat automat.

MODELUL ENTITATE ASOCIERE

Modelul entitate-asociere clasic

- Acest model a fost introdus de P. P. Chen in 1976 ([Ch 76]).
- O abordare de tip grafic a proiectarii bazelor de date
- A fost adoptat de comunitatea stiintifica precum si de producatorii de software in domeniu datorita simplitatii si expresivitatii sale.

ELEMENTELE MODELULUI

Modelul entitate-asociere permite reprezentarea informatiilor despre structura bazelor de date folosind trei elemente de constructie:

- Entitati
- Atribute
- Asocieri intre entitati

ENTITĂŢI (1)

- **Entitatile** modeleaza clase de obiecte concrete sau abstracte despre care se colecteaza informatii, au existenta independenta şi pot fi identificate in mod unic.
- Exemple de entitati:
  - Studenti,
  - Orase,
  - Angajati, etc.
- Ele definesc de obicei persoane, amplasamente, obiecte sau evenimente cu importanta informationala.

ENTITĂŢI (2)

- Membrii unei clase care formeaza o astfel de entitate poarta numele de instante ale acelei entitati.
- De remarcat ca in literatura de specialitate se defineste intii conceptul de multime de entitati (sau tip de entitati) pentru ca apoi sa adopte pentru usurinta exprimarii prescurtarea de entitate pentru acest concept.
- Deci entitatea este un obiect generic care reprezinta multimea tuturor instantelor sale.

CATEGORII DE ENTITATI

- Entitati independente (sau tari, eng. strong) sunt cele care au existenta independenta de alte entitati,
- Entitati dependente (sau slabe, eng. weak) sunt formate din instante care isi justifica incadrarea in clasa respectiva doar atita timp cit intr-o alta entitate (tata) exista o anumita instanta de care sunt dependente.

Exemplu

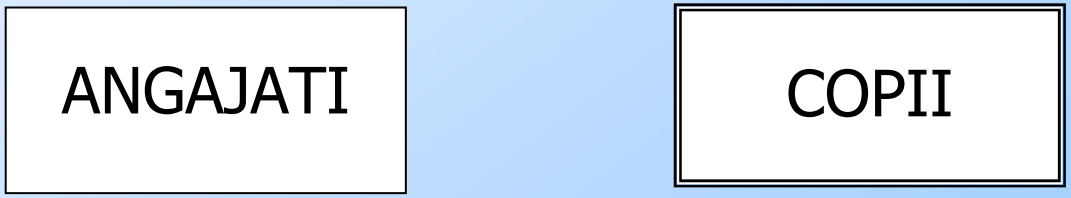
Intr-o baza de date de personal avem entitatile:

- ANGAJATI si
- COPII (ultima contine copii angajatilor companiei)

- Fiecare instanta a entitatii COPII depinde de o instanta a entitatii ANGAJATI
- Rezulta:
  - ANGAJATI: entitate independenta,
  - COPII: entitate dependenta

REPREZENTARE GRAFICA

- Entitatile se reprezinta grafic prin dreptunghiuri in care e inscris numele entitatii.
- Exemple:



ATRIBUTE

- **Atributele** modeleaza proprietati atomice distincte ale entitatilor.
- Exemplu: entitatea STUDENTI poate avea atributele
  - Matricola,
  - Nume,
  - Prenume,
  - Varsta,
  - Anul,
  - Grupa,
  - Domiciliu – e atomic sau nu?

ATRIBUTE (2)

- In procesul de modelare vor fi luate in considerare doar acele proprietati ale entitatilor care sunt semnificative pentru aplicatia respectiva.
- Exemplu: la entitatea STUDENTI nu vom avea atribute ca:
  - Talia sau
  - Culoarea_parului

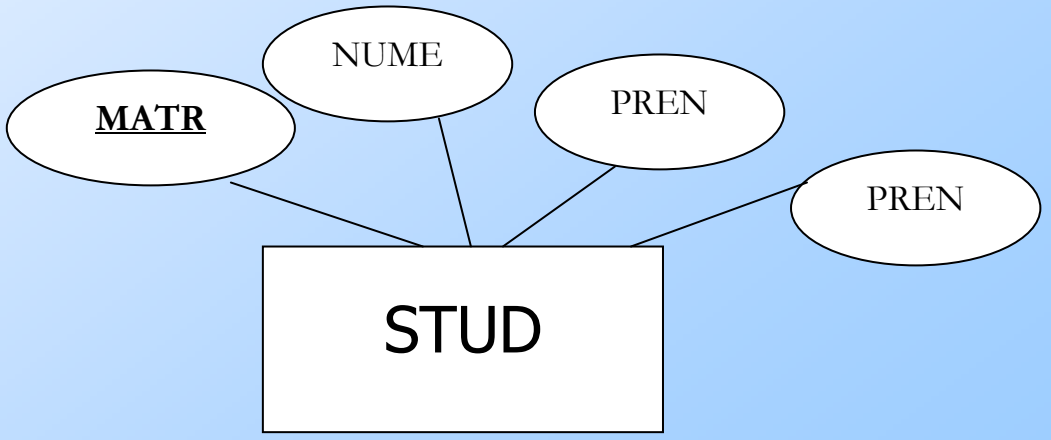
  acestea nefiind necesare pentru baza de date a universitatii (dar pot exista intr-o baza de date privind personalul militar).

CLASIFICARE ATRIBUTE

- atributele de identificare (formand impreuna identificatorul entitatii) reprezinta acea multime de atribute care permit distinctia intre instantele aceleiasi entitati .
- atributele de descriere (sau descriptori) sunt folositi pentru memorarea caracteristicilor suplimentare ale instantelor.
- Exemplu: Pentru entitatea STUDENTI
  - Matricola este atribut de identificare
  - celelalte atribute sunt descriptori

REPREZENTARE GRAFICA

- Atributele se reprezinta prin ovale sau cercuri in care e inscris numele atributului. Ele sunt conectate la entitatea pe care o caracterizeaza.
- Exemplu:

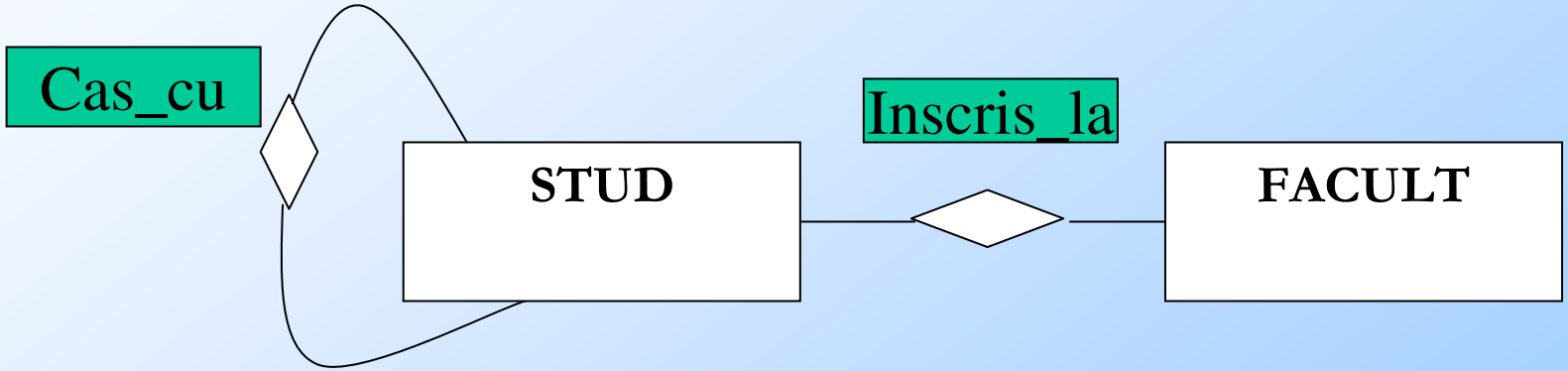


ASOCIERI

- **Asocierile** modeleaza interdependentele (sau legaturile) dintre clasele de obiecte reprezentate prin entitati.
  - Exemplu: intre STUDENTI şi FACULTATI se poate figura o asociere INSCRIS_LA care descrie impartirea studentilor pe facultati.
- In crearea diagramei nu vor fi luate in consideratie decit interdependentele care sunt necesare aplicatiei respective (pot exista şi alte asocieri care nu sunt semnificative pentru aplicatia proiectata)

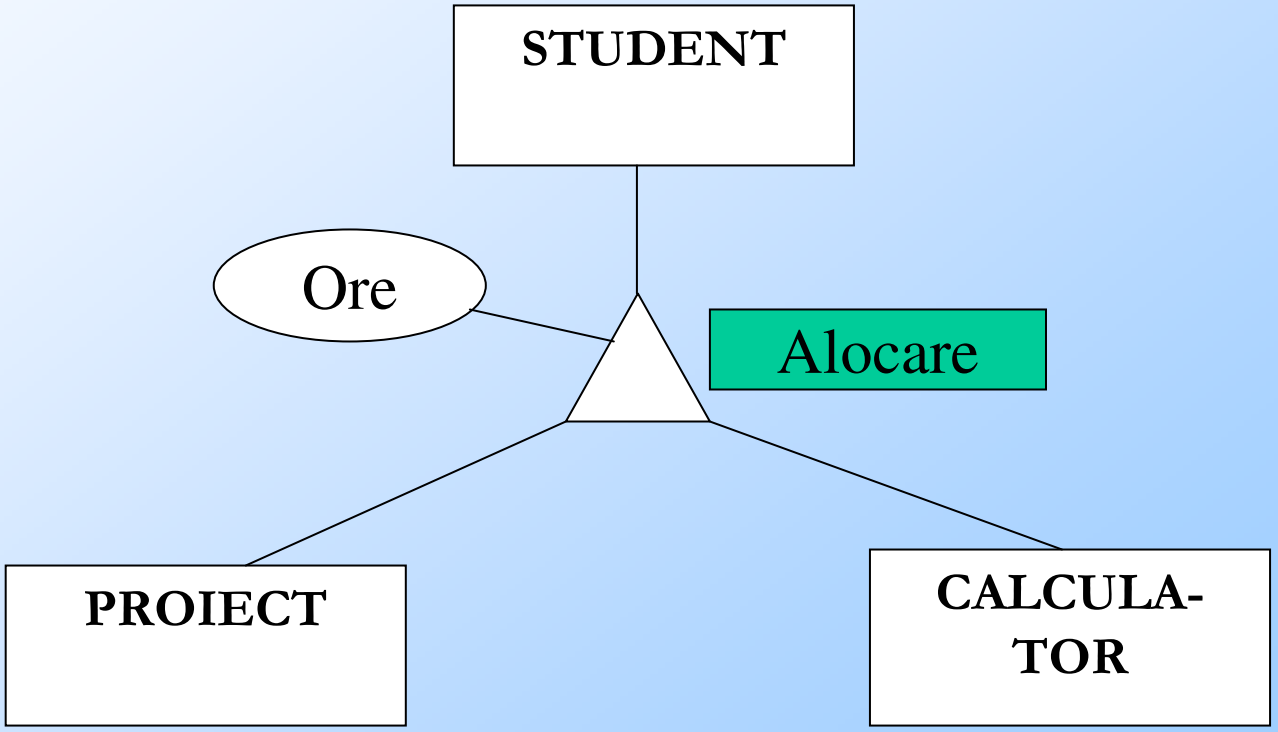
REPREZENTARE GRAFICA (1)

- Cand sunt asociate 1-2 entitati: romb



REPREZENTARE GRAFICA (2)

- Cand sunt asociate >3 entitati: poligon



EXTENSII ALE MODELULUI

- Modelul clasic are unele lipsuri in ceea ce priveste posibilitatea modelarii caracteristicilor asociate unor subclase modelate prin entitati.
- Pentru aceasta, la modelul original au fost adaugate doua noi concepte:
  - **ierarhia de generalizare** şi
  - **ierarhia de incluziune**.
- Prima defineste partitionarea instantelor unei entitati in **n** subclase diferite iar a doua permite clasarea unora dintre instantele unei entitati in **m** subclase care nu reprezinta o partitie in sens matematic.

