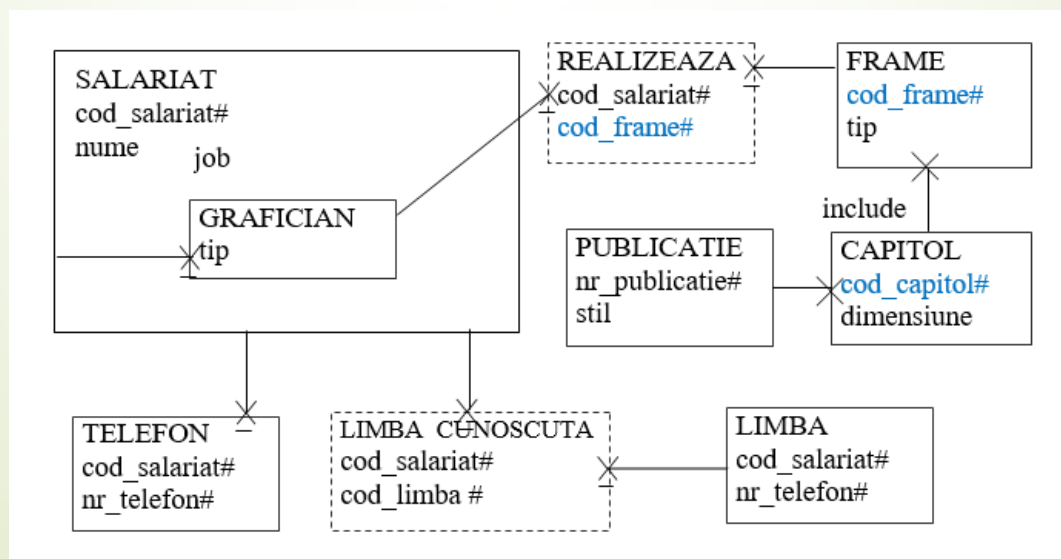
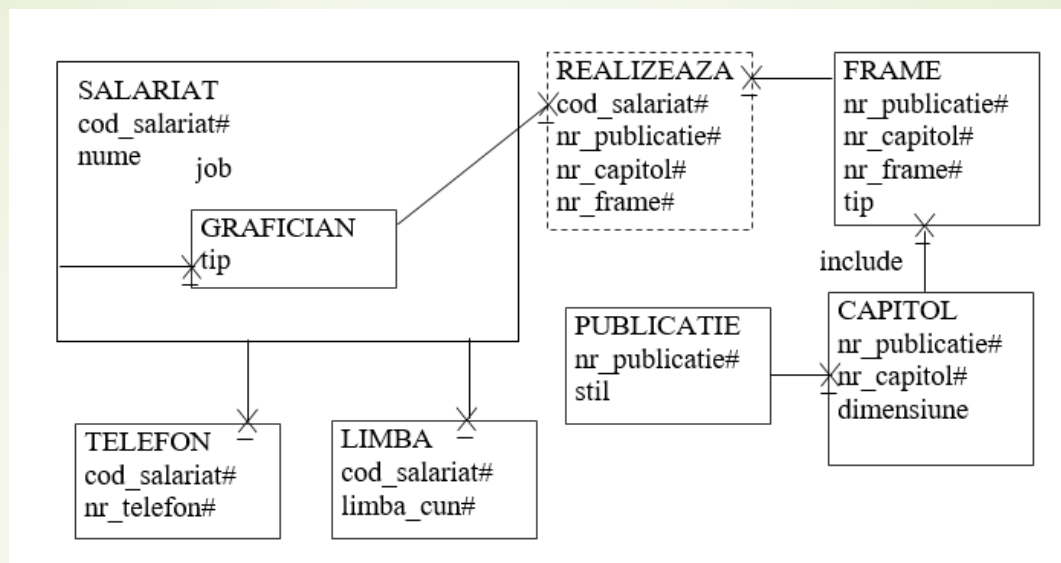




