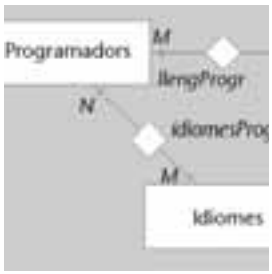
# Disseny conceptual i lògic de bases de dades

M. Elena Rodríguez González
Jaume Sistac i Planas

P01/11031/00005

Universitat Oberta de Catalunya



# Índex

# Introducció

En aquest mòdul estudiarem els aspectes més importants que estan relacionats amb el disseny de bases de dades (BD) relacionals clàssiques. Hi ha diverses alternatives per a efectuar el disseny d'una base de dades i a més, independentment de l'alternativa triada, el procés de disseny d'una base de dades es pot estructurar en diferents etapes, si bé l'objectiu cabdal és l'obtenció del disseny conceptual, el disseny lògic i el disseny físic de la base de dades.

A més, el disseny d'una base de dades ha de complir uns requisits de qualitat. En aquest sentit, estudiarem com la teoria de la normalització formalitza un conjunt d'idees simples que tenen com a objectiu garantir un bon disseny de bases de dades relacionals clàssiques.

# Objectius

Amb l'estudi dels materials associats a aquest mòdul, assolireu els objectius següents:

1. Valorar que el fet de tenir un sistema de gestió de bases de dades no significa que es puguin resoldre totes les qüestions sobre una empresa, sinó que és necessari dissenyar la base de dades adequada per a l'empresa.

2. Comprendre com el procés de disseny d'una base de dades es pot estructurar en diverses etapes.

3. Conèixer en profunditat els diferents models que s'empren en l'elaboració del disseny conceptual i el disseny lògic d'una base de dades relacional.

4. Ser conscient de les dificultats per a construir el disseny conceptual i el disseny lògic d'una base de dades en l'estat actual de la tecnologia.

5. Aprendre com la teoria de la normalització ajuda a detectar i resoldre redundàncies i anomalies en el disseny d'una base de dades.

6. Saber que hi ha alternatives diferents per a atacar el disseny d'una base de dades.

# 1. Disseny de bases de dades

A l'assignatura *Bases de dades I* trobareu una introducció al disseny de bases de dades, encara que aquí en canviarem una mica l'orientació o l'enfocament.

Vegeu una introducció al disseny de bases de dades en el mòdul "Introducció al disseny de bases de dades" i les representacions de les informacions en les bases de dades en el mòdul "Les dades: conceptes introductoris" de l'assignatura *Bases de dades I*.

Recordem que una base de dades (BD) serveix per a emmagatzemar les informacions (o, més ben dit, les representacions de les informacions) que s'utilitzen en el sistema d'informació d'una empresa o organització. Recordem també que una empresa està constituïda per tres subsistemes que s'anomenen **subsistema de producció, subsistema de decisió** i **subsistema d'informació**, i que aquest últim (que és el que realment ens interessa) recull (*inputs*), emmagatzema (*save*), processa amb programes i distribueix (*outputs*) totes les informacions rellevants i/o necessàries per als altres dos subsistemes, és a dir, que el subsistema d'informació d'una empresa és el pont entre la producció, que és la realització d'activitats que constitueixen l'objectiu d'una empresa, i la decisió, que és la planificació, coordinació i control de la producció.

**Abreugem *base de dades* amb la sigla BD.**

De les darreres afirmacions, els informàtics, i sobretot, els dissenyadors informàtics, hauríem de treure algunes conclusions importants:

**a) No toca decidir** als informàtics, com a tals, sobre l'empresa i la seva producció: per això hi ha el subsistema de decisió.

**b)** Un dissenyador no és pas un pintor (un pintor imagina i decideix), sinó més aviat un **fotògraf** de l'empresa. Per tant, no pot inventar o suposar característiques de l'empresa, sinó que ha de ser un transmissor fidel del que és la realitat de l'empresa que vol dissenyar.

**c)** Els elements d'un subsistema d'informació són les **dades** (representacions de les informacions), i els **processos** (o maneres de relacionar dades, o dades derivades, o càlculs, etc.).

## 1.1. Etapes del disseny de bases de dades

Parlar de les etapes o passos per a dissenyar una base de dades serà sempre quelcom d'artificiós, però concretar una mica ens pot ajudar a comprendre el procés del disseny de bases de dades. Detallarem deu etapes, de les quals les cinc primeres seran la tasca feixuga i essencial, i les cinc darreres seran feina d'afinament i detall, però no pas per això menys importants.

**1) Captura i abstracció dels requeriments dels usuaris**

En aquesta etapa es tracta de saber què vol l'usuari (o usuaris), és a dir, qui ha encarregat el disseny de la base de dades. Malauradament, moltes vegades, l'usuari només té una percepció genèrica d'allò que vol, i la missió del dissenyador serà fer palesos els vertaders desitjos de l'usuari.

El dissenyador no ha de decidir (ja ho hem explicat, no és un pintor, sinó un fotògraf). La seva tasca consisteix a ajudar l'usuari a descobrir què vol exactament. És una feina de paper i llapis, no pas fàcil, i que requereix entrevistar molts protagonistes de l'empresa, des del director fins al porter.

El dissenyador ha de fer un estudi d'oportunitats o de solucions variants sobre els diferents problemes, proposar-les amb independència i ajudar l'usuari a descobrir què desitja. Amb això podrà arribar a fer una anàlisi completa dels requeriments i, per tant, a concretar les especificacions de l'empresa o de l'organització.

**2) Elaboració d'un esquema conceptual amb l'ajut d'un model semàntic convenient**

En aquesta etapa, el dissenyador pot elaborar un esquema de les dades i de les associacions o interrelacions a partir del coneixement dels requeriments i especificacions d'allò que ha de fer. Això s'anomena **esquema conceptual de l'empresa o de l'organització**.

Per a confeccionar l'esquema conceptual, el dissenyador s'ajudarà de diferents "models de dades", enriquits avui pels anomenats *models semàntics de dades*, dels quals parlarem més endavant, i que incorporen més significat o semàntica per a fer una millor fotografia de l'empresa. Malgrat tot, en aquesta assignatura farem servir UML per a construir o elaborar l'esquema conceptual. En general, d'aquest procés se'n diu *elaborar el model conceptual*.

**3) Transformació del model conceptual a un model lògic general**

Quan ja s'ha obtingut un "model de dades", s'elabora un "model de bases de dades", és a dir, un model des del qual s'han implementat sistemes de gestió de bases de dades (SGBD). Aquests models genèrics s'anomenen *models lògics*, i són independents de les eines o implementacions.

