Disseny conceptual de bases de dades

Jordi Casas Roma

PID_00220512

Universitat Oberta de Catalunya

Índex

Introducció

El disseny conceptual de bases de dades és la segona etapa, després de l'anàlisi de requisits en el procés de disseny d'una base de dades. El disseny conceptual permet crear un esquema conceptual d'alt nivell i independent de la tecnologia d'implementació a partir de les especificacions i els requisits d'un problema del món real. En aquest procés cal extreure les necessitats i els requisits dels problemes i poder-los sintetitzar en un model visual que permeti representar les dades i les restriccions dels conceptes que es volen representar en el sistema d'informació.

En aquesta etapa s'obté una estructura de la informació independent de la tecnologia. En aquest moment del disseny encara no es té en compte quin tipus de base de dades s'utilitzarà (relacional, orientada a objectes, etc.). Per tant, tampoc no es té en compte quin sistema gestor de bases de dades s'utilitzarà o amb quin llenguatge concret s'implementarà la base de dades. En aquesta etapa el focus d'atenció se centra en l'estructura de la informació, sense resoldre de moment qüestions lligades a la tecnologia.

L'objectiu d'aquesta etapa és elaborar un model conceptual del problema. Aquest model es representa mitjançant algun model de dades d'alt nivell. Un dels models més coneguts és el model entitat-interrelació (model ER). Aquest model ha tingut una gran rellevància fins als nostres dies. Actualment, però, l'ús del llenguatge unificat de modelització (UML) ha pres una gran força en tots els àmbits de la computació. Inicialment es va desenvolupar per a modelitzar sistemes de programari, però la potència i la flexibilitat que té l'han convertit en un llenguatge ideal per a altres àmbits de la computació.

Concretament, en aquest mòdul explicarem els principis del disseny conceptual i l'aplicació d'aquest disseny mitjançant diagrames en llenguatge UML.

Objectius

En els materials didàctics d'aquest mòdul trobareu les eines indispensables per a assolir els objectius següents:

1. Conèixer els fonaments del disseny conceptual de bases de dades.

2. Entendre els elements bàsics de modelització com a mecanisme de representació conceptual de dades.

3. Estudiar els diagrames de classes UML com a eina per a representar models de dades.

4. Ser capaços de llegir i entendre diagrames de models conceptuals de bases de dades.

5. Aprendre a modelitzar, mitjançant diagrames UML, les necessitats d'un sistema d'informació a partir d'una descripció dels requisits inicials.

1. Introducció al disseny conceptual

El disseny conceptual és la segona etapa en el procés de disseny i implementació d'una base de dades, després de l'etapa inicial de recopilació i anàlisi de requisits. L'objectiu del disseny conceptual és crear un esquema conceptual per a la base de dades mitjançant un model de dades conceptual d'alt nivell i independent de la tecnologia a partir de l'anàlisi de requisits.

Un **esquema conceptual** és una descripció concisa dels requisits de dades per part dels usuaris i inclou descripcions detallades de les entitats que hi estan involucrades, les relacions entre aquestes entitats i les restriccions d'integritat que tenen. S'expressa mitjançant conceptes proporcionats per un model de dades d'alt nivell, que ha de ser fàcil d'entendre i no ha de contenir detalls d'implementació.

1.1. El disseny conceptual

L'etapa de disseny conceptual parteix de la recopilació i l'anàlisi de requisits obtinguda a partir de l'etapa prèvia feta amb els futurs usuaris de la base de dades, i té com a objectiu aconseguir un esquema conceptual de la base de dades que sigui consistent amb els requisits i les restriccions imposats pels problemes que cal resoldre.

L'esquema conceptual, a més, ha de servir com a referència per a verificar que s'han reunit tots els requisits i que no hi ha conflicte entre aquests requisits. Si alguns dels requisits inicials no es poden representar gràficament en el model conceptual cal afegir-los de manera textual, per assegurar-nos que queden recollits en el model conceptual i que es tindran en compte en les fases següents del disseny de la base de dades. Hi ha diferents llenguatges per a representar models conceptuals de dades. Algunes característiques importants per a representar models conceptuals de dades són aquestes:

- **Expressivitat**: el model ha de ser prou expressiu per a permetre representar els diferents conceptes del món real i les associacions entre aquests conceptes.

- **Simplicitat**: el model ha de ser simple. Ha de permetre als usuaris de la base de dades entendre els conceptes que s'hi expressen.

- **Representació diagramàtica**: el model ha de ser una notació diagramàtica fàcil d'entendre i que serveixi per a visualitzar l'esquema conceptual.

- **Formalitat**: la representació del model ha de ser precisa, formal i no pot presentar ambigüitats.

Satisfer aquestes característiques no és fàcil, i sovint algunes característiques entren en conflicte amb les altres.

A part de les característiques comentades, hem dit que un esquema conceptual és una descripció d'alt nivell i independent de la tecnologia. Aquestes dues característiques són importants per a un llenguatge que permeti definir un model conceptual. Hi ha moltes raons que fan que aquestes dues característiques siguin importants en el model conceptual, entre les quals es poden destacar les següents:

- L'ús del model conceptual permet als dissenyadors de bases de dades concentrar-se a expressar les propietats, les associacions i les restriccions de les dades sense preocupar-se dels detalls d'implementació i amb independència de les particularitats dels diferents sistemes gestors de bases de dades (SGBD[1]).

(1)SGBD és la sigla de *sistema gestor de bases de dades*.

- L'esquema conceptual esdevé un element de descripció estable del contingut de la base de dades.

- Un model d'alt nivell independent de les tecnologies específiques de cada SGBD permet més expressivitat i un caràcter més general.

- El model es pot utilitzar com a vehicle de comunicació entre els usuaris, els dissenyadors i els analistes de la base de dades.

1.1.1. Metodologies de disseny

A partir dels requisits recopilats i treballats durant la primera fase del disseny d'una base de dades es poden establir dues metodologies bàsiques per al disseny conceptual de bases de dades:

1) Metodologia centralitzada (***one shot***): en aquesta primera metodologia es fusionen tots els requisits de tots els grups d'usuaris i aplicacions recopilats en la primera fase en un sol conjunt abans de començar la fase de disseny. A partir d'aquest únic conjunt de requisits es desenvolupa un sol esquema conceptual per a tota la base de dades.

2) Metodologia d'integració de vistes: en aquesta metodologia es construeix un esquema (anomenat *vista*) per a cada conjunt de requisits de cada grup d'usuaris i es validen de manera independent. Posteriorment, en una subfase anomenada *integració de vistes* cal fusionar els diferents esquemes o vistes en un únic esquema conceptual global per a tota la base de dades.

La metodologia d'integració de vistes permet crear esquemes conceptuals més petits que són validats pels diferents grups d'usuaris de la base de dades, però afegeix la subfase d'integració de vistes per unificar els diferents esquemes. Aquesta subfase afegeix una complexitat notable i requereix una metodologia i unes eines específiques per a poder-se fer de manera correcta. Generalment, aquesta metodologia se sol aplicar al disseny de grans bases de dades.

En aquest material ens centrarem en la metodologia centralitzada.

1.1.2. Estratègies de disseny

A partir d'un conjunt de requisits, sia per a un únic usuari o per a un conjunt d'usuaris, cal dissenyar un esquema conceptual que satisfaci els requisits imposats. Per a dur a terme aquesta tasca, es poden utilitzar diferents estratègies. La majoria utilitzen un mètode incremental, és a dir, comencen per modelitzar algunes estructures principals derivades dels requisits i les van modificant i refinant en diferents iteracions del mateix procés.

Algunes de les estratègies més utilitzades per al disseny conceptual de bases de dades són les següents:

a) Estratègia descendent: es comença el procés amb un esquema que conté abstraccions d'alt nivell i es van aplicant successivament refinaments sobre aquestes abstraccions.

b) Estratègia ascendent: es comença el procés amb un esquema que conté els detalls de les abstraccions i es van combinant de manera successiva formant conjunts de més entitat o conceptes complexos.

c) Estratègia de dins cap a fora: és un cas concret de l'estratègia ascendent. Aquesta estratègia centra l'atenció en un conjunt central de conceptes que són molt evidents, i de mica en mica, inclou en l'esquema nous conceptes relacionats directament amb els conceptes existents.

d) Estratègia mixta: aquesta estratègia divideix els requisits segons una estratègia descendent i llavors construeix cadascuna de les divisions seguint una estratègia ascendent. Finalment es combinen les parts per a generar l'esquema complet.

1.2. El model ER

El **model entitat-interrelació**, o **model ER**, és un model conceptual de dades d'alt nivell i independent de la tecnologia. Aquest model i les variacions que té constitueixen els models més utilitzats pel disseny conceptual de les aplicaci-

ons de bases de dades. Això és deu, principalment, a la simplicitat i la facilitat d'ús. Els principals elements que inclou el model són les entitats, els atributs i les relacions entre entitats.

L'objectiu principal del model ER és permetre als dissenyadors reflectir en un model conceptual els requisits del món real que siguin d'interès per al problema.

Model ER

El nom original prové de la llengua anglesa, en què les sigles *ER* signifiquen *entity-relationship*. En llengua catalana trobem autors que ho traueixen com a *model entitat-relació* i d'altres que ho traueixen com a *model entitat-interrelació*. Tots dos es refereixen al mateix model.

Aquest model facilita el disseny conceptual d'una base de dades i és aplicable al disseny de qualsevol tipus de bases de dades (relacionals, orientades a objectes, etc.), ja que en l'etapa del disseny conceptual no es té en compte encara la tecnologia concreta que s'emprarà per a implementar la base de dades.

Peter Pin-Shan Chen

És científic i professor d'informàtica a la Universitat Estatal de Louisiana. Va obtenir el grau de doctor a la Universitat de Harvard el 1973. Posteriorment, el 1976, va desenvolupar el model ER, fet pel qual és conegut.

El model ER té l'origen en els treballs fets per Peter Chen el 1976. Posteriorment, molts altres autors han proposat variants i ampliacions per a aquest model. Per tant, en la literatura es poden trobar variacions en el model ER que poden diferir simplement en la notació diagramàtica o en alguns conceptes en què es basen per a modelitzar les dades.

El model ER permet reflectir aspectes relacionats amb l'estructura de les dades i de la integritat d'aquestes dades.

Per a representar el model ER s'ha emprat tradicionalment el **diagrama entitat-interrelació**, o **diagrama ER**. Aquest diagrama defineix una nomenclatura específica per a representar els diferents conceptes d'entitats i relacions que requereix el model. Tot i l'ampli ús d'aquest diagrama, darrerament es tendeix a emprar el llenguatge UML per a representar el model ER. En aquest text utilitzarem el llenguatge UML. Cal tenir en compte, però, que els conceptes són els mateixos i només canvia la manera de representar els conceptes relacionats amb les entitats, els atributs o les relacions.

1.3. El llenguatge UML

El **llenguatge unificat de modelització**[2] **(UML)** és un llenguatge de propòsit general per a modelitzar sistemes de programari. L'estàndard el va crear, i actualment el manté, l'Object Management Group. Es va afegir per primera vegada a la llista de tecnologies emprades per l'OMG el 1997, i des de llavors s'ha convertit en l'estàndard de la indústria per a modelitzar sistemes de programari.

[(2)]En anglès, *unified modeling language* (UML).

Object Management Group

L'Object Management Group (OMG) és un consorci dedicat a establir i promoure diverses especificacions de tecnologies orientades a objectes, com UML, XMI o CORBA. És una organització sense ànim de lucre que promou l'ús de la tecnologia orientada a objectes mitjançant guies i especificacions per a aquestes guies.

UML és un llenguatge gràfic dissenyat per a especificar, visualitzar, modificar, construir i documentar un sistema. Permet una visualització estàndard de diferents artefactes, entre altres, activitats, actors, lògiques de negoci i esquemes de bases de dades.

El llenguatge UML el va desenvolupar la companyia Rational Software Corporation durant els anys noranta. Després d'un període en què coexistien diferents tendències en la modelització orientada a objectes, la companyia unifica els esforços de tres pioners en aquesta àrea: James Rumbaugh, Grady Booch i Ivar Jacobson. El 1996 es crea el consorci UML Partners amb l'objectiu de completar l'especificació d'UML. El gener del 1997 es publica la versió 1.0 d'UML. Durant el mateix any es publica la versió 1.1, que accepta i adopta el consorci OMG. Finalment, la versió 2.0 es publica el 2005. En el moment d'escriure aquest material, la darrera versió és la 2.3, publicada el maig del 2010.

El llenguatge UML incorpora una gran quantitat de diagrames que permeten representar el model d'un sistema des de diferents perspectives. Podem trobar diagrames que fan referència a l'estructura (per exemple, diagrames de classe, diagrames de components o diagrames de paquets), diagrames referents al comportament (per exemple, diagrames d'activitat o diagrames de casos d'ús) o diagrames referents a la interacció (per exemple, diagrames de comunicació o diagrames de seqüència).

UML defineix nou tipus de diagrames dividits en dues categories:

- **Diagrames estructurals**: descriuen les relacions estructurals o estàtiques entre els diferents components del sistema.

- **Diagrames de comportament**: descriuen les relacions de comportament o dinàmiques entre els diferents components del sistema.

Pel que fa al disseny conceptual de bases de dades, ens interessa especialment el diagrama de classes, que permet representar informació del domini de discurs. Tot i això, a continuació veurem una breu descripció de cadascun d'aquests diagrames, tot i que l'abast dels diferents diagrames UML cau fora dels objectius d'aquest text.

Domini del discurs

El domini del discurs, univers del discurs, o simplement domini, és el conjunt de coses de què es parla en un determinat context.

Diagrames estructurals:

a) Els **diagrames de classes** són diagrames estàtics que descriuen l'estructura d'un sistema a partir de les classes del sistema, els atributs d'aquest sistema i les relacions que hi ha (també conegudes com a *associacions* en terminologia UML). Aquests diagrames són un dels principals blocs en el desenvolupament orientat a objectes, però també han demostrat una capacitat excel·lent per a modelitzar dades. Per aquest motiu, han esdevingut cada vegada més importants en el disseny conceptual de bases de dades.

b) Els **diagrames d'objectes** mostren les instàncies (o objectes del món real) i les relacions que tenen en un moment concret. Les instàncies d'un sistema es modifiquen al llarg de temps, és a dir, es creen instàncies noves, es destrueixen instàncies i es modifiquen certs valors de determinades instàncies. Per tant, els

diagrames d'objectes són com una fotografia que mostra una vista estàtica de les diferents instàncies d'un sistema i de les relacions entre aquestes instàncies en un moment de temps determinat.

c) Els **diagrames de components** il·lustren les organitzacions i les dependències entre els components del sistema. En entorns de bases de dades s'utilitzen per a modelitzar els espais de taula o les particions.

Vegeu també

Els espais de taula (*tablespaces*) es veuen en el subapartat 5.2 del mòdul "Disseny físic de bases de dades" d'aquesta assignatura.

d) Els **diagrames d'implantació** representen la distribució de components del sistema i la relació que tenen amb els components del maquinari disponible.

Diagrames de comportament:

a) Els **diagrames de casos d'ús** s'utilitzen per a modelitzar les interaccions funcionals entre els usuaris i el sistema.

b) Els **diagrames de seqüència** descriuen les interaccions entre diversos objectes en el transcurs del temps. Mostren el flux temporal de missatges entre diversos objectes.

c) Els **diagrames de col·laboració** representen les interaccions entre objectes com una sèrie de missatges en seqüència. Aquests diagrames centren l'atenció en l'organització estructural dels objectes que envien i reben missatges.

d) Els **diagrames d'estat** descriuen com canvia l'estat d'un objecte en resposta a diferents esdeveniments externs.

e) Els **diagrames d'activitat** presenten una vista dinàmica del sistema i modelitzen el flux de control d'activitat a activitat.

2. Elements bàsics de modelització

Un model conceptual permet definir informació de domini a partir de tipus d'entitats, tipus de relacions, atributs i restriccions d'integritat.

2.1. Tipus d'entitats

Entenem per **entitat** un objecte del món real, que té identitat pròpia i que és distingible de la resta d'objectes

Les entitats poden ser objectes amb existència física (per exemple, un cotxe o una persona) o objectes amb existència conceptual (per exemple, una feina o una assignatura d'un curs universitari).

Exemples d'entitats

Alguns exemples d'entitat són una poma, un cotxe o una persona. Altres exemples, que no són tangibles però que tenen identitat i són distingibles de la resta, són una comanda a un proveïdor, una petició de préstec en una biblioteca o una incidència municipal.

Entenem per **tipus d'entitat** l'abstracció que permet definir el conjunt d'objectes amb les mateixes propietats, comportament comú, semàntica comuna i idèntica relació amb els altres objectes.

El procés per a passar d'un conjunt d'entitats a un tipus d'entitat s'anomena **abstracció**. Aquest procés consisteix a eliminar les diferències o distincions entre les entitats per a poder observar aspectes comuns a totes aquestes entitats.

Exemples d'entitats

El tipus d'entitat *cotxe* es defineix com el concepte de *vehicle*, que descriu les parts comunes de diferents cotxes amb independència de la marca, el color, el model i altres característiques concretes. Un cotxe concret, per exemple el meu cotxe, és una instància o ocurrència del tipus d'entitat *cotxe*.

Utilitzarem la nomenclatura següent per a diferenciar aquests dos conceptes:

- El concepte que hem definit com a *entitat* també pot ser referenciat pels termes *objecte*, *instància* o *ocurrència*.
- El concepte que hem definit com a *tipus d'entitat* també pot ser referenciat pels termes *classe*, *entitat tipus* o *tipus*.

Terminologia UML

En la representació en diagrames UML, els tipus d'entitat s'anomenen *classes* i les entitats, *objectes*.

En la literatura es poden trobar autors que fan ús de les diferents terminologies, i per tant, cal tenir-les totes presents per a identificar de manera ràpida i unívoca a quin concepte es fa referència quan s'utilitza qualsevol d'aquests termes.

2.2. Atributs

Un **atribut** és una propietat que tenen totes les entitats d'un mateix tipus d'entitat i que permet representar-ne les característiques.

Atributs i relacions binàries

En teoria, un atribut no és més que un tipus de relació binària en què un dels participants és un tipus de dada (vegeu el subapartat següent). Això no obstant, per a simplificar el model i facilitar-ne la creació i la llegibilitat es tracten de manera diferent i més compacta.

Per a representar gràficament els tipus d'entitat i els seus atributs utilitzarem els diagrames de classes del model UML.

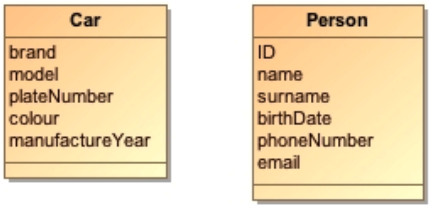
Diagrama de classes

El tipus d'entitat 'cotxe' té un conjunt d'atributs; per exemple: marca, model, matrícula, color i any de fabricació. Una entitat de *cotxe* té valors específics per a cadascun d'aquests atributs. Per exemple, un cotxe pot ser de la marca Fiat, model Punto, matrícula 3621-GHV, color negre i fabricat el 2010.

