Capítol 5:
Els dissenys conceptual i lògic

Contingut

5.1. Introducció

El *Unified Modeling Language (UML)* és una notació que combina les tres majors branques del disseny OO:

- El modelisme OMT de J. Rumbaugh
- L'anàlisi i el disseny OO de G. Booch
- L'Objectory de I. Jacobson

El Object Management Group (OMG) ha adoptat el UML com la notació estàndard per les metodologies d'objectes.

$\Rightarrow$ Ens centrarem en els elements del UML necessaris per modelar dades.

5.1. Introducció

Utilitzarem el *diagrama de classes* per expressar *l'esquema conceptual* (veure arquitectura de referència d'ANSI/SPARC), de forma no depenent del SGBD, durant el *disseny conceptual* de la BD. Després, mitjançant el disseny lògic, traduirem el UML a relacional i així aconseguirem un disseny depenent del SGBD.

5.2. Classes i atributs

Un diagrama de classes és un diagrama estructural o estàtic amb el que es modela l'estructura d'un sistema de classes.

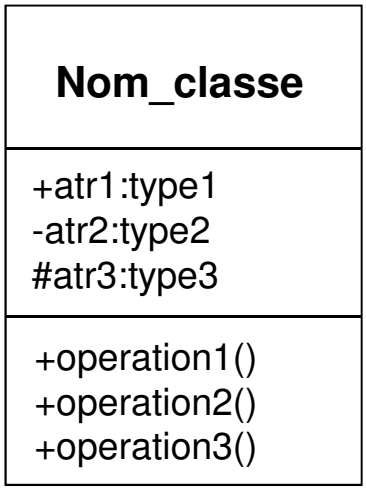
Una classe és una descripció d'un conjunt d'objectes que comparteixen els mateixos atributs, operacions, mètodes, interrelacions i semàntica.

Un atribut és un espai, amb nom, corresponent a un classificador (una interfície, tipus, classe, subsistema, BD o component) que descriu un rang de valors que les instàncies del classificador poden prendre.

5.2. Classes i atributs

Una *classe* es representa mitjançant un rectangle amb tres compartiments:

- El compartiment superior conté el *nom de la classe*.
- El compartiment del mig conté una *llista d'atributs* i les seves propietats.
- El compartiment inferior conté una *llista d'operacions* amb les seves propietats.



5.2. Classes i atributs

Existeixen tres símbols per expressar la visibilitat dels atributs:

\+ *Visibilitat pública*: qualsevol altra classe pot examinar o modificar directament el valor de l'atribut.

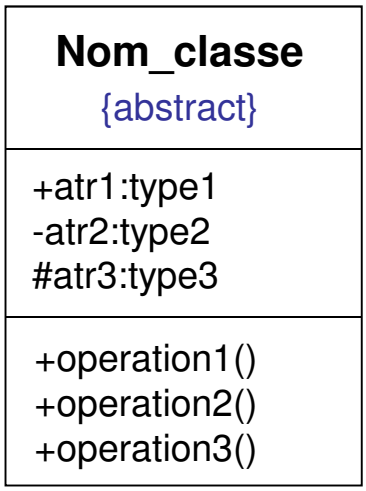
\# *Visibilitat protegida*: només els mètodes de la classe o d'una subclasse pública o protegida poden examinar o modificar directament el valor de l'atribut.

– *Visibilitat privada*: només els mètodes de la classe poden examinar o modificar directament el valor de l'atribut.

5.2. Classes i atributs

Una classe pot ser abstracta:

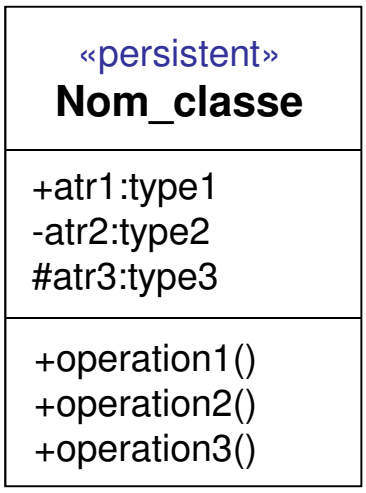
- És una classe que no té instàncies.
- Existeix per representar una abstracció comú que comparteixen diferents subclasses.
- Permet reduir la redundància.
- El seu nom es posa en “itàlics”, o bé afegint sota el nom de la classe el terme {abstract}.



5.2. Classes i atributs

Es pot estendre la semàntica d'un element del model amb els estereotipus *(stereotypes).*

Per indicar que una classe ha de ser persistent, per tant cal mapejar-la a la BD apropiada durant la implementació, s'utilitza l'esterotipus *«persistent».*



5.3. Interrelacions

Una interrelació és una connexió entre classes que relaciona una classe amb altres classes (o retorna a ella mateixa).

Al modelar dades, amb UML s'utilitzen dos tipus d'interrelació:

- Generalització: és una interrelació taxonòmica entre un element general i un altre de més específic. Permet definir la *jerarquia d'herència*.
- Associació: és una interrelació semàntica entre dos o més classificadors que implica connexions entre les seves instàncies. Mostra com un objecte d'una classe es relaciona amb altres objectes. Defineix la *jerarquia de contenció*.

5.3. Interrelacions

En moltes circumstàncies no està clar quan declarar un *atribut* o quan declarar una *associació*.

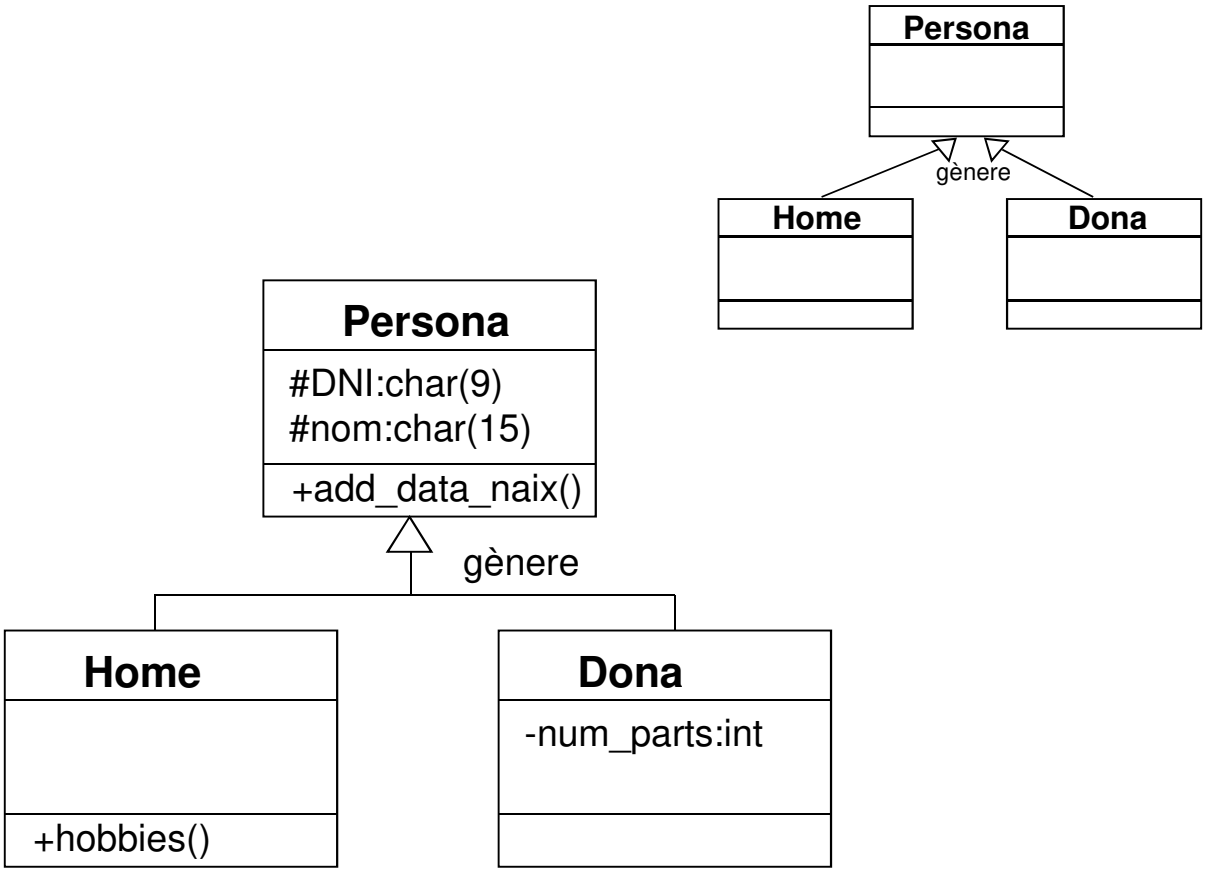
UML iguala atributs amb associacions: el tipus de dada de l'atribut és el nom d'una classe o d'un tipus.