**Abreugem *sistemes de gestió de bases de dades* amb la sigla SGBD.**

Tot i que més endavant en parlarem amb detall, podem avançar que en aquesta assignatura farem servir com a model lògic el model relacional clàssic. Per tant, en aquesta etapa, transformarem l'esquema conceptual obtingut amb l'UML en sentències estàndard d'SQL*, ja que, segons hem explicat en altres assignatures, l'SQL és l'estandardització del model relacional clàssic. En general, d'aquest procés se'n diu *construir el model lògic*.

Vegeu els models lògics de dades en el subapartat 1.2 d'aquest mòdul.

*** Per exemple, creació de taules, vistes, claus primàries, etc.**

**4) Aplicar la teoria de la normalització, si escau**

Atès que hem escollit el model relacional clàssic com a model lògic general, i aquest està íntimament lligat a la teoria de la normalització i de les formes normals, tindrà sentit aplicar aquesta teoria al model lògic obtingut.

La teoria de la normalització s'explica amb detall en l'apartat 2 d'aquest mòdul.

**5) Acomodació a l'SGBD de què disposem**

S'han fet nombroses "implementacions"* del model relacional clàssic, cadascuna amb particularitats notables i diferenciades que fan que el dissenyador hagi d'acomodar el model lògic obtingut a l'SGBD de què disposa l'empresa. Les particularitats de cada SGBD del mercat estan en els manuals que proporciona el fabricant, sobretot el destinat a l'administrador de la base de dades (ABD). En general, d'aquest procés se'n diu **construir el model físic**.

*** Com ara DB2, Oracle, Ingres, Informix, SQL-server, Access i moltes altres implementacions.**

**Abreugem administrador de bases de dades amb la sigla ABD.**

**6) Quantificació de volums de dades i freqüències de processos**

Aquí s'acaba la part feixuga i essencial del disseny de bases de dades, però encara s'ha de fer un afinament, ajust o millora de la base de dades dissenyada, que no és pas menys important. En primer lloc, hem de detectar si hi ha grans volums de dades (que en el model relacional clàssic es reflecteix per taules amb moltes tuples) a les quals haurem d'aplicar tècniques de fragmentació de taules que explicarem més endavant. També serà interessant de detectar si hi ha "processos crítics" amb alta freqüència i/o volatilitat, als quals aplicarem tècniques avançades d'implementació, algunes de les quals estan fora de l'abast d'aquesta assignatura.

Vegeu les tècniques de fragmentació de taules en el subapartat 7.2 del mòdul "Reconsideració dels models conceptual i lògic" d'aquesta assignatura.

**7) Consideracions sobre desnormalització, temps de resposta, integritat, seguretat, concurrència, recuperacions**

Aquestes consideracions pertanyen a diferents aspectes de l'estudi de les bases de dades. Per exemple, la desnormalització té a veure amb allò que s'explica respecte de la normalització; i el temps de resposta està molt relacionat amb els processos crítics.

Trobareu molta informació sobre integritat, seguretat, concurrència i recuperació, en aquesta assignatura i en l'assignatura *Tècniques avançades de bases de dades I*.

Respecte de la integritat, destaquem que és una tasca fonamental del dissenyador el fet d'assegurar-se que les restriccions* estiguin ben implementades, o de proposar per raons d'eficiència de substituir-les per d'altres que explicarem en el seu moment. Així mateix, és important que el dissenyador vetlli per la construcció de les transaccions (que dissenyi el nivell adequat d'aïllament) amb vista a la concurrència d'usuaris en la base de dades.

*** En anglès, *constraints*.**

**8) Afinament* de les estructures físiques d'emmagatzemament i dels paràmetres del sistema**

*** En anglès, *tuning*.**

Aquesta etapa és potser el nucli del model i l'esquema físic. Es tracta de saber si l'SGBD admet diferents formes d'emmagatzemament d'estructures físiques,

i de camins d'accés**, i d'escollir les més convenients segons les consultes que es fan o es faran a la base de dades. Però, sobretot, es tracta d'encertar amb els valors més convenients per als molts paràmetres que el constructor ha reservat a l'SGBD que ha construït.

**** Com ara accés seqüencial, índex en arbre B+ o *hash*, índex agrupat, estructures agrupades, etc.**

Els paràmetres del sistema només es poden encertar amb una lectura atenta dels manuals de l'ABD, amb molta experiència, i repetides proves per a constatar-ne el rendiment. Aquí no valen els coneixements generals de bases de dades; cal un coneixement profund de cada sistema (incloent-hi les diferents versions).

Exemples de paràmetres del sistema

El nombre i l'amplitud dels *buffers* i les pàgines, la grandària dels nusos dels arbres B+, el control de l'E/S i del paral·lelisme, els arxius de dades (*data files*), els arxius de control (*control files*), els *audit files*, etc., són alguns dels paràmetres del sistema.

**9) Control de rendiments**

Aquesta tasca complementa l'anterior. Hi ha moltes eines per a fer el control de rendiments. En primer lloc hi ha els monitors de rendiment, que amb menús desplegables i tot tipus d'histogrames, controlen les ocupacions de taules,espais per a taules* i arbres, del treball de la CPU, de la fluïdesa o embús de l'E/S, de degeneracions en *hashings* i agrupacions**, etc. A més, el pla de les consultes (que s'analitza en un altre mòdul) ajuda a millorar les consultes crítiques.

Vegeu el pla de les consultes en el subapartat 3.1 del mòdul "Disseny físic de bases de dades" d'aquesta assignatura.

*** En anglès, *tablespaces*.**
**** En anglès, *clusters*.**

**10) Informe final del dissenyador a l'administrador de la base de bades (ABD)**

Per acabar, el dissenyador ha d'escriure un informe que expliqui tot el que ha fet en aquella base de dades: els esquemes conceptual, lògic i físic, redundàncies volgudes i la manera com es controlen, degeneracions de *hash* i agrupacions, etc. Aquest escrit passa a mans del responsable del funcionament de la base de dades, que haurà de dissenyar tant polítiques ordinàries d'explotació i de manteniment com polítiques extraordinàries.

Exemples de polítiques

En el context de les bases de dades, una política és una manera d'actuar i comportar-se de la base de dades. Exemples:

- Polítiques d'explotació: arrencada i parada: qui i quan, *backups*: qui i quan, etc.
- Polítiques de manteniment: regeneració de *hash* i agrupacions, espais nous en taules i espais per a taules, etc.
- Polítiques extraordinàries: per a casos d'incendi, aiguats, atacs terroristes o altres.