IERARHIA DE INCLUZIUNE(1)

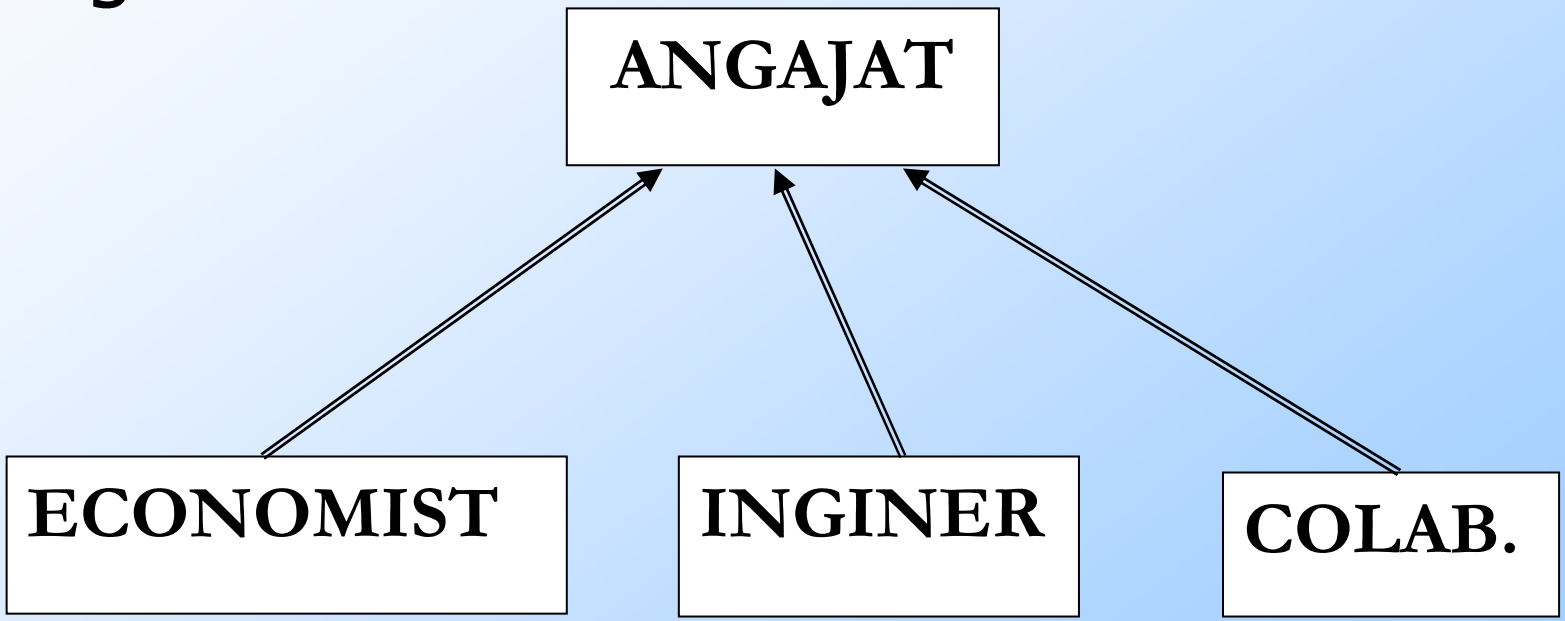
- ***Definitie :*** O entitate E1 este inclusa in entitatea E daca fiecare instanta a lui E1 este de asemenea o instanta a lui E.
- Exemplu: in cadrul entitatii ANGAJATI avem subclase modelate prin entitatile:
  - INGINERI,
  - ECONOMISTI si
  - COLABORATORI.

IERARHIA DE INCLUZIUNE(2)

- In cazul ierarhiei de incluziune entitatile fiu pot sa nu fie disjuncte doua cite doua: pentru exemplul dat, exista angajati ingineri si care sunt incadrati cu contract de colaborare.
- De asemenea reuniunea lor poate sa nu acopera in intregime entitatea tata: exista angajati care nu sunt nici ingineri, nici economisti si nici colaboratori.

REPREZENTARE GRAFICA

Sageti duble



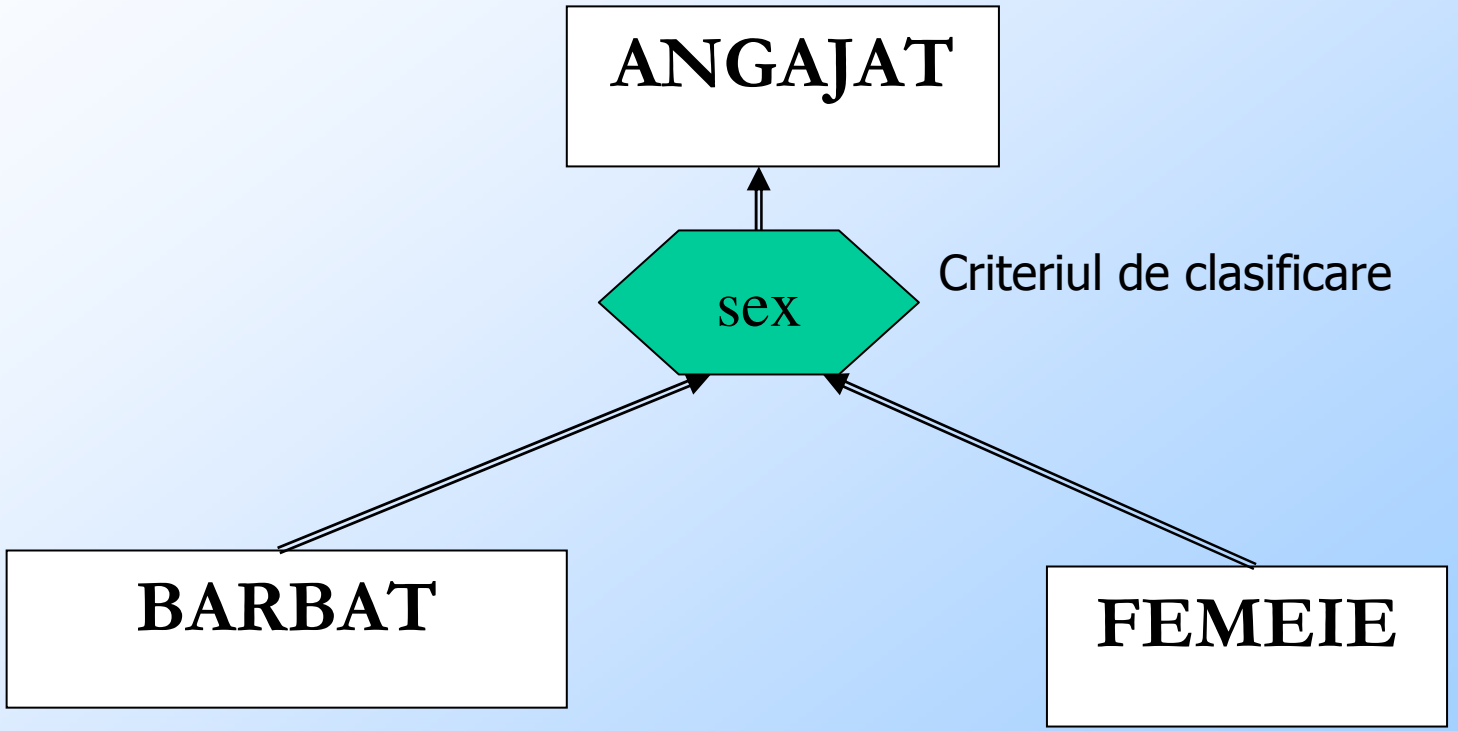
IERARHIA DE GENERALIZARE(1)

- ***Definitie (ierarhia de generalizare):*** O entitate E este generalizarea entitatilor E1, E2, ..., En daca orice instanta a lui E este de asemenea instanta in una şi numai una din entitatile E1, E2, ..., En.
- Un exemplu de generalizare este clasarea instantelor entitatii ANGAJATI in subclasele BARBATI şi FEMEI.
- Entitatile fiu reprezinta o clasificare a instantelor entitatii tata

IERARHIA DE GENERALIZARE(2)

- Caracteristica ierarhiei de generalizare este ca din punct de vedere matematic entitatile fiu reprezinta o partitie a entitatii tata:
  - a. $E1 \cup E2 \cup ... \cup En = E$ şi
  - b. $Ei \cap Ej = \varnothing$ pentru orice $i \neq j$ din intervalul 1..n