Es recomana especificar atributs només pels tipus de dades primitius dels llenguatges (integer, float, varchar, char, ...).

En canvi, per representar objectes que són instàncies de classes s'ha d'utilitzar sempre les associacions.

5.3. Interrelacions

5.3.1. Generalització (I)



Les *generalitzacions* es representen amb una fletxa amb l'apuntador blanc.

5.3. Interrelacions

5.3.1. Generalització (II)

- L'apuntador apunta a la classe més general, anomenada *superclasse* o *classe base*.
- La fletxa s'origina a la classe més especialitzada, que té per nom *subclasse* o *classe derivada*.
- Una *classe fulla* és una classe que ja no té més especialitzacions.
- Una *classe arrel* és una classe que no té res més general.
- Una superclasse amb diferents subclasses pot tenir un nom que s'utilitzi com a *discriminador*:
  - El discriminador es considera un atribut de la superclasse que identifica la subclasse.
  - No cal especificar-lo en la llista d'atributs de la superclasse.
  - Només consta com a etiqueta de la generalització.

5.3. Interrelacions

5.3.1. Generalització (III)

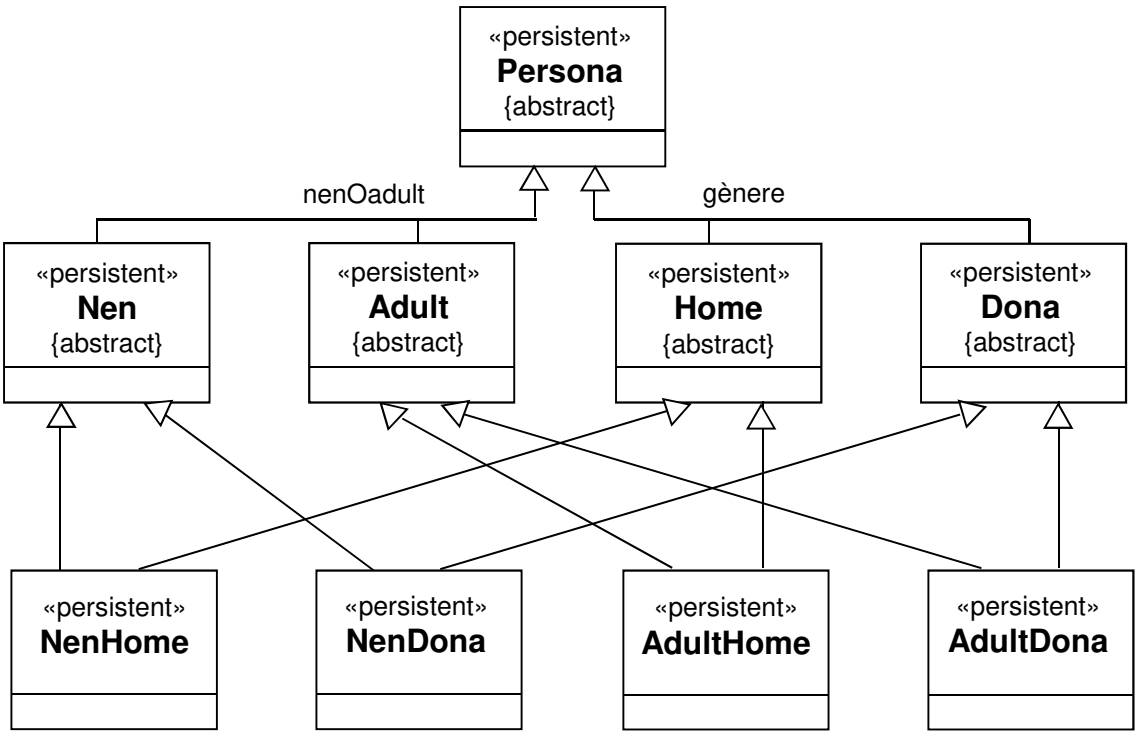
- *Polimorfisme*: és l'habilitat d'un llenguatge d'utilitzar el mateix nom per a diferents operacions. Hi ha dues opcions:
  - *Overloading*: les operacions tenen el mateix nom però diferent signatura.
  - *Overriding*: es defineix en una subclasse una operació que restringeix o canvia el comportament de l'operació que estava definida en la superclasse, però manté la mateixa signatura.

    Existeixen SGBDOO i SGBDOR que suporten ambdós conceptes (cal anar amb compte).

5.3. Interrelacions

5.3.1. Generalització (IV)

- *Herència Múltiple*: es dona quan existeixen classes que tenen més d'una superclasse com a pare. La raó d'això és la presència de més d'una dimensió en les dades.



5.3. Interrelacions

5.3.1. Generalització (V)

Restriccions d'especialització: (I)

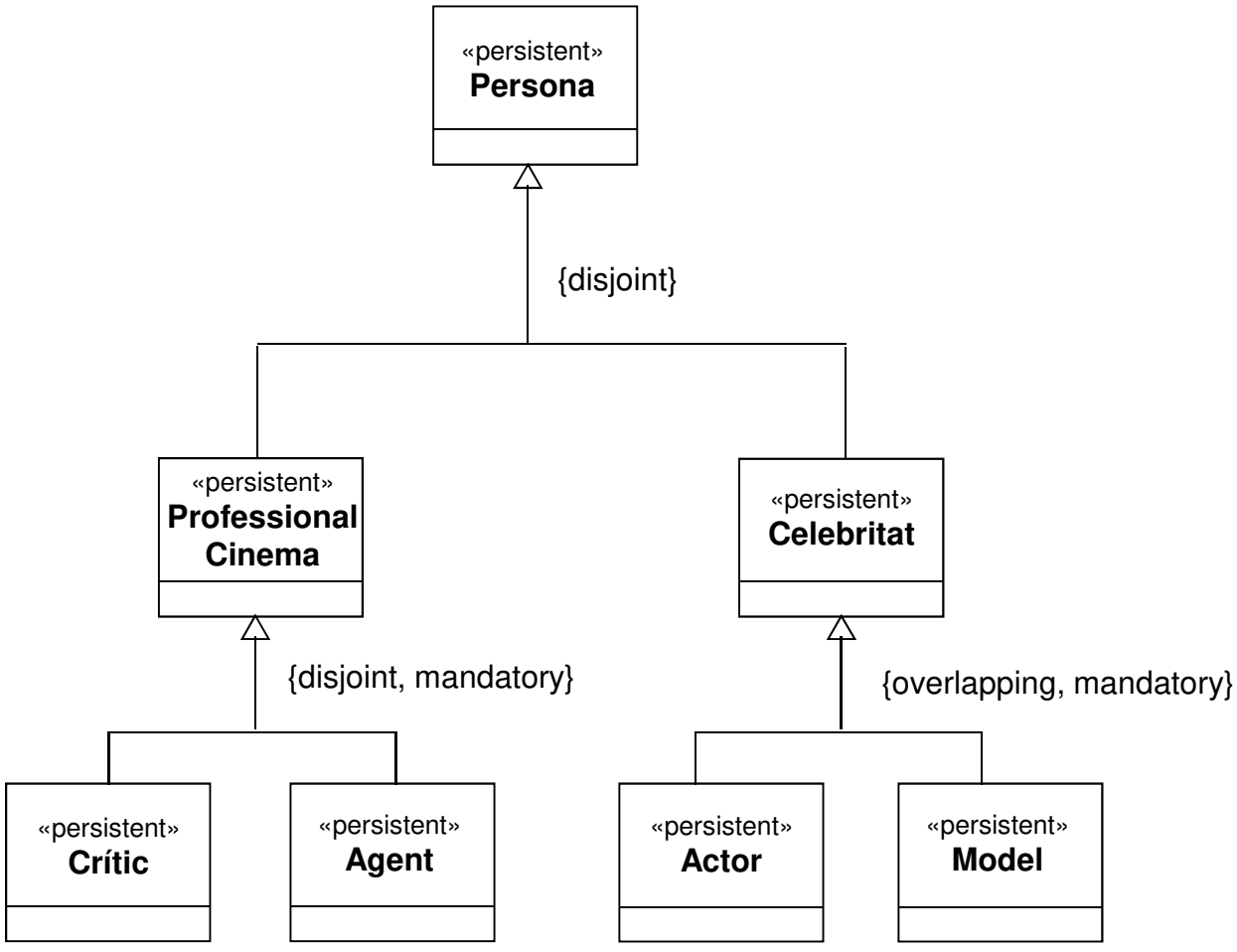
- s'utilitzen per expressar explícitament el significat d'una especialització.
- S'indiquen entre claus ({ }) i s'adjunten a la fletxa que representa la generalització.
- Els tipus de restriccions són:
  - *Disjoint*: les subclasses són disjuntes.
  - *Overlapping*: les subclasses es poden solapar.
  - *Mandatory*: la participació a les subclasses és obligatoria.
  - *Complete*: totes les subclasses possibles estan indicades.
  - *Incomplete*: totes les subclasses possibles encara no han estat indicades.