Heus aquí deu etapes que semblen imprescindibles per a fer un bon disseny de bases de dades. Però hem de fer una advertència final molt important: l'evolució o construcció d'un disseny no és pas una tasca lineal (és a dir, que després de l'etapa 2 ve la 3, i després de la 6 ve la 7), sinó que quasi constantment s'ha de tirar enrere, i després de l'etapa 3 potser haurem de tornar a completar la 1. Això es coneix amb el nom de *retroacció**, i és una característica molt important en el comportament dels dissenyadors.

*** En anglès, *feedback*.**

## 1.2. Sobre el model conceptual

L'elaboració del model conceptual és l'objectiu de l'etapa número 2. En l'etapa número 1 s'han conegut els requeriments de l'usuari i s'han concretat les especificacions de l'empresa, i ara es tracta d'escollir un "model de dades" d'entre els coneguts i estudiats per a organitzar requeriments i especificacions, de manera que estiguem més a prop d'un model, no ja "de dades", sinó "de bases de dades", és a dir, sobre els que s'estructuren els SGBD (etapa número 3).

Com sabeu, hi ha molts models de dades, començant pels fitxers clàssics i les seves modalitats. A continuació s'enumera una llarga evolució en què alguns estadis han donat lloc a implementacions de sistemes de gestió de bases de dades (són els models de bases de dades), però altres han quedat simplement com a models de dades, més o menys teòrics. Aquestes modelitzacions (en aquest cas, de l'empresa) són imatges més o menys simplificades dels trets essencials, diferencials i a vegades abstractes, de l'organització que pretenem dissenyar. Podríem pensar en aquestes modelitzacions com a fotografies més o menys difuminades de l'empresa, i en el model, com la màquina per a obtenir aquestes fotografies.

Vegeu els diferents models de dades introduïts en el mòdul "Evolució dels models de bases de dades i ..." d'aquesta assignatura.

Precisament després que l'any 1970 Codd proposà el model relacional, molt interessant pels seus components lògics i algebraics, els universitaris van buscar i proposar models de dades nous i amb un contingut de detalls més gran (o imatges més perfilades, o fotografies més precises). Aquests models s'han anomenat *models semàntics* pel fet que aporten més significat (semàntica) que el mateix model relacional.

La nostra meta serà, en principi, modelar sistemes d'informació d'empreses per a implementar-los en bases de dades "relacionals clàssiques", anomenades d'aquesta manera per a distingir-les d'altres bases de dades relacionals més avançades.

**Les bases de dades relacionals clàssiques també reben el nom de *bases de dades de Codd*.**

Per tant, el nostre **model lògic** derivarà de les bases de dades relacionals clàssiques, i el **model conceptual** haurà de ser més fi o detallat, atès que la implementació és sempre igual o més pobra que l'esquema conceptual desitjat.

Podríem pensar a fer servir el model conceptual més potent (o sigui, el que faci les fotografies més fines o exactes de l'empresa), i que els models lògic i físic implementin el que puguin; però això no seria tocar de peus a terra. Malauradament, avui dia la implementació està molt per darrere de les idees teòriques i les "màquines de fotografiar" molt precises de què disposem, els models semàntics, encara estan molt allunyats de les implementacions. L'ús d'un model conceptual més potent o fi, portaria a la "frustració econòmica" del dissenyador (gastaria molt de temps i diners en confegir el model conceptual, i després no hi podria implementar gaires elements) i a la "frustració psicològica" (tindria la sensació d'haver perdut el temps).

Malgrat tot, la tasca del disseny és dinàmica, és una tasca que caldrà acomodar contínuament i, si millora la potència d'implementació dels SGBD, s'haurà d'augmentar el model conceptual (la "potència o finesa de la fotografia"). Per aquest motiu, la nostra elecció del model conceptual està molt lligada pels volts de l'any 2001 (en què s'han escrit aquestes línies), i que potser serà dife-

rent d'anys a venir. En l'estat actual de la tecnologia optarem per escollir UML com a model conceptual.

Els models semàntics més populars

Com a referència, a continuació us donem una llista dels models semàntics més populars o coneguts avui en dia:

**1)** ABRIAL, de l'any 1974. Avui ja és història.

**2)** RM/T (Relational Model of Tasmania), de Codd, de l'any 1979, precisament nou anys després de descriure en una publicació l'RM/1 (Relational Model 1). L'RM/T té coses molt interessants, com els *surrogates* o els tipus d'interrelacions, que després han fet fortuna, però del qual Codd s'oblida en publicar el 1992 l'RM/2.

**3)** Entity-Relationship (E/R), de Chen, de l'any 1976. Senzill, només té, com el mateix nom indica, entitats i relacions entre entitats (*relationships*), representades respectivament per quadres i rombes. Ha tingut molts deixebles, tants que al novembre de l'any 2001 se celebrarà a Yokohama (Japó) el 20è. Congrés Internacional d'E/R. Per aquesta raó, s'han fet múltiples extensions al model E/R, que s'anomenen EER (*Extended Entity Relationships*), que fins fa poc mantenien el primer pensament de Chen, amb solament entitats i interrelacions.

Últimament, i per influència d'alguns llenguatges de programació, han aparegut EER orientades a l'objecte, d'entre les quals destaquen YOURDON i UML (*Unified Modeling Language*). Precisament aquesta última extensió és la que nosaltres adoptem per a construir el model conceptual. Tinguem present que el disseny de bases de dades abraça sobretot la part estàtica del sistema d'informació –estructures intraobjecte i interobjectes, dites també *classes* i *associacions*–, que en UML també s'anomena *model conceptual*, i deixa de banda altres aspectes de l'UML com el model de casos d'ús, el model de comportament i el model d'estats, que són més aviat la part dinàmica del sistema d'informació (o sigui, els aspectes que canvien amb el temps, com són els estats, transicions d'estats, esdeveniments, etc.).

**4)** DAPLEX, de Shipmann, de l'any 1980. És un model funcional, o sigui, un model en què tot són funcions matemàtiques. No és un model tan senzill com l'anterior, ni tan còmode de presentar als usuaris per a la seva validació, i amb una traducció al model relacional clàssic gens fàcil.

**5)** SDM, de Hammer & Mc. Leod, de l'any 1981. Moltes eines *case* del mercat en deriven, encara que té herències del model en xarxa.

**6)** NIAM de Verheijen, de l'any 1982. Té moltes semblances amb alguns models EER, però algunes restriccions s'expressen gràficament amb molta facilitat.

**7)** SHM i SHM+, de Smith & Smith, el primer de1977, i el segon de 1985.

**8)** TAXIS, de Mylopoulos, de l'any 1980.

**9)** ACM/PCM, de Brodie, de l'any 1982.

**10)** LDM, de Kuper, de l'any 1985.

Hi ha més models semàntics, encara que no tan coneguts. Cada vegada resulta més complex d'aplicar-los perquè consideren els aspectes dinàmics del disseny. No són fàcilment implementables amb les actuals tècniques de bases de dades.