# BAZE DE DATE

CURS 5

# Completări design BD

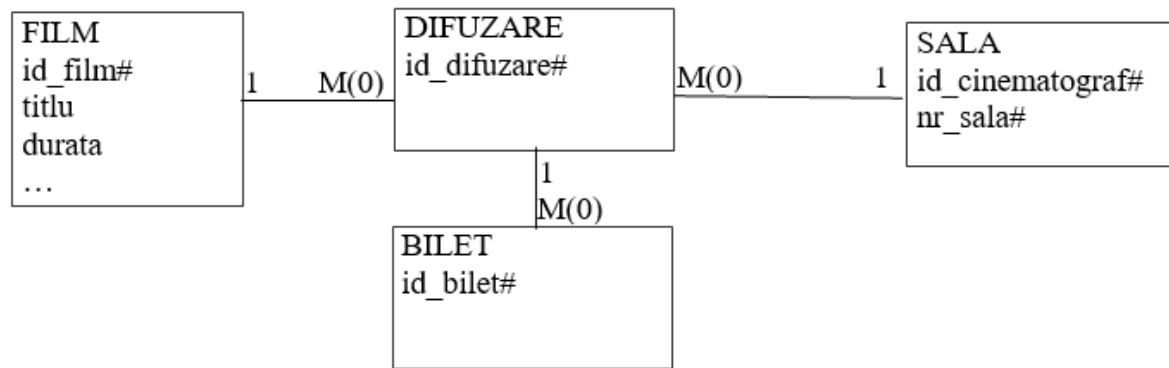
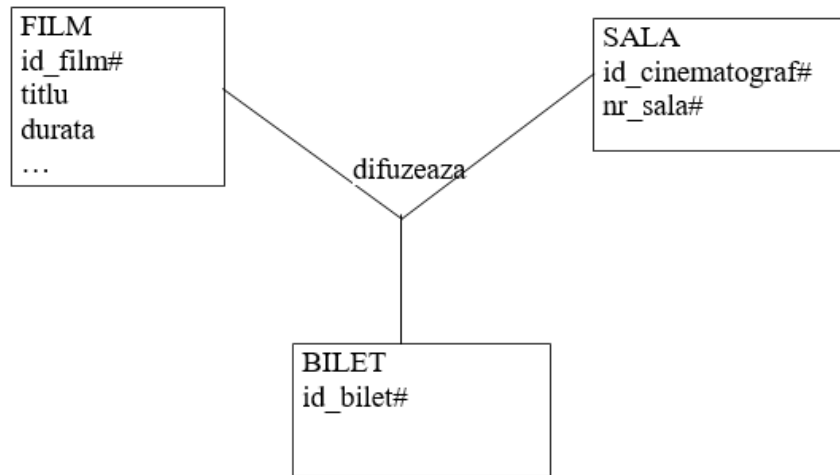


# Completări design BD

- Introducem o cheie primara artificială dacă cheia primară este compusă din multe attribute (în practică dacă sunt mai mult de doua attribute)
- Introducerea unei CP artificiale conduce la pierderea dependentei
- 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). In acest caz cheia primara a tabelului dependent nu va fi referită de nicio cheie externă.
- In cazul atributelor multiple alegem modelarea ca tabel dependent sau relație M:M (tabel asociativ)?
  - Dacă aceeași valoare poate aparține mai multor entități, atunci poate fi o opțiune de design relația M:M.


# Completări design BD

Ce varianta este mai optima si de ce?



# Modelare orientată pe obiecte cu *UML*

- Un **limbaj de modelare** reprezintă o modalitate de a **comunica** despre un sistem și de a **exprima** diversele modele produse în cadrul procesului de analiză și dezvoltare a sistemului.
  - Limbajul furnizează o **notație** care permite reprezentarea unui design.
- Ca orice limbaj, un limbaj de modelare are o **sintaxă** și 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** și 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, funcționând ca o modalitate de a comunica și de a exprima cunoștințe.