5.3. Interrelacions

5.3.1. Generalització (VI)

Restriccions d'especialització: (II)

Exemple



5.3. Interrelacions

5.3.1. Generalització (VII)

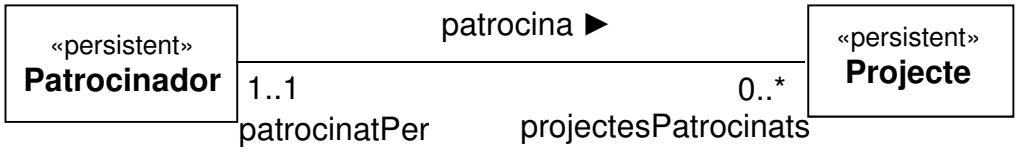
Puntualitzacions:

- En la representació per defecte de la generalització no s'indiquen explícitament ni discriminadors ni restriccions d'especialització.
- L'absència de la restricció *mandatory* implica que l'especialització és parcial (poden haver instàncies de la superclasse que no estiguin a cap subclasse)
- L'ús d'un discriminador sempre implica que l'especialització serà *disjoint* i *mandatory*.

5.3. Interrelacions

5.3.2. Associació (I)

Les *associacions binàries* són el tipus bàsic d'associació.



Es representen mitjançant una línia entre les dues classes associades.

A la línia s'hi poden afegir (opcionalment):

- *Nom de l'associació*: per indicar el significat de l'associació segons un sentit determinat (amb una fletxa s'indica el sentit en que s'ha de llegir l'associació).
- *Nom dels "rols" de cada classe participant*: permet indicar de quina forma participa en l'associació cadascuna de les classes.
- *Multiplicitat*: expressa les restriccions de cardinalitat i participació de la interrelació.

5.3. Interrelacions

5.3.2. Associació (II)

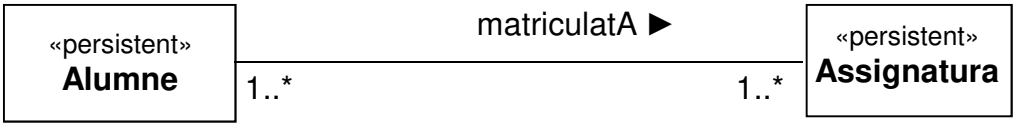
En la multiplicitat de les associacions s'acostuma a utilitzar una notació on s'indica el *mín* i el *màx* (relacionat com a mínim amb *mín* objectes i com a màxim amb *màx* objectes).

Les opcions de multiplicitat són:

0..*	relacionat amb zero o més elements
0..1	relacionat amb cap objecte o com a màxim 1
1..*	relacionat com a mínim amb 1 objecte
1..1	relacionat exactament amb 1 objecte
3..5	relacionat com a mínim amb 3 i com a màxim amb 5
1	igual que 1..1
*	igual que 0..*

5.3. Interrelacions

5.3.2. Associació (III)



On cal emmagatzemar la nota d'avaluació?

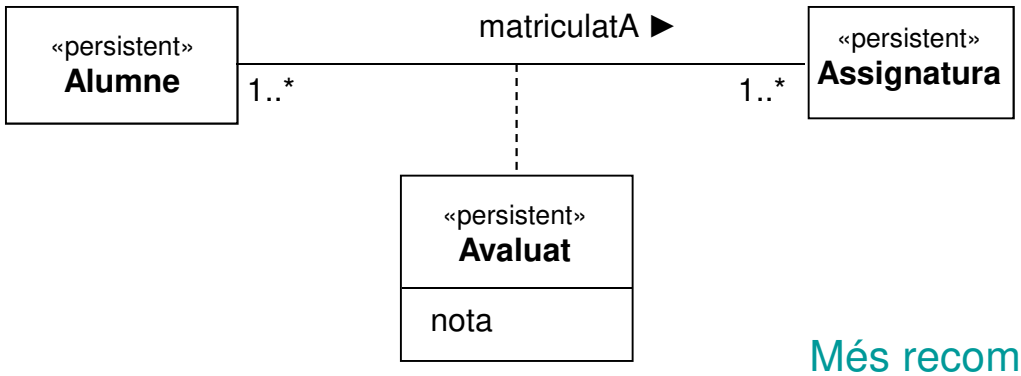
En determinades circumstàncies cal prendre en consideració un atribut que no és de cap de les dues classes associades, aleshores cal associar l'atribut a la interrelació entre les classes. Hi ha dues possibilitats:

- Classes d'associació
- Associacions reificades (mitjançant Reificació)

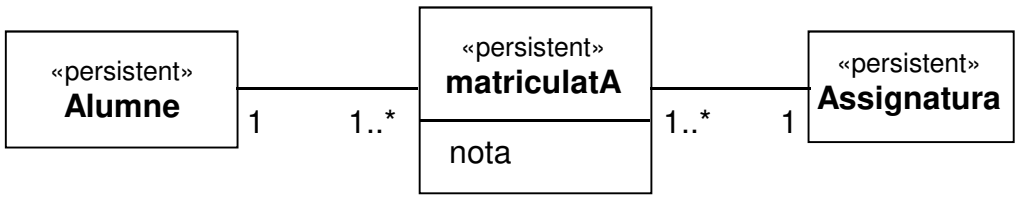
5.3. Interrelacions

5.3.2. Associació (IV)

Amb una classe d'associació ⇐ Més recomanable quan un alumne només matricula una assignatura un cop



Amb una associació reificada ⇐ Més recomanable quan un alumne matricula una assignatura diverses vegades

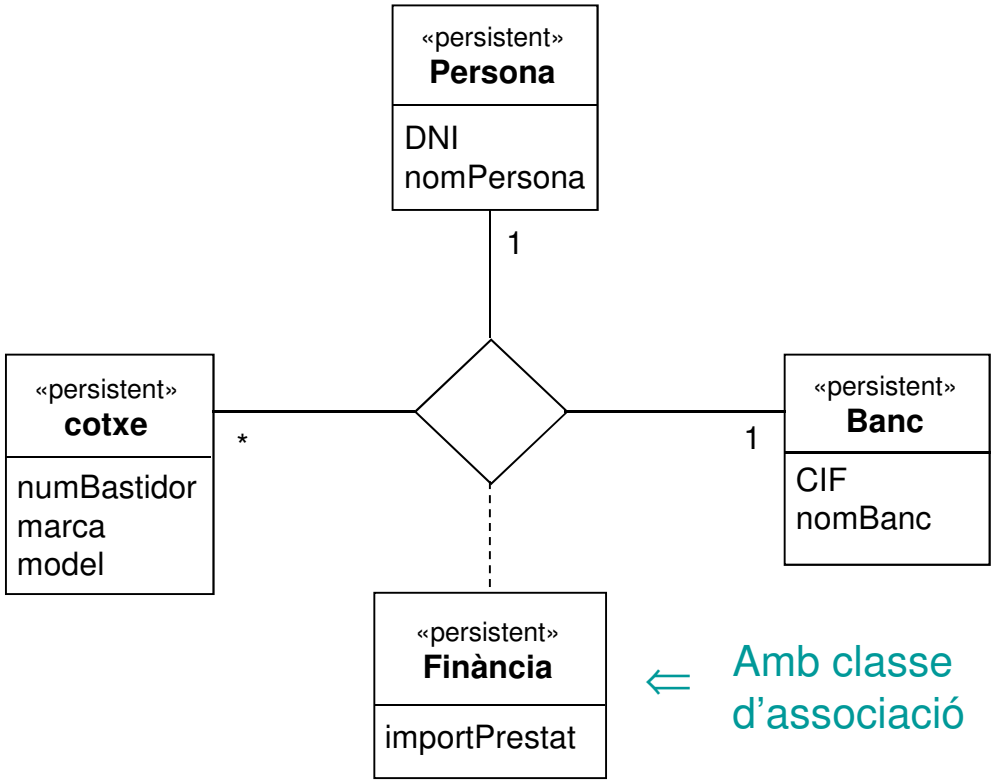


⇒ Gràcies a la reificació s'aconsegueix que una cosa que no s'acostumava a percebre com un objecte s'acabi modelant com una classe.