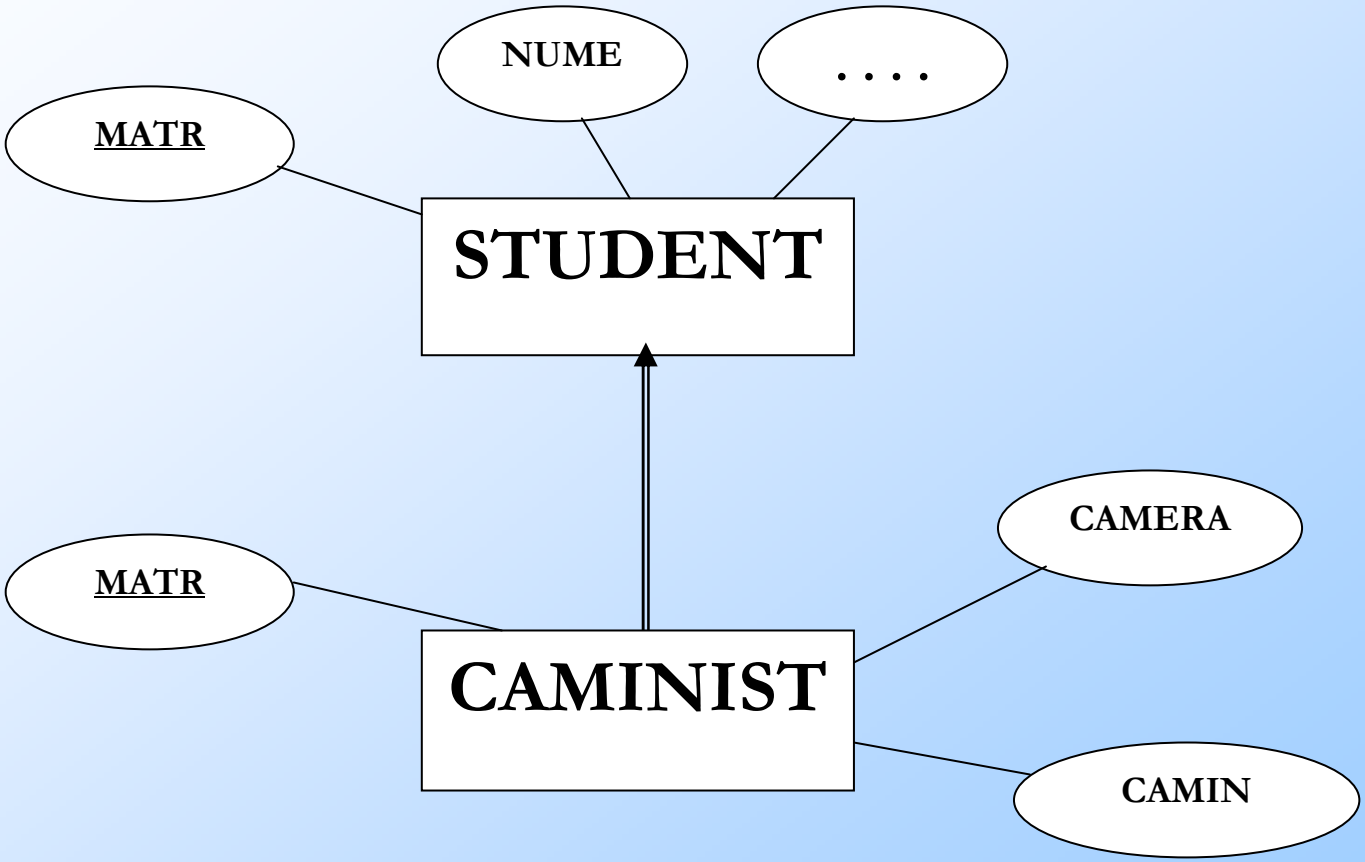
REPREZENTARE GRAFICA



CAND FOLOSIM IERARHII

- Ierarhiile de incluziune şi generalizare se folosesc doar in cazul in care pentru subclasele unor clase modelate prin entitati este nevoie de stocarea unor ***informatii suplimentare specifice***.
- *Exemplu*: Intr-o baze de date de studenti este nevoie de caminul si camera ocupata, dar doar pentru caministi.
- Acest fapt se poate modela printr-o **entitate suplimentare** CAMINIST aflata intr-o relatie de **incluziune** cu entitatea STUDENT. Ea va avea ca atribute de identificare pe cele ale tatalui iar ca atribute descriptive Caminul si Camera.

DIAGRAMA PT. EXEMPLU



CARACTERISTICI ELEMENTE

- Gradul unei asocieri
- Conectivitatea unei ramuri de asociere
- Obligativitatea unei ramuri de asociere
- Atribute asocieri
- Roluri pentru ramurile asocierilor

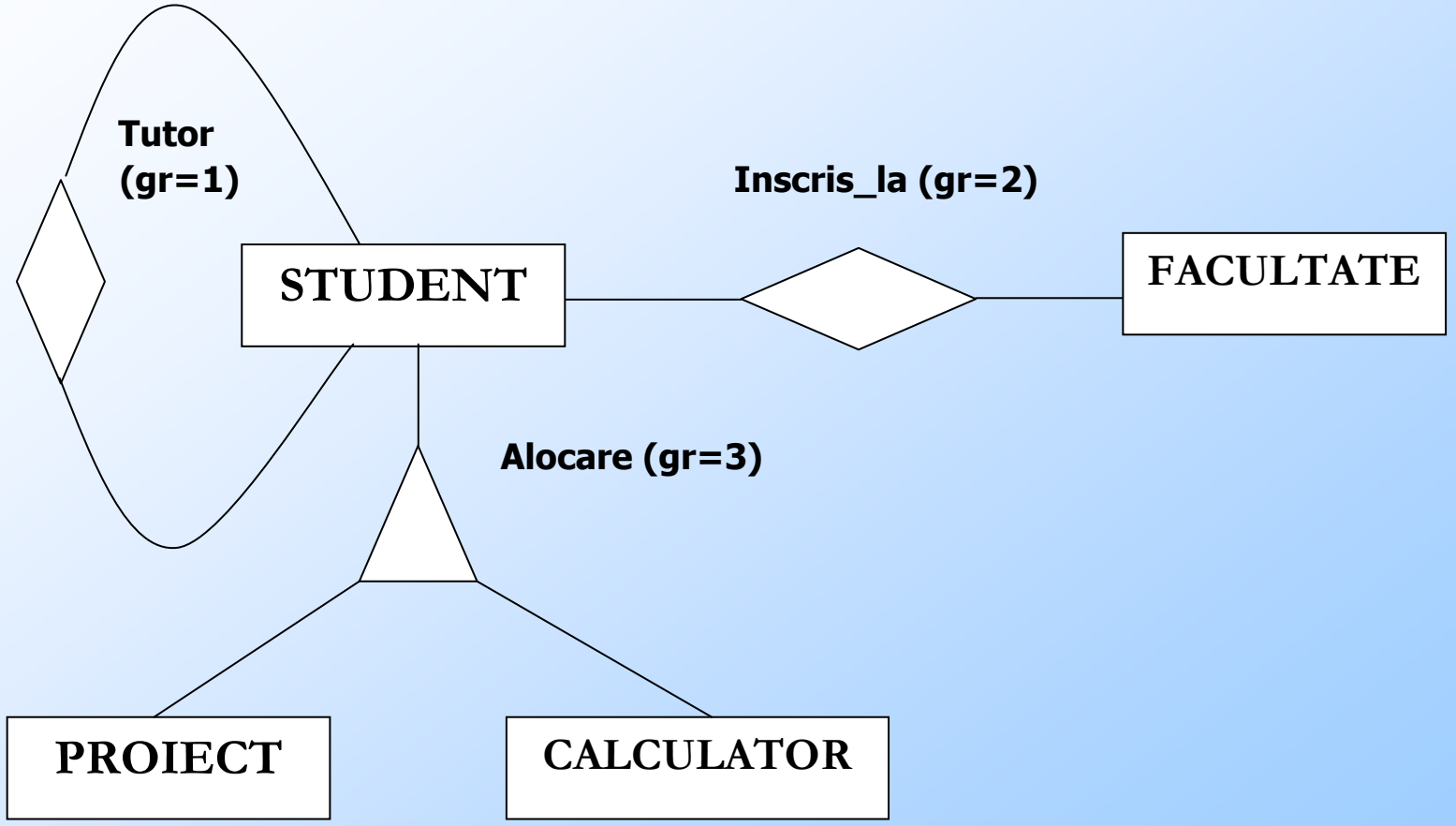
GRADUL ASOCIERII (1)

- Este o valoare numerica intreaga si este dat de numarul de entitati care participa la acea asociere.
- Poate avea deci valorile 1, 2, 3, 4, 5, ...
- Asocierile de grad 1, 2 şi 3 se mai numesc si asocieri ***unare***, ***binare*** si respectiv ***ternare***.

GRADUL ASOCIERII (2)

- Pentru exemplificare vom considera o baza de date continand informatii despre
  - studenti,
  - proiectele realizate de acestia,
  - calculatoarele pe care au alocate ore de lucru si
  - facultatile la care sunt inscrisi.
- De asemenea vom considera ca unii dintre studenti au un ***tutor*** care ii indruma, acesta fiind un student dintr-un an mai mare.

GRADUL ASOCIERII (3)

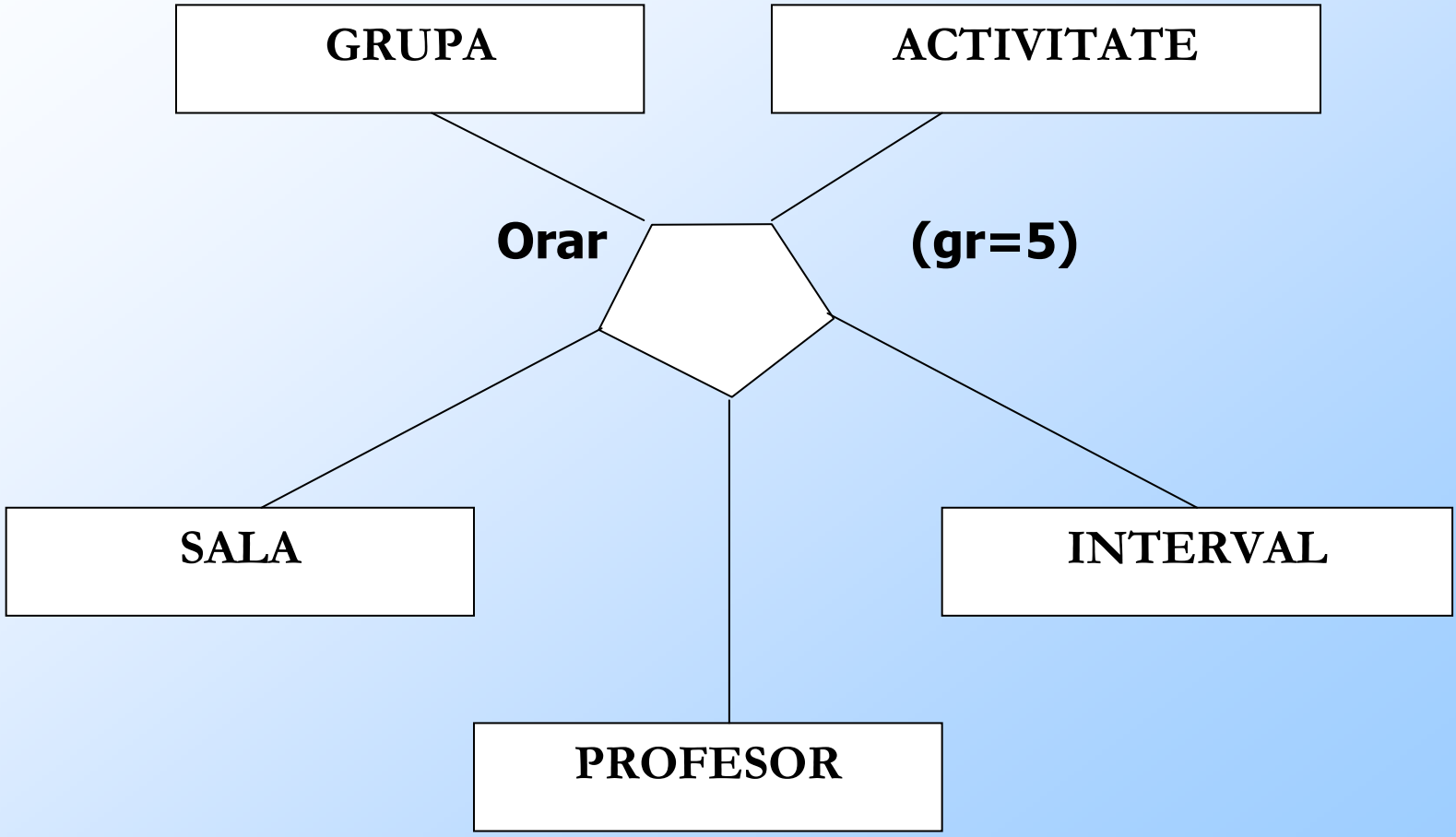


Obs: Nu sunt figurate atributele entitatilor

GRADUL ASOCIERII (4)