# Modelare orientată pe obiecte cu UML

- **UML** este un **limbaj grafic de modelare** pentru **specificarea, vizualizarea, construcția** și **documentarea** componentelor:
  - unui sistem *software* (pentru întreprinderi, telecomunicații, sisteme bancare și financiare, transporturi etc.)
  - sau pentru modelarea organizațională a unor sisteme *non-software* (din justiție, medicină, afaceri etc.).
- **UML** permite realizarea tuturor **documentelor** necesare înțelegerii modelului și diagramelor utilizate pe tot parcursul ciclului de viață al unui proces de realizare a unui sistem.
  - specificarea **cerințelor**, **arhitecturii** și **proiectării** sistemului; elaborarea **codului** sursă, planuri de dezvoltare și 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 ușor în limbaje de programare orientate pe obiecte (C++, Java, C#) sau în **baze de date relaționale**.
- ▶ Este posibilă nu numai generarea codului dintr-un model UML, dar și **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, entitățile (conceptele) se numesc **clase**, iar relațiile dintre ele se numesc **asocieri**.
- UML conține **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 componența ei intră **clase**, **stereotipuri** și **relațiile** 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 instanțe ale ei, fiind formată din obiecte și legături dintre ele.

# Modelare orientată pe obiecte cu *UML*

- **Diagrama cazurilor de utilizare** descrie **funcționalitatea** unui sistem, prezentând actorii externi, cazurile de utilizare identificate numai din punct de vedere al actorilor (comportamentul sistemului, așa cum este perceput de utilizatorii lui), precum și relațiile dintre actori și cazurile de utilizare.
  - Un actor poate fi orice sau oricine interacționează cu sistemul (trimite sau recepționează mesaje de la sistem sau schimbă informații cu acesta). Actorul are un rol în cadrul unui sistem, nu este un utilizator individual al acestuia și, din acest motiv, el este o entitate (o clasă), nu o instanță. Un caz de utilizare este inițiat mereu de un actor și 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 și relațiilor dintre acestea.
- **Diagrama de desfășurare** (de exploatare) indică arhitectura fizică pe care este implementat sistemul, calculatoarele, nodurile sistemului și conexiunile dintre ele.
- **Diagrama secvențelor** este o diagramă de interacțiune, care prezintă colaborarea dinamică dintre un număr de obiecte, punând accentul pe secvențele de mesaje trimise între acestea pe măsura trecerii timpului.
- **Diagrama de colaborare** este tot o diagramă de interacțiune, dar care, pe lângă interacțiunea dintre obiecte (schimbul de mesaje), prezintă obiectele și legăturile dintre ele.



# Modelare orientată pe obiecte cu *UML*

- **Diagrama de stare** descrie ciclul de viață al unui element (al obiectelor, subsistemelor și sistemelor), prin specificarea stărilor în care se găsește elementul și a evenimentelor care îi modifică starea.
- **Diagrama de activitate** prezintă activitățile și responsabilitățile elementelor sistemului, fiind utilizată pentru modelarea funcțiilor sistemului. Ea are ca elemente constitutive stări de acțiune și mesaje care vor fi trimise sau recepționate ca parte a acțiunii realizate.

# Modelare orientată pe obiecte cu *UML*

În UML, diagramele fac parte din două categorii.

- **Diagrame dinamice** sau comportamentale – descriu comportamentul și interacțiunile dintre diverse entități ale sistemului informatic.
  - diagramele de secvență, colaborare, stare și activitate.
- **Diagrame statice** sau structurale - descriu structura, responsabilitățile sistemului informatic, componentele executabile ale sistemului, locațiile fizice de execuție și nodurile de stocare a datelor.
  - diagramele claselor, obiectelor, cazurilor de utilizare, componentelor și diagramele de exploatare.



# Modelare orientată pe obiecte cu *UML*

- Diagramele UML pot fi desenate și administrate utilizând un utilitar **CASE** (*Computer Aided Software Engineering*).
  - Aceste utilitare sunt deosebit de utile în cazul unor diagrame complexe.
  - Totuși, dacă diagrama este prea complicată, atunci este necesară partiționarea 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 comunități** și regroupează o mulțime de elemente UML care se aplică unui context particular și care conservă metamodelul UML.
  - *IBM Rational Rose* a inclus un astfel de profil adaptat bazelor de date.
- Există o **diferență între un model** (de exemplu, modelul conceptual de date) și un **formalism** (în care este descris un model).
- Astfel, putem vorbi despre **modelarea conceptuală a datelor urmând formalismul entitate-relație sau formalismul UML**. Notația exprimă doar aspectul referitor la reprezentare.