5.3. Interrelacions

5.3.2. Associació (V)

A part de les associacions binàries es poden modelar *associacions n-àries*.(I)

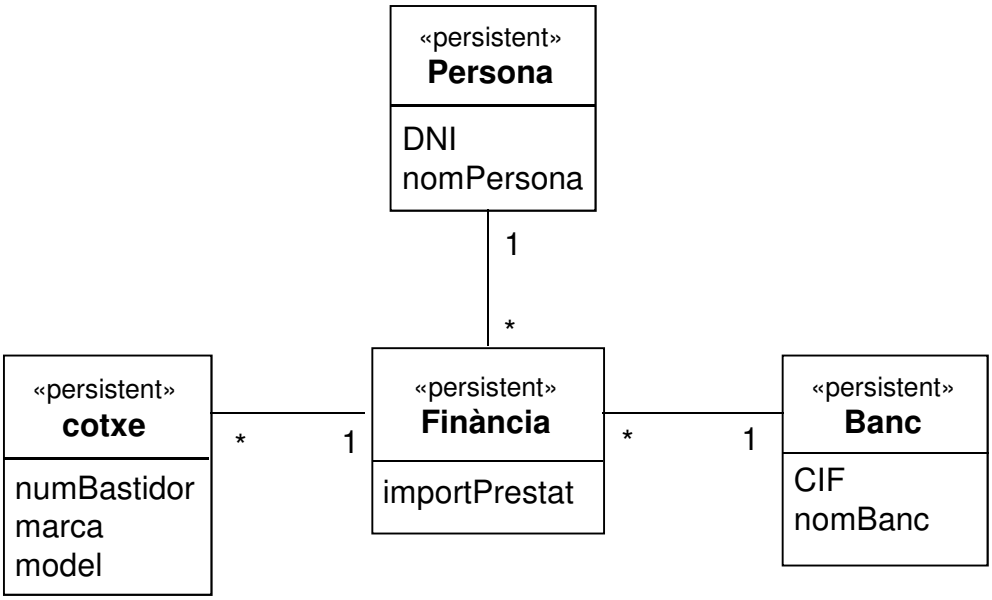


Es representen mitjançant un rombe, amb una línia des del rombe fins a cada classe participant.

5.3. Interrelacions

5.3.2. Associació (VI)

A part de les associacions binàries es poden modelar *associacions n-àries*.(II)



⇑

Amb associació reificada

5.3. Interrelacions

5.3.2. Associació (VII)

Formes addicionals d'associació: (I)

- Associació unidireccional (concepte de *navegabilitat*): permet restringir l'associació de forma unidireccional, així la implementació de l'associació s'emmagatzema només en un sentit.
- Associació d'agregació, permet modelar un objecte, conegut com la part, com a constituent d'un altre objecte (interrelació *tot-part*)
- Associació de composició: es tracta d'una associació d'agregació més forta, on la *part* només pot ésser associada a un únic *tot*.

5.3. Interrelacions

5.3.2. Associació (VIII)

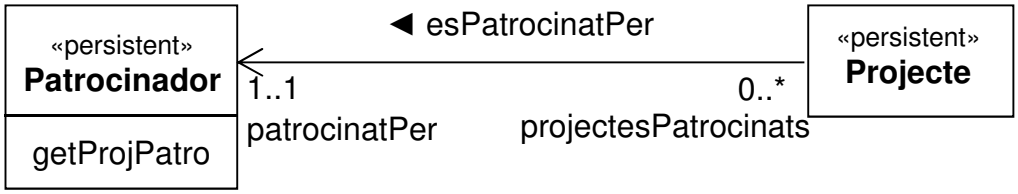
Formes addicionals d'associació: (II)

- Associació qualificada: permet modelar el concepte de taules de consulta o vectors (arrays).
- Restricció d'o exclusiu (xor): quan una classe té diferents associacions amb diferents classes, el xor permet indicar que en algun moment es pot utilitzar només una de les associacions.

5.3. Interrelacions

5.3.2. Associació (IX)

Associació unidireccional



S'afegeix una fletxa a la línia que representa l'associació per tal d'assenyalar la navegabilitat.

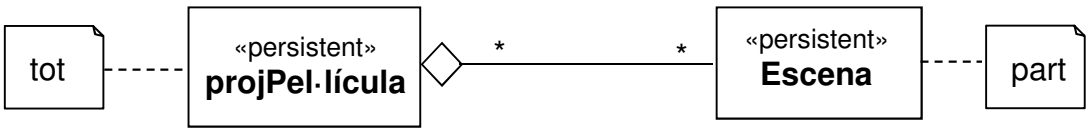
En aquest cas permet accedir de forma més eficient des del projecte al patrocinador que l'ha patrocinat. Per tal de garantir l'accés a la informació en sentit contrari, només cal que existeixi una operació que permeti obtenir les dades corresponents (getProjPatro).

Les associacions unidireccionals només s'utilitzen en associacions binàries que no siguin amb multiplicitat de molts-a-molts.

5.3. Interrelacions

5.3.2. Associació (X)

Associació d'agregació



S'afegeix una rombe blanc a la línia que representa l'associació per tal d'assenyalar la classe que representa el *tot.*

Característiques:

• no es pot posar un rombe al dos costats de la línia.

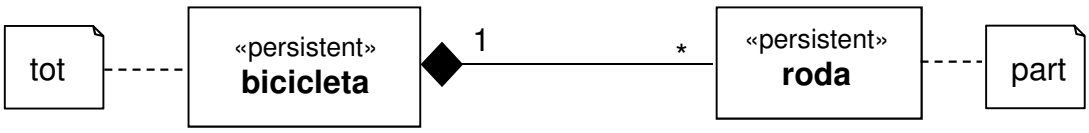
• es pot utilitzar en interrelacions transitives: A és part de B i B és part de C, per tant A és part de C.

• no pot ser antisimètrica: un objecte no pot ser part d'ell mateix. Tot i que una classe pot tenir una associació d'agregació amb ella mateixa per representar el concepte de recursivitat.

5.3. Interrelacions

5.3.2. Associació (XI)

Associació de composició



S'afegeix un rombe negre a la línia que representa l'associació per tal d'assenyalar la classe que representa el *tot.*

Característiques:

• la multiplicitat en el costat del rombe pot ser 1 o 0..1.

• cada instància del tot ha d'estar correctament connectada a les seves parts.

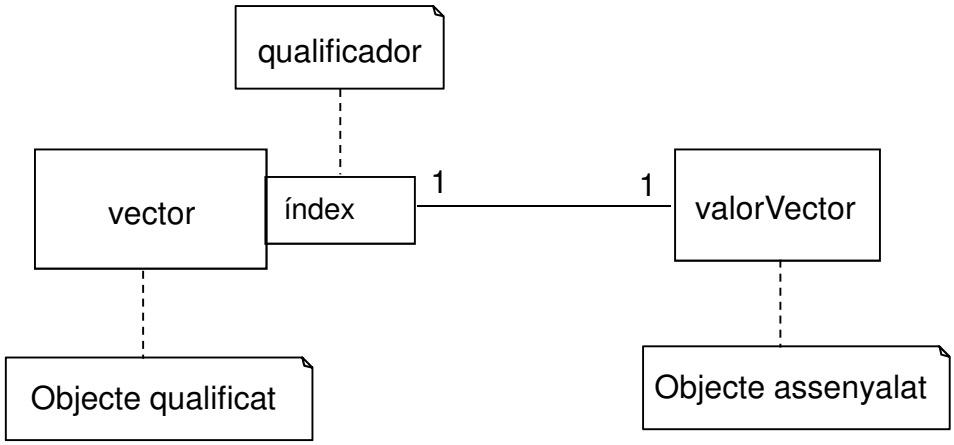
•Si la multiplicitat de la classe composada és 1, en cas que s'esborri un objecte composat caldrà esborrar també les parts o assignar-les a un altre objecte composat.