- Un exemplu de asociere de grad mai mare ca trei este orarul unui an de studiu al unei facultati.
- Acesta este o asociere intre urmatoarele entitati:
  - GRUPE. Fiecare grupa are un cod unic.
  - SALI. Salile sunt etichetate printr-un indicativ alfanumeric.
  - INTERVALE ORARE. Un interval orar este un triplet (Zi a saptamanii, De la ora, La ora)
  - ACTIVITATE. Este o activitate prezenta in orar (curs, laborator, seminar, proiect, la o disciplina).
  - PROFESOR. Este cadrul didactic titular pentru o activitate

GRADUL ASOCIERII (5)



CARACTERISTICI ELEMENTE

- Gradul unei asocieri
- Conectivitatea unei ramuri de asociere
- Obligativitatea unei ramuri de asociere
- Atribute asocieri
- Roluri pentru ramurile asocierilor

CONECTIVITATE (1)

- Este specifica fiecarei ramuri a unei asocieri şi poate avea una din urmatoarele doua valori: **unu** sau **multi** **(eng.: one / many)**.
- Determinarea ei pentru ramura spre o entitate E se face astfel: fixand arbitrar cite o instanta pentru celelalte entitati care participa la asociere se pune intrebarea: cate instante ale lui E pot fi conectate cu acestea?
- Daca poate fi cel mult una, conectivitatea ramurii este **unu**, altfel conectivitatea este **multi**.

CONECTIVITATE (2)

Pentru exemplul STUDENT, FACULTATE, PROIECT, CALCULATOR:

- Asocierea TUTOR este **unu-unu** sau **multi-uni** dupa cum un student poate fi tutor pentru un singur alt student sau pentru mai multi studenti de an inferior.
- Asocierea INSCRIS_LA este **multi-unu** (multi spre STUDENT) sau **multi-multi** dupa cum un student poate fi inscris la una sau mai multe facultati.

CONECTIVITATE (3)

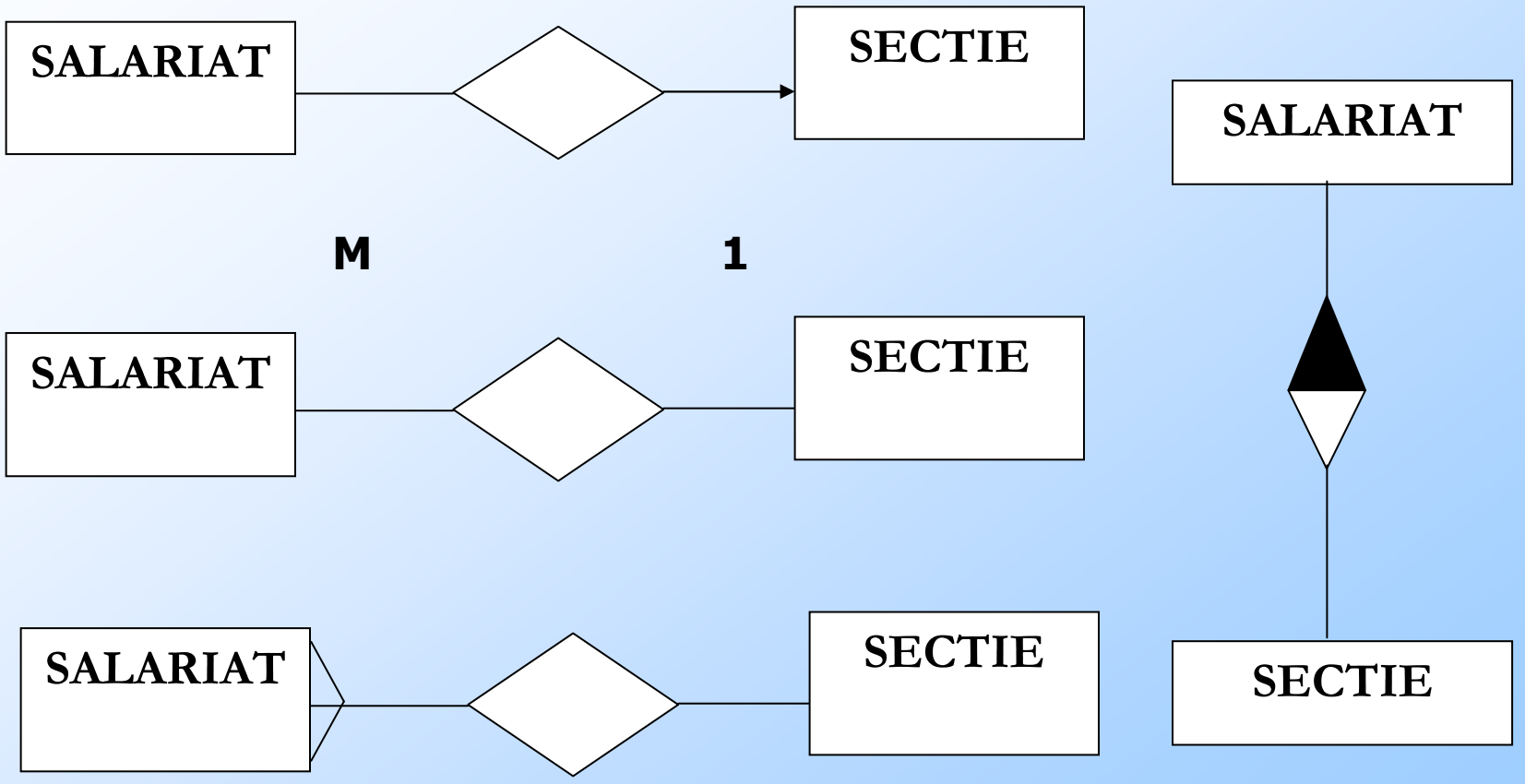
Asocierea ternara ALOCARE:

- Spre STUDENT: fiind dat un proiect si un calculator, citi studenti au ore alocate pe acel calculator pentru respectivul proiect? Presupunand ca mai multi studenti lucreaza pentru acelasi proiect pe acelasi calculator ramura va fi ***multi***.
- Spre PROIECT: fiind dat un student şi un calculator, la cite proiecte are acesta alocate ore pe acel calculator? Presupunand ca pentru fiecare proiect exista un calculator dedicat, ramura va fi ***unu***.
- Spre CALCULATOR: fiind dat un student şi un proiect, pe cate calculatoare are alocate acesta ore pentru realizarea proiectului? Presupunand ca la un proiect se lucreaza pe un singur calculator, ramura va fi ***unu***.

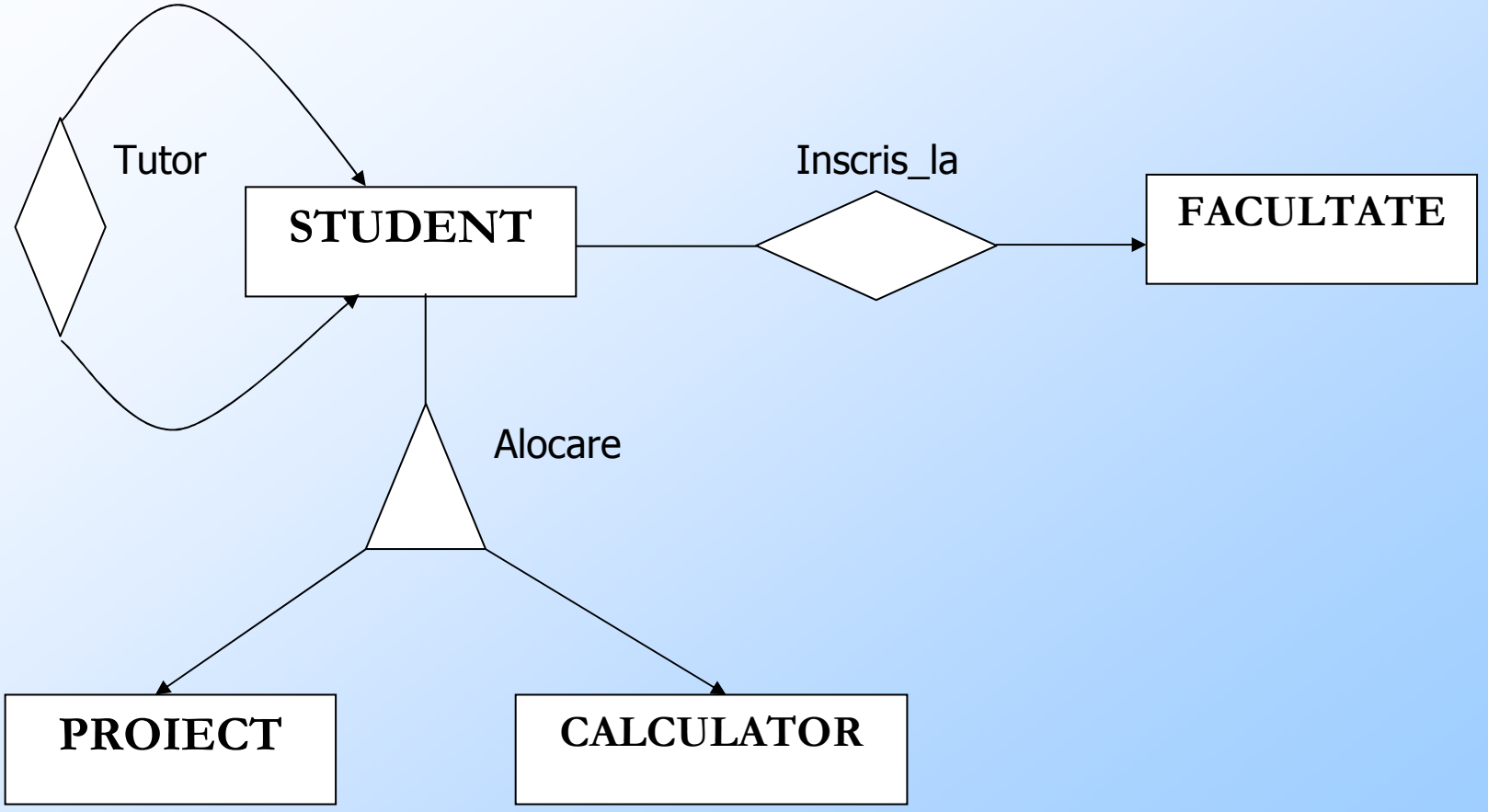
CONECTIVITATE (4)

- Deci asocierea ALOCARE este multi-unu-unu.
- Observam ca raspunsul la fiecare din cele trei intrebari se da in functie de realitatea modelata.
- Aceeasi asociere poate avea conectivitati diferite in cazuri diferite: daca exista chiar si un singur proiect la care un student are ore alocate pe mai mult de un calculator, ramura spre CALCULATOR va fi ***multi*** iar asocierea va fi multi-unu-multi.

