BAZE DE DATE

CURS 5

Observații temă

1. CNP-ul este o bună cheie candidat, dar este mai bine să introducem o CP artificială fiindcă ramane, totuși, greu de referit (pe lângă faptul că este o dată cu caracter personal).
2. În schemele relaționale se pun și cheile externe.
3. Nu definim subentități dacă nu avem fie atribute specifice, fie participare la relații diferite. În acest caz, va fi suficient un atribut (de exemplu, *tip_angajat*).
4. Legat de întrebarea "ce tabel are mai puține linii?" pentru a ști unde se plasează cheia externă, răspunsul îl găsim în specificația modelului sau chiar în alte legături (de exemplu, în modelul HR, avem și o relație 1:M între DEPARTMENTS și EMPLOYEES => știm că vor fi mai puține linii în DEPARTMENTS). Totuși, uneori numărul de linii este (aproximativ) același. Unde adaugăm cheia externă? Se poate oriunde, dar în practică alegem tabelul în care este mai important să regăsim valoarea respectivă (de exemplu, este mai important să avem atribut "id_locatie" la cinematograf decât "id_cinematograf" la locație).

Completări design BD

1. Introducem o CP artificială dacă cheia primară este compusă din multe (în general, în practică, >2) atribute sau atunci când nu avem o cheie candidat suficient de simplă (vezi observație CNP).

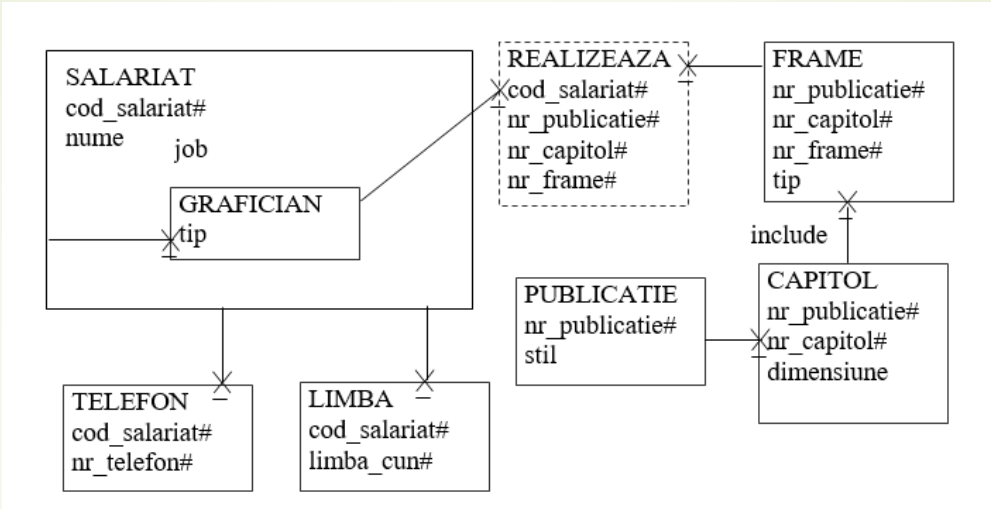
   Introducerea unei CP artificiale conduce la unele modificari ale modelului obținut:

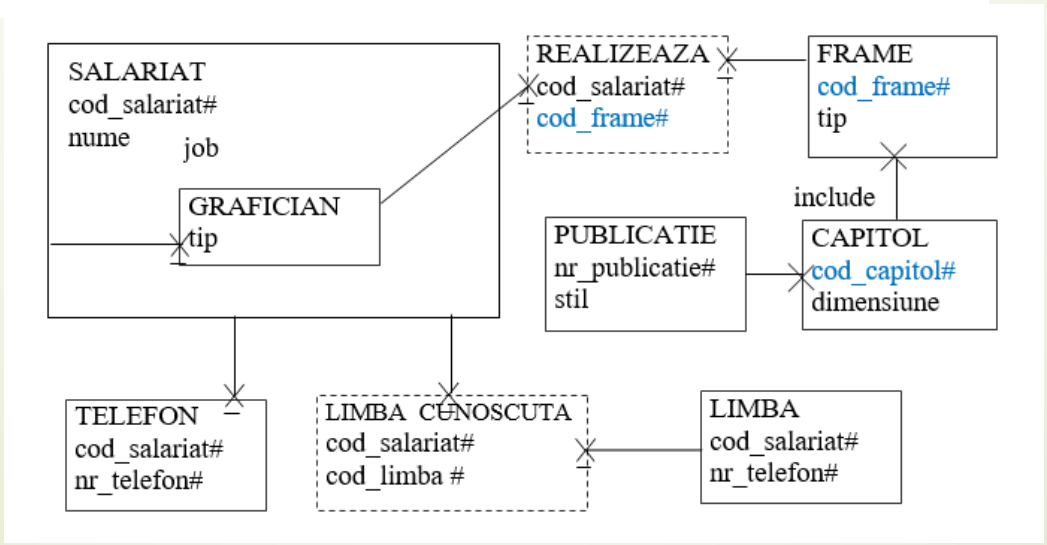
- se pierde noțiunea de dependență!
- se poate pune cheie artificială și pe tabelul asociativ, caz în care tabelul rămâne punctat în diagrama concepuală, însă fără dependențe (“x” simplu, nu subliniat). Totuși, știm precis că CP a tabelului dependent nu va fi referită de vreo cheie externă, deci poate rămâne compusă.