En el futur, a mesura que les implementacions d'SGBD siguin més fines o completes, potser haurem de canviar la nostra elecció de l'UML per una altra amb models semàntics més elaborats (algun dels enumerats en aquest paràgraf, o altres de nous que apareguin en el futur).

## 1.3. Sobre els models lògic i físic

Un cop hem aconseguit un model conceptual (un "model de dades"), cal que ens decidim per un "model de bases de dades", és a dir, organitzacions de da-

des que es puguin implementar en un ordinador amb un programari que s'anomena SGBD, i que estiguin dotades de qualitats com la persistència, la concurrència, la recuperació, la seguretat, la integritat, o sigui, tot el que ja coneixeu d'altres assignatures de bases de dades.

Vegeu els models de bases de dades introduïts en el mòdul "Evolució dels models de bases de dades i .." d'aquesta assignatura.

Ja hem fet una introducció a alguns "models de bases de dades", com són el model jeràrquic, el model en xarxa, el model relacional clàssic, el model orientat a l'objecte de les bases de dades anomenades *pures*, i el model orientat a l'objecte anomenat *model relacional amb objectes*. Aquest últim porta el terme *relational* perquè deriva del que nosaltres hem anomenat *model relacional clàssic* (de Codd), i no tenim pas intenció de confondre'ls.

*** En anglès, *object relational*.**

Els "models de bases de dades", en abstracte (prescindint de la implementació concreta), definiran el que n'hem dit el **model lògic**, representat generalment per un estàndard.

**ODMS és la sigla d'*Object Database Management Systems*.**

Avui, els "models de bases de dades" més representatius del mercat (cadascun amb nombroses implementacions) són el model relacional clàssic, el model orientat a l'objecte de les bases de dades pures, i el model relacional amb objectes.

**1)** Les **implementacions del model relacional clàssic*** són molt nombroses, amb una quota de mercat d'un 80%, i el seu estàndard és l'SQL.

*** Com per exemple DB2, Oracle, Ingres, Informix, SQL-Server, Sybase, Access, etc.**

**2)** Les **implementacions de les bases de dades pures orientades a l'objecte**** també són nombroses, però la seva quota de mercat és d'un 3 o un 5%, i sembla que no augmenta. L'estàndard d'aquestes hauria de derivar de l'ODMG v.3.0, però les diverses implementacions citades són més aviat prototips que es poden agrupar perquè tenen objectius comuns (com ara l'orientació a l'objecte, el fet que s'especifiquen amb llenguatges com Smalltalk, C++, Java, etc., als quals es dota de les qualitats de les bases de dades, com la persistència, la concurrència, la recuperació, la seguretat, etc.) que no pas implementacions que segueixin un estàndard o directives comunes.

**** Com ara Object-Store, O2, Gemstone, Orion, Vbase (Ontos), Objective/DB, Iris, Jasmine, poët, etc.**

**3)** Finalment, les **implementacions de les bases de dades relacional amb objectes** no són tantes. Les principals són Informix a partir de la versió 9, Oracle a partir de la versió 8, DB2 a partir de la versió 5, Sybase, etc.; i la seva quota de mercat és d'un 10 a 15%, amb tendència a augmentar en detriment de les bases de dades relacionals clàssiques. El seu estàndard és l'SQL, sobretot des que l'SQL:1999 ha estat enriquit amb construccions, sentències, dominis, tipus i funcions definides per l'usuari, etc., a fi d'estendre el model relacional clàssic al model relacional amb objectes.

En aquesta assignatura hem exclòs el model de les bases de dades "pures" per la seva limitació de mercat i especialització d'aplicacions. Hem adoptat com a model lògic el model relacional clàssic expressat amb sentències SQL (el seu

estàndard), i com a model físic, qualsevol de les seves implementacions físiques (de diferents fabricants).

Al **model físic** s'expressaran les particularitats i singularitats típiques dels sistemes de gestió de bases de dades esmentats.

Això no impedeix que, en un futur tal volta no llunyà, ens decantem per escollir com a model lògic el model relacional amb objectes i, en conseqüència, com a model físic, el corresponent a les seves diferents implementacions físiques.

Per a acabar aquest subapartat, assenyalem que l'estudi i disseny de models físics és una tasca molt especialitzada que exigeix el coneixement profund de l'SGBD concret, que requereix un estudi aprofundit dels manuals que proporciona el constructor i que demana llarga experiència i molts jocs de proves per a acomodar-se a l'SGBD que tenim i treure'n tot el rendiment que sigui possible.

### 1.4. Transformació del model conceptual al model lògic

Per a fer la transformació del model conceptual (en UML) al model lògic (en el model relacional clàssic) incloem un esquema general que us pot ajudar i que creiem bastant explicatiu si es coneixen prou l'UML i l'SQL (que és l'estàndard del model relacional clàssic).

Vegeu l'esquema de transformació del model conceptual al model lògic relacional clàssic a l'annex d'aquest mòdul.

Abans d'explicar l'esquema esmentat, farem cinc observacions que creiem importants:

**a) OID enfront de PK**: amb l'UML cada classe component del model té instàncies que es diferencien per un identificador d'objecte (OID) que el sistema proporciona automàticament. Nosaltres hem triat com a model lògic el model relacional clàssic que distingeix les diferents tuples o ocurrències mitjançant claus primàries (PK). Per tant, en el nostre cas una primera i important tasca serà determinar per a cada classe l'atribut o els atributs que constituiran la clau primària. La clau primària és un element essencial del model relacional clàssic, i la primera complicació en la traducció des de l'UML.

**L'acrònim OID denota *Object IDentifier*.**

**La sigla PK denota *Primary Key*.**

**La clau primària també es coneix com a *clau externa*, per a distingir-la de la clau interna o OID.**

**b) Operacions exclusives per a cada classe**: un altre element diferenciador entre l'UML i el model relacional clàssic és que les operacions anomenades *mètodes**, que l'UML especifica per a cada classe, només s'apliquen a aquesta classe. En canvi, en el model relacional clàssic, les funcions**, els procediments*** i, en general, els procediments emmagatzemats que s'utilitzen moltes vegades per a implementar les operacions es poden aplicar, en principi, a tota la base de dades i no a classes o taules en particular.