5.3. Interrelacions

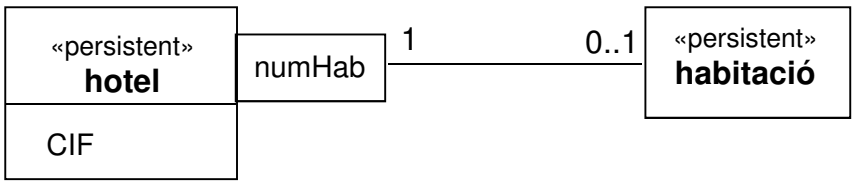
5.3.2. Associació (XII)

Associació qualificada

Notació utilitzada:



Exemple:

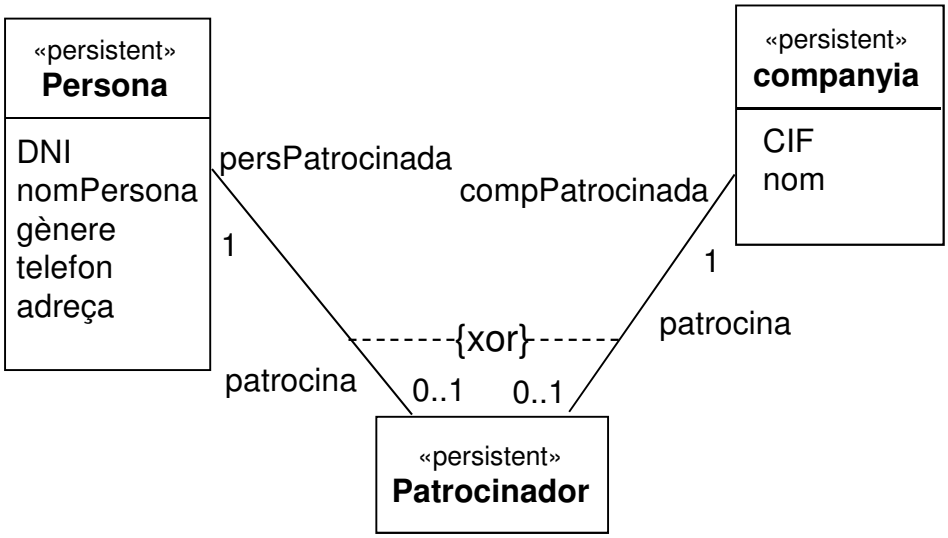


5.3. Interrelacions

5.3.2. Associació (XIII)

Restricció xor

Exemple:

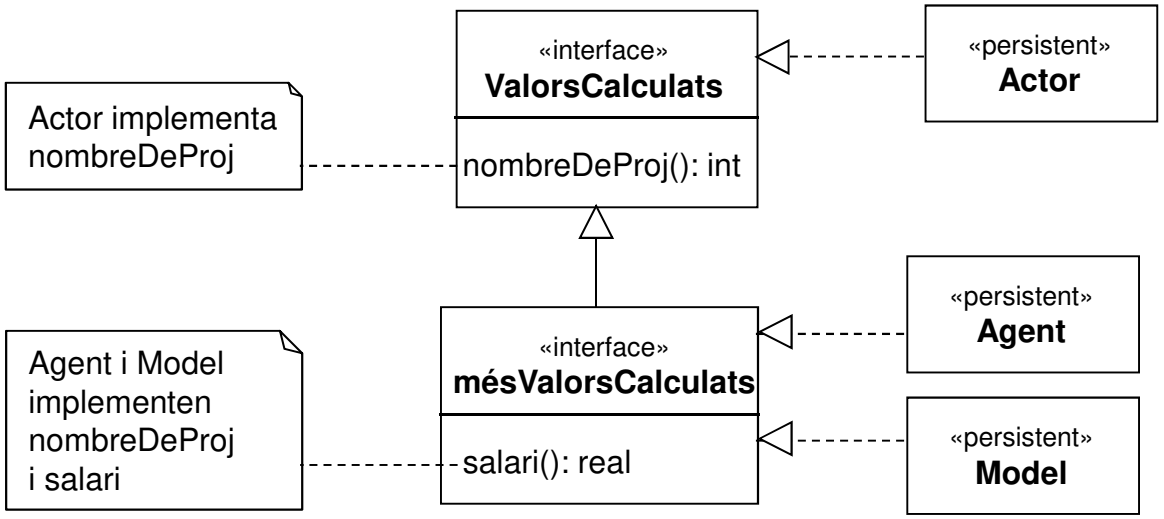


Tot patrocinador patrocina o una Persona o una companyia, encara que cada Persona i cada companyia puguin o no tenir Patrocinador

5.4. Altres

Interrelació de realització i interfície

- La interfície s'utilitza per especificar operacions com a operacions abstractes. Les operacions abstractes permeten demostrar el mateix comportament en classes pertanyents a diferents jerarquies.
- Les classes que necessiten implementar l'operació abstracta de la interfície han de participar en una interrelació de realització amb la interfície.



5.5. Traducció dels diagrames de classes d'UML al model relacional

Classes (I)

Les classes, sempre hi quan no formin part d'una jerarquia, es tradueixen en una relació.

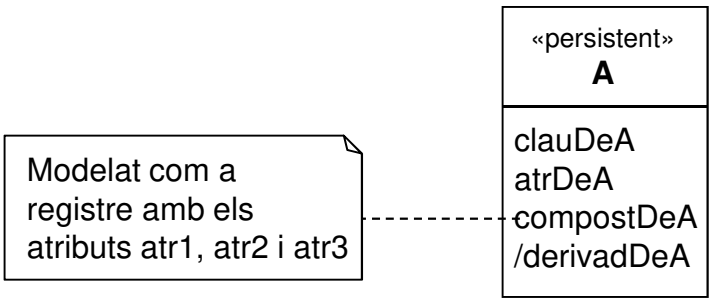
Una classe que té atributs simples, atributs compostos i atributs derivats; es tradueix en una relació que conté els atributs simples i els components simples dels atributs compostos. Pel que fa als atributs derivats hi ha dos opcions:

- Incloure l'atribut derivat com a atribut simple i utilitzar un *trigger* per actualitzar-lo en el moment en que s'actualitzin les dades a partir de les que es deriva.
- Definir una vista que inclogui el càlcul de l'atribut derivat a partir de l'accés a les dades des de les que es deriva.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Classes (II)

Exemple



a(<u>clauDeA</u>, atrDeA, atr1, atr2, atr3, derivadDeA)

i el trigger corresponent

o,

a(<u>clauDeA</u>, atrDeA, atr1, atr2, atr3)

i la vista corresponent

5.5. Traducció dels diagrames de classes d'UML al model relacional

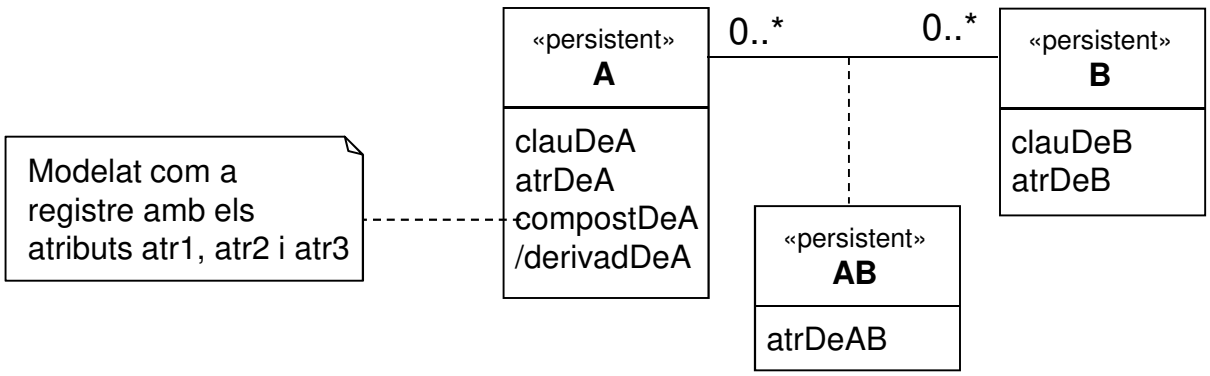
Associacions binàries bidireccionals (I)

- L'associació es tradueix en una relació.
- Com atributs de la relació cal incloure els atributs que formen la clau primària de cada una de les classes relacionades.
- Caldrà explicitar les restriccions de clau forana corresponents a les claus primàries que s'han inclòs a la relació.
- La clau primària de la relació resultant dependrà de la multiplicitat de l'associació i de la semàntica de l'organització.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (II)

Exemple I

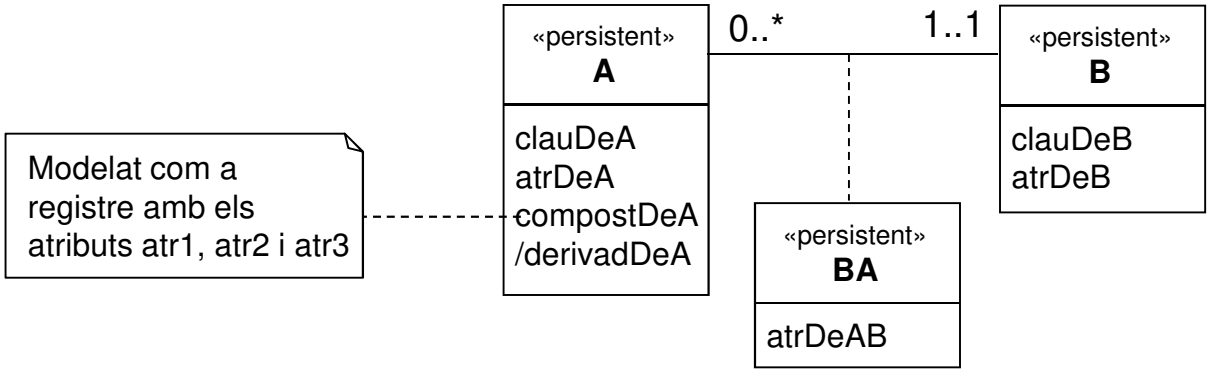


a(clauDeA, atrDeA, atr1, atr2, atr3)
b(clauDeB, atrDeB)
ab(clauDeA, clauDeB, atrDeAB)
foreign key (clauDeA) references a(clauDeA),
foreign key (clauDeB) references b(clauDeB)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (III)