2. Observație atribute multiple: alegem modelarea ca tabel dependent sau relație M:M (tabel asociativ)?

- Dacă o aceeași valoare poate aparține mai multor entități, atunci poate fi o opțiune de design relația M:M.

Completări design BD



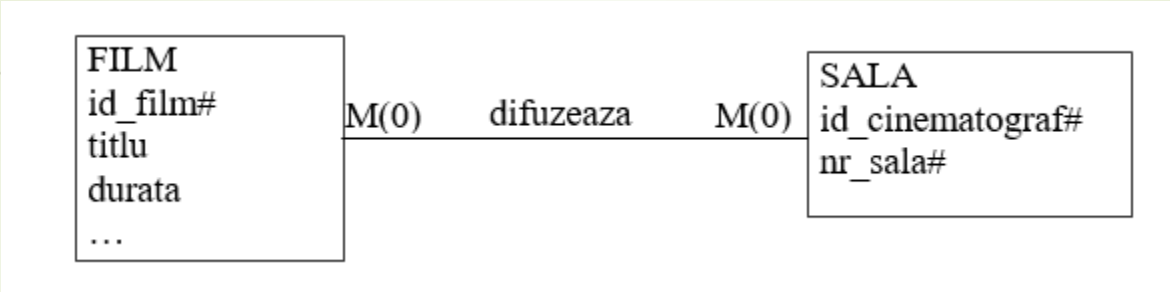


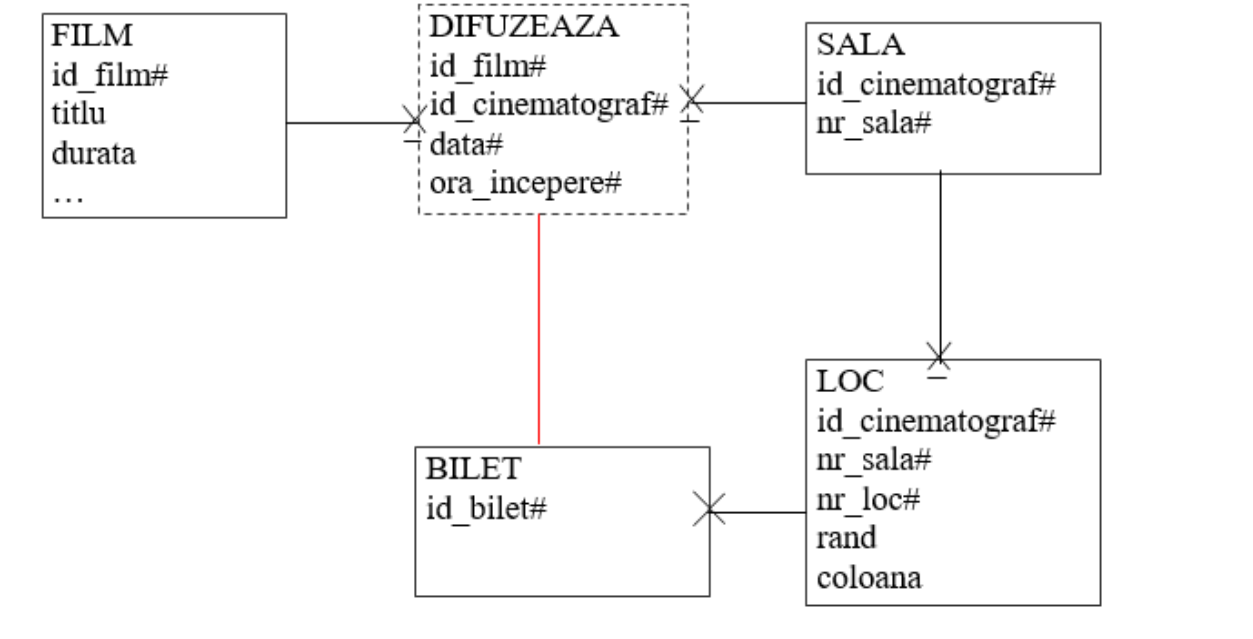
Completări design BD