De manera similar, el tipus d'entitat 'persona' té un conjunt d'atributs; per exemple: identificació (en el cas d'Espanya és el document nacional d'identitat o DNI), nom i cognoms, data de naixement, número de telèfon i correu electrònic. Una entitat de 'persona' pot ser, per exemple, una persona amb DNI 33551058D, nom Fred i cognom Smith, nascut l'1 de març del 1968, amb número de telèfon 85542154 i correu electrònic fredsmith@mydom.com.

La figura 1 mostra un exemple de representació dels tipus d'entitat 'cotxe' (*Car*) i 'persona' (*Person*).

Figura 1. Exemples dels tipus d'entitat 'cotxe' (*Car*) i 'persona' (*Person*)



En els diagrames de classes del model UML representarem els tipus d'entitat (anomenats *classes* en els diagrames UML) mitjançant un rectangle amb tres seccions. En la primera secció indicarem el nom del tipus d'entitat. Els atributs es representen com una llista en la segona secció, i la tercera secció permet representar certes restriccions sobre les dades o operacions, que no utilitzarem en el disseny conceptual de bases de dades.

La taula 1 mostra un possible exemple d'un conjunt d'entitats de persona.

Taula 1. Exemple de representació d'entitats del tipus d'entitat 'persona' (*Person*)

ID	name	surname	birthDate	phoneNumber	email
33941857B	John	Smith	12/10/1987	936542798	johnsmith@exemple.com
77854623Q	Mary	Jane	17/02/1972	696275345	maryjane@exemple.com

ID	name	surname	birthDate	phoneNumber	email
96725466Z	Fred	Sanchez	07/09/2001	649785467	fredsanchez@exemple.com

2.2.1. Representació dels atributs

Els diagrames UML presenten una nomenclatura concreta per a especificar les propietats de cadascun dels atributs:

```
visibilitat nom [etiquetes] [: tipus] [multiplicitat] [= valor inicial]
```

Definim cada element:

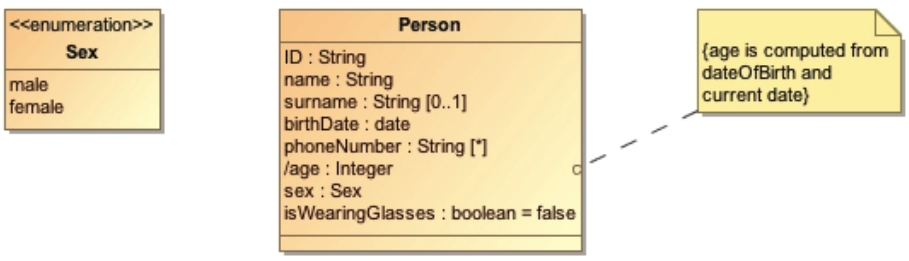
- Visibilitat: visibilitat de l'atribut (*public*, *protected*, *package* o *private*). Normalment aquest concepte no s'aplica en disseny conceptual de bases de dades.
- Nom: denominació de l'atribut.
- Etiquetes: etiquetes opcionals per a indicar certs conceptes relacionats amb l'atribut.
- Tipus: tipus o domini de les dades.
- Multiplicitat: nombre d'ocurrències de l'atribut.
- Valor inicial: valor per defecte.

Com ja s'ha vist en alguns exemples, es poden utilitzar notes associades als tipus d'entitats o atributs per a indicar restriccions especials o qualsevol casuística que el dissenyador consideri oportú de remarcar.

Tipus d'entitat

La figura 2 mostra un tipus d'entitat en què podem veure un atribut marcat a opcional (*surname*), un atribut derivat (*age*), un atribut de tipus enumeració (*sex*), un atribut amb valor inicial especificat (*isWearingGlasses*) i un atribut multivalor (*phoneNumber*). Una nota associada a l'atribut *age* indica la fórmula de càlcul d'aquest atribut derivat.

Figura 2. Exemple del tipus d'entitat 'persona' (*Person*)



Atributs de tipus enumeració

Un atribut de tipus enumeració presenta un conjunt tancat de valors que l'atribut pot prendre. Per exemple, es poden definir els dies de la setmana com un atribut de tipus enumeració, en què el conjunt de possibles valors és dilluns, dimarts, dimecres, dijous, divendres, dissabte i diumenge.

2.2.2. Domini dels atributs

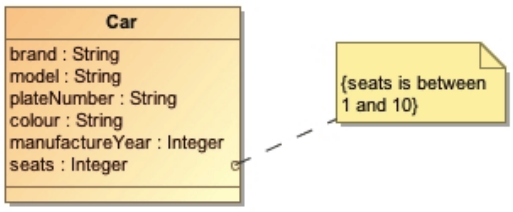
El conjunt de possibles valors legals que pot prendre un atribut s'anomena **domini de l'atribut**. Aquest conjunt restringeix els possibles valors que pot prendre l'atribut en qualsevol de les entitats. Generalment s'especifiquen a partir dels tipus de dades bàsics disponibles en la majoria dels llenguatges de programació, com enter, real, cadena de text, booleà o tipus enumerat.

Domini dels atributs

La figura 3 mostra el tipus d'entitat 'cotxe' (*Car*), en què s'especifica el domini per a cadascun dels atributs que té.

En aquest exemple hem definit un nou atribut per a indicar el nombre de places d'un cotxe (*seats*), que correspon al domini dels enters i que està limitat per un valor mínim d'1 plaça i un valor màxim de 10 places (per a aquest exemple es considera que no hi ha cotxes que tinguin un nombre de places superior a 10). Per a indicar aquesta restricció sobre el domini en UML es poden utilitzar les notes textuals o recórrer a un dels llenguatges que permeten definir restriccions sobre els models UML, com per exemple OCL.

Figura 3. Exemple dels tipus d'entitat cotxe (*Car*) amb el domini dels atributs



Reflexió

L'ús de llenguatges com OCL, per la seva complexitat, cau fora de l'abast d'aquest text, i per tant, aquí utilitzarem una nota que indiqui de manera textual la restricció associada a l'atribut en qüestió.

Object Constraint Language (OCL)

L'OCL és un llenguatge declaratiu que permet definir regles que s'apliquen a models UML. Entre d'altres, permet descriure restriccions sobre el domini dels atributs. En aquest mòdul no utilitzarem aquest llenguatge, ja que comporta una dificultat afegida, i utilitzarem les notes textuals per a indicar restriccions en el domini i consideracions similars.

2.2.3. Atributs compostos i atributs atòmics

Un **atribut compost** és aquell que es pot dividir en atributs més bàsics amb significat independent. Un **atribut atòmic** o **simple** és aquell que no és divisible sense perdre el significat.

Atribut compost

Un exemple d'atribut compost és el concepte d''adreça postal': l'atribut d'adreça postal amb valor "Avinguda Santa Cristina, 75, 08280 Barcelona" es pot descompondre en els atributs *carrer*, *número*, *codi postal* i *ciutat*. D'aquesta manera, cada nou atribut manté significat independent i el conjunt de tots plegats identifica una adreça postal. Un exemple d'atribut simple és el nom d'una població. Encara que aquest atribut sigui compost, per exemple "Arenys de Mar", no es pot dividir sense perdre el significat original.

En llenguatge UML, els atributs compostos es representen mitjançant un nou tipus d'entitat.

2.2.4. Atributs monovalor i atributs multivalor

Un atribut monovalor és aquell que té un sol valor per a cada entitat. En contraposició a aquest concepte, un atribut multivalor és aquell que pot tenir més d'un valor per a la mateixa entitat.

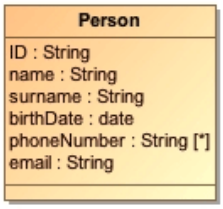
Atributs monovalor i multivalor

Un exemple d'atribut monovalor és l'atribut 'data de naixement' d'una persona, ja que una persona només pot haver nascut en una única data.

Un exemple d'atribut multivalor és l'atribut 'telèfon' d'una persona, ja que es pot donar el cas que una persona tingui zero, un o més d'un telèfon (per exemple, un telèfon fix i un telèfon mòbil).

La figura 4 mostra el tipus d'entitat 'persona' (*Person*) amb l'atribut 'telèfon' convertit en atribut multivalor.

Figura 4. Exemple del tipus d'entitat 'persona' (*Person*) amb un atribut multivalor



Per a indicar la **cardinalitat** d'un atribut multivalor utilitzarem:

- Un número per a indicar el nombre exacte de valors que ha de tenir l'atribut.
- L'interval *min..max* per a indicar el nombre mínim i màxim de valors que pot tenir l'atribut.
- El símbol * per a indicar que l'atribut pot tenir indefinits valors (zero o més).

Significat del símbol *

Per a indicar que l'atribut 'telèfon' (*phoneNumber*) del tipus d'entitat 'persona' (*Person*) de la figura 4 pot tenir entre 0 i 5 valors cal indicar "phoneNumber: String[0..5]" en la definició del tipus d'entitat.

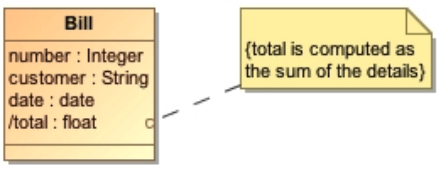
2.2.5. Atributs derivats

Un **atribut derivat** és aquell que es calcula a partir d'altres atributs. Per tant, un atribut derivat porta associat un procediment de càlcul per a obtenir el valor a partir d'altres atributs, que poden ser derivats o no. Un atribut no derivat és aquell al qual cal assignar un valor.

Atributs derivats

Suposem la classe 'factura' (*Bill*) amb els atributs 'número de factura' (*number*), 'client' (*customer*), 'data de la factura' (*date*) i 'import total de la factura' (*total*). Per a obtenir la informació de l'import total de la factura, hem de crear l'atribut 'import total' (*total*) com a atribut derivat, calculat a partir de la suma de cadascuna de les línies dels detalls de la factura, tal com es mostra en la figura 5.

Figura 5. Exemple d'atribut derivat



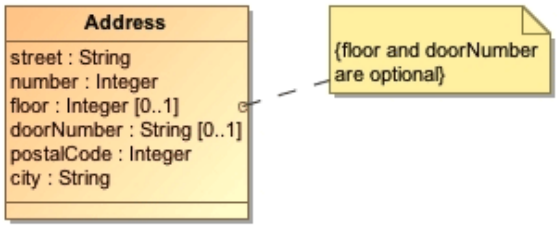
2.2.6. El valor NULL

El valor NULL és un valor especial dissenyat per a indicar que un atribut és desconegut o no s'aplica en una entitat concreta. Per a cada atribut, es pot indicar si accepta un valor NULL o no. En cas que s'especifiqui que l'atribut no accepta un valor NULL, totes les entitats han de donar un valor per a aquest atribut.

Atribut amb valor NULL

La figura 6 mostra el tipus d'entitat 'adreça' (*Address*), que permet definir l'adreça física d'una casa, un pis o un local. Cal fer notar que en les cases unifamiliars no té sentit l'atribut de 'número de pis i porta'. Els atributs que indiquen el 'número de pis' (*floor*) i 'porta d'escala' (*doorNumber*) especifiquen una multiplicitat de 0 o 1, és a dir, que per a cada entitat poden tenir un valor associat o tenir valor no definit (NULL).

Figura 6. Exemple d'atribut amb valor NULL



2.3. Claus

Una **clau candidata** és un conjunt d'atributs mínim que permet identificar de manera única totes les entitats d'un tipus d'entitat.

Un tipus d'entitat pot tenir una o més claus candidates. És necessari, però, que tingui almenys una clau candidata que permeti identificar de manera unívoca totes les entitats.

Una clau candidata pot estar formada per un o més atributs. En el cas d'estar formada per diversos atributs rep el nom de *clau candidata composta*. En aquest cas, la combinació dels diferents valors dels atributs ha de ser única per a identificar de manera unívoca a totes les entitats d'un mateix tipus d'entitat. Una clau candidata composta ha de ser mínima, és a dir, ha d'incloure el nombre mínim d'atributs que permetin identificar de manera unívoca les entitats. La definició de clau candidata es pot veure com una restricció sobre les dades, ja que prohibeix que dues entitats tinguin el mateix valor en els atributs que formen la clau candidata.

El dissenyador de la base de dades escull una de les claus candidates per a ser la clau primària. Aquesta és la clau escollida per a identificar les entitats de manera unívoca. Com a clau candidata no pot contenir valors duplicats, però a més a més, de manera implícita, no pot contenir valors NULL en cap entitat.

En UML no hi ha cap notació per a indicar les claus candidates ni la clau primària. En aquest text prenem un conveni per a indicar les claus candidates i les claus primàries. Les claus candidates es marquen amb l'etiqueta < *Un* >, en què n ($n > 0$) agrupa cadascun dels conjunts de claus candidates. Per tant, si dos o més atributs tenen la mateixa etiqueta, per exemple < *U*1 >, entenem que es tracta d'una clau candidata composta per tots els atributs amb aquesta etiqueta. La clau primària es marca amb l'etiqueta < *P* > per a distingir-la de la resta de claus.

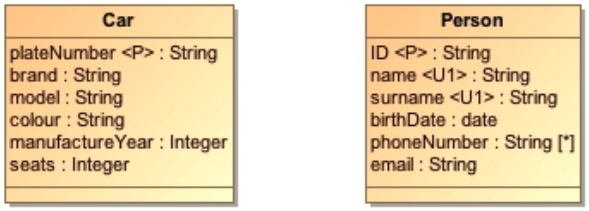
Altres autors s'estimen més indicar les claus amb l'etiqueta $\{U_n\}$ i $\{P\}$ o la utilització de restriccions textuals com a mètode de notació. Tot i que el significat és el mateix per a les diferents notacions, en aquest text utilitzarem la notació esmentada inicialment.

Tipus d'entitat amb clau primària i claus candidates

La figura 7 mostra exemples dels tipus d'entitat 'cotxe' i 'persona' amb les marques de la clau primària i de les claus candidates, en cas que n'hi hagi.

El tipus d'entitat 'persona' (*Person*) té dues claus candidates: per un costat tenim la clau formada per l'atribut 'ID' i per l'altre tenim la clau formada pels atributs 'nom' (*name*) i 'cognoms' (*surname*). Per a aquest exemple suposem que no hi poden haver dues persones amb els mateixos nom i cognoms. Tot i això, escollim com a clau primària l'atribut 'ID'. El tipus d'entitat 'cotxe' (*Car*) només té una clau candidata, formada per l'atribut 'matrícula' (*plateNumber*), que per tant és escollida per a ser clau primària.

Figura 7. Exemples dels tipus d'entitat 'cotxe' (*Car*) i 'persona' (*Person*) amb la clau primària i les claus candidates marcades



2.4. Tipus de relacions

Anomenem **tipus de relacions** les associacions entre tipus d'entitats, i **relacions** les associacions entre les entitats.

Els tipus de relacions indiquen generalment les relacions en què poden participar les entitats d'un tipus d'entitat, i amb quines condicions.

Utilitzarem la nomenclatura següent per a diferenciar aquests dos conceptes:

- El concepte que hem definit com a **tipus de relació** també pot ser referenciat pels termes **interrelació**, **relació** o **associació**.
- El concepte que hem definit com a **relació** també pot ser referenciat pels termes **instància** o **ocurrència** de la relació, **interrelació** o **associació**.

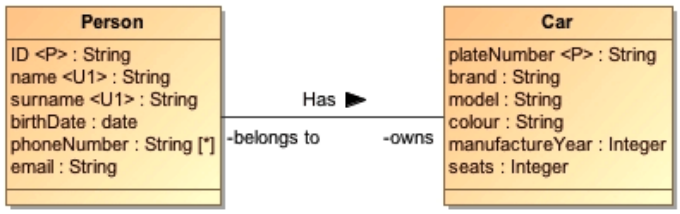
Terminologia UML

En la representació en diagrames UML, els tipus de relacions s'anomenen *associacions* i les relacions, *instàncies de les associacions*.

Exemple de tipus de relació

Entre els tipus d'entitats 'cotxe' (*Car*) i 'persona' (*Person*) podem establir un tipus de relació de propietat, que indiqui que una persona posseeix un cotxe. Per a representar aquesta informació hem de crear en el diagrama conceptual de la figura 8 un nou tipus de relació anomenat *Has*, per a indicar que una persona té un cotxe.

Figura 8. Exemple de tipus de relació *Has* entre els tipus d'entitat 'persona' (*Person*) i 'cotxe' (*Car*)



Tal com es pot veure en la figura 8, en un tipus de relació hi poden aparèixer diferents etiquetes, totes amb la finalitat de facilitar la comprensió de la relació entre les entitats:

- Els tipus de relació gairebé sempre tenen una **etiqueta de tipus de relació** que indica el significat de l'associació entre els dos tipus d'entitats. Aquesta etiqueta indica l'acció entre els dos tipus d'entitats, i generalment sol ser un verb. Per a especificar-ne més el significat, de manera opcional es pot indicar la direcció del tipus de relació. Això facilita la lectura de l'acció o de la relació que hi ha entre els dos tipus d'entitat. En la figura 8 podem llegir que "una persona té un cotxe". Tot i que no és gaire habitual, quan aquesta relació és prou evident es pot considerar que no és necessari indicar-ho explícitament i se'n pot obviar l'etiqueta o la direcció.

- També es pot donar la situació en què calgui definir una mica més el paper que té en la relació cadascuna de les entitats. En aquests casos es pot definir una etiqueta en cada extrem del tipus de relació que indica el paper que té cadascuna de les entitats en la relació i que ajuda a explicar el significat de la relació. Aquestes etiquetes s'anomenen **etiquetes de rol** o **noms de rol**. En termes generals no solen ser necessàries i no sempre es posen, però en alguns casos poden ajudar a definir el paper de les entitats en la relació. La figura 8 mostra els noms de rols per als dos tipus d'entitats que hi apareixen. Utilitzant les etiquetes de rol, podem llegir que "una persona posseeix (*owns*) un cotxe" i, de manera inversa, que "un cotxe és propietat (*belongs*

to) d'una persona". És necessari que tot tipus de relació tingui definits els rols dels participants o el nom del tipus de relació.

En alguns casos els tipus de relacions, en primera instància, es modelitzen com a atributs. Una anàlisi posterior amb més detall ajuda a veure que el que s'havia modelitzat en primer moment com un atribut és un tipus de relació devers un tipus d'entitat nou o existent.

Disseny d'un nou tipus de relació

Suposem que volem modelitzar el tipus d'entitat empleat (*Employee*) per emmagatzemar dades dels empleats d'una empresa. Entre altres atributs, ens interessa indicar a quin departament de l'empresa pertany cada empleat. Una primera aproximació suggereix un disseny en què s'inclou l'atribut 'departament' en el tipus d'entitat 'empleat', tal com mostra la figura 9. Però refinant aquest model, és possible que ens interessi tenir els diferents departaments categoritzats, i fins i tot ens podria interessar emmagatzemar certa informació relativa al departament. En aquest nou disseny es crea un nou tipus d'entitat per als departaments i s'estableix un tipus de relació entre els empleats i els departaments (tipus d'entitat *Department*) que indica a quin departament pertany cada empleat. La figura 10 mostra el nou disseny.