REPREZENTARE GRAFICA



EXEMPLU


Tutor
STUDENT
Inscris_la
FACULTATE
Alocare
PROIECT
CALCULATOR

CARACTERISTICI ELEMENTE

- Gradul unei asocieri
- Conectivitatea unei ramuri de asociere
- Obligativitatea unei ramuri de asociere
- Atribute asocieri
- Roluri pentru ramurile asocierilor

Obligativitate (1)

- Ca şi conectivitatea, aceasta se determina pentru fiecare ramura şi poate avea doar una din urmatoarele valori: **obligatorie** sau **optionala**.
- Determinarea ei pentru ramura spre o entitate E se face astfel: fixand arbitrar cite o instanta pentru celelalte entitati care participa la asociere se pune intrebarea: este obligatoriu sa existe o instanta a lui E asociata cu acestea?
- Daca raspunsul este 'Da' ramura este **obligatorie** altfel este **optionala**.

Obligativitate (2)

- In exemplul anterior ramurile asocierilor TUTOR si ALOCARE sunt optionale iar cele ale asocierii INSCRIS_LA sunt obligatorii deoarece:
- Pentru asocierea INSCRIS_LA: nu exista studenti care nu sunt inscrisi la nici o facultate si nici facultati fara studenti inscrisi.
- Pentru asocierea TUTOR: exista studenti care
  - nici nu au un tutor
  - nici nu sunt tutori pentru alti studenti

Obligativitate (3)

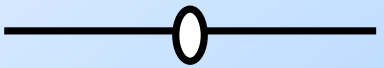
Pentru asocierea ALOCARE:

- - *Un student la un proiect* poate sa nu aiba alocate ore pe nici calculator (de exemplu in cazul unui proiect la o materie umanista)
- - *Un student si un calculator* respectiv *un calculator si un proiect* pot sa nu fie asociati prin alocare de ore de lucru (de exemplu pentru calculatoarele din birourile cadrelor didactice)

Obligativitate (4)

- Obligativitatea se modeleaza pentru definirea unui criteriu de integritate specificand posibilitatea de aparitie a ***valorilor nule***.
- La transformarea diagramei EA in model relational atributele tabelelor care modeleaza informatia reprezentata de asocieri pot avea sau nu valori nule dupa cum ramurile acestora sunt optionale sau obligatorii.

REPREZENTARE GRAFICA

	
Ramura obligatorie	Ramura optionala

CARACTERISTICI ELEMENTE

- Gradul unei asocieri
- Conectivitatea unei ramuri de asociere
- Obligativitatea unei ramuri de asociere
- Atribute asocieri
- Roluri pentru ramurile asocierilor

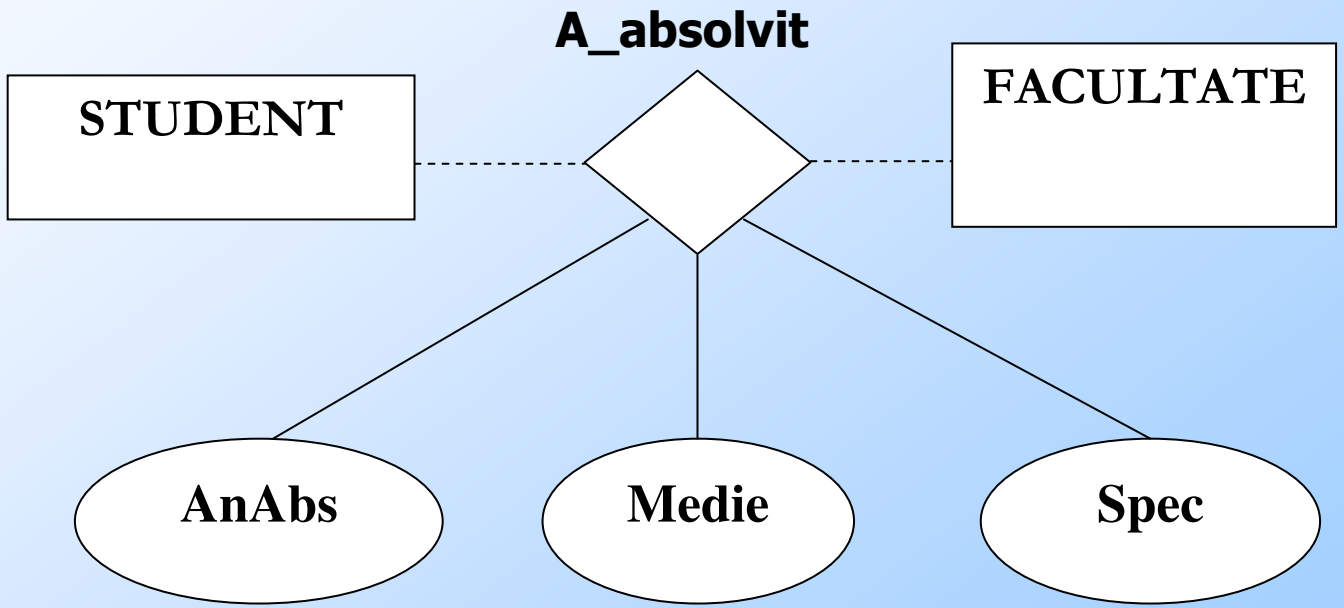
ATRIBUTE ASOCIERI (1)

- In unele cazuri o anumita informatie descriptiva nu este asociata cu o clasa de obiecte ci cu un ansamblu de clase diferite modelate fiecare prin entitati.
- Informatia va fi modelata ca un atribut al asocierii dintre entitatile respective.
- Exemplu: cazul unei asocieri multi-multi A_ABSOLVIT intre entitatile STUDENT şi FACULTATE care contine informatii privind facultatile absolvite anterior de unii studenti.

ATRIBUTE ASOCIERI (2)

- In acest caz informatii ca anul absolvirii, media, specializarea nu pot fi conectate nici la STUDENT (pentru ca un student poate fi absolventul mai multor facultati in ani diferiti, cu medii diferite, etc.) si din motive similare nici la FACULTATE.
- Ele descriu asocierea unui student cu o facultate si de aceea vor fi atasate asocierii A_ABSOLVIT.
- Toate atributele unei asocieri sunt atribute descriptive, neexistand in acest caz un identificator al asocierii.

ATRIBUTE ASOCIERI (3)



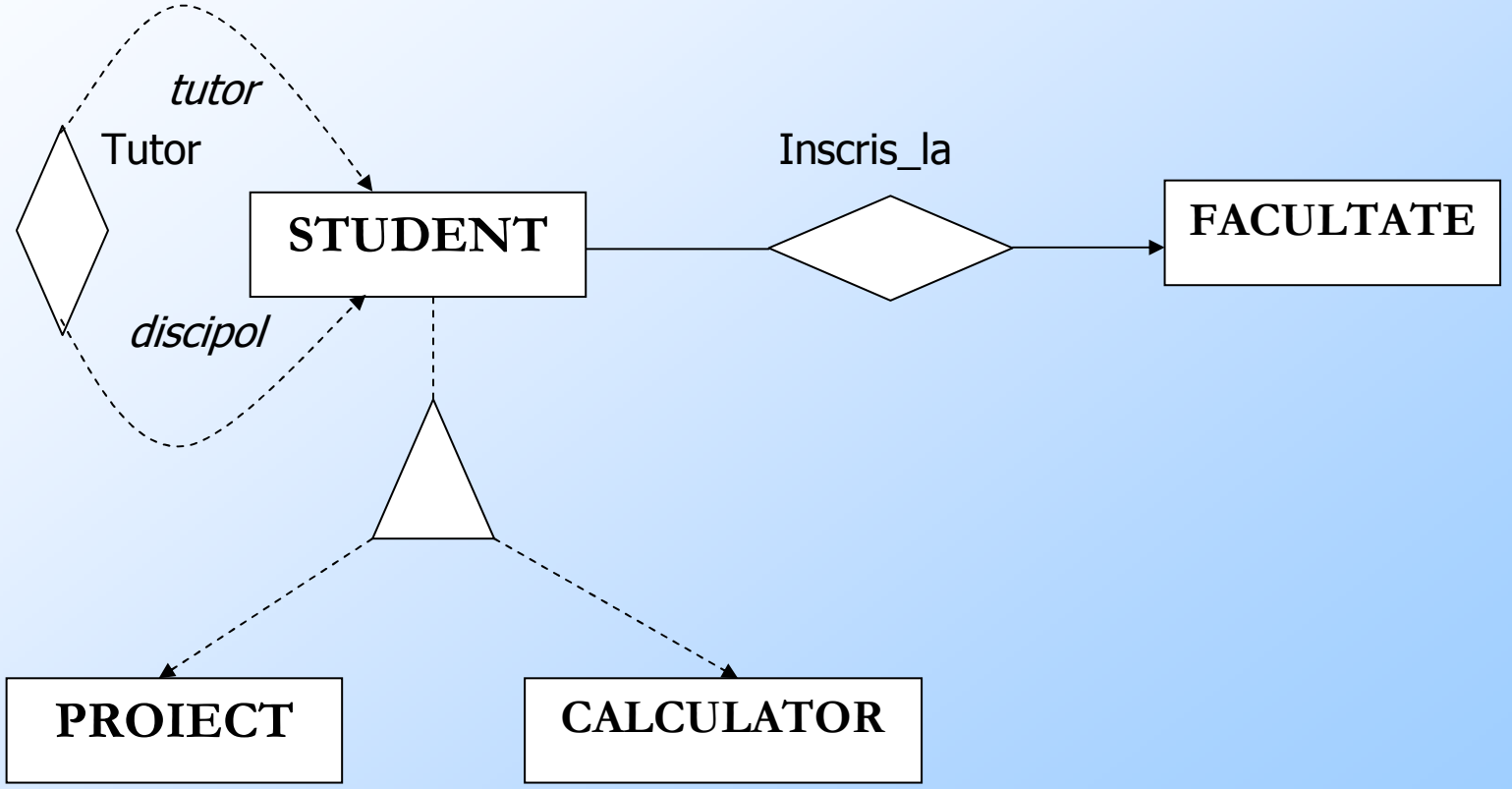
CARACTERISTICI ELEMENTE

- Gradul unei asocieri
- Conectivitatea unei ramuri de asociere
- Obligativitatea unei ramuri de asociere
- Atribute asocieri
- Roluri pentru ramurile asocierilor

ROLUL RAMURII (1)

- In cazul in care de la o asociere pornesc mai multe ramuri catre aceeasi entitate, fiecareia dintre acestea i se poate asocia un rol.
- Acesta arata semnificatiile diferite pe care le are aceeasi entitate in cadrul asocierii respective.
- In cazul asocierii TUTOR cele doua ramuri pot fi etichetate de exemplu cu **tutor** si **discipol** aratand ca instante diferite ale aceleiasi entitati au rolurile respective

ROLUL RAMURII (2)



CRITERII DE MODELARE

- Clasificarea in entitati şi atribute
- Identificarea ierarhiilor de generalizare şi incluziune
- Identificarea asocierilor
- Integrarea vederilor.