3. Uneori ne aflăm în situația în care "apar" legături cu tabele asociative în diagrama conceptuală => pas înapoi în diagrama E/R pentru definirea corectă:

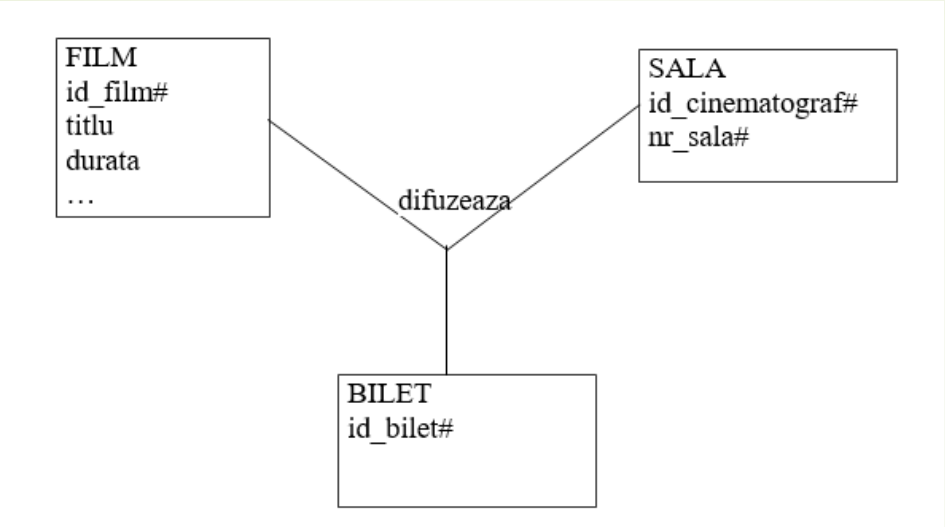
- Definirea unei relații de tip 3
- Considerarea tabelului asociativ ca entitate (pentru numele căreia găsim un substantiv potrivit)

Completări design BD

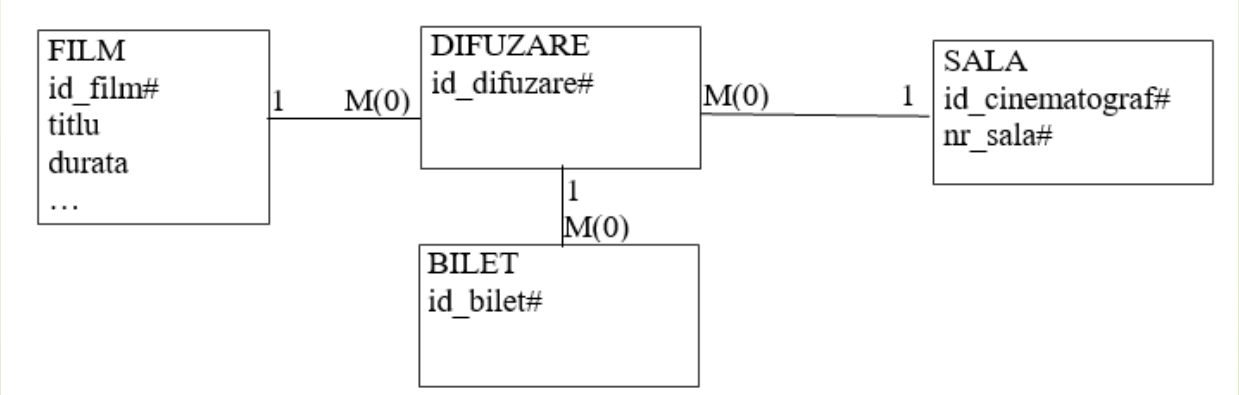




Completări design BD



Sau (mai bine, în acest caz – de ce?):