Exemple II

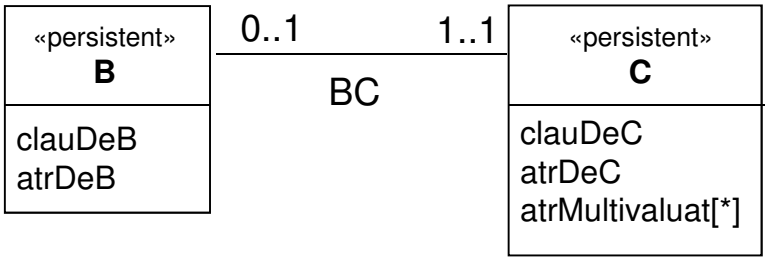


a(clauDeA, atrDeA, atr1, atr2, atr3)
b(clauDeB, atrDeB)
ba(clauDeA, clauDeB, atrDeAB)
foreign key (clauDeA) references a(clauDeA),
foreign key (clauDeB) references b(clauDeB),
constraint notNullB not null (clauDeB)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (IV)

Exemple III

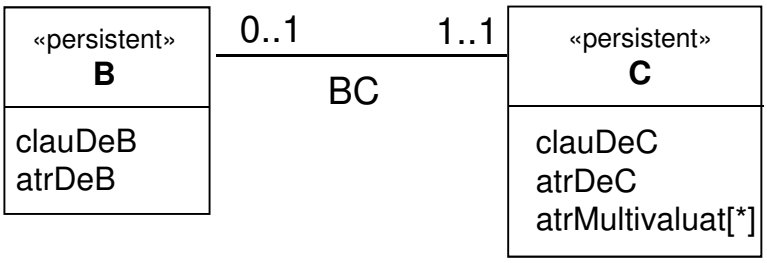


b(<u>clauDeB</u>, atrDeB)
c(<u>clauDeC</u>, atrDeC)
multiC(<u>claudeC,atrMultivaluat</u>)
foreign key (clauDeC) references c(clauDeC)
bc(<u>clauDeB</u>, clauDeC) *(clauDeC també es clau candidata)*
foreign key (clauDeB) references b(clauDeB),
foreign key (clauDeC) references c(clauDeC),
constraint notNullCandidateKey not null (clauDeC),
constraint candidateKeyConstraint unique (clauDeC)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (V)

Exemple III (segueix)



Per garantir la participació total de B en BC (1..1):

```
create assertion totsBaBC
        check (not exists (select * from b
                           where b.clauDeB not in
                                 (select bc.clauDeB from bc)));
```

Si el SGBD no soporta el *check* per múltiples taules, es pot:

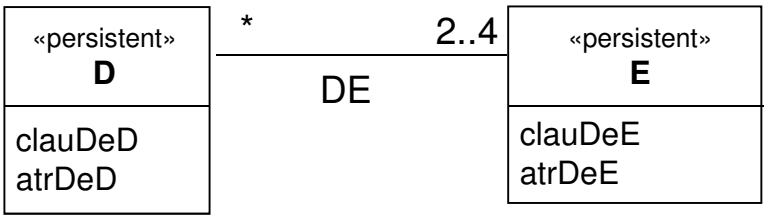
```
create view nototsBaBC as
        select * from b
        where b.clauDeB not in (select bc.clauDeB
                                from bc);
```

(el mateix caldria fer per la participació total de A a AB i a BA)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (VI)

Exemple IV

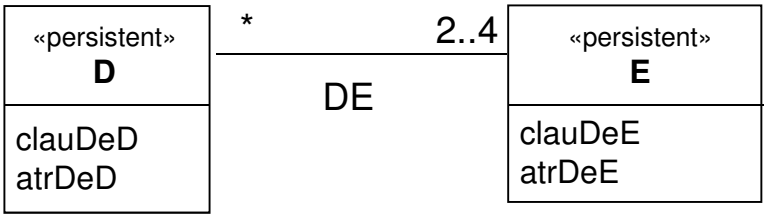


d(<u>clauDeD</u>, atrDeD)
e(<u>clauDeE</u>, atrDeE)
de(<u>clauDeD, clauDeE</u>)
foreign key (clauDeD) references d(clauDeD),
foreign key (clauDeE) references e(clauDeE)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (VII)

Exemple IV (segueix)



Per garantir que D participa a DE entre 2 i 4 vegades:

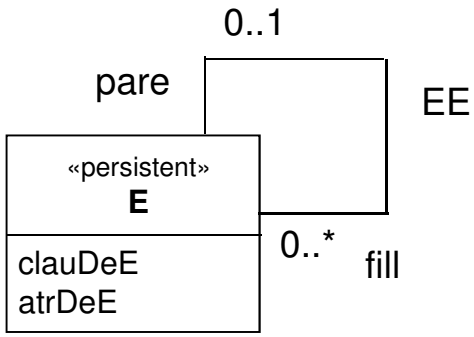
```
create view contarDaDE(clauDeD, conteADE) as
            (select clauDeD, count(*) from de
             group by clauDeD)
             union
            (select clauDeD, 0 from d
             where d.clauDeD not in (select de.clauDeD
                                     from de))

create assertion DaDE
            check (not exists (select *
                               from contarDaDE
                               where conteADE < 2
                                     or conteADE > 4 ));
```

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries bidireccionals (VIII)

Exemple V



e(<u>clauDeE</u>, atrDeE)
ee(<u>clauDeEfill</u>, clauDeEpare)
foreign key (clauDeEfill) references e(clauDeE),
foreign key (clauDeEpare) references e(clauDeE)

5.5. Traducció dels diagrames de classes d'UML al model relacional

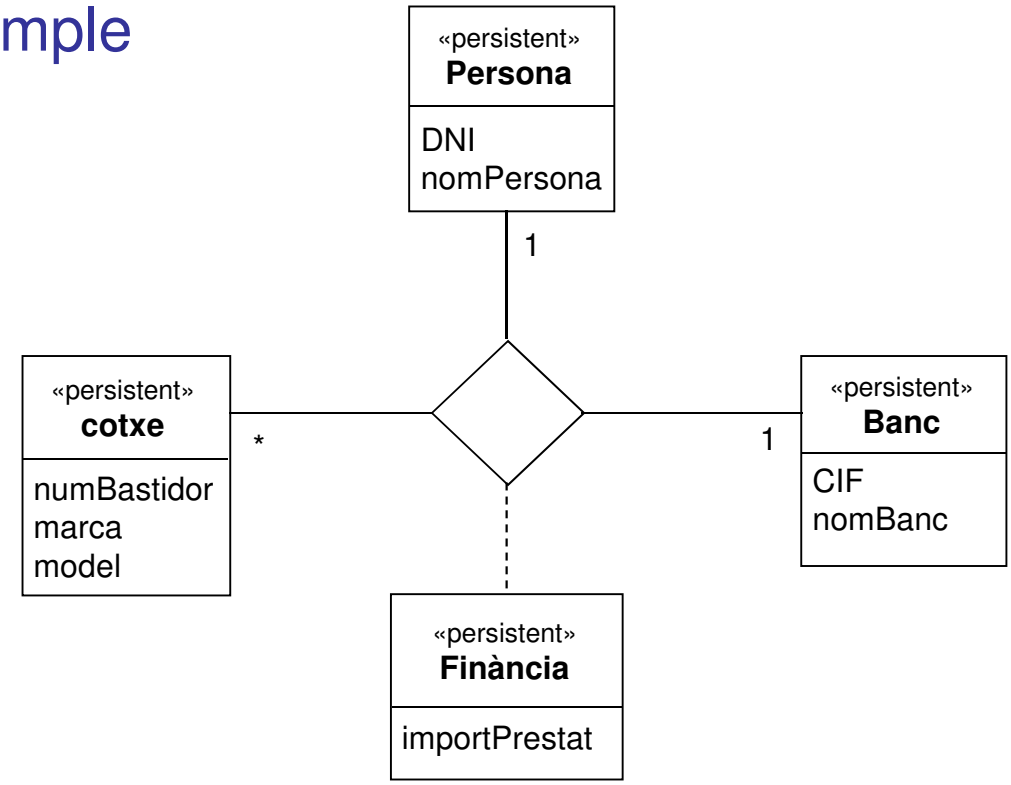
Associacions n-àries (I)