*** En anglès, *methods*.**
**** En anglès, *functions*.**
***** En anglès, *procedures*.**

**c) Format de relacions enfront del format d'SQL* (amb *CREATE TABLE*, etc.)**: en els esquemes de transformació, que trobareu a l'annex, representarem les taules del model relacional com a relacions, és a dir, escriurem:

```
<nom de la taula> (atribut1, atribut2, ..., atributn)
```

quan en el fons s'haurien d'haver escrit de la manera següent:

```
create table <nom>
       (atribut1 <domini1>, atribut2 <domini2>, ...,
       atributn <dominin>)
```

atès que el model lògic s'ha d'expressar amb SQL. A més, les *primary key*, *foreign key*, *not null*, *unique*, *check*..., s'hauran d'explicitar segons la sintaxi de l'SQL estàndard, i això és responsabilitat del dissenyador. Les sentències *create* són, doncs, bàsiques perquè el dissenyador transformi les classes de l'UML en taules de l'SQL. També cal parar atenció a la traducció de les restriccions. Algunes es poden implementar amb la sentència *create table* (vegeu els quadres **3 i 4b**), però altres requeriran sentències amb SQL-PSM (funcions, procediments, disparadors), o bé programes amb SQL-Host o també amb CLI.

**La tendència actual**

Avui hi ha una gran tendència a utilitzar l'hoste Java, que a SQL s'anomena SQLJ i també al CLI, el qual, derivat de Java, s'anomena JDBC.

**d) Alternativa gràfica**: el quadre **1** conté un resum molt sintètic, del qual volem destacar les associacions binàries en UML, que es tradueixen de la manera següent:

Vegeu els quadres de l'esquema de transformació del model conceptual al model lògic relacional clàssic a l'annex d'aquest mòdul.

- Si són *M*:*N* (molts a molts), en dues taules SQL corresponents a les dues classes components, més una tercera taula lligam de les dues anteriors, amb una clau primària composta per les dues claus primàries de les dues taules anteriors, que a més són claus foranes respecte de les claus primàries anteriors (i que hem representat per dues fletxes).

- Si són 1:*N* (u a molts) es tradueixen en simplement dues taules, però que a la banda de *N* tindran una clau forana respecte de la clau primària de la banda d'1. El cas *M*:*N* s'explicita en el quadre **2**, i el cas 1:*N*, en els quadres **3** i **4** (encara que aquests últims estan més explicats en el punt **e**).

El quadre **5** conté les traduccions de les associacions binàries reflexives, amb traducció paral·lela o similar a les *M*:*N* i 1:*N* anteriors. Els quadres **6a** i **6b**, contenen la traducció del model conceptual al model lògic d'associacions *N*-àries en general, encara que només hem representat les ternàries, amb tots els casos possibles, si bé **6b** representa casos poc freqüents.

**Les associacions binàries reflexives també s'anomenen *associacions recursives*.**

El quadre **9** és la representació de la traducció de la generalització/especialització, i es detalla en un altre mòdul.

Vegeu la traducció de la generalització/especialització en el mòdul "Reconsideració dels models conceptual i lògic obtinguts" d'aquesta assignatura.

Finalment els quadres **7** i **8** expliquen amb exemples la traducció de l'agregació composta i de les classes associatives de l'UML al model relacional clàssic. Observeu com s'assemblen les representacions en l'UML de les associacions dels quadres **2**, **8** i la part esquerra del **7**, i alhora, la gran diferència en la traducció d'aquestes tres representacions al model relacional clàssic. Per aquest motiu hem adoptat una alternativa gràfica, no pas estàndard, però que pot ajudar al dissenyador-traductor a no confondre's en aquests casos i a documentar més bé les traduccions.

Fixeu-vos que les classes pròpiament associatives requereixen una traducció bastant complexa, exposada en els quadres **8a** i **8b**, que exigeix tres taules (*A*, *B* i *C*) amb claus primàries simples, i dues taules més (*R* i *S*), *R* amb clau primària composta pels dos atributs de les dues primeres taules (*A* i *B*), i *S*, de clau primària composta pels tres atributs de les tres primeres taules, i a més, que dos d'aquests atributs siguin clau forana dels atributs d'*R* (fet que equival a dir que les tuples de la taula composta *S* no són qualssevol, sinó només subconjunts dels de *R*). Al quadre **7** (traducció de l'agregació composta de l'UML), la transformació és molt particular.

Observeu que a la banda de *N* hi hem posat un doble requadre, i que la clau primària està composta per l'atribut identificador d'aquesta entitat més l'atribut identificador de l'entitat de la banda d'1 (el pare), de manera que, a més de clau primària, serà clau forana. És per això, que hem anomenat les entitats doble-requadrades del model relacional clàssic *entitats dèbils*, en el benentès que en l'UML no hi ha entitats dèbils.

**e) Sobre els valors nuls**: el que segueix fa referència als quadres **3** i **4**, associacions binàries 1:*N* i 1:1 en l'UML, la traducció de les quals és bàsicament dues taules d'SQL, una amb una clau forana respecte de la clau primària de l'altra (la de la banda d *N*, si és 1:*N*, o una a elecció, si és 1:1). Aquestes claus foranes tindran sempre els valors de la clau referenciada, o simplement, valors nuls.

Els valors de les claus foranes en les associacions 1:*N* i 1:1 en l'UML

En el quadre **3** de l'annex hem dibuixat diferents dissenys possibles d'associacions 1:*N*. Per exemple, que un departament tingui *N* empleats (que poden ser d'1 a *N*, o de 0 a *N* empleats) i que, a més, un empleat pertany a 1 departament (que pot ser 0 –o cap–, o 1 i només 1 departament). Aquests quatre casos estan representats al marge esquerre. Un 0 n'indica la multiplicitat, o sigui, els departaments que tenen cap o *N* empleats, o els empleats que pertanyen a cap o 1 departament. Similarment s'ha fet en el quadre **4**. En aquests casos, la traducció seran claus foranes amb valors nuls.

Els valors nuls són un problema si ocupen espai addicional a disc, o si es malbarata temps de procés. No ocupen un espai especial en el disc, atès que els diferents camps de cada tupla (que representen els atributs) sempre tenen un descriptor del camp abans de la representació del valor, que inclou un bit per

a indicar si el valor és nul o no. Si no és nul, el valor va a continuació, però si és nul, el valor s'omet i es passa al descriptor següent.

Per tant, els valors nuls no són un problema quant a l'espai ocupat al disc, però sí, respecte del temps que empren en processar les consultes en què intervenen.

**Processament de consultes amb valors nuls**

Penseu en la consulta de valors que pren un atribut d'una taula de mil tuples de les quals 750 tenen valor nul en aquest camp: es processaran mil tuples per a obtenir només 250 valors resultants.

En conclusió, els valors nuls, poden ser un problema, no tant per l'espai ocupat, com pel temps invertit en les consultes. Per a solucionar aquest punt (que als quadres s'indica amb l'expressió "o bé" encerclada), crearem una tercera taula (a l'estil de les traduccions de les associacions *M*:*N*) que no contindrà les tuples amb valors nuls. Aquesta solució és cara tant en disseny com en l'execució per part de l'SGBD (representa una nova taula), i només té sentit d'aplicar-la quan el nombre de valors nuls sigui elevat respecte dels valors possibles.