Figura 9. Exemple de disseny del departament com un atribut del tipus d'entitat 'empleat' (*Employee*)

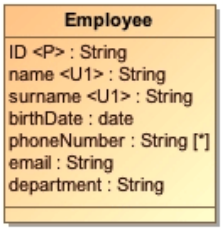


Figura 10. Exemple de disseny del departament com un nou tipus d'entitat 'departament' (*Department*) relacionat amb el tipus d'entitat 'empleat' (*Employee*)

Es defineix el **grau d'un tipus de relació** com el nombre de tipus d'entitats que participen en l'associació.

Els tipus de relacions es classifiquen, segons el grau que tenen, en:

- **Tipus de relacions binàries**: associacions en què intervenen dos tipus d'entitats.

- **Tipus de relacions ternàries**: associacions en què intervenen tres tipus d'entitats.

- **Tipus de relacions *n*-àries**: associacions en què intervenen *n* tipus d'entitats. Tot i que els tipus de relacions binàries i ternàries són un cas

concret dels tipus de relacions *n*-àries, aquests tipus es fan servir per a denotar relacions amb un grau superior a tres.

2.4.1. Tipus de relacions binàries

Es defineix un **tipus de relació binària** com l'associació entre dos tipus d'entitats.

Tipus de relació binària

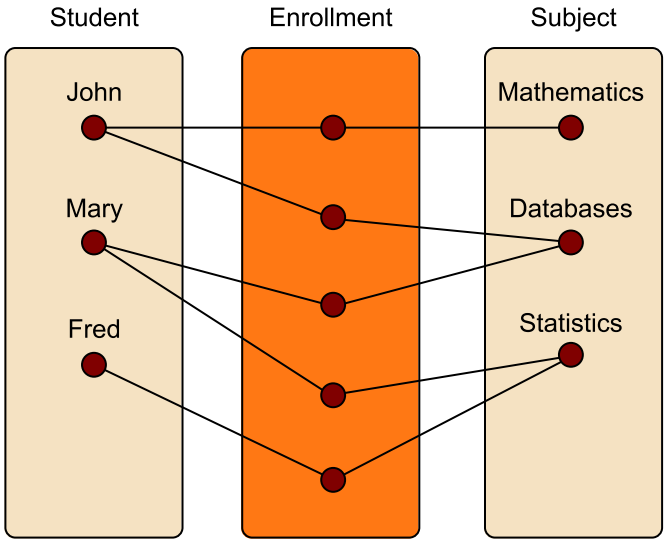
Figura 11. Exemple del tipus de relació binària 'matrícula' (*Enrollment*) entre els tipus d'entitat 'estudiant' (*Student*) i 'assignatura' (*Subject*)

La figura 11 mostra el tipus de relació 'matrícula' (*Enrollment*) entre els tipus d'entitat 'estudiant' (*Student*) i 'assignatura' (*Subject*). Aquesta relació indica en quines assignatures s'han matriculat cadascun dels estudiants.

Observació

En alguns diagrames d'exemple dels tipus de relacions obviarem els atributs i les característiques que no siguin necessaris per a explicar el concepte que ens interessi, a fi de simplificar els diagrames i centrar-nos en els tipus de relacions.

Figura 12. Exemple en el nivell d'entitats de la relació 'matrícula' (*Enrollment*)



Continuant amb l'exemple de la figura 11, la figura 12 mostra la mateixa relació, però a escala d'entitats. És a dir, aquest exemple mostra una possible associació entre les dades de les dues entitats mitjançant la relació *Enrollment*. Es pot veure, per exemple, que l'estudiant *John* està matriculat en les assignatures *Mathematics* i *Databases*.

Connectivitat

La **connectivitat d'un tipus de relació** expressa el tipus de correspondència que hi ha entre els tipus d'entitats que participen en un tipus de relació. En el cas dels tipus de relació binaris, expressa el nombre d'ocurrències d'un tipus d'entitat amb què es pot associar una ocurrència de l'altre tipus d'entitat segons el tipus de relació.

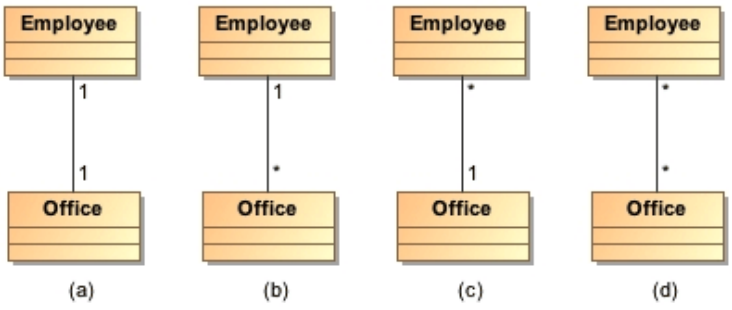
Un tipus de relació binària pot tenir tres tipus de connectivitat:

1) Connectivitat un a un (1:1 o 1..1): indica que cada entitat d'un tipus d'entitat es relaciona només amb una de les entitats de l'altre tipus. En la figura 13a es pot veure un exemple de connectivitat 1:1, en què, segons el model, cada empleat té un despatx i a cada despatx només hi ha un empleat. La connectivitat 1:1 es denota posant un 1 a cada banda del tipus de relació.

2) Connectivitat un a molts (1:*N* o 1..*): indica que cada entitat d'un tipus d'entitat es relaciona amb diverses entitats de l'altre tipus, però les entitats d'aquest segon tipus només es relacionen amb una única entitat del primer tipus. En la figura 13b es pot veure un exemple de connectivitat 1:*N*, en què la *N* és a la banda del tipus d'entitat 'despatx'. Segons el model, cada empleat té diversos despatxos, però a cada despatx només hi ha un empleat. En la figura 13c es pot veure la relació inversa. En aquest cas la *N* és a la banda del tipus d'entitat empleat. Segons el model, cada empleat té un únic despatx, però en un mateix despatx hi poden conviure diversos empleats. La connectivitat 1:*N* es denota posant un 1 a una banda del tipus de relació i una *N* o * a l'altra banda.

3) Connectivitat molts a molts (*M*:*N* o *..*): indica que cada entitat d'un tipus d'entitat es relaciona amb diverses entitats de l'altre tipus. En la figura 13d es pot veure un exemple de connectivitat *M*:*N*, en què, segons el model, cada empleat té diversos despatxos i a cada despatx hi ha diversos empleats. La connectivitat *M*:*N* es denota posant una *M* o * a una banda del tipus de relació i una *N* o * a l'altra banda.

Figura 13. Exemples de connectivitat en els tipus de relacions binàries entre els tipus d'entitat 'empleat' (*Employee*) i 'despatx' (*Office*)



Aquestes restriccions les determinen els requisits de cada problema i són els usuaris de la base de dades els qui han de conèixer les restriccions implícites en els problemes.

Quan es parla de connectivitat "a molts" es pot indicar de diferents maneres, segons el matís del problema. Farem servir:

- Un número per a indicar el nombre exacte d'entitats que poden intervenir en la relació.

- L'interval *min..max* per a indicar el nombre mínim i màxim d'entitats que poden intervenir en la relació.

- L'etiqueta *N* o ***** per a indicar que un nombre indefinit (zero o més) d'entitats poden participar en la relació.

Connectivitat "a molts"

La figura 14 mostra el tipus de relació 'matrícula' (*Enrollment*) entre el tipus d'entitat 'estudiant' (*Student*) i 'assignatura' (*Subject*), en què s'indica que un estudiant ha d'estar matriculat a un mínim de dues assignatures i un màxim de cinc.

Figura 14. Exemple del tipus de relació 'matrícula' (*Enrollment*) en què s'indica el nombre mínim i màxim d'instàncies que poden intervenir en la relació

Dependència d'existència o restriccions de participació

En alguns casos, les entitats d'un tipus d'entitat no poden existir si no estan relacionades amb les entitats d'un altre tipus d'entitat mitjançant una relació binària determinada. En aquests casos, es diu que el segon tipus d'entitat és un tipus d'entitat obligatori en la relació. En un altre cas, es diu que és un tipus d'entitat opcional en la relació.

Tipus d'entitat obligatòria i opcional

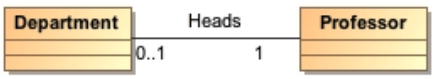
Tipus d'entitat obligatòria: per a designar aquest mateix concepte també es pot dir que el tipus d'entitat té una participació total en l'associació o que presenta una dependència d'existència respecte a l'associació.

Tipus d'entitat opcional: per a designar aquest mateix concepte també es pot dir que el tipus d'entitat té una participació parcial en l'associació.

Dependència d'existència en els tipus de relacions binàries

La figura 15 mostra un model en què apareix un tipus de relació binària que expressa la relació de 'direcció de departament' (*Heads*) entre els tipus d'entitat 'professor' (*Professor*) i 'departament' (*Department*). El tipus d'entitat 'professor' és obligatori en la relació 'direcció de departament'. Això indica que no hi pot haver un departament que no tingui un professor que fa de director de departament. El tipus d'entitat 'departament', en canvi, és opcional en la relació 'direcció de departament'. Pot ser que hi hagi un o més professors que no tinguin la responsabilitat de dirigir cap departament.

Figura 15. Exemple de dependència d'existència en els tipus de relacions binàries



Aquesta dependència d'existència només és aplicable en els tipus de relacions binàries. No es dóna en els tipus de relacions *n*-àries.

En els diagrames UML s'utilitza la cardinalitat del tipus de relació per a expressar l'obligatorietat o no de cadascun dels tipus d'entitats que participen en l'associació. Considerant els tipus de connectivitat que hem vist, cal aplicar:

- **Tipus d'entitat amb cardinalitat 1**: un tipus d'entitat connectada amb cardinalitat 1 a una associació expressa l'obligatorietat tot indicant una cardinalitat exactament igual a 1..1, mentre que si és opcional en l'associació s'indica amb la cardinalitat 0..1.

- **Tipus d'entitat amb cardinalitat *N***: un tipus d'entitat connectada amb cardinalitat *N* a una associació expressa l'obligatorietat tot indicant una cardinalitat 1..*, mentre que si és opcional en l'associació s'indica amb la cardinalitat 0..*.

Mètodes abreujats de cardinalitats:

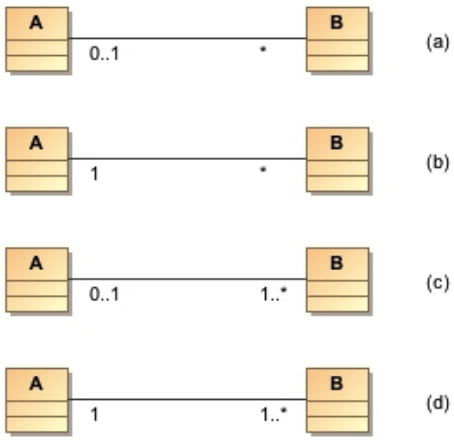
- La cardinalitat 1 és una manera abreujada de referir-se a la cardinalitat 1..1.

- La cardinalitat * és una manera abreujada de referir-se a la cardinalitat 0..*.

Obligatorietat en la dependència d'existència

La figura 16 mostra les quatre possibilitats d'expressar l'obligatorietat de les classes A i B en un tipus de relació binària 1:*N*. En la figura 16 veiem en 16a el cas en què els dos tipus d'entitats són opcionals, en 16b el cas en què el tipus d'entitat A és obligatori, en 16c el cas en què el tipus d'entitat B és obligatori i en 16d el cas en què els dos tipus d'entitat són obligatoris.

Figura 16. Exemples d'obligatorietat en la dependència d'existència



Igual que les restriccions de connectivitat, aquestes restriccions són inherents als problemes que es volen resoldre i és a partir del recull de requisits dels usuaris de la base de dades, que es podrà determinar com cal procedir en cada cas.

2.4.2. Tipus de relacions ternàries

Es defineix un **tipus de relació ternària** com l'associació entre tres tipus d'entitats.

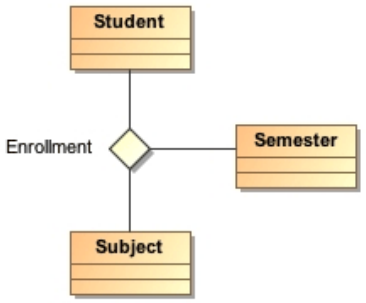
Hi ha situacions en què no n'hi ha prou de modelitzar una associació entre dos tipus d'entitats per a representar un concepte, i cal modelitzar l'associació entre tres tipus d'entitats per a representar el concepte desitjat.

Tipus de relació ternària

Seguint un exemple vist abans, no n'hi ha prou de modelitzar el tipus de relació 'matrícula' (*Enrollment*) entre els tipus d'entitats 'estudiant' (*Student*) i 'assignatura' (*Subject*) per

a representar la realitat dels estudiants que suspenen una assignatura i han de tornar a cursar-la en un semestre posterior. En aquests casos, una entitat estudiant pot tenir zero, una o més d'una matrícula per a una mateixa entitat assignatura. El requisit és que només es pot matricular una vegada cada semestre. Per tant, cal introduir el tipus d'entitat 'semestre' (*Semester*) en l'exemple anterior i modificar el model perquè la relació 'matrícula' associï els tres tipus d'entitat. La figura 17 mostra el nou model.

Figura 17. Exemple de tipus de relació ternària entre els tipus d'entitat 'estudiant' (*Student*), 'assignatura' (*Subject*) i 'semestre' (*Semester*)



Connectivitat

En la connectivitat d'un tipus de relació ternària hi ha implicats tres tipus d'entitats, i cadascun pot prendre la connectivitat d'"un" o "molts".

Un tipus de relació ternària pot tenir quatre tipus de connectivitat:

- Connectivitat *M*:*N*:*P* o *..*..*.
- Connectivitat *M*:*N*:1 o *..*..1.
- Connectivitat *M*:1:1 o *..1..1.
- Connectivitat 1:1:1 o 1..1..1.

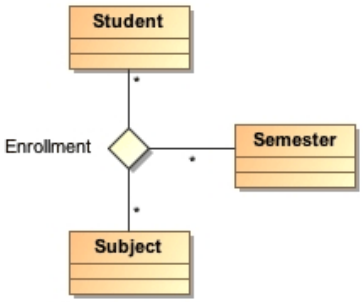
Per a decidir com s'ha de connectar cadascun dels tipus d'entitat a l'associació, cal fixar "a una" les altres entitats i llavors preguntar-se si és possible connectar amb la relació "a una" o "a moltes" entitats del primer tipus d'entitat.

Connectivitat en un tipus de relació ternària

Vegem com definim la connectivitat en un tipus de relació ternària mitjançant l'exemple que il·lustra la figura 18.

En la figura podem veure els tres tipus d'entitat, 'estudiant' (*Student*), 'assignatura' (*Subject*) i 'semestre' (*Semester*), i el tipus de relació 'matrícula' (*Enrollment*).

Figura 18. Exemple de connectivitat en un tipus de relació ternària



Per a determinar el grau de connectivitat de cada tipus d'entitat en l'associació hem de fixar "a una" les instàncies de la resta de tipus d'entitats i ens hem de preguntar si és possible connectar amb la relació "a una" o "a moltes" instàncies del primer tipus d'entitat.

En primer lloc, per exemple, ens preguntem si amb referència a la relació 'matrícula' i donats una assignatura i un semestre concrets es poden tenir "un" o "molts" estudiants. La resposta és que es poden tenir "molts" estudiants, ja que hi poden haver molts estudiants que es matriculen d'una mateixa assignatura en el mateix semestre. Per tant, el tipus d'entitat 'estudiant' participa amb grau *N* en la relació 'matrícula'.

En segon lloc, ens preguntem, per exemple, si donats un estudiant i una assignatura concrets es poden tenir "un" o "molts" semestres. La resposta és que es poden tenir "molts" semestres, ja que un estudiant es pot matricular més d'una vegada en diferents semestres abans de superar una assignatura. Per tant, el tipus d'entitat 'semestre' participa amb grau *N* en la relació 'matrícula'.

Finalment, ens preguntem si donats un estudiant i un semestre concrets es poden tenir "una" o "moltes" assignatures. La resposta és que es poden tenir "moltes" assignatures, ja que un estudiant es pot matricular de diverses assignatures en un mateix semestre. Per tant, el tipus d'entitat 'assignatura' també participa amb grau *N* en la relació matrícula.

Per tant, veiem que el tipus de relació 'matrícula' té cardinalitat *M*:*N*:*P* o *..*..*.

2.4.3. Tipus de relacions *n*-àries

Es defineix un **tipus de relació *n*-ària** com l'associació entre tres o més tipus d'entitats.

El tipus de relació ternària és un cas concret de tipus de relació *n*-ària. El que hem vist en el subapartat anterior es pot generalitzar per als tipus de relacions *n*-àries. Per tant, en una relació *n*-ària hi poden haver $n + 1$ tipus de connectivitat, ja que cadascuna de les n entitats pot estar connectada "a un" o "a molts" tipus en l'associació.

Connectivitat en un tipus de relació quaternària

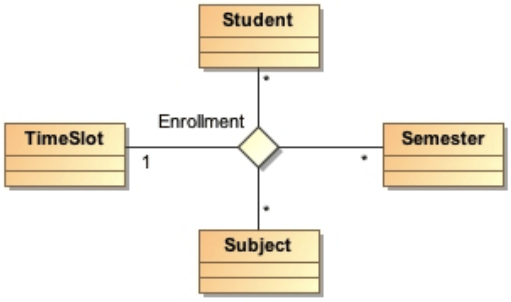
La figura 19 presenta una ampliació de l'exemple anterior, en què es presentaven els tipus d'entitat 'estudiant' (*Student*), 'assignatura' (*Subject*) i 'semestre' (*Semester*) associats mitjançant el tipus de relació 'matrícula' (*Enrollment*).

En aquest cas volem afegir el concepte de *torn* o *franja horària* en l'esquema, per indicar que una mateixa assignatura és de torn de matí o de tarda. Per a modelitzar aquest concepte, hem creat el tipus d'entitat 'franja horària' (*TimeSlot*).

Per a determinar el grau de connectivitat del nou tipus d'entitat seguirem el mateix procés vist per al cas del tipus de relacions ternàries. En aquest cas, ens preguntem si amb referència al tipus de relació 'matrícula' i donats un estudiant, un semestre i una assignatura concrets es poden tenir "una" o "moltes" franges horàries. La resposta és que només

es pot tenir "una" franja horària. Per tant, el tipus d'entitat de 'franja horària' o 'torn' (*TimeSlot*) participa amb grau 1 en aquest tipus de relació.

Figura 19. Exemple de connectivitat en un tipus de relació quaternària



2.4.4. Tipus de relacions reflexives o recursives

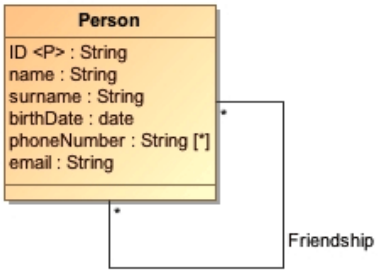
Un **tipus de relació reflexiva** o **recursiva** és un tipus de relació binària en la qual un tipus d'entitat es relaciona amb si mateixa.