Modelare orientată pe obiecte cu *UML*

- Un **limbaj de modelare** reprezintă o modalitate de a **comunica** despre un sistem şi de a **exprima** diversele modele produse în cadrul procesului de analiză şi dezvoltare a sistemului.
  - Limbajul furnizează o **notaţie** care permite reprezentarea unui design.
- Ca orice limbaj, un limbaj de modelare are o **sintaxă** şi o **semantică**.
  - Pentru semantica modelului trebuie aleasă sintaxa cea mai expresivă.
  - De cele mai multe ori, un limbaj de modelare este un limbaj grafic (diagramatic).

Modelare orientată pe obiecte cu *UML*

- *The Unified Modelling Language* (**UML**) este un limbaj **vizual** de **modelare** şi de **comunicare** despre sistem.
- UML nu este un limbaj de programare.
- UML este un limbaj pentru modelarea orientată pe obiecte, cu suport pentru modelare vizuală, funcţionând ca o modalitate de a comunica şi de a exprima cunoştinţe.

Modelare orientată pe obiecte cu *UML*

- **UML** este un **limbaj grafic de modelare** pentru **specificarea**, **vizualizarea**, **construcţia** şi **documentarea** componentelor:
  - unui sistem *software* (pentru întreprinderi, telecomunicaţii, sisteme bancare şi financiare, transporturi etc.)
  - sau pentru modelarea organizaţională a unor sisteme *non-software* (din justiţie, medicină, afaceri etc.).
- **UML** permite realizarea tuturor **documentelor** necesare înţelegerii modelului şi diagramelor utilizate pe tot parcursul ciclului de viaţă al unui proces de realizare a unui sistem.
  - specificarea **cerinţelor**, **arhitecturii** şi **proiectării** sistemului; elaborarea **codului** sursă, planuri de dezvoltare şi de **management** al proiectului.

Modelare orientată pe obiecte cu *UML*

- UML **nu** este un limbaj de programare, dar modelele exprimate în UML pot fi implementate uşor în limbaje de programare orientate pe obiecte (C++, Java, C#) sau în **baze de date relaţionale**.
  - Este posibilă nu numai generarea codului dintr-un model UML, dar şi **ingineria inversă**, constând în construirea dintr-un cod dat a unui model UML.

Modelare orientată pe obiecte cu *UML*

- **Sintaxa limbajului UML implică diagrame, în timp ce semantica lui se bazează pe paradigma orientării pe obiecte.**
  - Din punct de vedere al modelării orientate pe obiecte, entităţile (conceptele) se numesc **clase**, iar relaţiile dintre ele se numesc **asocieri**.

- UML conţine mai multe tipuri de diagrame, fiecare reflectând unul sau mai multe dintre tipurile de vizualizări posibile asupra unui sistem *software*.

Modelare orientată pe obiecte cu *UML*

- **Diagrama claselor** descrie structura unui sistem în general. În componenţa ei intră **clase**, **stereotipuri** şi **relaţiile** dintre acestea.
- **Diagrama obiectelor** descrie structura unui sistem la un anumit moment. Acest tip de diagramă este o variantă a diagramei claselor care, în locul unei clase, prezintă mai multe instanţe ale ei, fiind formată din obiecte şi legături dintre ele.

Modelare orientată pe obiecte cu *UML*

- **Diagrama cazurilor de utilizare** descrie **funcţionalitatea** unui sistem, prezentând actorii externi, cazurile de utilizare identificate numai din punct de vedere al actorilor (comportamentul sistemului, aşa cum este perceput de utilizatorii lui), precum şi relaţiile dintre actori şi cazurile de utilizare.
  - Un actor poate fi orice sau oricine interacţionează cu sistemul (trimite sau recepţionează mesaje de la sistem sau schimbă informaţii cu acesta). Actorul are un rol în cadrul unui sistem, nu este un utilizator individual al acestuia şi, din acest motiv, el este o entitate (o clasă), nu o instanţă. Un caz de utilizare este iniţiat mereu de un actor şi furnizează o valoare actorului.