## 1.5. Alternatives de disseny

Per acabar aquesta introducció, destacarem algunes alternatives al disseny de bases de dades:

**1)** Quant al model conceptual, hem elegit l'UML. Respecte del model lògic, hem escollit el model relacional clàssic. Finalment, el model físic dependrà de les implementacions i versions futures. En aquest cas, utilitzarem taules i tipus clàssics (identificats per PK) i també tipus estructurats i les noves taules de SQL:1999 (com són les *create table N of type M*, identificades per OID).

**Els models escollits**

Ja hem dit que podríem haver triat d'altres models semàntics com a models conceptuals, i potser es farà així en el futur. Quant al model lògic, en un futur proper, hi haurà el model relacional amb objectes com a alternativa al model relacional clàssic.

**2)** Quan estudiarem la teoria de la normalització, derivarem una altra manera de dissenyar, interessant des del punt de vista teòric, però que nosaltres no adoptarem. Aquesta alternativa aprofita que Java és orientat a l'objecte per a introduir orientació a l'objecte en l'SQL mitjançant SQL-Host, on l'hoste és Java, i s'anomena SQLJ. Hi ha tres propostes inicials a SQL:1999 que s'aniran desenvolupant progressivament en els anys a venir, i que són les que esmentem a continuació:

La teoria de la normalització s'estudia en l'apartat 2 d'aquest mòdul.

**a)** SQLJ0, que proposa incloure sentències SQL en programes escrits en Java, i que un precompilador s'encarrega de traduir. És pròpiament l'SQL-host, amb Java com a hoste.

*** També s'anomenen *procediments incrustats* a la base de dades.**

**b)** SQLJ1, que vol convertir procediments emmagatzemats escrits en Java a procediments emmagatzemats d'SQL*. Oracle ho té ja incorporat i mig automatitzat, mitjançant Jdeveloper.

*** En anglès, *mapping*.**

**c)** SQLJ2 (encara llunyà), que associaria classes i mètodes de Java amb taules, tipus i procediments d'SQL amb un mapatge* que es pretén que fos automàtic. Això equivaldria a tenir autèntics SGBD orientats a l'objecte, o sigui, tindríem un mapatge d'un model conceptual orientat a l'objecte (com l'UML) en un

model lògic (el relacional amb objectes amb SQLJ2) també orientat a l'objecte, que ja disposaria d'identificadors d'objecte (OID) i de mètodes (operacions) exclusius per a cada classe. Aquest és el futur esperançador, que ja veurem si serà proper o llunyà, però al qual convindrà estar atent.

**3)** Finalment, voldríem assenyalar que no és el mateix dissenyar un sistema d'informació nou (o partint des de zero) que dissenyar un sistema d'informació per a una empresa que resultés de l'absorció o fusió de diverses empreses, nacionals o multinacionals, on cadascuna tenia el seu sistema d'informació propi. Fins i tot es podria tractar de la fusió d'uns quants departaments autònoms d'una mateixa empresa, segurament antiquada, que vol funcionar a partir d'ara com una empresa unificada.

Un cas extrem

El disseny del sistema d'informació seria molt complicat en una situació en què els SGBD dels diversos elements que es fusionessin tinguessin models lògics diferents, com ara que un fos una base de dades jeràrquica (per exemple, un IMS d'IBM), un altre, una base de dades en xarxa (per exemple, un DBMS de Digital) i una tercera, una base de dades relacional (com Oracle).

Les situacions plantejades es podrien agreujar si els SGBD dels elements fusionats tinguessin el mateix model lògic (per exemple, del model relacional clàssic), però fossin de fabricants (o implementacions) diferents, per exemple un DB2, un Ingres, i un Oracle.

Tot el disseny de bases de dades que hem estudiat, o estudiarem en aquesta assignatura, és per a sistemes d'informació de nova creació, o sigui, partint des de zero. Per a tots els altres casos que hem imaginat, la solució està en el que s'anomena *integració de vistes*. La **integració de vistes** és una disciplina que encara està en estudi i evolució, i que requereix gent molt experimentada i multidisciplinar, atès que s'han d'agermanar diferents criteris, cultures i coneixements tècnics molt variats. Aquest tema, cau fora dels límits d'aquesta assignatura.

Un exemple particular d'integració de vistes

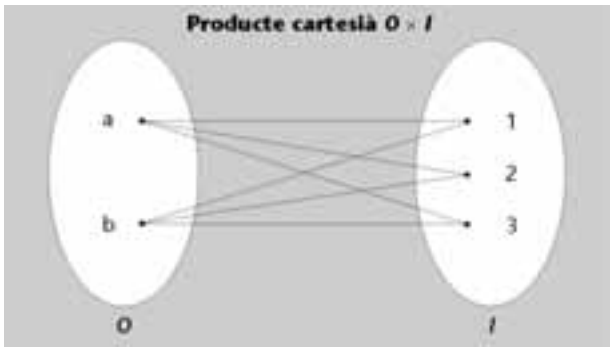
Un cas particular que podríem afrontar amb els coneixements adquirits és el d'un equip de dissenyadors que es constituís per a fer el disseny d'una base de dades per a una empresa molt gran i que s'encarregaria dels diferents components de l'empresa (com el departament de comptabilitat, el de producció, el de magatzem, etc.) per separat. Després, sota la direcció d'un dissenyador expert, s'ajuntarien tots els requeriments i les especificacions trobades per a dissenyar la gran empresa com una unitat. Aquest també és un cas d'integració de vistes, en què les vistes no vénen donades, sinó programades o fixades pel mateix equip de dissenyadors.

# 2. Normalització en bases de dades relacionals

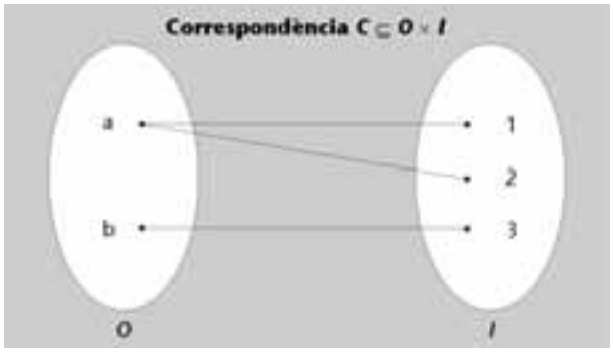
## 2.1. Conceptes previs d'àlgebra de conjunts

Recordarem alguns conceptes bàsics d'àlgebra de conjunts que són necessaris per a la correcta assimilació de la teoria de la normalització. Els conceptes que definim tot seguit són el producte cartesià, les correspondències i les funcions.