- L'associació es tradueix en una relació igual que si fos binària.
- Com atributs de la relació cal incloure els atributs que formen la clau primària de cada una de les classes relacionades més els atributs de les classes d'associació.
- Caldrà explicitar les restriccions de clau forana corresponents a les claus primàries que s'han inclòs a la relació.
- La clau primària de la relació resultant dependrà de la multiplicitat de l'associació i de la semàntica de l'organització.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions n-àries (II)

Exemple



cotxe(numBastidor, marca, model)
persona(DNI, nomPersona)
banc(CIF, nomBanc)
finància(numBastidor, DNI, CIF, importPrestat)

5.5. Traducció dels diagrames de classes d'UML al model relacional

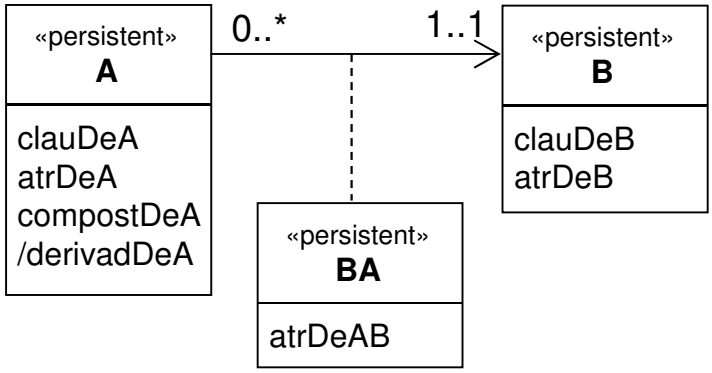
Associacions binàries unidirecionals (I)

- La traducció de l'associació binària bidireccional en una relació, permet navegar d'una classe a l'altra i al inrevés.
- En l'associació binària unidireccional s'indica que la implementació només ha de contemplar un sentit de navegació.
- No es tradueix en una nova relació sinó que s'afegeix la clau primària d'una classe a la relació que s'obté de la traducció de l'altra classe.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries unidirecionals (II)

Exemple (I)

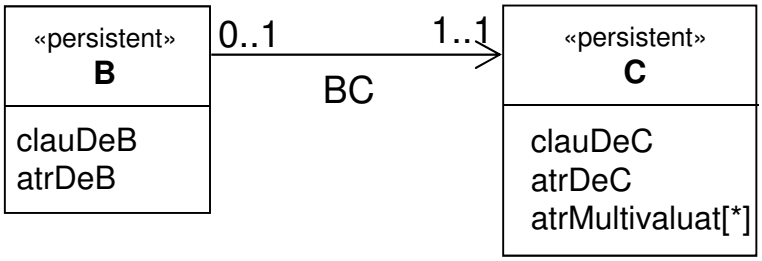


b(<u>clauDeB</u>, atrDeB)
a(<u>clauDeA</u>, atrDeA, atr1, atr2, atr3, clauDeB, atrDeAB)
foreign key (clauDeB) references b(clauDeB),
constraint participacioTotalDeAaBA not null (clauDeB)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associacions binàries unidirecionals (II)

Exemple (II)

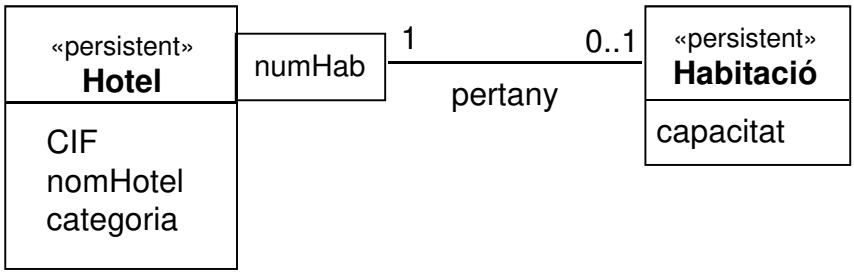


c(<u>clauDeC</u>, atrDeC)
b(<u>clauDeB</u>, atrDeB, *clauDeC*) *(clauDeC també es clau candidata)*
foreign key (clauDeC) references c(clauDeC),
constraint noNullClauCandidata not null (clauDeC),

5.5. Traducció dels diagrames de classes d'UML al model relacional

Associació qualificada

Exemple



Traduida com a unidireccional:

hotel(CIF, nomHotel, categoria)
habitació(CIF, numHab, capacitat)
foreign key (CIF) references hotel(CIF)

Traduida com a bidireccional:

hotel(CIF, nomHotel, categoria)
habitació(CIF, numHab, capacitat)
foreign key (CIF) references hotel(CIF)
pertany(*CIF*, numHab, identificadorClauHotel)
foreign key (CIF) references hotel(CIF)
foreign key (identificadorClauHotel) references hotel(CIF)

5.5. Traducció dels diagrames de classes d'UML al model relacional

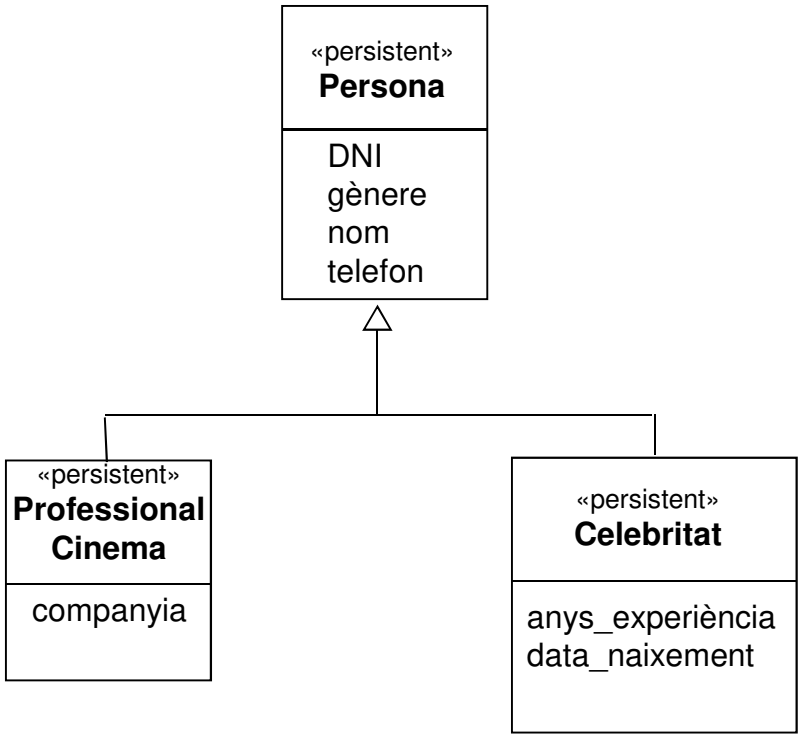
Jerarquies de classes

Hi ha tres formes principals de traduir les jerarquies de classes a relacions:

1. Creant una relació per a cada classe, incloent tant les superclasses com les subclasses.
2. Creant una relació només per a cada subclasse.
3. Aplanant la jerarquia.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Creant una relació per a cada classe (I)



persona(<u>DNI</u>, gènere, nom, telefon)
professional_cinema_BDE(<u>DNI</u>, companyia)
foreign key (DNI) references persona(DNI)
celebritat_BDE(<u>DNI</u>, anys_experiència, data_naixement)
foreign key (DNI) references persona(DNI)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Creant una relació per a cada classe (II)

persona(DNI, gènere, nom, telefon)
professional_cinema_BDE(DNI, companyia)
foreign key (DNI) references persona(DNI)
celebritat_BDE(DNI, anys_experiència, data_naixement)
foreign key (DNI) references persona(DNI)