Modelare orientată pe obiecte cu *UML*

- **Diagrama componentelor** (diagrama de implementare) descrie structura fizică a codului în termenii componentelor de cod şi relaţiilor dintre acestea.
- **Diagrama de desfăşurare** (de exploatare) indică arhitectura fizică pe care este implementat sistemul, calculatoarele, nodurile sistemului şi conexiunile dintre ele.
- **Diagrama secvenţelor** este o diagramă de interacţiune, care prezintă colaborarea dinamică dintre un număr de obiecte, punând accentul pe secvenţele de mesaje trimise între acestea pe măsura trecerii timpului.
- **Diagrama de colaborare** este tot o diagramă de interacţiune, dar care, pe lângă interacţiunea dintre obiecte (schimbul de mesaje), prezintă obiectele şi legăturile dintre ele.

Modelare orientată pe obiecte cu *UML*

- **Diagrama de stare** descrie ciclul de viaţă al unui element (al obiectelor, subsistemelor şi sistemelor), prin specificarea stărilor în care se găseşte elementul şi a evenimentelor care îi modifică starea.
- **Diagrama de activitate** prezintă activităţile şi responsabilităţile elementelor sistemului, fiind utilizată pentru modelarea funcţiilor sistemului. Ea are ca elemente constitutive stări de acţiune şi mesaje care vor fi trimise sau recepţionate ca parte a acţiunii realizate.

Modelare orientată pe obiecte cu *UML*

În UML, diagramele fac parte din două categorii.

- **Diagrame dinamice** sau comportamentale – descriu comportamentul şi interacţiunile dintre diverse entităţi ale sistemului informatic.
  - diagramele de secvenţă, colaborare, stare şi activitate.
- **Diagrame statice** sau structurale - descriu structura, responsabilităţile sistemului informatic, componentele executabile ale sistemului, locaţiile fizice de execuţie şi nodurile de stocare a datelor.
  - diagramele claselor, obiectelor, cazurilor de utilizare, componentelor şi diagramele de exploatare.

Modelare orientată pe obiecte cu *UML*

- Diagramele UML pot fi desenate şi administrate utilizând un utilitar CASE (*Computer Aided Software Engineering*).
  - Aceste utilitare sunt deosebit de utile în cazul unor diagrame complexe.
  - Totuşi, dacă diagrama este prea complicată, atunci este necesară partiţionarea ei în mai multe diagrame sau reprezentarea la un nivel superior de abstractizare.
- Exemple de utilitare CASE care permit realizarea diagramelor UML sunt reprezentate de: *Microsoft Office* ***Visio***, *IBM Rational Rose Professional Data Modeler*, ***Lucidchart***, *Altova UModel*, *Borland Together*, *Visual Paradigm for UML*, *ArgoUML* etc.

Modelare orientată pe obiecte cu *UML*

- Interesul pentru suportul **modelării bazelor de date cu ajutorul UML** a condus la crearea unor **profile specifice**.
  - Un profil constituie o propunere a unei comunităţi şi regrupează o mulţime de elemente UML care se aplică unui context particular şi care conservă metamodelul UML.
  - *IBM Rational Rose* a inclus un astfel de profil adaptat bazelor de date.
- Există o diferenţă între un model (de exemplu, modelul conceptual de date) şi un formalism (în care este descris un model).
- Astfel, putem vorbi despre **modelarea conceptuală a datelor urmând formalismul entitate-relaţie sau formalismul UML**. Notaţia exprimă doar aspectul referitor la reprezentare.

Modelare orientată pe obiecte cu *UML*

- Pe lângă **formalismul entitate-relaţie**, considerat **standardul *de facto* pentru modelarea datelor**, o opţiune alternativă este oferită de către UML.
- Acesta include **primitive pentru modelarea datelor**, iniţial concepute pentru reprezentarea structurii claselor unei aplicaţii orientate obiect, dar care pot fi folosite pentru specificarea modelului de date al domeniului unei aplicaţii.
  - În particular, diagramele de clase UML pot fi utilizate ca alternativă la diagramele entitate-relaţie.