Es presenta una relació entre dues entitats del mateix tipus d'entitat. Igual que en la resta de tipus de relacions binàries, poden tenir connectivitat 1:1, 1:*N* o *M*:*N* i poden expressar dependència d'existència.

Tipus de relació recursiva binària

Un clar exemple d'un tipus de relació recursiva binària és el concepte d'*amistat* entre dues persones. En aquest cas, tenim el tipus d'entitat 'persona' (*Person*) i el tipus de relació 'amistat' (*Friendship*) que uneix dues entitats de 'persona'. La figura 20 en mostra el model conceptual. La connectivitat de l'associació és *M*:*N*, ja que una persona pot tenir diversos amics i, alhora, hi pot haver diverses persones que la tenen d'amiga.

Figura 20. Exemple de tipus de relació recursiva

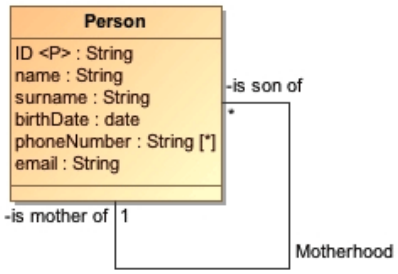


Generalment no es fa referència explícita al rol de cada entitat en una relació, ja que normalment cada entitat participa una única vegada en una relació i se sobreentén el rol que té. Però en les relacions recursives pot ser que el rol de cada entitat no sigui tan clar. Per tant, si es considera oportú, es pot indicar el rol de cada associació entre l'entitat i la relació.

Un altre exemple de tipus de relació recursiva

La figura 21 mostra un tipus de relació recursiva que descriu la relació de maternitat entre les persones. Podem veure que una persona pot tenir zero o més fills, i que una persona només pot tenir una mare. Les etiquetes o noms de rol, en aquest cas, poden ajudar a comprendre el rol que té cada entitat en aquesta relació recursiva.

Figura 21. Exemple de tipus de relació recursiva amb noms de rol



2.5. Tipus d'entitats associatives

Les relacions també poden tenir atributs que permetin descriure'n propietats interessants. Aquests atributs es defineixen i segueixen les mateixes regles que en el cas dels atributs de les entitats.

Un **tipus d'entitat associativa** és el resultat de considerar una relació entre entitats com si fos una entitat. El nom que té es correspon amb el nom de la relació sobre la qual es defineix.

Els tipus d'entitats associatives afegeixen noves funcionalitats a les relacions. Algunes de les principals són les següents:

1) Permeten definir atributs que fan referència a la relació. És a dir, són específics de l'associació que hi ha entre les dues instàncies dels dos tipus d'entitat que formen la relació.

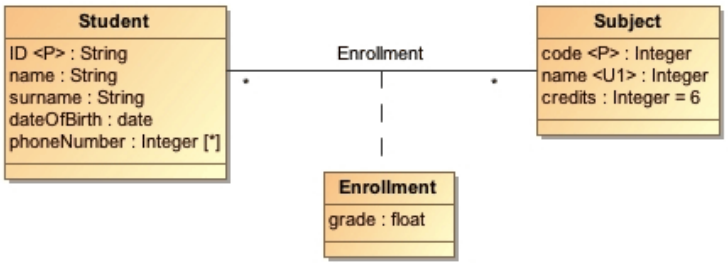
2) Permeten que el tipus d'entitat associativa, i per tant la relació entre els tipus d'entitats, participi en altres relacions amb altres tipus d'entitats o amb altres tipus d'entitats associatives.

Tipus d'entitat associativa

La figura 22 mostra el tipus de relació 'matrícula' (*Enrollment*) entre els tipus d'entitat 'estudiant' (*Student*) i 'assignatura' (*Subject*). Aquesta relació indica en quines assignatures estan matriculats cadascun dels estudiants. L'entitat associativa conté l'atribut 'nota' (*grade*), que indica la nota que ha obtingut un estudiant en una assignatura. Per a cada relació entre un estudiant i una assignatura hi ha una relació 'matrícula' amb un atribut 'nota'.

En el cas de situar l'atribut 'nota' (*grade*) en l'entitat 'estudiant' (*Student*), diem que un estudiant sempre tindrà la mateixa nota de totes les assignatures de què es matriculi. En el cas de situar l'atribut en l'entitat 'assignatura' (*Subject*), en canvi, diem que tots els estudiants tenen la mateixa nota en una mateixa assignatura. Per tant, l'única manera de modelitzar la realitat de manera correcta és considerar l'atribut 'nota' com un atribut de la relació entre aquests dos tipus d'entitat.

Figura 22. Exemple del tipus d'entitat associativa matrícula (*Enrollment*)

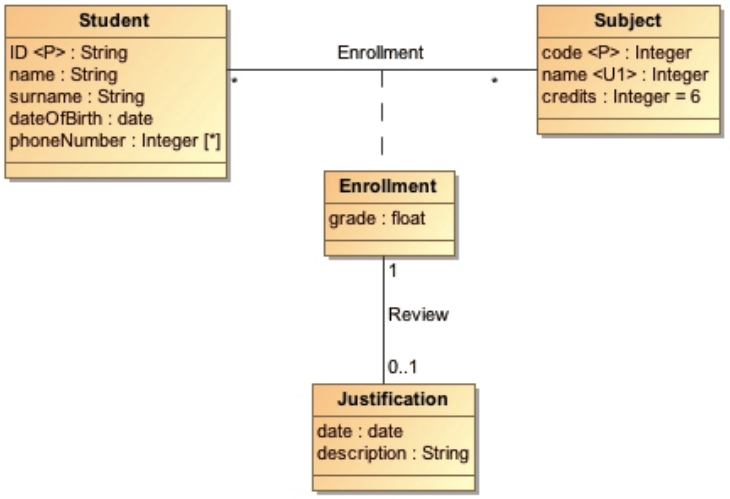


Els tipus d'entitats associatives permeten establir relacions amb altres entitats i també amb altres entitats associatives, és a dir, permeten crear relacions entre relacions.

Tipus de relació entre una entitat i una entitat associativa

Suposem que volem afegir a l'exemple anterior el concepte de *justificació* o *retorn personalitzat* de la nota. És a dir, volem expressar que en alguns casos, per exemple quan ho sol·liciti, l'estudiant rebrà una descripció en què es justifica la puntuació que ha obtingut i els errors que ha comès en una assignatura determinada. La figura 23 mostra el model conceptual amb el nou requisit. Podem veure com el nou tipus d'entitat 'justificació' (*Justification*) indica la data i la descripció de la justificació i es relaciona amb algunes matriculacions. Aquest punt és rellevant, ja que, atesa la cardinalitat del tipus de relació 'revisió' (*Review*), es pot deduir que hi poden haver parelles estudiant-assignatura sense una justificació.

Figura 23. Exemple de tipus de relació entre una entitat i una entitat associativa



Cal fer notar que aquesta mateixa situació no es pot modelitzar mitjançant un tipus de relació ternària.

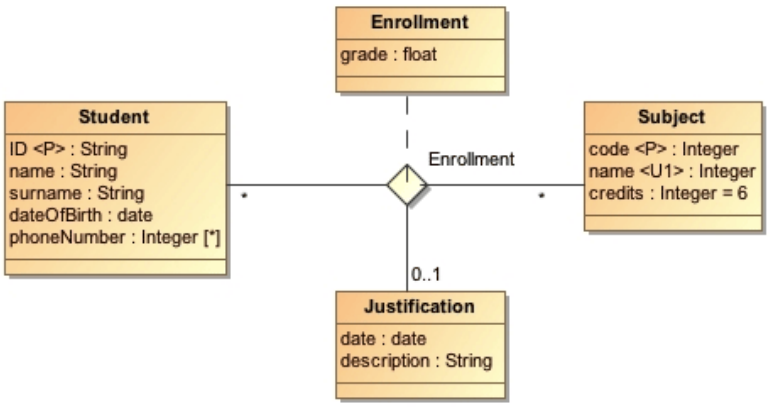
Error d'expressió

La figura 24 mostra el model conceptual emprant un tipus de relació ternària. En aquest cas el significat del model és diferent del que plantejàvem en l'exemple anterior. Aquest model ens diu que per a cada estudiant (*Student*) i assignatura (*Subject*) hi pot haver zero o una justificació (*Justification*). Però també indica que donades una justificació i una assignatura hi poden haver molts estudiants, afirmació que és falsa segons els requisits inicials. Igualment, el model indica que donat un estudiant i una justificació hi poden haver moltes assignatures, afirmació que també és falsa. Si modifiquem les cardinalitats dels tipus d'entitat 'estudiant' i 'assignatura' i les establim a 1, llavors no podrem representar de manera correcta que un estudiant pot estar matriculat en diverses assignatures.

El problema rau en el fet que intentem representar amb un tipus de relació ternària un model que és binari. Cal fer notar que el concepte de *justificació* fa referència a la relació

que hi ha entre les entitats 'estudiant' i 'assignatura', i per això cal que sigui modelitzat com un tipus d'entitat que es relaciona amb el tipus d'entitat associativa que sorgeix del tipus de relació entre els dos tipus d'entitats.

Figura 24. Exemple d'error d'expressió sobre l'exemple de la figura 23



És important que quedi clar que no es representa el mateix concepte amb un tipus de relació ternària i un tipus de relació binària amb una entitat associativa.

2.6. Tipus d'entitats dèbils

Es defineix un **tipus d'entitat forta** com un tipus d'entitat que posseeix un conjunt d'atributs suficients per a formar claus candidates i identificar de manera inequívoca a totes les seves instàncies.

Tipus d'entitats fortes

També es poden anomenar *tipus d'entitats propietàries, dominants* o *pares*.

L'existència de les instàncies d'un tipus d'entitat forta no depèn de cap altre tipus d'entitat o tipus de relació. És a dir, el significat de les entitats fortes no necessita cap relació amb altres entitats que en complementi el significat. Per tant, es poden veure aquestes entitats com aquelles que tenen existència pròpia i ben definida.

Tipus d'entitats dèbils

També es poden anomenar *tipus d'entitats subordinades* o *filles*.

Es defineix un **tipus d'entitat dèbil** com un tipus d'entitat els atributs de la qual no la identifiquen completament, sinó que només la identifiquen de manera parcial. Aquestes entitats han de participar en una relació amb altres entitats que ajudin a identificar-les de manera unívoca.

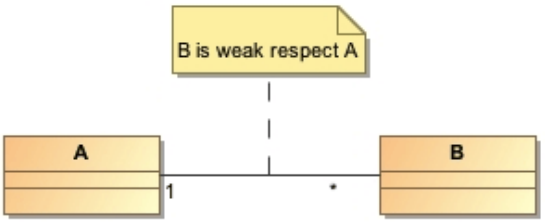
Un **tipus d'entitat dèbil**, per tant, necessita una associació amb un tipus d'entitat que li permet identificar de manera única les seves instàncies. Aquest tipus de relació rep el nom de *tipus de relació identificativa* del tipus d'entitat dèbil. El tipus d'entitat dèbil ha de tenir una restricció de participació total (dependència d'existència) respecte al tipus de relació identificativa, ja que si no és així hi ha instàncies que no es podran identificar de manera única.

Tota entitat dèbil ha de formar part d'una relació binària amb connectivitat 1:*N*, en què l'entitat dèbil ha de ser al costat *N*. Aquesta relació li permetrà identificar-se completament.

Les entitats dèbils no poden sobreviure si s'elimina l'entitat forta amb la qual estan relacionades mitjançant la relació identificativa. Per tant, cal tenir en compte que, si s'elimina una entitat forta, cal eliminar les entitats dèbils que hi estan relacionades.

En els diagrames UML un tipus d'entitat dèbil s'indica mitjançant la dependència d'existència que té devers el tipus d'entitat forta. El tipus d'entitat dèbil, a causa de la dependència d'existència o restricció de participació, necessita mantenir una relació 1:*N* amb el tipus d'entitat forta. Per tant, sempre mantindrà una relació amb una cardinalitat igual a 1 devers la banda del tipus d'entitat forta. La figura 25 mostra la notació per a indicar un tipus d'entitat dèbil en els diagrames UML.

Figura 25. Notació UML per a indicar un tipus d'entitat dèbil (*B*)



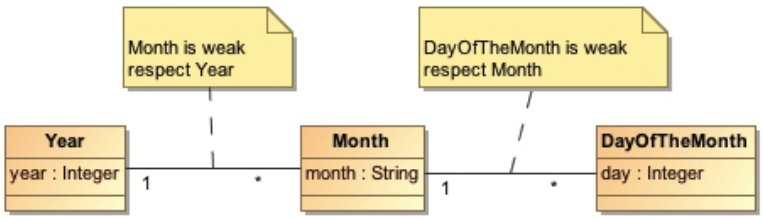
Cal fer notar que un tipus de relació identificativa no sol tenir atributs, ja que en les relacions 1:*N* sempre es poden posar els atributs en el tipus d'entitat dèbil, és a dir, a la banda amb cardinalitat *N*.

Un tipus d'entitat pot ser fort en una determinada associació i, al mateix temps, ser dèbil en una altra associació.

Relativitat del tipus de relació

La figura 26 mostra el tipus d'entitat 'mes de l'any' (*Month*), que és dèbil respecte a la relació amb el tipus d'entitat 'any' (*Year*) i forta respecte a la relació amb el tipus d'entitat 'dia del mes' (*DayOfTheMonth*).

Figura 26. Exemple de tipus d'entitat forta respecte a una relació i, al mateix temps, dèbil respecte a una altra relació



Una **clau parcial** en un tipus d'entitat dèbil és el conjunt d'atributs que permet identificar de manera única totes les instàncies del tipus d'entitat dèbil que estan relacionades amb la mateixa instància del tipus d'entitat forta.

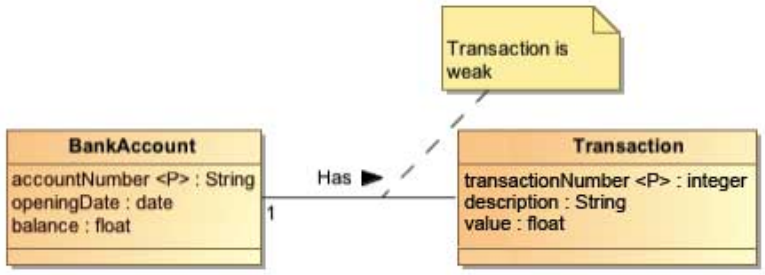
En el pitjor dels casos, la clau parcial és el conjunt de tots els atributs del tipus d'entitat dèbil.

Es considera que la clau primària del tipus d'entitat dèbil està formada per la clau primària del tipus d'entitat forta i la clau parcial del tipus d'entitat dèbil. D'aquesta manera es pot referenciar cadascuna de les instàncies del tipus d'entitat dèbil de manera única.

Notació única de dos tipus d'entitats dèbils

Si considerem el concepte de *compte corrent* en una entitat bancària podem afirmar que és un tipus d'entitat forta, ja que posseeix el conjunt d'atributs necessaris per a identificar de manera única cadascuna de les seves instàncies. En canvi, si considerem el concepte de *moviment* o *transacció* ens adonem que només té sentit i pot existir si està associat a un compte bancari. Per tant, per a modelitzar aquesta situació es pot crear un tipus d'entitat forta, *BankAccount*, i un tipus d'entitat dèbil, *Transaction*, com es mostra en la figura 27. La clau primària del tipus d'entitat *BankAccount* és l'atribut *accountNumber*. En el tipus d'entitat *Transaction* la clau parcial és l'atribut *transactionNumber*, que indica el número de transacció. Per tant, la clau primària del tipus d'entitat dèbil és composta i està formada pels atributs {*accountNumber*, *transactionNumber*}.

Figura 27. Exemple de tipus d'entitat dèbil (*Transaction*) i la relació identificativa amb el tipus d'entitat forta (*BankAccount*)



2.7. Opcions de disseny

En algunes situacions no és trivial determinar si un concepte ha de ser modelitzat com un tipus d'entitat, un atribut o un tipus de relació. Cal tenir en compte que normalment no hi ha un únic diagrama conceptual per a un problema concret, i que hi poden haver diferents conceptualitzacions que donin lloc a diferents diagrames, tots correctes.

A continuació donarem alguns consells que poden ser útils durant el disseny conceptual d'una base de dades:

- El procés de disseny conceptual d'una base de dades es pot enfocar com un procés iteratiu de refinament, en què es crea un primer disseny inicial

i es va refinant de manera iterativa fins a aconseguir el nivell de disseny desitjat.

- És freqüent modelitzar un concepte com un atribut en un primer moment, i en refinaments posteriors fins a convertint-lo en un tipus de relació. Modelitzar l'atribut com una relació a un tipus d'entitat nou o existent permet categoritzar els valors d'aquest concepte i, a més a més, permet que es puguin afegir nous atributs relacionats amb el tipus d'entitat nou o existent, ara o en el futur.

- També és habitual que un atribut existent en diversos tipus d'entitats sigui convertit en un nou tipus d'entitat. A continuació cal establir relacions des dels tipus d'entitats que contenien aquest atribut cap al nou tipus d'entitat.

- També hi pot haver el refinament invers. En aquest cas, es pot eliminar un tipus d'entitat i representar el concepte com un atribut si només afecta un (o molt pocs) tipus d'entitat.

D'atribut a tipus d'entitat

Si els tipus d'entitats *Estudiant*, *Professor* i *Curs* tenen l'atribut *departament*, és interessant crear el nou tipus d'entitat *Departament* amb un únic atribut que indica el nom del departament, i crear tres tipus de relacions binàries entre *Estudiant*, *Professor* i *Curs*, i el nou tipus d'entitat *Departament*.

2.8. Criteris d'assignació de noms

En el disseny de l'esquema conceptual d'una base de dades és important fer una bona elecció de noms per als tipus d'entitats, atributs i tipus de relacions que hi apareixen, ja que no sempre és senzill poder determinar el paper que té cadascun en un model conceptual. Cal escollir noms que transmetin de la millor manera possible el significat de les diferents estructures de l'esquema.

Els criteris emprats en aquest text són una opció, tot i que no hi ha normativa i poden variar respecte d'altres autors.

En els punts següents establim la base de la nomenclatura en els diagrames UML:

- Etiquetar els tipus d'entitats utilitzant noms simples en singular, sempre que sigui possible. Si és necessari, es poden utilitzar fins a dos o tres noms. Cal emprar la grafia Pascal per a escriure el nom dels tipus d'entitats.

- Etiquetar els atributs utilitzant noms simples en singular i evitant que hi aparegui el nom del tipus d'entitat. Si és necessari, es poden utilitzar fins a dos o tres noms o la combinació d'un nom i un o dos adjectius. Cal emprar la grafia Camel per a escriure el nom dels atributs.