ENTITATI SI ATRIBUTE (1)

- Desi definitia notiunilor de entitate, atribut, asociere este destul de simpla, in practica modelarii apar dificultati in clasificarea diverselor informatii intr-una din aceste categorii.
- De exemplu in cazul sediilor unei banci localizate in diverse orase: obiectul ORAS este entitate distincta sau atribut descriptiv al entitatii SEDIU?

ENTITATI SI ATRIBUTE (2)

- Pentru a putea clasifica corect informatiile, exista citeva reguli care trebuie respectate şi pe care le prezentam in continuare.
- Prima regula da un criteriu general de impartire in entitati şi atribute,
- Urmatoarele doua semnaleaza exceptii iar ultimele doua reguli au un caracter mai putin normativ ci mai degraba orientativ.

Regula 1

- ***Regula 1.*** Entitatile au informatii descriptive, pe cand atributele nu poseda astfel de informatii.
- Daca exista informatii descriptive despre o anumita clasa de obiecte, aceasta va fi modelata ca o entitate.
- In cazul in care pentru pentru acea clasa de obiecte nu este nevoie decit de un identificator (codul, denumirea, etc), ea va fi modelata ca un atribut.

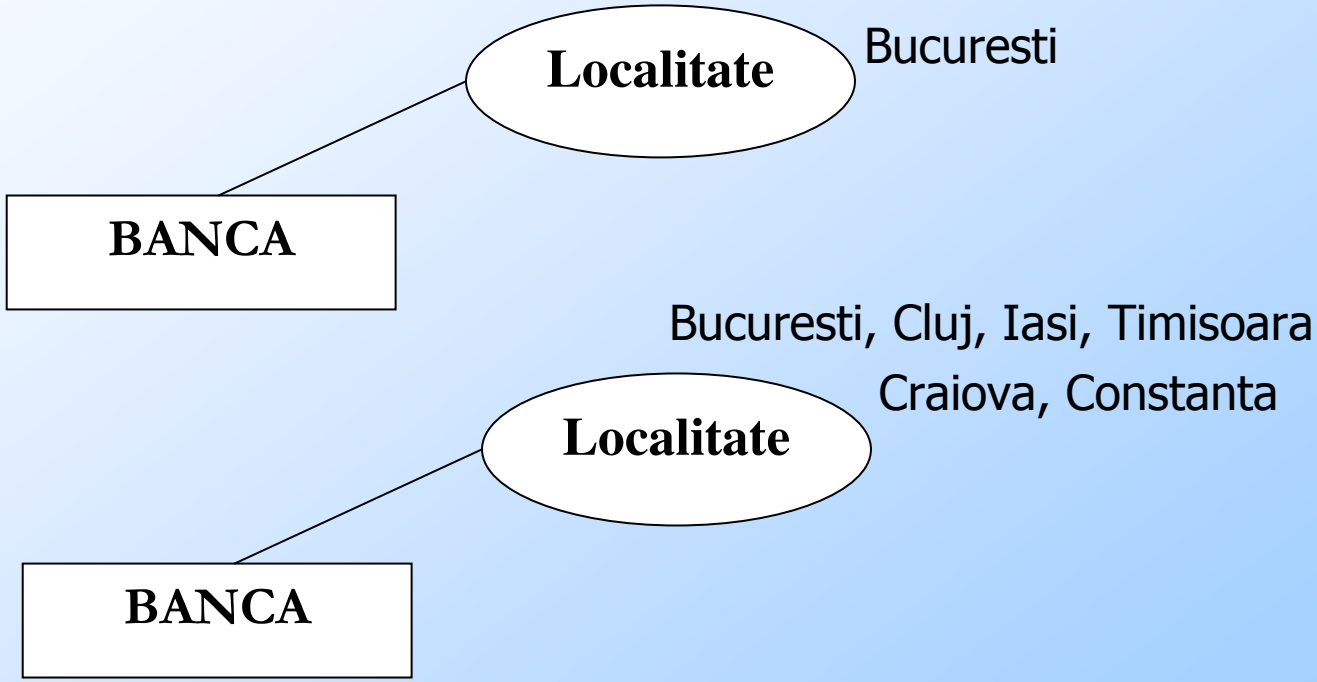
Exemplu

- Despre un ORAS este necesara stocarea in baza de date unor informatii ca JUDET, POPULATIE, etc. atunci ORAS va fi o entitate.
- Daca singura informatie necesara este numele sau atunci NUME_ORAS va fi un atribut al altei entitati.

Regula 2

- ***Regula 2.*** Atributele multivalorice vor fi reclasificate ca entitati.
- Daca la o valoare a unui identificator corespund mai multe valori ale unui descriptor, acesta va fi clasat ca entitate.
- De exemplu, in cazul unei baze de date privind localizarea in teritoriu a unor banci, daca se memoreaza informatii doar despre banci care au un singur sediu, LOCALITATE este atribut al entitatii BANCA.

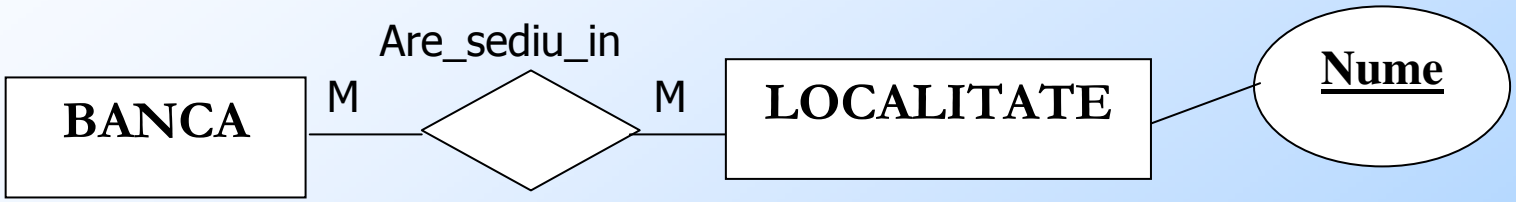
Exemplu


Localitate
Bucuresti
BANCA
Bucuresti, Cluj, Iasi, Timisoara
Craiova, Constanta
Localitate
BANCA

Exemplu - continuare

- Daca insa se memoreaza informatii despre banci care au sucursale şi filiale in diverse localitati, deci pentru o singura banca (o valoare a identificatorului entitatii BANCA) avem mai multe localitati in care aceasta are sedii (mai multe valori ale descriptorului LOCALITATE), atunci LOCALITATE va fi entitate distincta desi nu are decat un singur atribut.
- Pentru a modela localizarea sediilor in diverse localitati intre cele doua entitati va exista o asociere binara multi-multi numita de exemplu ARE_SEDIU_IN.

Exemplu - continuare

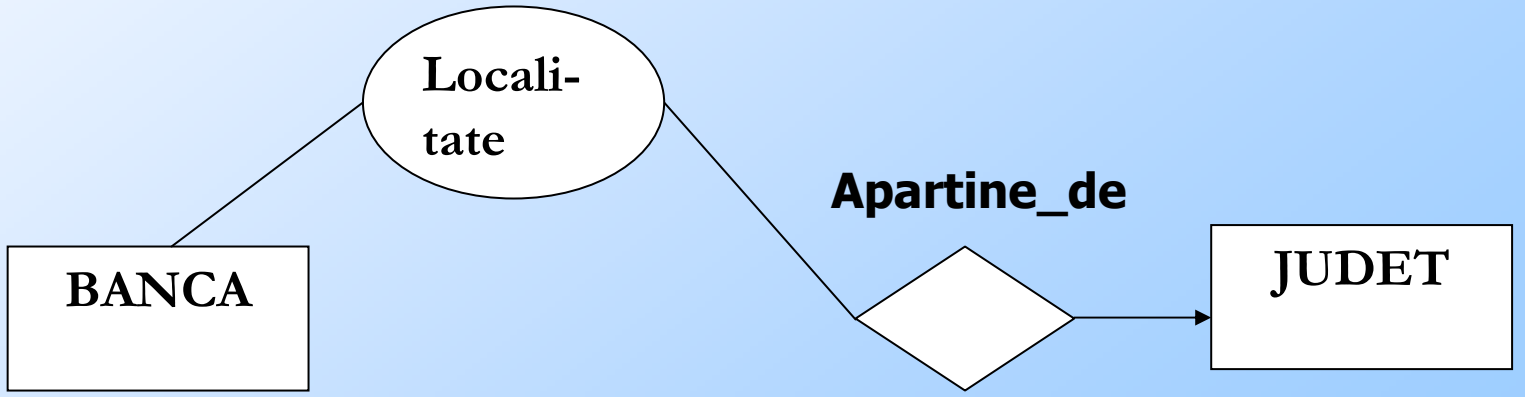


Regula 3

- ***Regula 3.*** Atributele unei entitati care au o asociere multi-unu cu o alta entitate vor fi reclasificate ca entitati.
- Asa cum am vazut asocierile pot lega doar entitati. Daca un descriptor al unei entitati este intr-o relatie multi-unu cu o alta entitate acel descriptor va fi trecut in categoria entitatilor.