Modelare orientată pe obiecte cu *UML*

- **Diferenţa majoră** dintre o diagramă de clase UML şi o diagramă entitate-relaţie este reprezentată de **diferenţa dintre o clasă şi o entitate**: o clasă este o generalizare a noţiunii de entitate, care permite proiectantului să specifice nu numai atribute, ci şi funcţii (numite metode) aplicabile instanţelor clasei.
- UML este mai general decât modelul entitate-relaţie
  - iar proiectantul poate exploata diagramele de clase UML pentru a realiza **aceeaşi specificaţie** pe care o poate obţine cu ajutorul modelului entitate-relaţie.

Modelare orientată pe obiecte cu *UML*

- Tabelul următor stabileşte echivalenţele dintre formalismele modelului entitate-relaţie şi al notaţiei UML:

Entitate-relație	UML
Tip entitate	Clasă
Asociere (relație)	Asociere (relație)
Entitate	Obiect
Cardinalitate	Multiplicitate
Model conceptual de date	Diagramă de clase

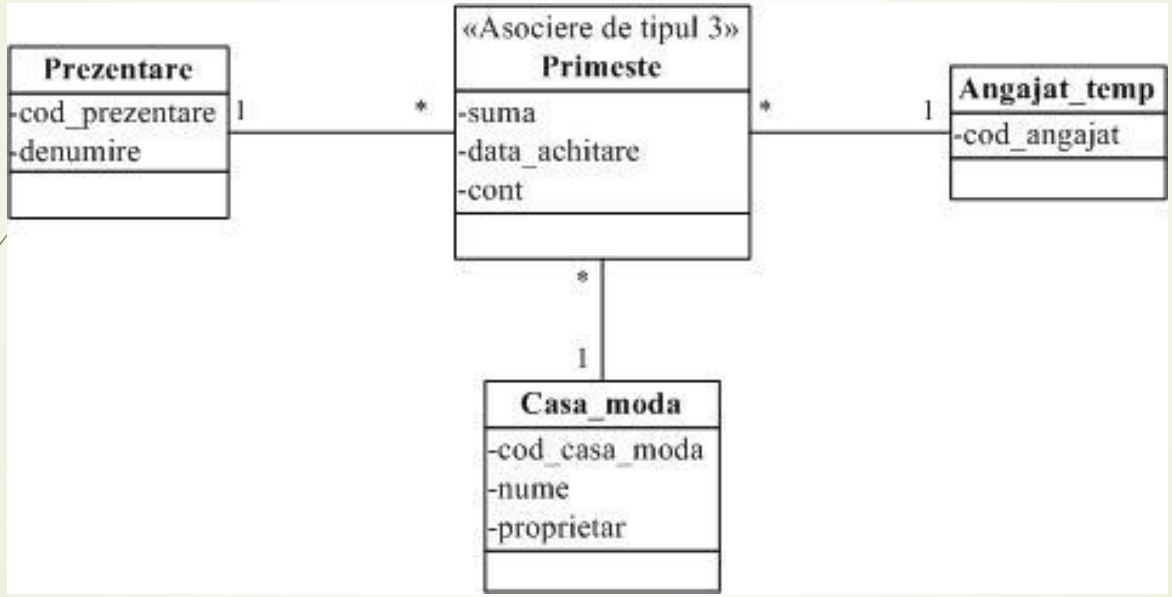
Modelare orientată pe obiecte cu *UML*

- Există **două forme de agregare**: compusă şi partajată.
- **Compunerea** exprimă o relaţie de apartenenţă mai puternică, apărând atunci când relaţia este de tipul „compune” sau „face parte din”. Reprezentarea compunerii se face prin intermediul unui romb plin, poziţionat lângă clasa care reprezintă întregul. Multiplicitatea extremităţii agregat nu poate depăşi 1.
- **Agregarea** partajată presupune că un obiect poate face parte din mai multe agregate, iar întregul se poate modifica în timp.O astfel de relaţie se reprezintă prin intermediul unui romb gol, poziţionat lângă clasa care reprezintă întregul.