- En el cas d'atributs booleans se sol utilitzar el nom o noms volguts precedits de *es* (de la forma verbal *és*, o *is* en la versió anglesa) per a donar l'èmfasi de valor booleà que respon que sí o que no davant d'una pregunta. Per exemple: *esNegreta* o *isBold*.

Exemples de grafies possibles

Grafia Pascal: la primera lletra de l'identificador i la primera lletra de les següents paraules s'escriuen en majúscula. La resta, en minúscula. Per exemple: BackColor.

Grafia Camel: la primera lletra de l'identificador s'escriu en minúscula i la primera lletra de les següents paraules concatenades, en majúscula. La resta, en minúscula. Per exemple: backColor.

- Etiquetar els tipus de relació utilitzant verbs en tercera persona del singular. En cas de requerir etiquetes compostes, se sol utilitzar un verb seguit d'una preposició i un altre verb. Les etiquetes dels tipus de relacions segueixen la grafia Pascal.

Les etiquetes per a identificar els rols de les entitats en les relacions segueixen la grafia Camel.

Com a pràctica general, a partir d'una descripció funcional dels requisits de la base de dades, els noms que apareixen tendeixen a ser candidats de noms per als tipus d'entitats que calgui definir, mentre que els verbs tendeixen a ser utilitzats per als tipus de relacions de l'esquema conceptual. Els noms dels tipus de relacions s'escullen per tal que es puguin llegir d'esquerra a dreta i de dalt a baix, com a criteri general. D'aquesta manera es facilita la lectura de les relacions. Hi poden haver excepcions a aquestes normes, i en aquests casos ens podem ajudar dels noms de rols o podem indicar la direcció del tipus de relació per facilitar la comprensió de les relacions i el paper de les entitats en cadascuna d'aquestes relacions. Els atributs solen ser noms extrets de les descripcions dels objectes definits com a tipus d'entitats.

2.9. Exemple complet

En aquest apartat hem vist els elements bàsics del disseny conceptual segons el model ER. Abans de continuar amb els elements avançats en l'apartat 3, pot resultar molt il·lustratiu veure una aplicació pràctica dels conceptes vistos fins ara. Per a facilitar l'anàlisi d'un exemple més o menys real, plantejarem a continuació una descripció i un conjunt de requisits i restriccions que han de permetre implementar un model conceptual d'una base de dades destinada a la gestió universitària. Aquest model és una reducció d'un model real i no es vol adaptar a la gestió real d'una universitat. Només vol servir de model d'exemple per als conceptes vistos fins ara.

A continuació enumerem els diferents aspectes dels requisits dels usuaris que cal tenir en compte en fer el disseny conceptual de la futura base de dades:

1) La universitat està formada per diferents departaments. Cada departament està format per un conjunt de professors i està dirigit per un únic professor. Dels departaments ens interessa el nom i el nombre de professors.

2) De cada professor ens interessa poder emmagatzemar-ne les dades personals, com, per exemple, nom, DNI, edat i si és un home o una dona. Dels professors que són directors de departament ens interessa poder identificar la data en què van començar a exercir aquest càrrec.

3) En la universitat s'imparteixen un conjunt d'assignatures. De cadascuna d'aquestes assignatures, ens interessa saber-ne el codi, el nom, el nombre de crèdits, la descripció i els prerequisits, és a dir, altres assignatures que cal haver cursat anteriorment. A més, cal tenir en compte que una assignatura pertany a un únic departament i és impartida per un únic professor.

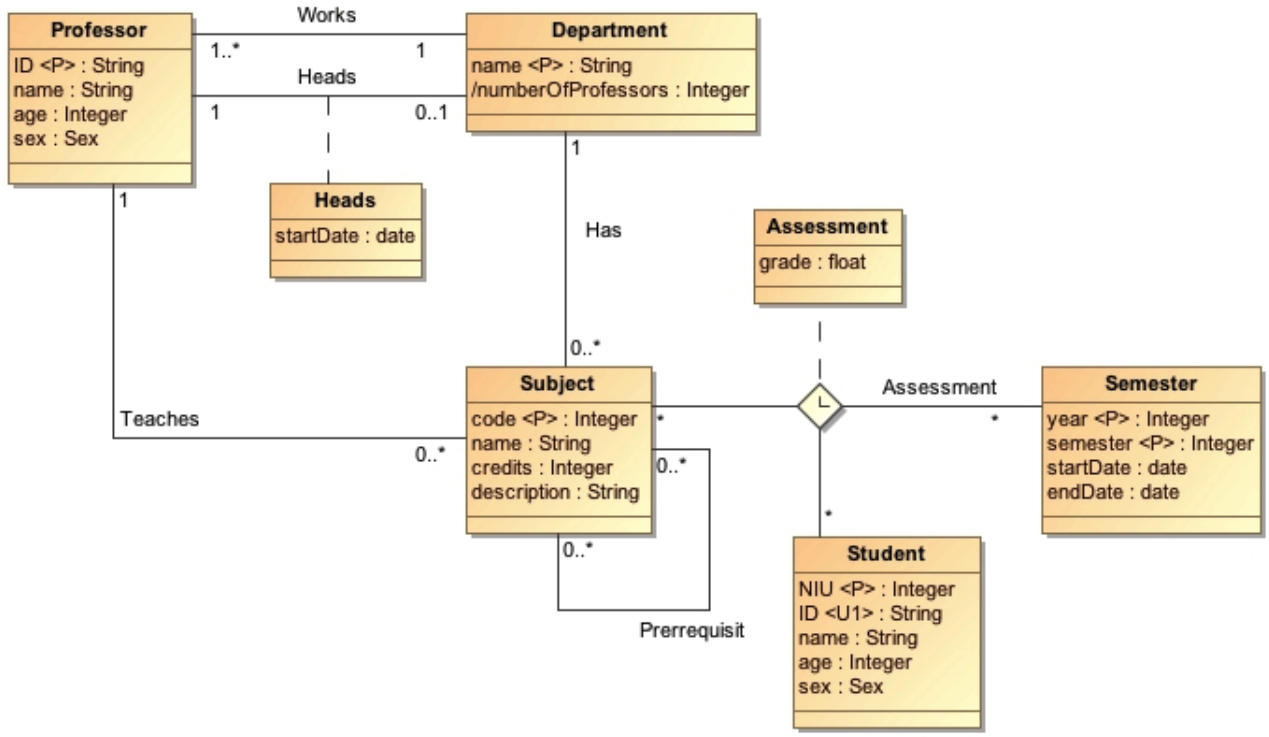
4) Els estudiants són una part molt important de la base de dades. D'aquests estudiants es vol emmagatzemar informació sobre les dades personals (nom, DNI, edat, sexe) i el número d'identificació universitari (conegut com a NIU).

5) Cada estudiant pot estar matriculat de més d'una assignatura en cada semestre. I per cadascuna de les assignatures en què està matriculat cada semestre, n'hem de poder emmagatzemar la nota final.

6) També es vol deixar constància de les dates d'inici i final de cada semestre.

A partir dels requisits expressats, la figura 28 mostra un diagrama del model conceptual. Com hem comentat, aquest model no és únic, i hi poden haver diferents aproximacions i conceptualitzacions vàlides per a una mateixa representació del món real.

Figura 28. Diagrama del model conceptual per a la base de dades de gestió universitària



A continuació comentem els aspectes més rellevants d'aquest model conceptual:

- En el tipus d'entitat 'departament' (*Department*), l'atribut 'nombre de professors' (*numberOfProfessors*) és un atribut derivat, ja que es calcula a partir del nombre de professors que estan vinculats a cada departament.

- Els tipus d'entitat 'departament' i 'professor' (*Professor*) tenen dos tipus de relacions per a indicar, per un costat, en quin departament treballa cada professor i per a determinar, per l'altre, quin professor és el director de cada departament. La connectivitat de la primera associació indica que un professor ha de pertànyer a un departament, i que els departaments han de tenir com a mínim un professor. La segona relació és un tipus d'entitat associativa i conté l'atribut 'data d'inici' (*startDate*) per a indicar la data en què el professor comença la tasca com a cap de departament. Un professor pot impartir cap, una o més d'una assignatura, com indica la connectivitat del tipus de relació amb el tipus d'entitat 'assignatura' (*Subject*).

- El concepte de *prerequisit* d'una assignatura s'ha modelitzat com un tipus de relació recursiva que associa diferents entitats d'assignatures. Tal com es desprèn de la connectivitat, una assignatura pot tenir cap o múltiples prerequisits i ser alhora prerequisit de cap o múltiples assignatures.

- El semestre s'ha modelitzat com un tipus d'entitat (*Semester*).

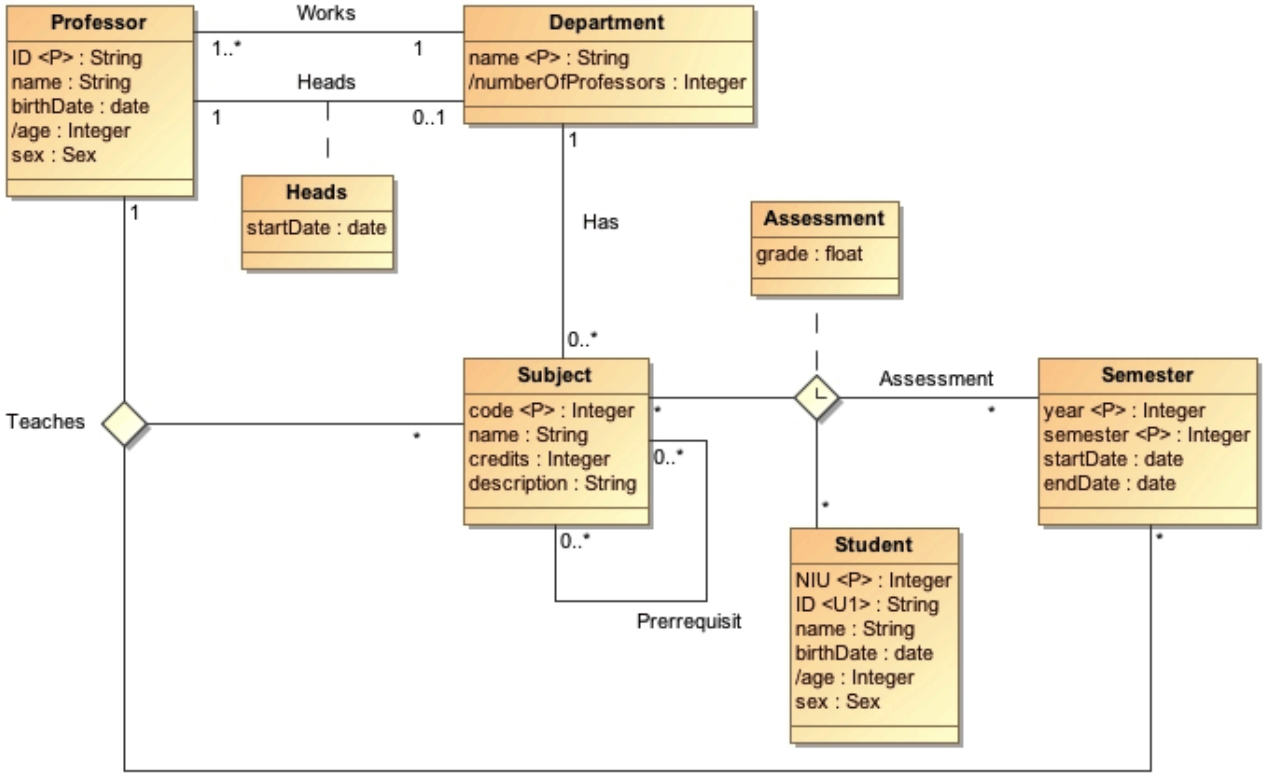
- El tipus de relació ternària entre els tipus d'entitat 'assignatura', 'semestre' i 'estudiant' (*Student*) determina la nota obtinguda per cada estudiant com a qualificació de cada assignatura en què està matriculat cada semestre. Tal com s'expressava en els requisits, un estudiant pot estar matriculat de múltiples assignatures en un semestre i es pot matricular de la mateixa assignatura en diversos semestres.

Però aquest model conceptual també presenta algunes deficiències, que comentem a continuació:

- Segons el model, un professor imparteix un conjunt d'assignatures, però no es manté relació amb cadascun dels semestres. És a dir, el tipus de relació 'imparteix' (*Teaches*) indica el professor que imparteix una assignatura, però no permet saber quin professor l'havia impartit en semestres anteriors.

- En els tipus d'entitat 'professor' i 'estudiant' s'emmagatzema l'edat mitjançant un atribut numèric que conté el valor per a cada professor o estudiant en un moment determinat. Això implica que cada any caldrà actualitzar la base de dades i sumar 1 als valors d'aquest atribut. Lògicament, aquest model no és l'adient, i caldria emmagatzemar la data de naixement dels professors i els estudiants i crear un atribut derivat que indiqui l'edat (*age*) dels professors o els estudiants de manera automàtica.

Figura 29. Diagrama del model conceptual amb les correccions aplicades



La figura 29 mostra el model conceptual modificat per a corregir les deficiències detectades. El tipus de relació 'imparteix' (*Teaches*) s'ha convertit en un tipus de relació ternària. D'aquesta manera, podem indicar quin professor és responsable de cada assignatura en cada semestre.

3. Elements avançats de modelització

Els conceptes vistos fins ara permeten representar una gran quantitat dels esquemes de bases de dades en aplicacions tradicionals. Però per a poder aproximar de manera més precisa algunes característiques del món real i fomentar la reusabilitat, s'han desenvolupat múltiples ampliacions i complements als elements vistos fins ara. En aquest apartat veurem algunes d'aquestes ampliacions i d'aquests complements que permeten representar realitats difícilment expressables utilitzant els elements de modelització vistos fins ara.

3.1. Generalització/especialització

En alguns casos, es poden trobar entitats que tinguin algunes característiques pròpies i diferenciades d'altres entitats del mateix tipus d'entitat, i que interessi poder representar aquestes diferències en el model conceptual.

Relació és-un o és-una

Una relació de generalització/especialització a vegades també s'anomena *relació* "és-un" o "és-una" per la manera de fer referència al concepte. Per exemple, un estudiant "és una" persona, un professor "és una" persona.

Exemples de generalització/especialització de tipus d'entitats

Dins un entorn universitari podem modelitzar el tipus d'entitat 'persona', que conté atributs com ara el nom, els cognoms, el DNI, la data de naixement, el sexe, el telèfon o el correu electrònic. Però dins d'aquest tipus d'entitat podem trobar algunes entitats amb característiques pròpies que ens interessa modelitzar. Les entitats que corresponen a estudiants tenen algunes característiques pròpies rellevants (número d'identificació universitari, any de primera matrícula, etc.) que cal incloure en el model. D'altra banda, podem definir altres entitats que pertanyin al col·lectiu de professors i sobre els quals volem representar altres tipus d'atributs (departament al qual pertanyen, especialitat, etc.) o al col·lectiu de personal administratiu de la universitat, que té altres requisits (càrrec, secció, etc.). És a dir, totes les entitats tenen un conjunt de característiques comunes important, però també tenen algunes diferències entre si que cal poder representar en el model de base de dades.

La **generalització/especialització** és una relació entre un tipus d'entitat general, anomenat *superclasse* o *tipus d'entitat pare*, i un tipus d'entitat més específic, anomenat *subclasse* o *tipus d'entitat fill*.

Aquesta relació permet definir els tipus d'entitats en dos nivells: en el primer nivell hi trobem el tipus d'entitat general (*superclasse*), que conté les característiques comunes a totes les entitats, i en el segon nivell trobem els tipus d'entitats que contenen les característiques pròpies de les diferents especialitzacions (*subclasses*).

Qualsevol entitat d'una subclasse ha de ser també entitat de la superclasse. En sentit contrari no és imprescindible, és a dir, hi poden haver entitats de la superclasse que no pertanyin a cap subclasse.

Els atributs o relacions de la superclasse es propaguen cap a les subclasses. És el que s'anomena **herència de propietats**.

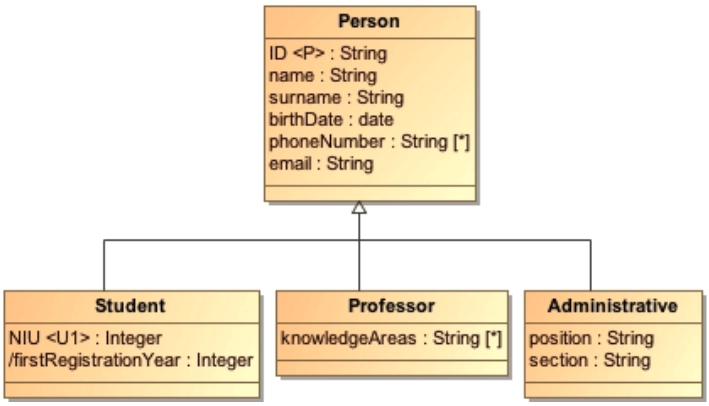
Els atributs de les subclasses reben el nom d'**atributs específics** o **atributs locals**. Només les entitats que pertanyen a una subclasse determinada tenen valor per a aquests atributs. Per contra, qualsevol entitat que pertanyi a qualsevol de les subclasses ha de tenir valor pels atributs que es defineixen en la superclasse. En notació UML aquesta relació es representa mitjançant una fletxa buida que surt de les subclasses i que es dirigeix cap a la superclasse.

Model de generalització/especialització

Seguint l'exemple anterior, la figura 30 mostra un model de generalització/especialització per a representar els diferents tipus de persones que podem trobar en l'entorn universitari que hem descrit. La superclasse de la relació és el tipus d'entitat 'persona' (*Person*), que inclou les dades bàsiques que ens interessa emmagatzemar per a qualsevol persona del model. A continuació trobem les tres subclasses descrites abans, que contenen la informació rellevant dels estudiants (*Student*), professors (*Professor*) i personal d'administració (*Administrative*).

Per tant, qualsevol entitat de la subclasse 'estudiant' té tots els atributs de la superclasse 'persona' i a més els atributs específics o locals.

Figura 30. Exemple de generalització/especialització



Les subclasses s'anomenen **especialitzacions de la superclasse**, i la superclasse s'anomena **generalització de les subclasses**.

En el moment de fer el disseny del model, es pot optar per una estratègia descendent (*top-down*) i començar dissenyant la superclasse com a tipus d'entitat principal, i posteriorment refinar el disseny afegint les característiques específiques de les subclasses. Aquesta metodologia de disseny rep el nom de **procés d'especialització**.

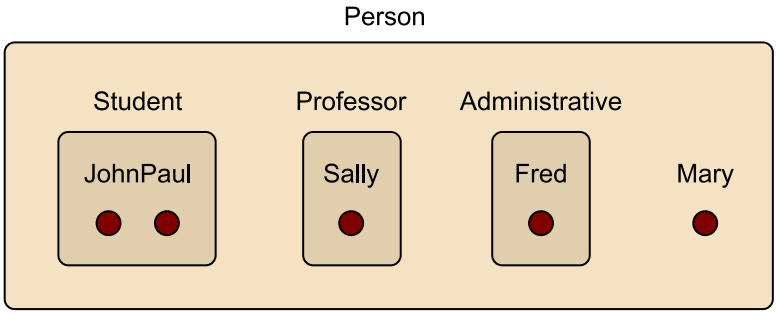
L'estratègia contrària, basada en una metodologia ascendent (*bottom-up*), consisteix a dissenyar les subclasses i posteriorment, en el moment d'adonar-se de les característiques comunes entre aquestes subclasses, definir la superclasse que les uneix. Aquesta metodologia rep el nom de **procés de generalització**.