# Modelare orientată pe obiecte cu *UML*

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

# Modelare orientată pe obiecte cu *UML*

- **Diferența majoră** dintre o diagramă de clase UML și o diagramă entitate-relație este reprezentată de **diferența dintre o clasă și o entitate**: o clasă este o generalizare a noțiunii de entitate, care permite proiectantului să specifice nu numai attribute, ci și funcții (numite metode) aplicabile instanțelor clasei.
- **UML este mai general decât modelul entitate-relație**
  - iar proiectantul poate exploata diagramele de clase UML pentru a realiza **aceeași specificație** pe care o poate obține cu ajutorul modelului entitate-relație.

# Modelare orientată pe obiecte cu *UML*

- Tabelul următor stabilește echivalențele dintre formalismele modelului entitate-relație și al notației UML:

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

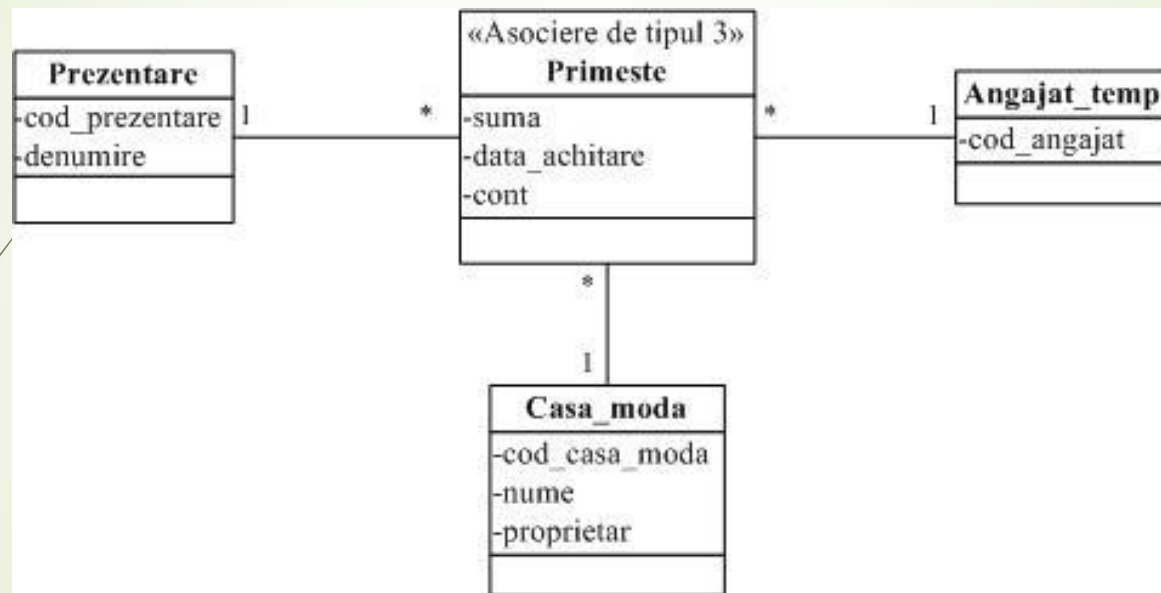
# Modelare orientată pe obiecte cu UML

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

# Modelare orientată pe obiecte cu UML

- Relațiile 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 proprietăți specifice domeniului problemei.
  - Reprezentarea grafică a unui stereotip se face prin plasarea numelui său, încadrat între caracterele „<<” și „>>” deasupra numelui unui alt element.

# Modelare orientată pe obiecte cu *UML*



Asociere UML de tipul 3 cu stereotip.

Diagrama de clase corespunzătoare unei restricții a modelului.