Modelare orientată pe obiecte cu *UML*

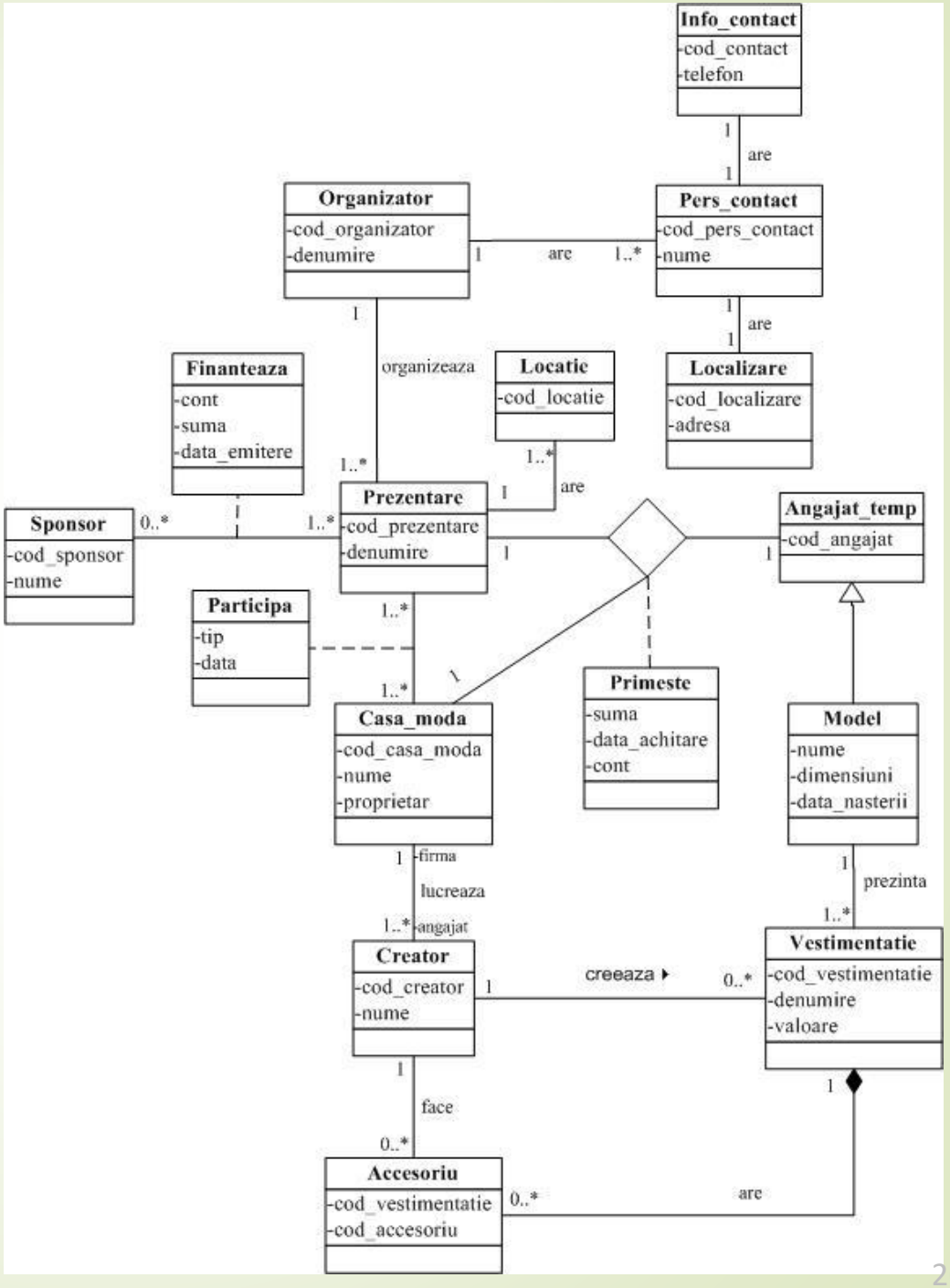
- Relaţiile de grade strict mai mari decât 2 sunt reprezentate în UML cu ajutorul unui romb sau prin intermediul unei clase cu **stereotip**.
- Stereotipurile constituie un mecanism de extensibilitate al UML.
  - Ele permit extinderea vocabularului UML astfel încât să poată fi definite noi elemente ale modelului, derivate din cele existente dar cu proprietăţi specifice domeniului problemei.
  - Reprezentarea grafică a unui stereotip se face prin plasarea numelui său, încadrat între caracterele „<<” şi „>>” deasupra numelui unui alt element.

Modelare orientată pe obiecte cu *UML*



Asociere UML de tipul 3 cu stereotip.

Diagrama de clase corespunzătoare unei restricţii a modelului.



TEMA

- Gestiunea unui lanț de depozite en-gros: diagrama E/R, diagrama conceptuală, schemele relaționale.