Relació entre les entitats en una generalització/especialització

La figura 31 mostra una possible relació entre les entitats de l'exemple anterior. Podem veure que hi ha cinc entitats de la superclasse 'persona' (*Person*), amb els noms *John*,

Mary, *Paul*, *Fred* i *Sally*, dues de les quals són estudiants i pertanyen a la subclasse 'estudiant' (*Student*), una és un professor (*Professor*) i una altra és administratiu (*Administrative*). L'entitat *Mary* no pertany a cap de les subclasses, i només pertany a la superclasse.

Figura 31. Exemple de la relació entre les entitats en una generalització/especialització

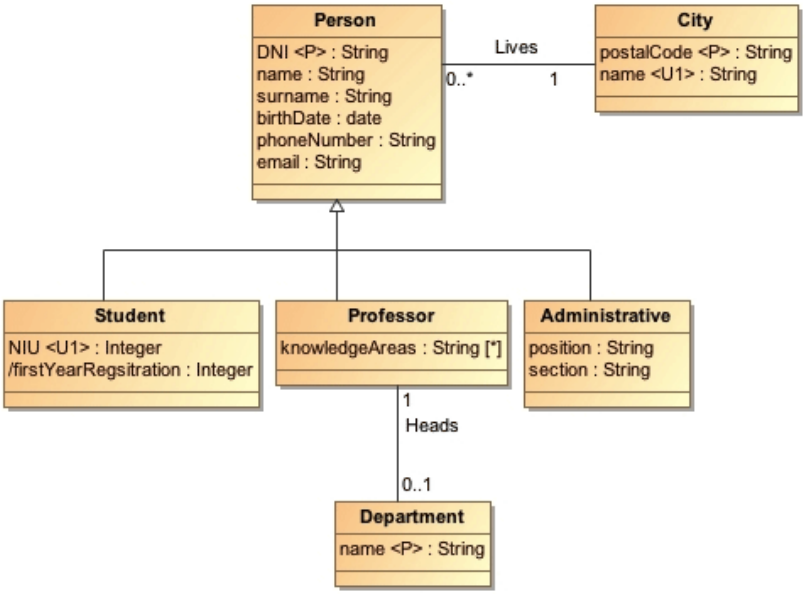


Tant les superclasses com les subclasses poden intervenir en relacions sense cap tipus de restricció.

Generalització/especialització amb tipus de relacions

En la figura 32 es pot veure com la superclasse 'persona' (*Person*) participa en la relació 'viu' (*Lives*) amb l'entitat 'població' (*City*) i la subclasse 'professor' (*Professor*) participa en la relació 'dirigeix' (*Heads*) amb l'entitat 'departament' (*Department*). Evidentment no té sentit considerar que qualsevol entitat de la superclasse 'persona' pot dirigir un departament i cal indicar en el model conceptual que només aquelles persones que són entitats de la subclasse 'professor' poden dirigir un departament.

Figura 32. Exemple de generalització/especialització amb tipus de relacions



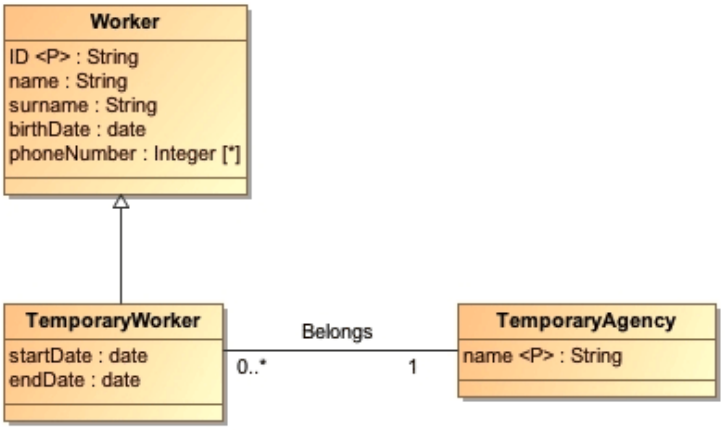
Hi ha dos motius principals que indueixen a l'ús de les relacions de generalització/especialització entre tipus d'entitats:

1) El primer motiu es deu al fet que en determinades situacions cal modelitzar casos en què alguns atributs només són aplicables a algunes de les entitats, però no a tota la superclasse. Per tant, es defineix una relació de generalització/especialització que permet agrupar aquestes entitats en una subclasse i afe-

gir els atributs necessaris, mentre continuen formant part de la superclasse i compartint la resta d'atributs amb totes les entitats de la superclasse. La figura 32 mostra un exemple d'aquest cas.

2) La segona situació es produeix quan cal modelitzar tipus de relacions que només es poden establir amb algunes entitats de la superclasse. Per exemple, suposem que hem creat un tipus d'entitat 'treballador' (*Worker*) per a modelitzar els treballadors d'una fàbrica i volem establir un tipus de relació que indiqui de quina empresa de treball temporal provenen. Lògicament, aquesta relació només és aplicable als treballadors que siguin temporals i no als treballadors fixos de l'empresa. Per a modelitzar aquesta situació podem crear una subclasse per a identificar els 'treballadors temporals' (*TemporaryWorker*) i que relacioni amb el tipus d'entitat de les empreses temporals (*EmploymentAgency*). D'aquesta manera només els treballadors que pertanyin a la subclasse podran estar relacionats amb l'empresa de treball temporal. La figura 33 mostra un esquema que exemplifica aquest cas.

Figura 33. Exemple de generalització/especialització que incorpora relacions amb la subclasse



Per tant, un procés de generalització/especialització permet:

- Crear una taxonomia de tipus d'entitats, tot definint un conjunt d'entitats tipus específiques (subclasses) a partir del tipus d'entitat genèrica (superclasse).

- Establir atributs específics addicionals a cada tipus d'entitat específic (subclasse).

- Establir tipus relacions addicionals entre els tipus d'entitats específics (subclasses) i altres tipus d'entitat.

Observació

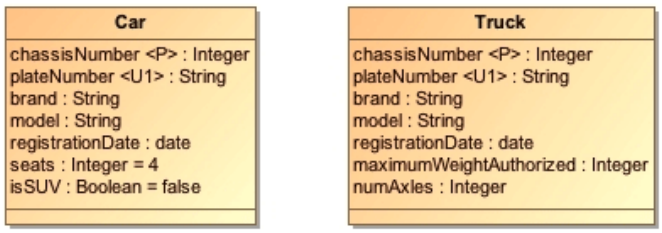
A partir d'aquest punt, anomenarem *superclasse* els tipus d'entitat genèrics, i *subclasse*, els tipus d'entitat específics.

Generalització de tipus d'entitat

Suposem que volem modelitzar una base de dades per a mantenir informació sobre els vehicles que té una empresa. En un moment del disseny apareix el tipus d'entitat 'cotxe' (*Car*), de la qual volem emmagatzemar el número de bastidor, la matrícula, la marca, el model, la data de matriculació, el nombre de places disponibles i si és un vehicle tot camí o no. Més endavant, en la mateixa fase de disseny, ens apareix el tipus d'entitat 'camió' (*Truck*), de la qual volem emmagatzemar el número de bastidor, la matrícula, la

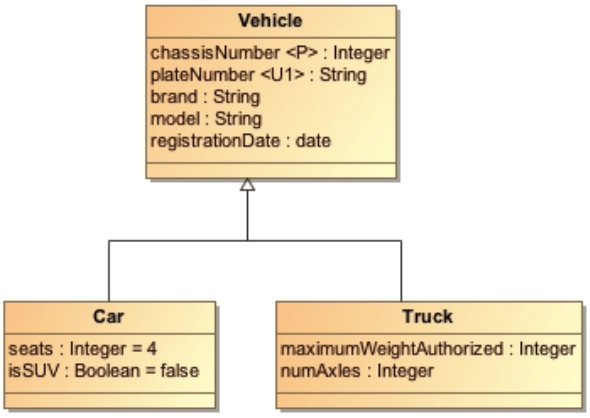
marca, el model, la data de matriculació, el pes màxim autoritzat de càrrega i el nombre d'eixos del camió. La figura 34 mostra el disseny conceptual dels dos tipus d'entitats.

Figura 34. Exemple dels tipus d'entitat 'cotxe' (*Car*) i 'camió' (*Truck*)



En aquest punt, però, el dissenyador s'adona que els dos tipus d'entitat comparteixen una bona part dels atributs i que seria interessant generalitzar aquesta part comuna per obtenir una nova superclasse que permeti identificar un vehicle, sia camió o cotxe. Per tant, es crea la superclasse 'vehicle' (*Vehicle*), que conté tots els atributs comuns (el número de bastidor, la matrícula, la marca, el model i la data de matriculació), i llavors es creen dues subclasses amb els atributs propis de cadascuna. La figura 35 mostra el disseny conceptual després del procés de generalització.

Figura 35. Exemple de generalització dels tipus d'entitat 'cotxe' (*Car*) i 'camió' (*Truck*)



3.1.1. Restriccions en la generalització/especialització

Hi ha tres tipus de restriccions bàsiques en els processos de generalització/especialització.

1) Pertinença definida per atribut o usuari

En alguns casos la superclasse conté un conjunt d'atributs que indiquen, per a cada entitat, a quina de les subclasses pertany. En aquests casos s'anomenen **subclasses de predicat definit** o **subclasses de condició definida**.

Exemple de pertinença definida per atribut o usuari

Continuant amb l'exemple anterior dels vehicles, podem especificar a quina subclasse pertany cada entitat a partir d'un atribut que defineix el tipus de vehicle (*vehicleType*) de la superclasse, que prengui valor igual a *car* en cas que l'entitat hagi de pertànyer a la subclasse *Car* o que prengui valor igual a *truck* en cas que hagi de pertànyer a la subclasse *Truck*.

En cas que totes les subclasses tinguin una condició de pertinença en el mateix atribut de la superclasse, l'atribut rep el nom d'**atribut definitori** o **discriminador** i la relació de generalització/especialització es diu que és d'**atribut de-**

finit. En cas que no hi hagi cap atribut que permeti identificar la pertinença de les entitats a les diferents subclasses es diu que és una generalització/especialització **definida per l'usuari**.

En UML, l'atribut discriminador, si n'hi ha, s'indica en la fletxa de la relació de generalització/especialització. La figura 37 mostra un exemple de generalització/especialització d'atribut definit.

2) Restricció d'exclusivitat

Una altra restricció que es pot aplicar a les generalitzacions/especialitzacions és la **restricció d'exclusivitat**, que indica si les subclasses han ser disjuntes entre si. Dit d'una altra manera, expressa si una mateixa entitat pot pertànyer a més d'una subclasse. En aquest sentit la generalització/especialització pot ser:

- **Disjunta**[3]: una entitat només pot pertànyer a una única subclasse.
- **Encavalcada**[4]: una entitat pot pertànyer a més d'una subclasse al mateix temps.

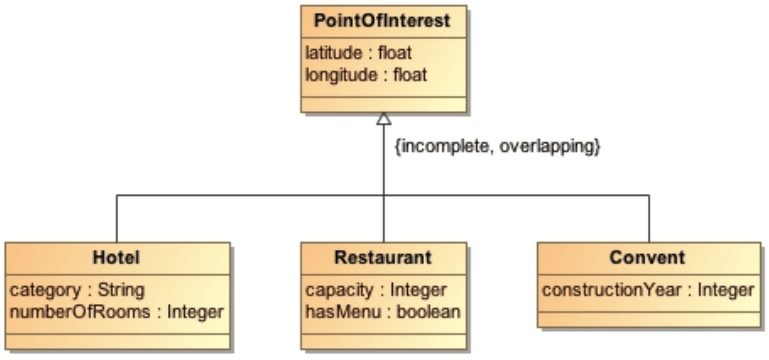
(3) En anglès, *disjoint*.

(4) En anglès, *overlapping*.

Generalització/especialització encavalcada

La figura 36 mostra un exemple de generalització/especialització encavalcada. En aquest model podem veure la superclasse 'punt d'interès' (*PointOfInterest*), que permet emmagatzemar informació sobre diferents punts d'interès turístic d'una zona. Aquest concepte es pot especialitzar en diferents subclasses. Per exemple, podem dir que els hotels (*Hotel*), els restaurants (*Restaurant*) i els convents (*Convent*) són especialitzacions de 'punt d'interès'. Cal tenir en compte que hi pot haver hotels que també són restaurants o convents que fan funcions d'hotel. Això implica que una mateixa entitat pot pertànyer a més d'una subclasse al mateix temps. Per tant, podem dir que aquesta generalització/especialització és de tipus encavalcada.

Figura 36. Exemple de generalització/especialització encavalcada



3) Restricció de participació

Finalment, la darrera de les restriccions referents a la generalització/especialització s'anomena **restricció de participació** i indica l'obligatorietat de les entitats de pertànyer a alguna de les subclasses o no. Seguint aquest enfocament es pot establir la classificació següent:

- **Total**[5]: tota entitat de la superclasse ha de pertànyer a alguna de les subclasses.

(5)En anglès, *complete*.

- **Parcial**[6]: hi poden haver entitats de la superclasse que no pertanyin a cap de les subclasses.

(6)En anglès, *incomplete*.

En una generalització/especialització amb restricció de participació total, totes les entitats han de pertànyer a una o més subclasses. Per tant, totes les entitats pertanyen a alguna de les subclasses i no hi poden haver entitats que pertanyin exclusivament a la superclasse. En aquests casos es diu que la superclasse és un **tipus d'entitat abstracta**, ja que no conté cap entitat. Els tipus d'entitats abstractes en UML s'indiquen amb el nom del tipus d'entitat en lletra cursiva.

Si en una generalització/especialització no s'indiquen les restriccions, per defecte s'assumeix que es tracta d'una relació **definida per l'usuari**, **encavalcada** i **parcial**.

La figura 37 mostra la nomenclatura per a identificar tots dos casos en UML. En la mateixa relació s'indica entre claus {} els conceptes de *total* o *parcial* (*complete*/*incomplete*) i *disjunta* o *encavalcada* (*disjoint*/*overlapping*).

Generalització/especialització disjunta parcial i completa

Les figures 37 i 38 mostren un esquema similar, amb una diferència important en el disseny.

En la figura 37 podem veure la superclasse 'vehicle' (*Vehicle*) i les subclasses 'cotxe' (*Car*) i 'camió' (*Truck*) amb una herència disjunta i incompleta. Això ens indica, en primer lloc, que una entitat no pot pertànyer a les dues subclasses al mateix temps, és a dir, que no hi pot haver un vehicle que sigui cotxe i camió alhora, i, en segon lloc, que algunes entitats pot ser que no pertanyin a cap de les subclasses, és a dir, que hi poden haver instàncies de vehicles que no són cotxes ni camions.

En la figura 38, en canvi, es presenta una herència similar, però en aquest cas és completa. El fet que sigui completa implica que la superclasse no podrà contenir entitats; per tant, serà un tipus d'entitat abstracta. És a dir, que qualsevol entitat de 'vehicle' haurà de ser un cotxe o un camió.

Figura 37. Exemple de generalització/especialització disjunta (*disjoint*) i parcial (*incomplete*)

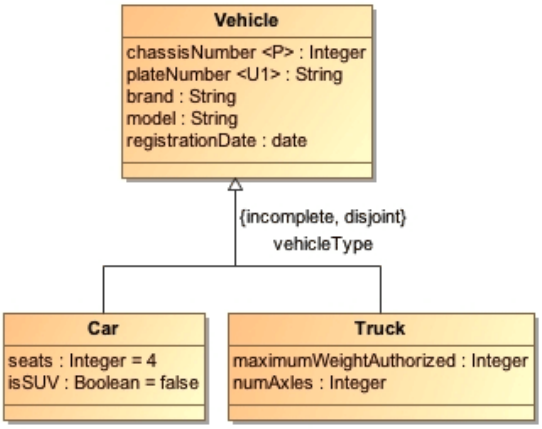
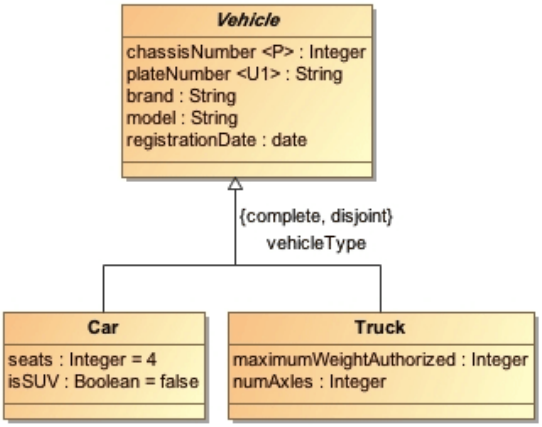


Figura 38. Exemple de generalització/especialització disjunta (*disjoint*) i completa (*complete*)



Els conceptes de *restriccions d'exclusivitat* i de *participació* són independents entre si, i per tant, es poden combinar per a donar lloc a quatre possibles restriccions de generalització/especialització:

- Disjunta i total
- Disjunta i parcial
- Encavalcada i total
- Encavalcada i parcial

3.1.2. Herència simple i múltiple

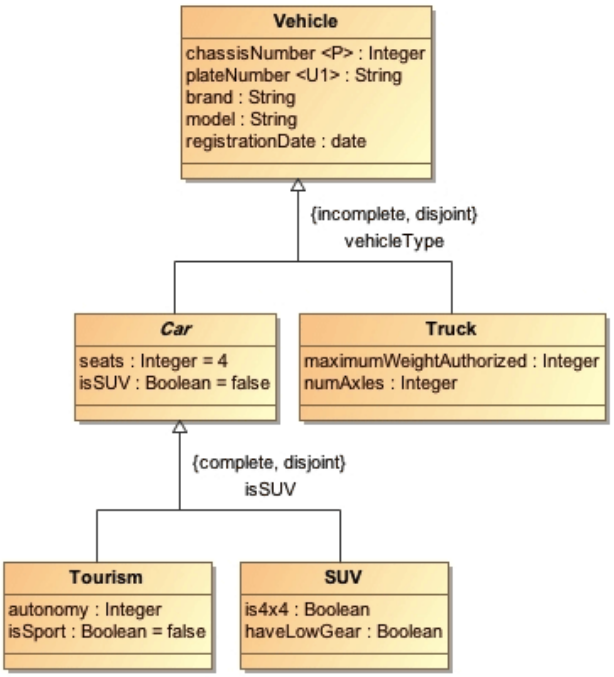
Una subclasse pot tenir alhora subclasses que en permetin refinar més els conceptes.