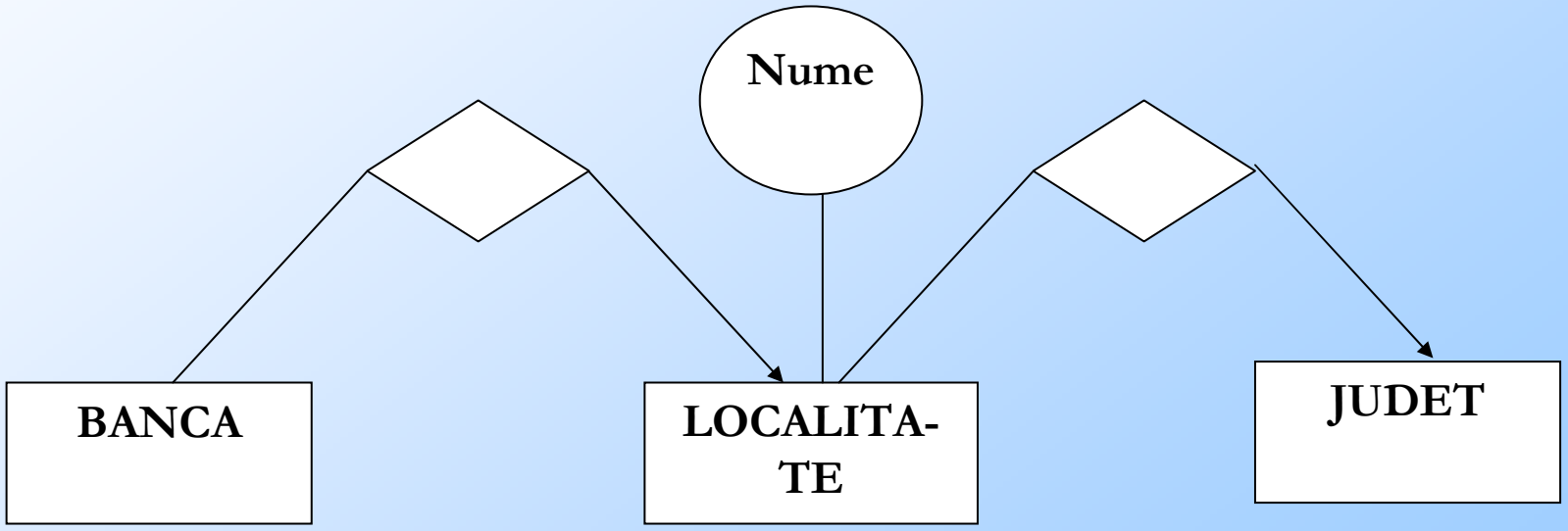
Exemplu

- Daca avem entitatile BANCA avand ca atribut descriptiv monovaloric LOCALITATE şi JUDET, daca se doreste modelarea apartenentei la judete a localitatilor va exista o asociere multi-unu intre atributul LOCALITATE şi entitatea JUDET.



Exemplu - continuare

- In acest caz LOCALITATE va fi reclasificata ca entitate desi nu sunt necesare alte informatii in afara numelui localitatii.



Regula 4

- ***Regula 4.*** Atributele vor fi atasate la entitatile pe care le descriu in mod nemijlocit.
- De exemplu, UNIVERSITATE va fi atasat ca atribut al entitatii FACULTATE şi nu al entitatilor STUDENT sau PROFESOR.

Regula 5

- ***Regula 5.*** Folosirea identificatorilor compusi va fi evitata.
- Identificatorul unei entitati este acea submultime de atribute ale acesteia care identifica in mod unic fiecare instanta a sa.
- In model relational pentru atributele de acest fel se construiesc de regula structuri de cautare rapida (indecsi) care functioneaza cu atat mai lent cu cat complexitatea indecsului creste.

Regula 5 - continuare

- Daca identificatorul unei entitati este compus din mai multe atribute care sunt toate identificatori in alte entitati, acea entitate se elimina (informatia va fi modelata sub forma unei asocieri intre acele entitati).
- Daca identificatorul unei entitati este compus din mai multe atribute care nu sunt toate identificatori in alte entitati, exista doua solutii:
- Entitatea respectiva se elimina şi este inlocuita prin alte entitati si asocieri (cu pastrarea informatiei modelate).
- Entitatea respectiva ramine (dar scade viteza).

Entitati si atribute - CONCLUZII

- Se vede ca procedura clasificarii obiectelor in entitati şi atribute este iterativa:
- se face o prima impartire conform primei reguli
- parte din atributele astfel obtinute se reclasifica in entitati conform regulilor 2 si 3
- se face o rafinare finala conform regulilor 4 si 5.

CRITERII DE MODELARE

- Clasificarea in entitati şi atribute
- Identificarea ierarhiilor de generalizare şi incluziune
- Identificarea asocierilor
- Integrarea vederilor.

IDENTIFICARE IERARHII (1)

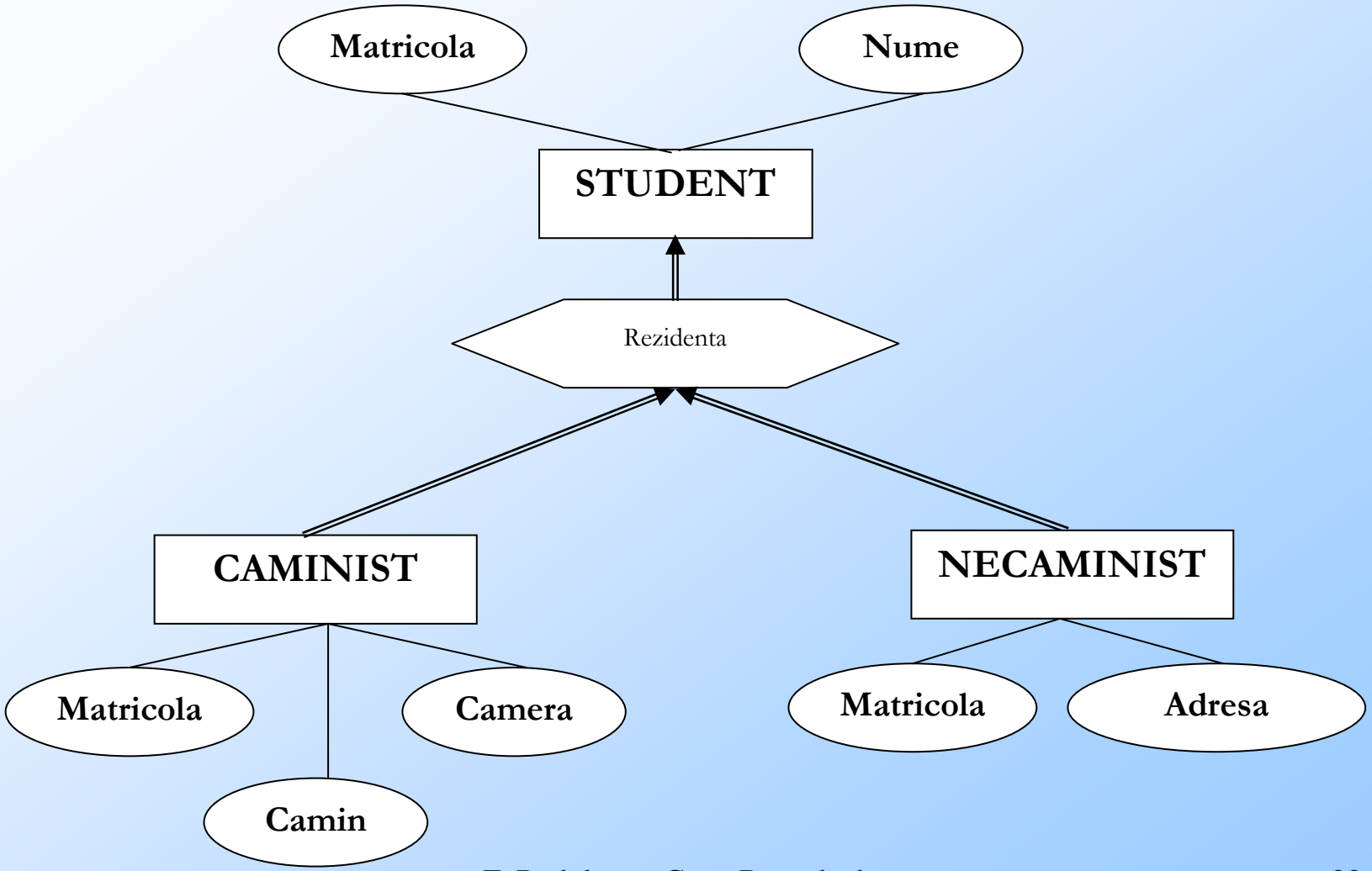
- In cazul in care despre anumite subclase ale unei clase de obiecte exista informatii specifice, clasa şi subclasele (care la pasul anterior au fost catalogate ca **entitati**) sunt interconectate intr-o ierarhie de incluziune sau generalizare, dupa cum este cazul.
- La acest pas se face şi o reatasare a atributelor pentru evitarea redundantei

IDENTIFICARE IERARHII (2)

Rearanjare atribute:

- La entitatea tata vor fi atasate atributele care formeaza **identificatorul** şi descriptorii care modeleaza informatii specifice **intregii clase**.
- La entitatile fiu vor fi atasate atributele de **identificare** (aceleasi ca ale tatalui) şi atributele care modeleaza informatii **specifice doar acelei subclase** de obiecte.

Exemplu


Matricola
Nume
STUDENT
Rezidenta
CAMINIST
NECAMINIST
Matricola
Camera
Camin
Matricola
Adresa

IERARHII - CONCLUZII

- Rezulta urmatoarele reguli:

1. Tatal şi fii unei ierarhii au acelasi identificator.
2. Descriptorii care apar şi la tata şi la fii se elimina de la fii.
3. Descriptorii care apar la toti fii unei ierarhii de generalizare şi nu apar la tata se muta la tata.

CRITERII DE MODELARE

- Clasificarea in entitati şi atribute
- Identificarea ierarhiilor de generalizare şi incluziune
- Identificarea asocierilor
- Integrarea vederilor.

IDENTIFICARE ASOCIERI (1)