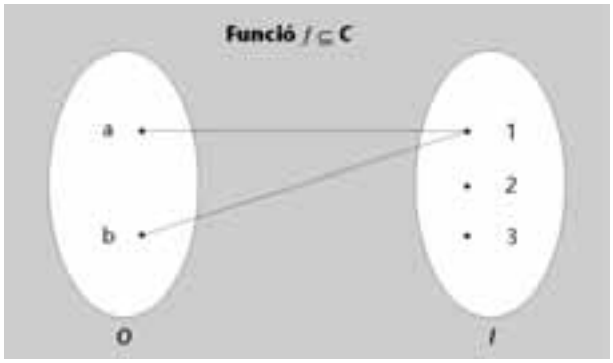
Si $O$ és un conjunt d'elements, que anomenarem *conjunt d'originals*, i $I$ és un altre conjunt d'elements, que anomenarem *conjunt d'imatges*, el **producte cartesià de $O \times I$** és el conjunt de tots els parells ordenats $(o, i)$ tal que $o \in O$ i $i \in I$:




Qualsevol subconjunt $C$ del producte cartesià és una **correspondència**:

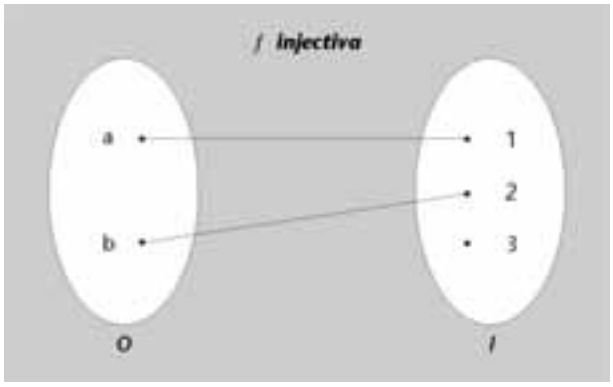



Les **funcions** (o **aplicacions**) són un subconjunt de les correspondències, en què es verifica que cada element del conjunt $O$ d'originals es relaciona amb un, i només un, element del conjunt $I$ d'imatges:

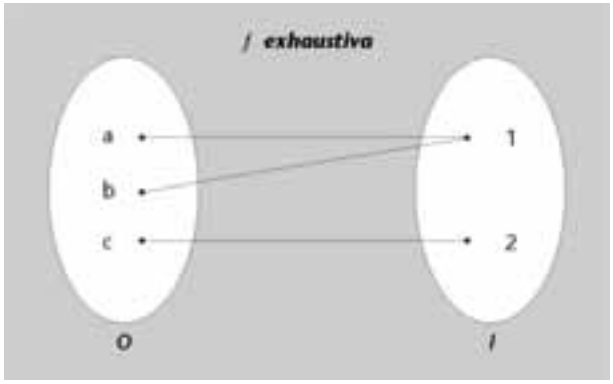


Tenim diferents categories de funcions. A nosaltres, concretament, ens interessa recordar-ne tres, les funcions injectives, les funcions exhaustives i les funcions bijectives:

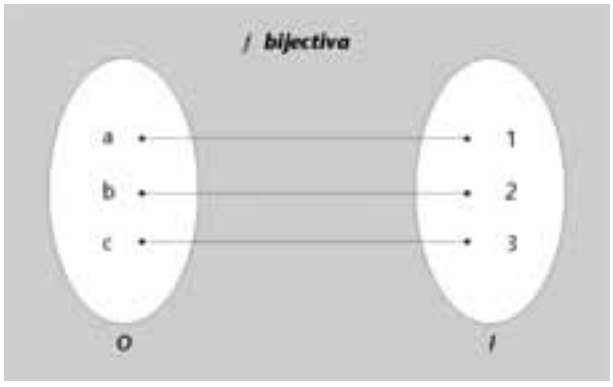
**1)** Si *O* és un conjunt d'originals i *I* és un conjunt d'imatges, una **funció injectiva** és una funció *f* tal que cada element del conjunt *I* es relaciona, com a molt, amb un element del conjunt *O*.




**2)** Si *O* és un conjunt d'originals i *I* és un conjunt d'imatges, una **funció exhaustiva** és una funció *f* tal que cada element del conjunt *I* es relaciona, com a mínim, amb un element del conjunt *O*.




**3)** Si *O* és un conjunt d'originals i *I* és un conjunt d'imatges, una **funció bijectiva** és una funció *f* tal que cada element del conjunt *I* es relaciona, amb un, i només un, element del conjunt *O*.



## 2.2. Conceptes bàsics d'àlgebra relacional

El model relacional proporciona una estructura de les dades per a representar informació del món real que consisteix en un conjunt de relacions. Cadascuna d'aquestes relacions, que representa un conjunt d'entitats del món real amb característiques comunes, es compon de la intensió (o esquema de la relació) i de l'extensió.

Els conceptes d'*intensió* i *extensió d'una relació, atribut, domini i tupla* s'estudien a l'assignatura *Bases de dades I*.

La **intensió** d'una relació consisteix en un nom de relació $R$ i un conjunt d'atributs $\{ A_1, A_2, \ldots, A_n \}$. Cada **atribut** $A_i$ és el nom del paper que exerceix un domini $D_i$ determinat en un esquema de relació. Cada **domini** representa el conjunt de valors atòmics vàlids que pot prendre un atribut.

D'altra banda, l'**extensió** d'una relació amb esquema $R(A_1, A_2, \ldots, A_n)$ és un conjunt de tuples $t_i$ (amb $i = 1, 2, \ldots, m$), i cada tupla $t_i$ és un conjunt de valors $<v_1, v_2, \ldots, v_n>$ tal que $\forall j, v_j \in D_j$.

Tota la informació que conté una base de dades s'ha de poder identificar d'alguna manera. En el cas particular d'una base de dades relacional, utilitzem la clau primària de la relació per a identificar de manera unívoca cadascuna de les tuples que componen l'extensió d'una relació. A continuació recordem ràpidament el concepte de *clau primària* d'una relació, i també altres conceptes relacionats com, per exemple, el concepte de *clau candidata* i *clau alternativa* d'una relació.

Els conceptes de *clau candidata*, *clau primària* i *clau alternativa* s'estudien a l'assignatura *Bases de dades I*.

Donada una relació $R$ amb esquema $R(A_1, A_2, \ldots, A_n)$, un subconjunt $C$ d'atributs d'$R$ és **clau candidata**, si $C$ conté el nombre mínim d'atributs necessaris per a identificar de manera unívoca totes les tuples d'$R$.

El dissenyador de la base de dades, d'entre totes les claus candidates de la relació $R$, n'escull una com a **clau primària** de la relació.