L'**herència simple** en un procés de generalització/especialització es dóna quan tots els tipus d'entitats involucrades en la relació només tenen una superclasse, és a dir, hi ha un únic tipus d'entitat pare o classe arrel. Com a resultat d'aquesta condició es genera una estructura en forma d'arbre.

Herència simple

La figura 39 mostra una relació d'herència simple, en què a partir de la superclasse arrel 'vehicle' (*Vehicle*) i de l'atribut discriminador 'tipus de vehicle' (*vehicleType*) es refinen dues subclasses segons si el vehicle és un cotxe (*Car*) o un camió (*Truck*). Al mateix temps, el tipus d'entitat 'cotxe' s'especialitza en dues subclasses, segons si és un turisme (*Tourism*) o un vehicle tot camí (*SUV*). El tipus d'entitat 'cotxe' és superclasse en aquesta nova relació i alhora subclasse en la relació amb el tipus d'entitat 'vehicle'. Cal fer notar, a més, que el tipus d'entitat 'cotxe' esdevé abstracta, ja que l'especialització és completa.

Figura 39. Exemple d'herència simple



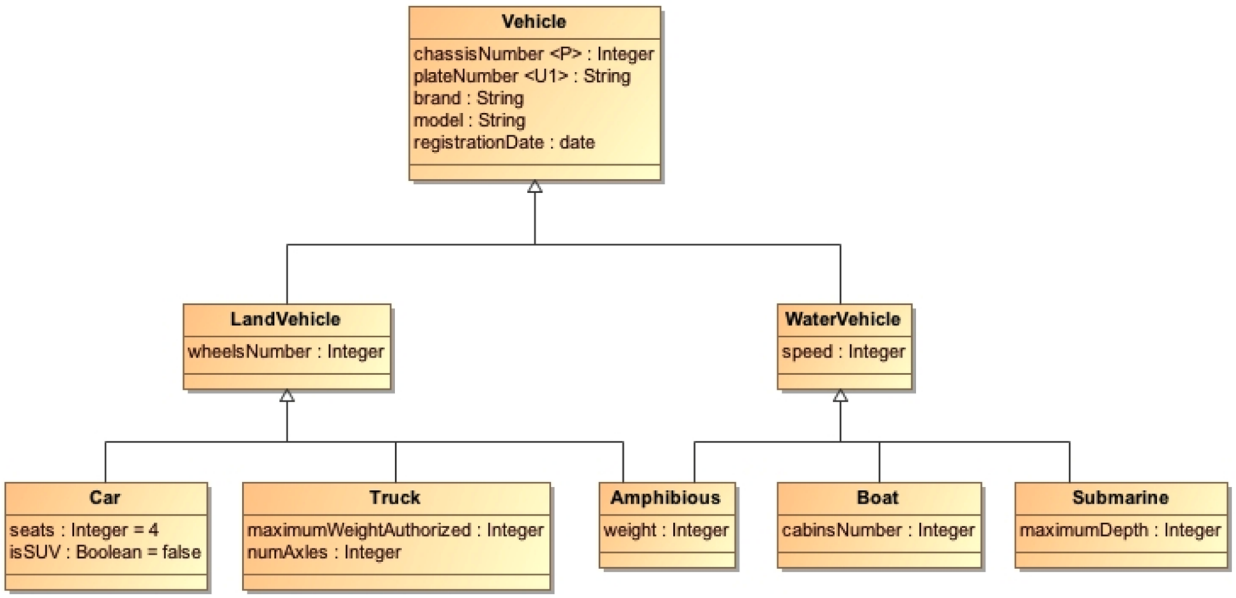
El tipus d'entitat origen de la jerarquia, en aquest cas el tipus d'entitat 'vehicle', és la **superclasse arrel** o **tipus d'entitat base**. Les subclasses que no tenen altres subclasses, en aquest cas les classes 'turisme', 'tot camí i 'camió' són **classes filla**.

L'**herència múltiple** en un procés de generalització/especialització es dóna quan algun dels tipus d'entitat involucrats en la relació té dues o més superclasses. En aquests casos s'obté una estructura tipus graf dirigit.

Herència múltiple

La figura 40 mostra una relació d'herència múltiple. A partir del tipus d'entitat base, en aquest exemple el tipus d'entitat 'vehicle' (*Vehicle*), creem una especialització en què indiquem si es tracta d'un vehicle terrestre (*LandVehicle*) o aquàtic (*WaterVehicle*). Els vehicles terrestres es poden especialitzar en cotxes (*Car*), camions (*Truck*) o amfibis (*Amphibious*). Els vehicles amfibis es poden desplaçar per terra i per medis aquàtics, i per tant, aquest tipus d'entitat també és una especialització del tipus d'entitat 'vehicle aquàtic', juntament amb els tipus d'entitats 'vaixell' (*Boat*) i 'submarí' (*Submarine*).

Figura 40. Exemple d'herència múltiple



El tipus d'entitat 'amfibi' s'anomena *subclasse compartida*, ja que té més d'una superclasse o tipus d'entitat pare. Cal fer notar que si un atribut o relació que s'origina en una mateixa superclasse és heretat més d'una vegada per camins diferents de l'entramat, només es pot incloure una sola vegada en la subclasse compartida. És a dir, els atributs i les relacions de la classe 'vehicle' només poden aparèixer una única vegada en la subclasse compartida.

És important adonar-se que l'existència d'una única subclasse compartida provoca que la relació sigui d'herència múltiple en lloc de simple.

Observació

Una relació d'herència múltiple no pot contenir cicles.

3.1.3. Classificació múltiple

Un **model de classificació múltiple** és aquell que permet que una entitat sigui instància de dos tipus d'entitats, E_1 i E_2, de manera que E_1 no és subclasse de E_2, E_2 no és subclasse de E_1 i no hi ha cap tipus d'entitat E_3 que sigui subclasse de E_1 i E_2.

Les relacions de generalització/especialització encavalcades permeten classificació múltiple, ja que una entitat pot pertànyer a diferents subclasses alhora. La figura 36 mostra un exemple de generalització/especialització encavalcada que permet classificació múltiple.

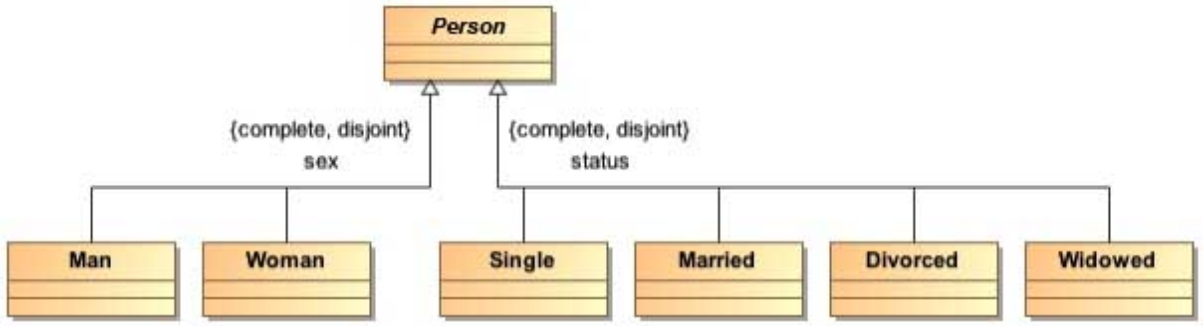
Un altre tipus de classificació múltiple es dóna quan la superclasse té dues o més jerarquies d'especialització segons diferents discriminants. Cada entitat pot pertànyer a diferents subclasses alhora.

Generalització/especialització amb classificació múltiple

Partint del tipus d'entitat 'persona' (*Person*) podem establir un procés d'especialització per a definir el seu estat civil, que indiqui si la persona és soltera (*Single*), casada (*Marri-*

ed), divorciada (*Divorced*) o vídua (*Widowed*). D'altra banda, una persona també es pot especialitzar, segons el sexe, en 'home' (*Man*) o 'dona' (*Woman*). La figura 41 mostra el model conceptual descrit.

Figura 41. Exemple de generalització/especialització amb classificació múltiple



3.2. Agregació i composició

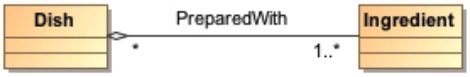
L'**agregació** és un tipus de relació entre entitats que indica una relació de "part-tot" entre les instàncies.

En aquesta associació es distingeix la part que representa el "tot", i que rep el nom de **compost**, i les diferents "parts" que el componen, i que reben el nom de **components**.

Agregació

La figura 42 mostra un exemple d'agregació entre un plat (*Dish*) i els ingredients amb què s'ha preparat (*Ingredient*). Com es pot veure en el model conceptual, un plat ha d'estar compost per un o més ingredients, mentre que un mateix ingredient pot estar inclòs en qualsevol nombre de plats.

Figura 42. Exemple d'agregació



La distinció entre tipus de relació i agregació és subtil, i sovint subjectiva. En general es considera que cal que hi hagi una certa relació d'assemblatge, sia física o lògica, entre les entitats per a parlar d'*agregació* en lloc de *tipus de relació*. L'única restricció que afegeix l'agregació respecte al tipus de relació és que les cadenes d'agregacions entre entitats no poden formar cicles.

La **composició** és un tipus concret d'agregació en el qual la multiplicitat del tipus d'entitat composta ha de ser com a màxim 1 i els components presenten dependència d'existència vers el compost que tenen.

És a dir, la composició imposa dues restriccions respecte de l'agregació:

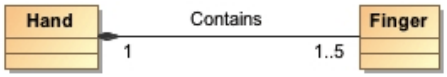
- Cada component pot estar-hi present en un únic compost (la multiplicitat del tipus d'entitat composta ha de ser com a màxim 1).

- Si s'elimina l'element compost, cal eliminar tots els components vinculats a aquest element (els components presenten dependència d'existència vers el compost que tenen).

Composició

Podem pensar en el conjunt de dits que formen una mà. La figura 43 mostra el model conceptual en què es representa la composició d'una mà (tipus d'entitat *Hand*) com a conjunt de dits (tipus d'entitat *Finger*) i en què cada dit només pot pertànyer a una sola mà.

Figura 43. Exemple de composició



Observació

Cal destacar que l'agregació es marca amb un rombe buit, mentre que la composició es marca amb un rombe negre.

A diferència de l'agregació, en la composició hi ha una forta dependència entre la part composta i la part component, de manera que els components no poden viure sense el compost.

3.3. Restriccions d'integritat

3.3.1. Restriccions en els tipus d'entitat

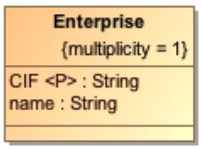
La **multiplicitat de tipus d'entitat** estableix un rang de possibles cardinalitats per a les entitats d'un determinat tipus d'entitat. És a dir, indica quantes instàncies d'un determinat tipus d'entitat poden aparèixer en la base de dades.

Generalment, i per defecte, aquesta restricció pren el valor indefinit, però algunes vegades pot ser interessant poder determinar un interval en el nombre d'ocurrències d'un tipus d'entitat, especialment en els casos en què només hi pot haver una entitat.

Tipus d'entitat amb multiplicitat 1

Suposem que volem modelitzar una base de dades per a gestionar petites o mitjanes empreses. A més de tota la informació de clients, proveïdors, productes i altres coses, cal emmagatzemar les dades de l'empresa mateixa (CIF, nom, adreça, etc.) per poder mostrar aquesta informació en factures, llistes i altres coses. Una manera de modelitzar aquest requisit és crear un tipus d'entitat que permeti emmagatzemar aquesta informació. En aquest cas, cal especificar una multiplicitat de tipus d'entitat igual a 1, ja que només hi ha d'haver una entitat amb les dades de l'empresa mateixa. La figura 44 mostra el diagrama UML d'aquest tipus d'entitat.

Figura 44. Tipus d'entitat amb multiplicitat 1



3.3.2. Restriccions en els atributs

En la definició dels atributs hem vist algunes restriccions bàsiques associades als atributs. Aquestes restriccions són les següents:

- **Restriccions de domini**: són les restriccions associades al conjunt de possibles valors legals que pot prendre un atribut. En aquest sentit, es pot especificar el tipus de dades que utilitzarà l'atribut (cadena de text, enter, real, booleà, etc.) i les restriccions addicionals específiques per a cada tipus de dades (per exemple, la longitud en el cas de cadenes de text o un interval vàlid en el cas de valors enters).

- **Atributs derivats**: són atributs que calculen el valor a partir d'altres atributs i que, per tant, implícitament són atributs de només lectura des del punt de vista de l'usuari o de l'aplicació. El seu valor, però, pot canviar si canvia el valor dels atributs de què depenen.

Restriccions de canviabilitat

Aquesta restricció indica si els valors d'un atribut poden canviar o no.

Hi ha quatre valors permesos:

- 'Canviable' (*changeable* o *unrestricted*): és el valor per defecte. Indica que es permet qualsevol canvi sobre l'atribut.

- 'Congelat' (*frozen* o *readOnly*): indica que una vegada s'ha assignat valor a l'atribut no es podrà modificar ni eliminar.

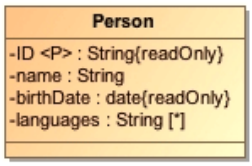
- 'Només-afegir' (*addOnly*): indica que una vegada s'ha assignat valor a l'atribut no es podrà modificar ni eliminar. Però, en cas de ser un atribut multivalor, sí que es podran assignar nous valors per a aquest atribut.

- 'Només-eliminar' (*removeOnly*): indica que l'única operació permesa és eliminar el valor de l'atribut.

Restriccions de canviabilitat en els atributs

La figura 45 mostra el tipus d'entitat *Person*, en què podem veure un atribut no restringit o canviable (*name*) i dos atributs congelats o de només lectura (*ID* i *birthDate*).

Figura 45. Exemple de restriccions de canviabilitat en els atributs



3.3.3. Restriccions en els tipus de relacions

A part de les restriccions en la multiplicitat en els tipus de relacions, és possible expressar altres restriccions sobre les relacions en els diagrames UML. Tot i que la majoria de casos queden coberts amb les restriccions que hem vist fins ara, aquestes noves restriccions permetran una expressivitat més gran del model conceptual. A continuació n'esmentem algunes de les més rellevants.

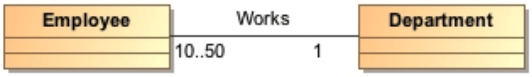
a) Restriccions de cardinalitat *min..max*

A part de totes les restriccions referents a la cardinalitat de les relacions que hem vist, és possible indicar un valor o rang concret en la cardinalitat d'una relació. Per a indicar un rang en UML, s'expressa el valor mínim seguit de dos punts i el valor màxim (*min..max*).

Exemple

La figura 46 modelitza una situació en què un empleat (*Employee*) només pot pertànyer a un únic departament (*Department*) i cada departament ha de tenir entre 10 i 50 empleats.

Figura 46. Exemple de restricció de cardinalitat *min..max*



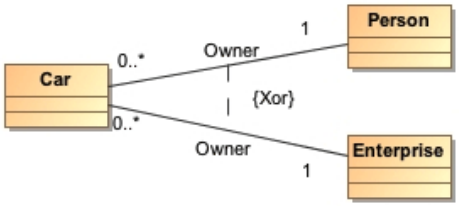
b) Restricció *xor*

Aquesta restricció es pot aplicar quan hi ha diverses relacions lligades a un mateix tipus d'entitat. La restricció indica que una entitat només pot participar en un dels tipus de relació units per aquesta restricció.

Exemple

Un cotxe pot pertànyer a una persona o a una empresa. Per a modelitzar aquesta situació, la figura 47 mostra com els tipus de relacions que uneixen els tipus d'entitat 'cotxe' (*Car*) amb els tipus d'entitat 'persona' (*Person*) i 'empresa' (*Enterprise*) estan relacionats amb la restricció *xor*, que indica que un cotxe ha de ser d'una persona o d'una empresa, però no de tots dos alhora.

Figura 47. Exemple de restricció *xor*



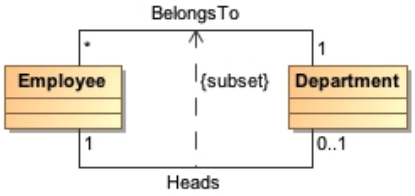
c) Restricció *subset*

Aquesta restricció indica que un tipus de relació és un subconjunt d'un altre tipus de relació.

Exemple de restricció *subset*

El cap d'un departament ha de pertànyer a aquest departament. Per a indicar aquesta restricció, podem utilitzar el concepte *subset* tal com es mostra en la figura 48.

Figura 48. Exemple de restricció *subset*



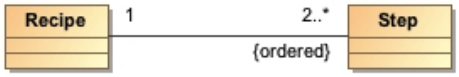
d) Restricció *ordered*

Aquesta restricció indica que hi ha una relació d'ordre en l'associació entre els tipus d'entitats.

Exemple de restricció *subset*

Una recepta de cuina està formada per un conjunt de dos o més passos que han de seguir un determinat ordre. Per a indicar aquesta restricció, podem utilitzar el concepte *ordered* tal com es mostra en la figura 49.

Figura 49. Exemple de restricció *ordered*



e) Restriccions de canviabilitat

Aquesta restricció indica si els valors de l'extrem d'una relació poden canviar o no.

Hi ha quatre valors permesos:

- 'Canviable' (*changeable* o *unrestricted*): és el valor per defecte. Indica que es permet qualsevol canvi sobre l'associació.

- 'Congelat' (*frozen* o *readOnly*): indica que una vegada la relació ha estat creada no es podrà modificar ni eliminar. Tampoc no es podran crear noves relacions.

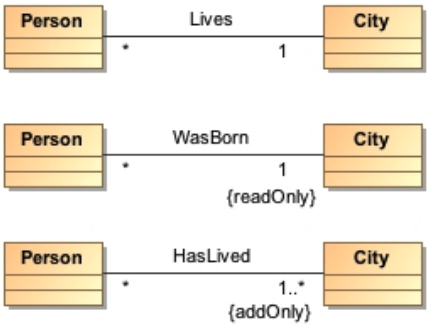
- 'Només-afegir' (*addOnly*): indica que una vegada la relació ha estat creada no es podrà modificar ni eliminar. Però sí que es podran crear noves relacions.

- 'Només-eliminar' (*removeOnly*): indica que l'única operació permesa és eliminar el valor de la relació.

Restriccions de canviabilitat en els tipus de relacions

Per a poder comparar diferents exemples relacionats amb la canviabilitat, la figura 50 ens mostra tres situacions diferents en què intervenen els mateixos tipus d'entitats. En la primera, representem la ciutat on viu una persona, que evidentment pot canviar. Per tant, l'extrem de l'associació que connecta amb el tipus d'entitat 'ciutat' (*City*) és canviable o no restringida (opció per defecte). En la segona situació, representem el lloc de naixement d'una persona, i en aquest cas etiquetem com a congelat l'extrem d'aquesta associació. Finalment, si considerem les ciutats on ha viscut una persona, etiquetem amb l'atribut 'Només-afegir', ja que la llista de ciutats on ha viscut una persona pot augmentar, però no modificar o eliminar valors existents.