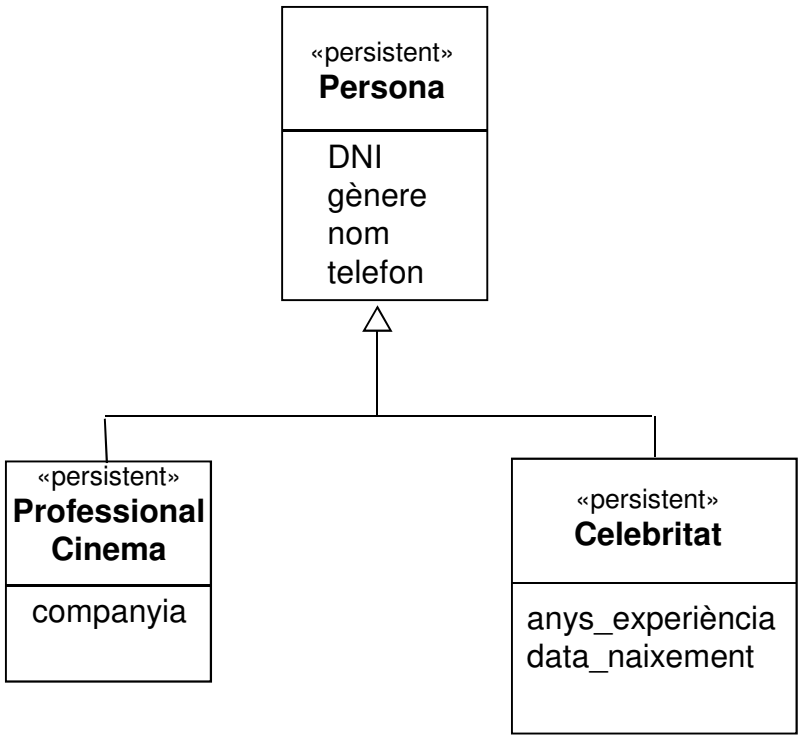
Amb les relacions s'ha definit l'estructura pròpia de cada classe. Cal utilitzar vistes per disposar de les traduccions de les subclasses amb els atributs propis i els heretats:

```
create view professional_cinema as
   select pe.DNI, pe.gènere, pe.nom, pe.telefon,
          pr.companyia
   from persona pe, professional_cinema_BDE pr
   where pe.DNI = pr.DNI;

create view celebritat as
   select pe.DNI, pe.gènere, pe.nom, pe.telefon,
          c.anys_experiència, c. data_naixement
   from persona pe, celebritat_BDE c
   where pe.DNI = c.DNI;
```

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Creant una relació només per a cada subclasse (I)



professional_cinema(DNI, gènere, nom, telefon, companyia)
celebritat(DNI, gènere, nom, telefon, anys_expe, data_naix)

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Creant una relació només per a cada subclasse (II)

professional_cinema(DNI, gènere, nom, telefon, companyia)
celebritat(DNI, gènere, nom, telefon, anys_expe, data_naix)

Amb les relacions s'ha definit l'estructura de cada subclasse, tant amb els atributs especialitzats com els heretats. Cal utilitzar una vista per disposar de la traducció de la superclasse:

```
create view persona as
    select DNI, gènere, nom, telefon
    from professional_cinema
    union
    select DNI, gènere, nom, telefon
    from celebritat;
```

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Creant una relació només per a cada subclasse (III)

Cal tenir present que al no existir físicament la superclasse:

- no es poden suportar especialitzacions parcials.
- si l'especialització no és disjunta hi ha redundància.

$\Rightarrow$ s'utilitza aquesta traducció en especialitzacions disjuntes i totals (la superclasse és abstracta).

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Aplanant la jerarquia en una única relació (I)

- Existirà una única relació on s'emmagatzemarà tota la jerarquia.
- Caldrà que tant els atributs de la superclasse com els de les subclasses apareguin en la relació. A més a més, s'afegirà els atributs de tipus que siguin necessaris per indicar la subclasse a la que correspon cada tupla.
- Els atributs específics d'una subclasse prendran valor nul per a totes les tuples que no corresponguin a la subclasse.
- Es definirà una vista per cada una de les classes de la jerarquia.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Jerarquies de classes – Aplanant la jerarquia en una única relació (II)

la relació seria:

jerarquia_persona(<u>DNI</u>, gènere, nom, telefon, companyia, anys_experiència, data_naixement, subtipus)

i les vistes:

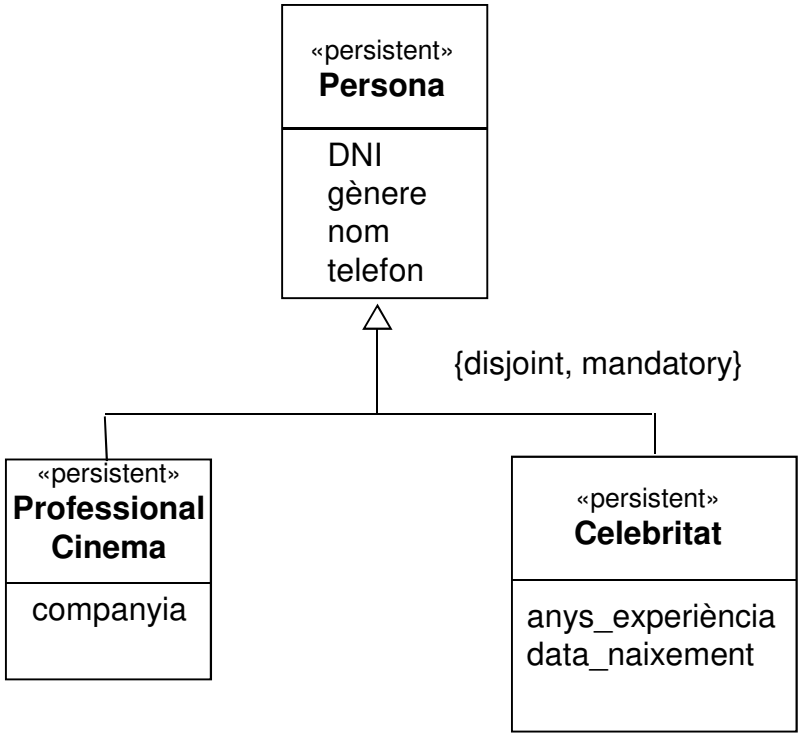
```
create view persona as
    select DNI, gènere, nom, telefon
    from jerarquia_persona;

create view professional_cinema as
   select DNI, gènere, nom, telefon, companyia
   from jerarquia_persona
   where subtipus = 'p';

create view celebritat as
   select DNI, gènere, nom, telefon, anys_experiència,
          data_naixement
   from jerarquia_persona
   where subtipus = 'c';
```

5.5. Traducció dels diagrames de classes d'UML al model relacional

Restriccions d'especialització (I)



5.5. Traducció dels diagrames de classes d'UML al model relacional

Restriccions d'especialització (II)

Restricció "*disjoint*"

Per comprovar que no hi hagi cap tupla que sigui a la vegada professional del cinema i celebritat cal fer:

```
create assertion especialitzacióDisjointPersona as
    check (not exits
    (select *
      from persona pe, professional_cinema pr, celebritat c
      where pe.DNI = pr.DNI and pe.DNI = c.DNI));
```

⇒ així es garanteix que les relacions corresponents a les subclasses siguin disjuntes.

5.5. Traducció dels diagrames de classes d'UML al model relacional

Restriccions d'especialització (III)

Restricció "*mandatory*"

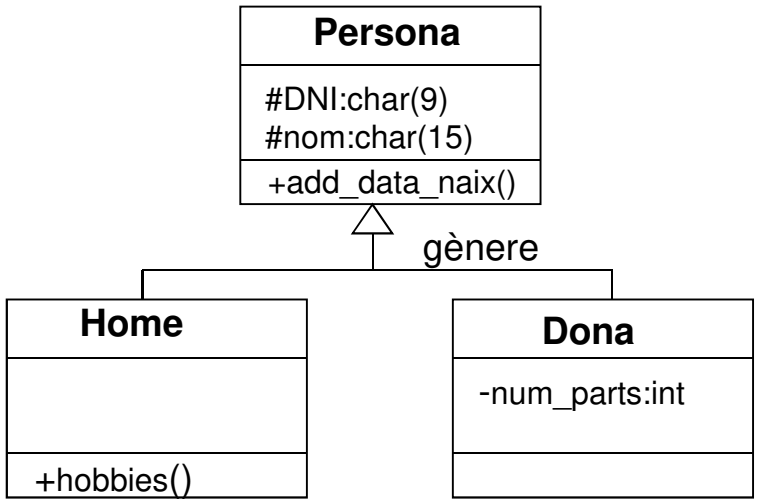
Per comprovar que no hi hagi cap tupla corresponent a la superclasse que no aparegui a cap subclasse:

```
create assertion especialitzacióTotalPersona as
    check (not exits
    (select *
      from persona pe
      where pe.DNI not in (select pr.DNI
                           from professional_cinema pr
                           union
                           select c.DNI
                           from celebritat c );
```

5.5. Traducció dels diagrames de classes d'UML al model relacional

Restriccions d'especialització (IV)

Restricció "*atribut-defined*"



Per quan hi ha discriminador; cal coherència entre el valor de l'atribut i la instanciació dels objectes a les subclasses

```
create assertion especialitzacioAttrdefinedPersona as
   check (not exists
   (select *
    from persona p, home h, dona d
    where ((p.DNI = h.DNI and p.gènere <> 'H') or
              (p.DNI not in (select DNI from home) and p.gènere ='H'))
              or
             ((p.DNI = d.DNI and p.gènere <> 'D') or
              (p.DNI not in (select DNI from dona) and p.gènere='D'));
```