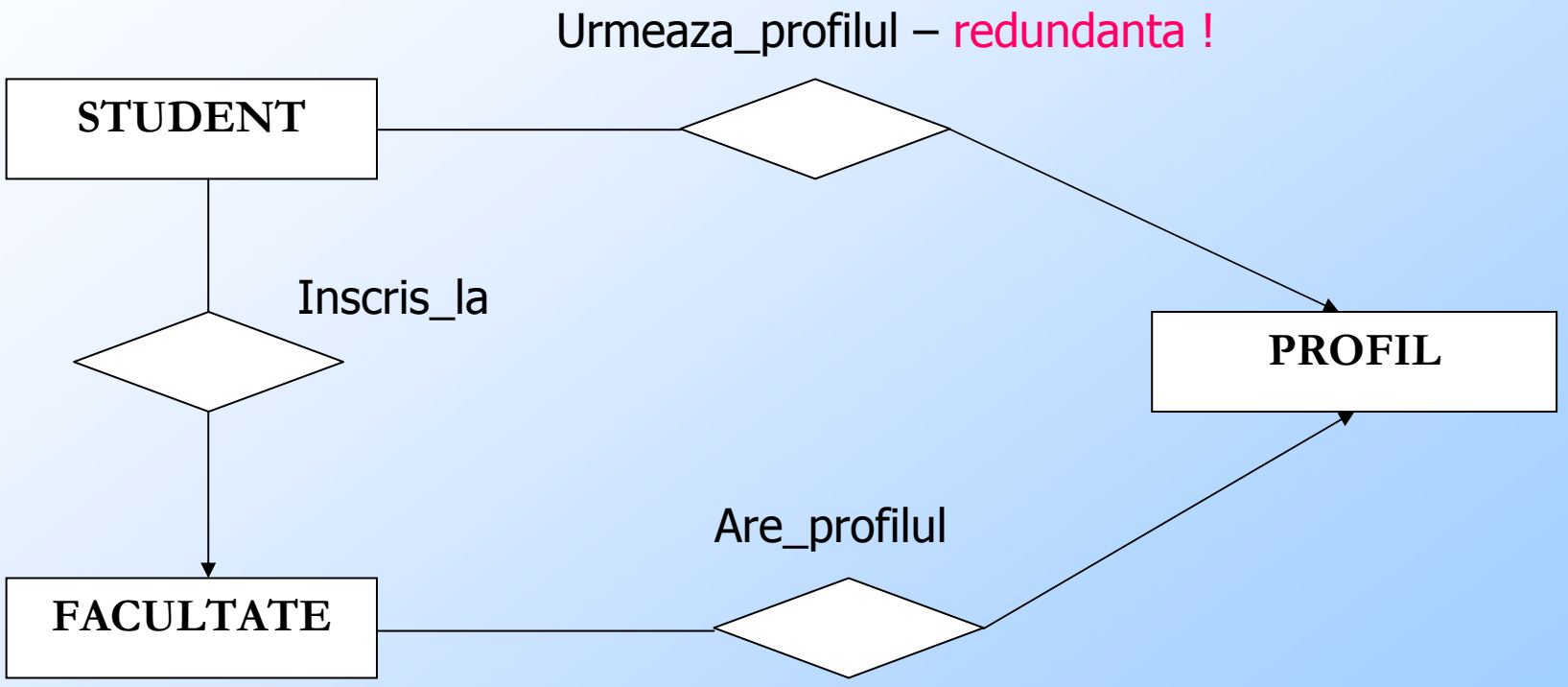
- In aceasta etapa se trateaza informatiile care nu au fost clasificate ca entitati sau atribute ci reprezinta interdependente intre clase de obiecte.
- Ele sunt modelate ca asocieri intre entitati.
- Pentru fiecare asociere se specifica gradul, conectivitatea, obligativitatea si daca este cazul si atributele asocierii precum si rolurile ramurilor sale.

IDENTIFICARE ASOCIERI (2)

- Ca şi in cazul clasificarii in entitati şi atribute, existe citeva reguli de urmat in operatia de definire a asocierilor:

1. Eliminarea asocierilor redundante
2. Evitarea Asocierilor de grad mai mare ca 2

ASOCIERI REDUNDANTE



ASOCIERI REDUNDANTE-cont.

- In acest exemplu, asocierea INSCRIS_LA modeleaza apartenenta fiecarui student la o facultate a unui institut de invatamint superior.
- Fiecare facultate are un profil unic descris de asocierea ARE_PROFILUL. Ambele asocieri sunt multi-unu in sensul STUDENT→FACULTATE→PROFIL.
- Deoarece asocierile multi-unu sunt asemanatoare unor functii, din compunerea lor putem afla profilul la care este inscris fiecare student.
- Rezulta ca asocierea URMEAZA_PROFILUL este redundanta şi trebuie eliminata.

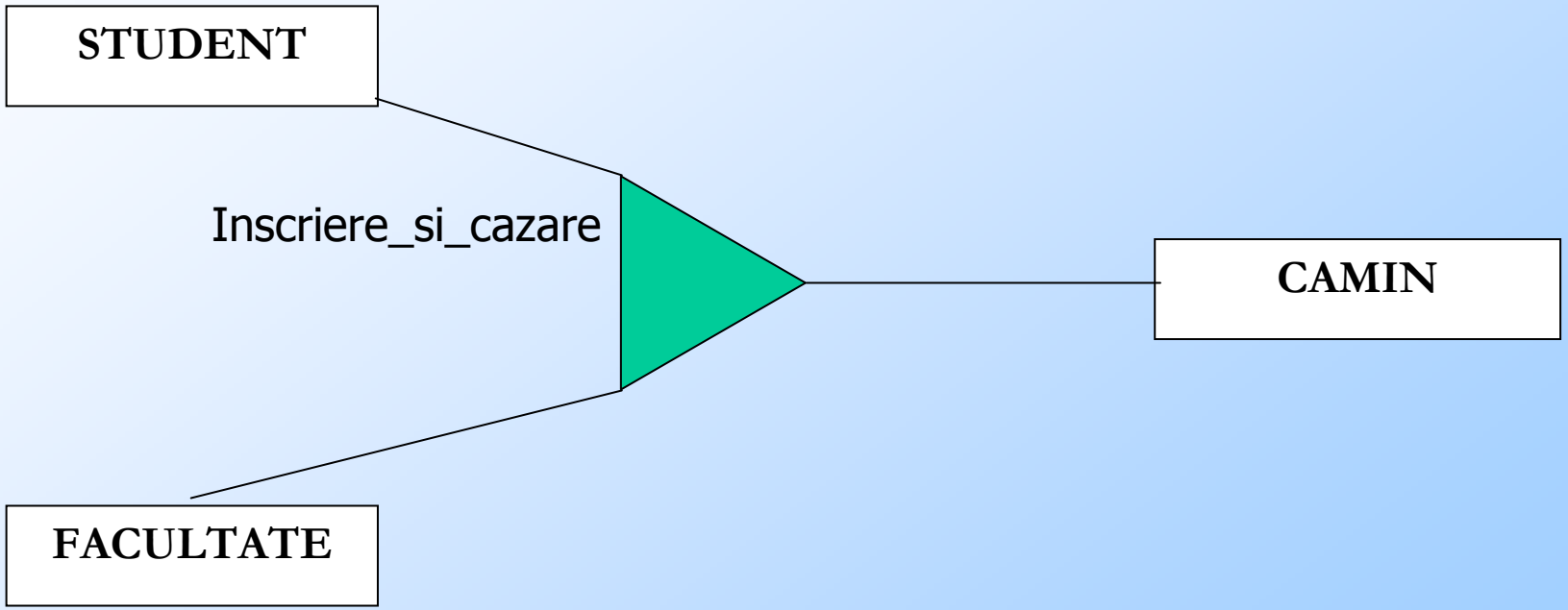
ASOCIERI DE GRAD > 2

- Asocierile de grad > 2 se folosesc doar atunci cand sunt strict necesare.
- Este de multe ori posibil ca o aceeasi informatie sa fie modelata ca o asociere ternara sau ca un ansamblu de asocieri binare si unare.
- In cazul acesta, este de preferat ca sa se opteze pentru a doua varianta.

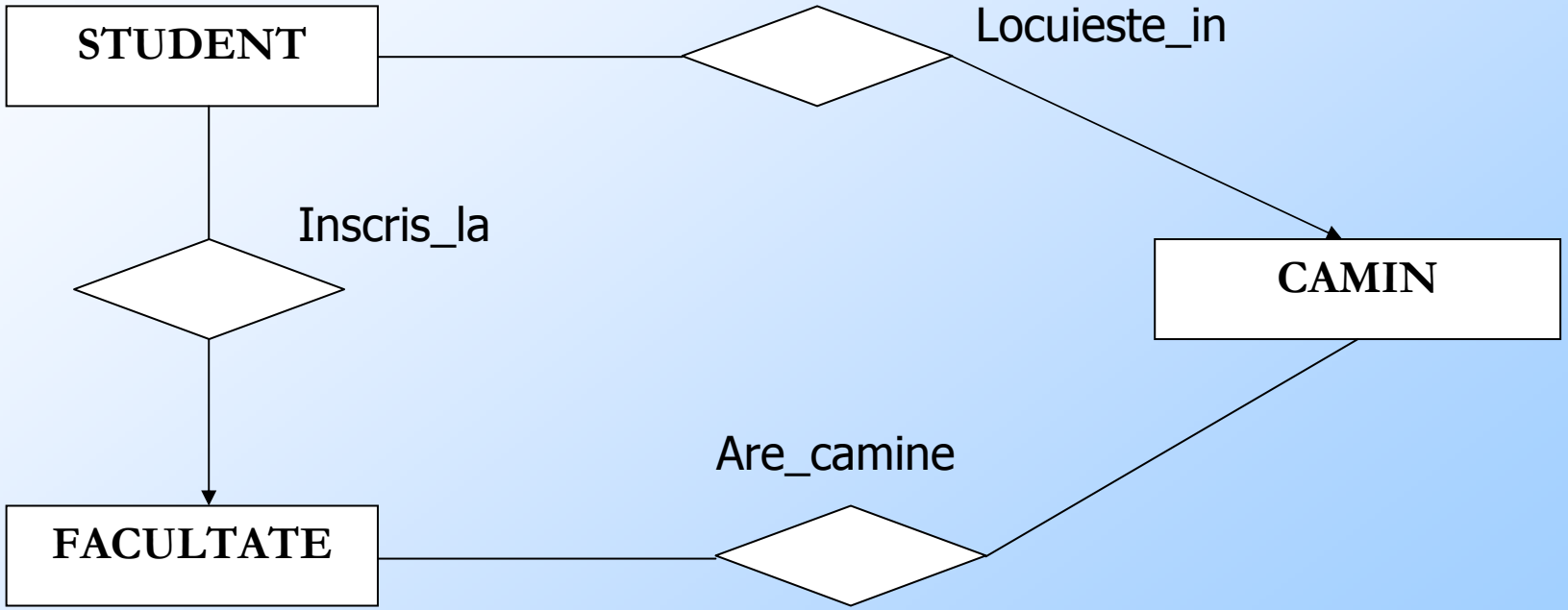
ASOCIERI DE GRAD > 2 – cont.

- Doar cand asocierile binare nu pot modela intreaga semnificatie dorita se va opta pentru asocieri de grad mai mare ca doi.
- Aceasta cerinta deriva din faptul ca la trecerea in modelul relational asocierile de grad superior devin scheme de relatii (tabele) de sine statatoare, marind numarul de tabele din baza de date pe cand cele de grad unu şi doi (cu exceptia celor multi-multi) nu au acest efect.

Exemplu – modelare gresita



Exemplu – modelare corecta



CRITERII DE MODELARE

- Clasificarea in entitati şi atribute
- Identificarea ierarhiilor de generalizare şi incluziune
- Identificarea asocierilor
- Integrarea vederilor

INTEGRAREA VEDERILOR

- In cazul proiectarii bazelor de date complexe, activitatea se desfasoara uneori de catre mai multe colective simultan, fiecare modeland o portiune distincta a bazei de date.
- Deoarece in final trebuie sa se obtina o singura diagrama a bazei de date, dupa terminatea modelarii pe portiuni diagramele rezultate sunt integrate eliminandu-se redundantele si inconsistentele.

INTEGRAREA VEDERILOR – cont.

Probleme:

1. Obiecte cu aceeasi semnificatie au nume diferite in diagramele de detaliu
2. Obiecte cu semnificatii diferite au acelasi nume in diagramele de detaliu

- Rezolvare: redenumire si “lipirea” diagramelor intr-una singura pe baza elementelor comune

Sfârşitul Capitolului 2