Figura 50. Exemples de restriccions de canviabilitat en els tipus de relacions



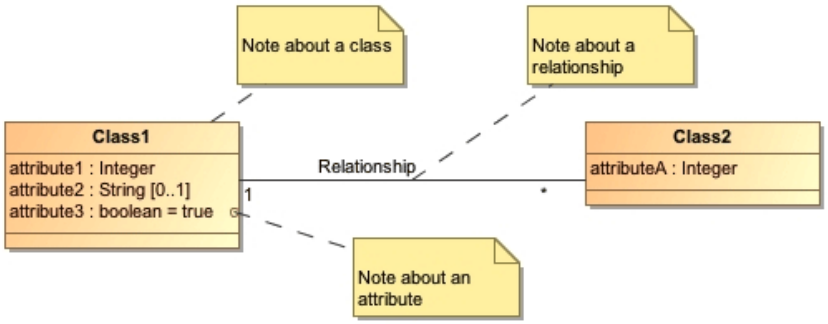
3.3.4. Altres restriccions

En l'anàlisi de requisits moltes vegades es poden trobar requisits o restriccions de caràcter semàntic. Aquestes restriccions depenen completament del problema, i cal deixar-ne constància. Generalment, s'utilitzen notes lligades als tipus d'entitats, atributs o tipus de relacions a què fan referència per a deixar-ne constància.

Ús de notes per a indicar altres restriccions

La figura 51 mostra tres notes que permeten notacions de qualsevol tipus per a clarificar conceptes del model, associades a un tipus d'entitat, un atribut o un tipus de relació.

Figura 51. Exemple d'ús de notes per a indicar requisits o restriccions del model



3.4. Modelització de dades històriques

Les bases de dades modelitzen l'estat d'alguns aspectes del món exterior. Generalment, només modelitzen un estat del món real –l'estat actual–, i no emmagatzemen informació sobre els estats anteriors. Quan es produeix un canvi d'estat en el món real, la base de dades s'actualitza i es perd la informació sobre els estats anteriors. En algunes aplicacions, però, és important poder emmagatzemar i recuperar informació sobre els estats anteriors del sistema.

Bàsicament hi ha dos tipus de requisits a l'hora de modelitzar el temps associat a un estat del món real:

1) Instant de temps: en algunes accions només cal assenyalar en quin instant de temps han ocorregut. Per a modelitzar aquest tipus d'activitat, es pot afegir una marca de temps que permeti identificar en cada instància el moment de temps que hi ha associat. Aquest comportament es pot modelitzar de dues maneres:

- Afegint un atribut de tipus 'data i hora'. Cada instància indica el valor de temps en aquest atribut.

- Afegint una relació amb el tipus d'entitat 'data' (*Date*), que conté un atribut de tipus 'data i hora'.

2) Interval de temps: altres accions tenen un temps de vida. És a dir, són actives durant un interval de temps específic. En aquests casos cal indicar l'interval de temps en què és vàlida cadascuna de les entitats. Aquest comportament es pot modelitzar de dues maneres:

- Si es vol aplicar aquest criteri sobre un concepte que està modelitzat com un tipus d'entitat, es poden afegir dos atributs de tipus data i hora en l'entitat per a indicar l'inici i el final de l'interval.

- Si es vol aplicar aquest criteri sobre un concepte que està modelitzat com una relació, cal afegir en la relació un tipus d'entitat 'data' (*Date*) que determinarà els temps inicial i final de l'interval.

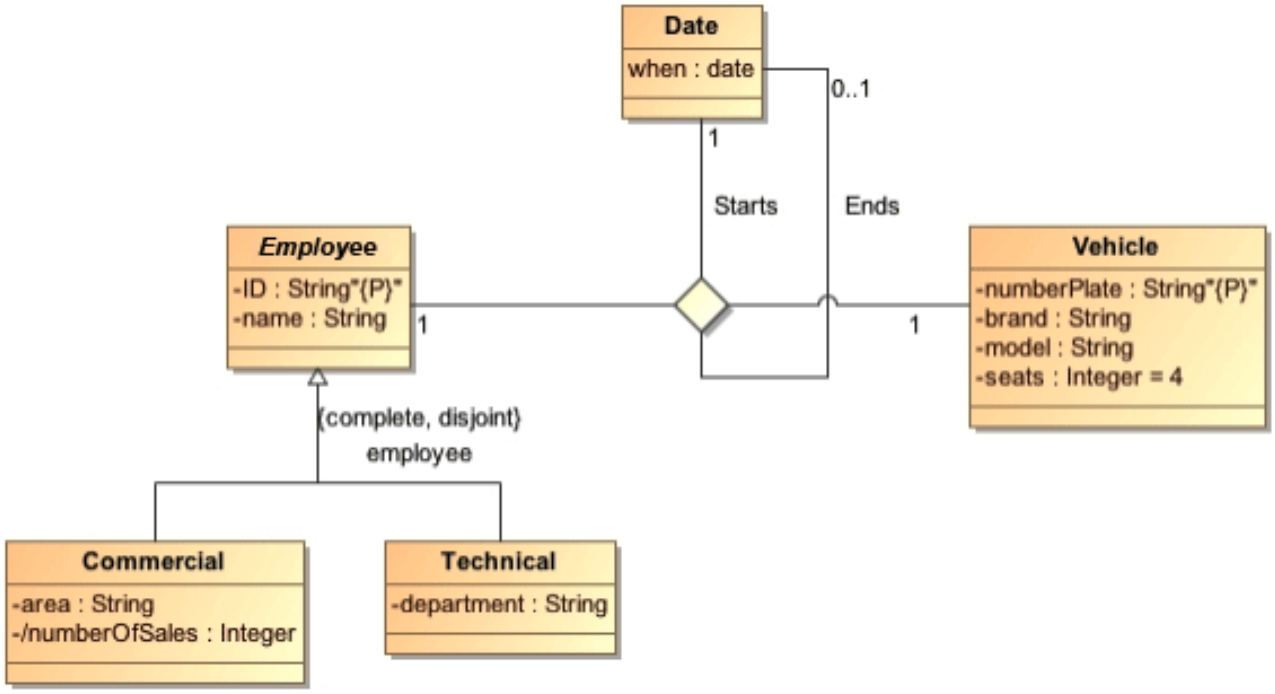
Modelització de dades històriques

La lectura d'un sistema de monitoratge d'una fàbrica anirà prenent diferents lectures en diferents estats de temps. Ens interessa guardar el valor de la lectura i el moment de temps en què s'ha produït aquesta lectura.

En altres situacions ens interessa definir, no un instant de temps, sinó un interval de temps. Per exemple, suposem que una empresa té un conjunt de vehicles amb diferents característiques i un conjunt de comercials i personal tècnic que en fa ús. Segons les tasques i les visites de cada dia, cadascun dels comercials i els tècnics pot necessitar un vehicle diferent. Per tant, a aquesta empresa li interessarà saber quin cotxe ha estat fent servir cadascun dels comercials i els tècnics en cada moment (per exemple, en cas de rebre una sanció de trànsit). Per a modelitzar aquesta situació cal definir un tipus de relació quaternària en què intervenen un comercial, un vehicle i dues entitats de temps (una que

indica l'inici de l'associació i una altra que indica quan l'associació deixa de ser vàlida). La figura 52 mostra un model que implementa la situació descrita.

Figura 52. Exemples de situació amb modelització de dades històriques



La taula 2 mostra, a tall d'exemple, una possible configuració d'entitats de l'exemple anterior en un instant de temps concret.

Taula 2. Exemple de representació d'entitats del model de la figura 52

ID	name	numberPlate	starts	ends
33941857B	John	8754-GFD	2001-05-12 09:25	2001-05-12 17:36
33941857B	John	1258-CGC	2001-05-17 11:13	--
15488574Q	Mary	8754-GFD	2001-05-10 15:11	2001-05-10 19:47
15488574Q	Mary	1258-CGC	2001-05-13 07:02	2001-05-14 14:41
15488574Q	Mary	1126-BMR	2001-05-16 20:14	--
25486257F	Fred	1126-BMR	2001-05-11 09:13	2001-05-11 18:56

3.5. Exemple complet

A continuació plantegem un exemple complet per a poder veure un esquema conceptual en què apareguin alguns dels elements vistos en aquesta secció. Continuarem amb l'exemple de l'apartat anterior, que es basa en el disseny d'una base de dades per a la gestió universitària.

A continuació enumerem els diferents aspectes dels requisits dels usuaris que cal tenir en compte en fer el disseny conceptual de la base de dades:

1) La universitat està formada per diferents departaments. Cada departament està format per un conjunt de professors i és dirigit per un únic professor. Dels departaments ens interessa el nom i el nombre de professors, així com la seva localització física, l'adreça, tenint en compte que una mateixa adreça pot existir en més d'una ciutat.

2) De cada professor ens interessa poder emmagatzemar-ne les dades personals, com, per exemple, nom, DNI, data de naixement, sexe, direcció i números de telèfon. Dels professors que són directors de departament ens interessa poder identificar la data en què van començar a exercir aquest càrrec.

3) En la universitat s'imparteixen un conjunt d'assignatures. De cadascuna d'aquestes assignatures, interessa saber-ne el codi, el nom, el nombre de crèdits, la descripció, si té laboratori associat i els prerequisits, és a dir, altres assignatures que cal haver cursat anteriorment. A més, cal tenir en compte que una assignatura pertany a un únic departament.

4) Els departaments estan agrupats en escoles o facultats. Per exemple, el departament de matemàtiques pertany a la facultat de ciències. Un departament pertany a una única escola o facultat.

5) Els estudiants són una part molt important de la base de dades. D'aquests estudiants es vol emmagatzemar informació sobre les dades personals (nom, DNI, data de naixement, sexe, adreça i números de telèfon) i el número d'identificació universitària (conegut com a NIU).

6) Cada estudiant pot estar matriculat de diverses assignatures en cada semestre. I per cadascuna de les assignatures de què està matriculat cada semestre n'hem de poder emmagatzemar una nota final.

7) També es vol deixar constància de les dates d'inici i final de cada semestre.

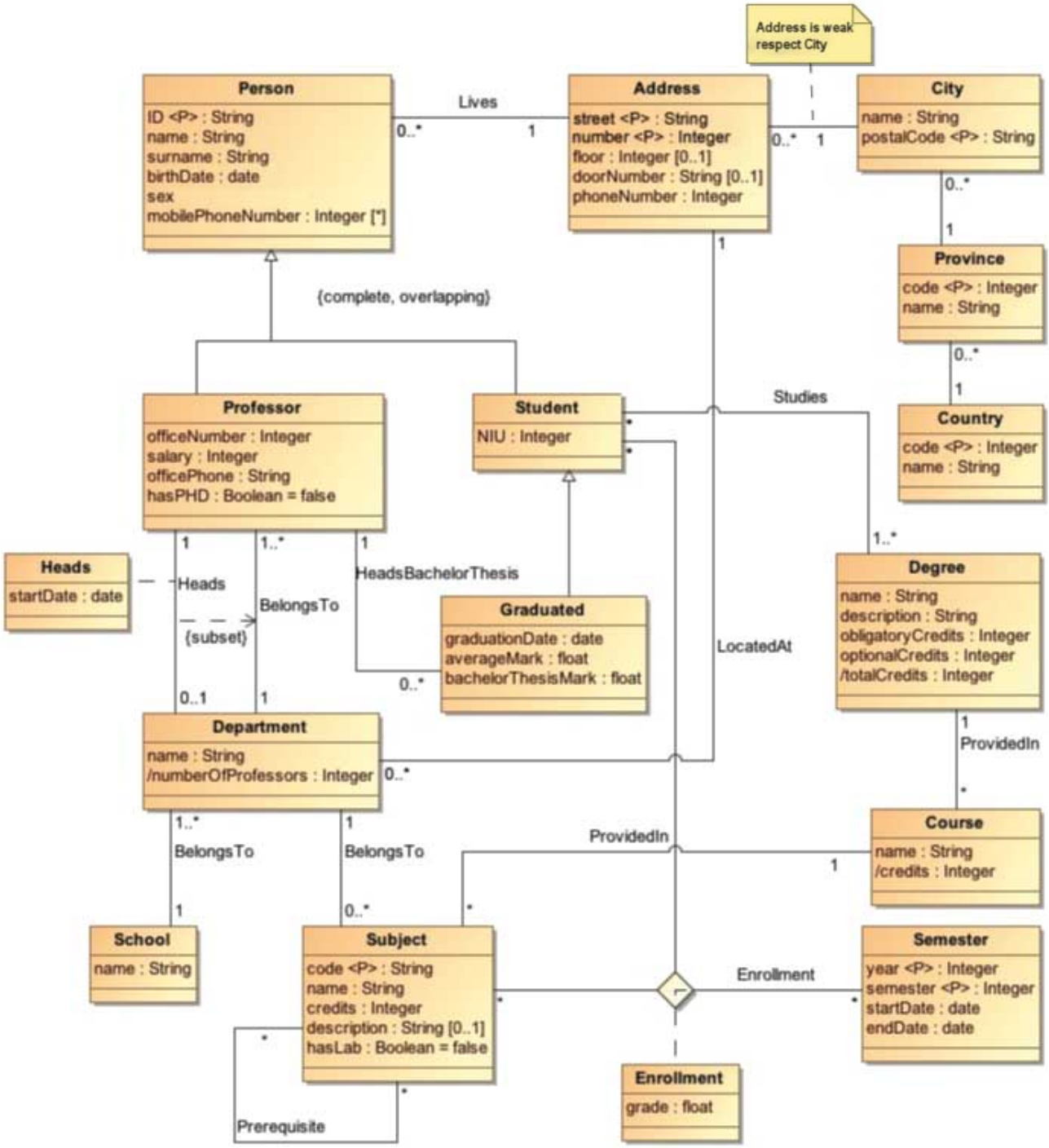
8) Les assignatures s'agrupen en els diferents cursos que formen cadascun dels graus que ofereix la universitat. Per exemple, l'assignatura d'Àlgebra pertany al primer curs del grau de Matemàtiques. Sobre cadascun del cursos només ens interessa emmagatzemar el conjunt d'assignatures que el formen. Sobre cadascun dels graus, ens interessa emmagatzemar informació referent al nombre de crèdits (obligatoris, opcionals i totals) i una descripció.

9) Els estudiants estudien un o més graus.

10) Quan un estudiant es gradua, elabora un projecte de final de carrera, sobre el qual ens interessa emmagatzemar informació sobre la data en què s'ha presentat, la nota obtinguda i el professor que l'ha dirigit.

A partir dels requisits expressats, la figura 53 mostra un diagrama del model conceptual. Com hem comentat, aquest model no és únic, sinó que hi poden haver diferents aproximacions i interpretacions vàlides per a una mateixa representació del món real.

Figura 53. Diagrama del model conceptual per a la segona aproximació de la base de dades de gestió universitària



Resum

En aquest mòdul hem vist el procés de disseny conceptual. Aquest procés és una de les etapes del disseny de bases de dades, concretament, és la segona etapa, i es fa després de l'anàlisi de requisits.

El disseny conceptual permet crear un esquema conceptual d'alt nivell i independent de la tecnologia d'implementació a partir de les especificacions i els requisits d'un problema del món real.

L'enfocament d'aquest material ens ha permès veure les bases del disseny conceptual emprant els diagrames UML com a sistema de notació.

En la primera part d'aquest material hem vist una introducció breu en què hem detallat les bases, els objectius i els requisits del disseny conceptual i dels diagrames en llenguatge UML.

En la segona part hem vist els elements bàsics de modelització en el disseny conceptual. Entre els elements principals cal destacar els tipus d'entitats, els atributs i els tipus de relacions. Tot i que aquests tres elements formen la base principal del model conceptual, també hem vist els tipus d'entitats associatives i els tipus d'entitats dèbils, que permeten una riquesa més gran en la representació del model conceptual.

Finalment, en la tercera part d'aquest material hem vist alguns dels elements avançats en el disseny conceptual. Els elements vistos en la part anterior no permeten representar situacions que trobem en el món real i que volem poder representar en el nostre model. Fruit d'aquesta necessitat, s'amplia el model per a incloure conceptes com la generalització/especialització, l'agregació o la composició, que permeten més riquesa en la representació del model conceptual. Per finalitzar aquest text, ens hem referit breument a les restriccions d'integritat bàsiques i a la modelització de dades històriques

Glossari

atribut *m* Propietat que interessa representar d'un tipus d'entitat.

classe *f* Nom que reben els tipus d'entitats en el model UML.

connectivitat *f* Expressió del tipus de correspondència entre les entitats d'una relació.

disseny conceptual *m* Etapa del disseny d'una base de dades que obté una estructura de la informació de la futura base de dades independent de la tecnologia que es vol emprar.

disseny lògic *m* Etapa del disseny d'una base de dades que parteix del resultat del disseny conceptual i el transforma de manera que s'adapti al model de sistema gestor de bases de dades amb el qual es vol implementar la base de dades.

entitat *f* Objecte del món real que podem distingir de la resta d'objectes i del qual ens interessen algunes propietats.

generalització/especialització *f* Construcció que permet reflectir que hi ha un tipus d'entitat general, anomenada *superclasse*, que es pot especialitzar en diferents tipus d'entitats més específiques, anomenades *subclasses*.

grau d'una relació *m* Nombre d'entitats que associa la relació.

llenguatge unificat de modelització *m* Llenguatge gràfic per a modelitzar, visualitzar, especificar, construir i documentar sistemes de programari o de bases de dades.
sigla **UML**
en unified modeling language

model entitat-interrelació *m* Model de dades d'alt nivell àmpliament utilitzat per al disseny conceptual de les aplicacions de bases de dades. L'objectiu principal del model ER és permetre als dissenyadors reflectir en un model conceptual els requisits del món real.

sistema gestor de bases de dades *m* Tipus de programari específic dedicat a servir de interfície entre la base de dades, l'usuari i les aplicacions que utilitzen que la utilitzen.
sigla **SGBD**
en database management system (DBMS)

tipus d'entitat associativa *m* Tipus d'entitat resultant de considerar una relació entre entitats com una nova entitat.

tipus d'entitat dèbil *m* Tipus d'entitat els atributs de la qual no la identifiquen completament, sinó que només la identifiquen de manera parcial.

tipus de relació *m* Associació entre entitats.

tipus de relació recursiva *m* Associació a la qual alguna entitat està associada més d'una vegada.

Bibliografia

Elmasri, Ramez; Navathe, Shamkant B. (2007). *Fundamentos de sistemas de bases de datos* (5a ed.). Madrid: Pearson Educación.

Teorey, Toby J. (2008). *Database design: Know it all*. Burlington: Morgan Kaufmann.

Camps, R.; Cassany M. J.; Conesa, J.; Costal, D.; Figuls, D.; Martín, C.; Rius, A.; Rodríguez, M. E.; Urpí, T. (2011). *Ús de bases de dades*. FUOC.

Date, C. J. (2001). *Introducción a los sistemas de bases de datos*. Madrid: Pearson Educación.

Rumbaugh, James; Jacobson, Ivar; Booch, Grady (2007). *El lenguaje unificado de modelado. Manual de referencia* (2a ed.). Madrid: Pearson Educación.