Les claus candidates no escollides com a clau primària s'anomenen **claus alternatives**.

Farem servir la convenció de subratllar la clau primària d'una relació.

A continuació mostrem un exemple de relació *Estudiants* amb la seva intensió i extensió:

| Estudiants | | | | |
|---|---|---|---|---|
| **DNI** | **Nom** | **DataNaixement** | **Sexe** | **AnyAdmissio** |
| 74233384 | Carles Pérez | 13.12.1967 | home | 1999 |
| 46742366 | Anna Puig | 31.03.1966 | dona | 1990 |
| 48001122 | Joan Rius | 13.12.1967 | home | 2000 |

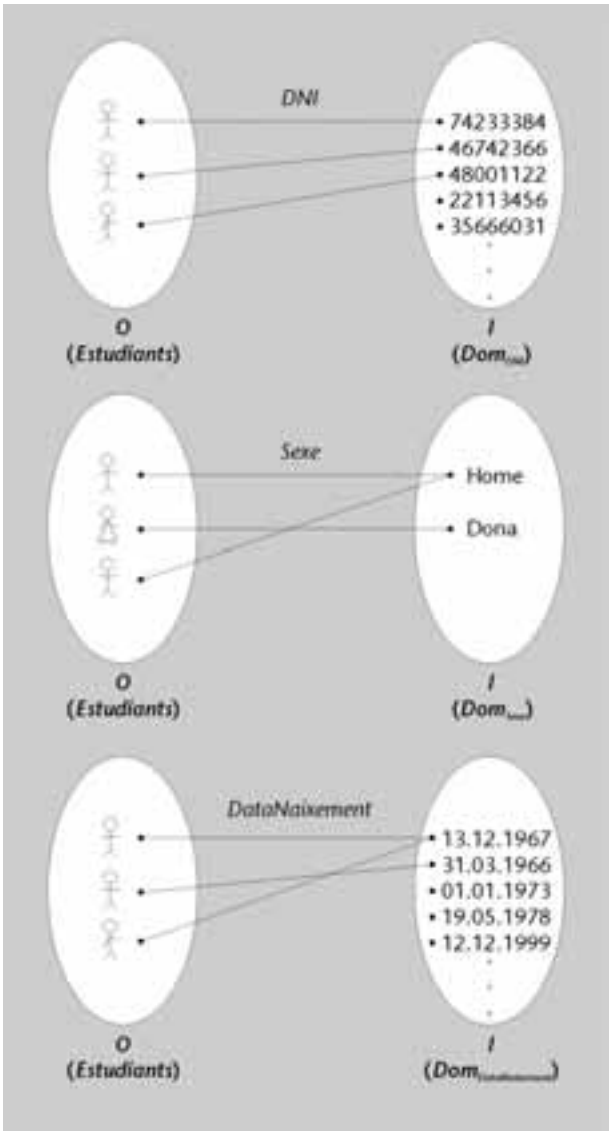
**Intensió**

**Extensió**

Bàsicament, la intensió de la relació *Estudiants* és representar informació personal rellevant per al conjunt d'estudiants d'una universitat. Per exemple, estem interessats a guardar el DNI dels estudiants, el nom, la data de naixement, el sexe, i l'any d'admissió a la universitat.

D'altra banda, l'extensió de la relació *Estudiants* representa el conjunt d'estudiants que pertanyen a la universitat (evidentment en mostrem només alguns). Per acabar, l'atribut *DNI* ens permet d'identificar de manera unívoca les diferents tuples de la relació *Estudiants* i és, per tant, clau candidata. Atès que és l'única clau candidata d'*Estudiants*, és alhora la clau primària de la relació.

En el marc de l'àlgebra de conjunts podem veure cada atribut d'una relació com una funció entre el conjunt d'entitats del món real que representa la relació i el domini associat a cada atribut. La figura següent mostra diversos exemples d'atributs considerats en el marc de l'àlgebra de conjunts:




**1)** El primer exemple mostra l'atribut *DNI* com una funció injectiva entre el conjunt d'estudiants i el domini de l'atribut *DNI*; els valors de l'atribut *DNI*

que no estan relacionats amb cap estudiant representen valors vàlids per a l'atribut *DNI* que podrien correspondre, per exemple, a altres persones que no són estudiants.

**2)** El segon exemple, mostra l'atribut *Sexe* com una funció exhaustiva entre el conjunt d'estudiants i el domini de l'atribut *Sexe*.

**3)** Finalment, el tercer exemple mostra l'atribut *DataNaixement* com una funció entre el conjunt d'estudiants i el domini de l'atribut *DataNaixement*. És important que ens adonem que aquesta darrera funció no és ni injectiva, ni exhaustiva, ni bijectiva.

Com ja sabeu, sobre una base de dades podem imposar restriccions o regles d'integritat. Aquestes restriccions regulen els possibles estats vàlids, és a dir, els estats íntegres d'una base de dades. A continuació presentem un nou tipus de restricció, les dependències funcionals.

Una **dependència funcional** és una restricció sobre una relació amb esquema $R(A_1, A_2, \ldots, A_n)$ denotada com $\{ X \} \rightarrow \{ Y \}$, on $\{ X \}$ i $\{ Y \}$ són subconjunts *de* $\{ A_1, A_2, \ldots, A_n \}$, que garanteix que donat un valor de $\{ X \}$, aquest determina de manera única el valor de $\{ Y \}$, és a dir, cada valor de $\{ X \}$ té associat un, i només un, valor de $\{ Y \}$. En aquest cas, diem que $\{ Y \}$ **depèn funcionalment** de $\{ X \}$ o, alternativament, $\{ X \}$ **determina funcionalment** $\{ Y \}$.

Intentarem d'explicar el significat de la definició de dependència funcional mitjançant l'exemple de relació que es mostra a continuació:

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1 | 100 | Reus |
| 1 | a2 | 150 | Reus |
| 2 | a1 | 200 | Vic |
| 2 | a2 | 250 | Vic |
| 3 | a2 | 100 | Vic |

En la relació d'esquema *Subministraments*(*CodiProv*,*CodiArticle*,*Quantitat*,*CiutatProv*) la clau primària és formada pels atributs *CodiProv* i *CodiArticle*. Aquesta relació representa quins articles subministren els diferents proveïdors i en quina quantitat. A més, també enregistra la ciutat de cada proveïdor.

Alguns exemples de dependències funcionals que existeixen en aquesta relació serien:

$$\{ CodiProv, CodiArticle \} \rightarrow \{ Quantitat \}$$
$$\{ CodiProv, CodiArticle \} \rightarrow \{ CiutatProv \}$$

Atès que { *CodiProv*, *CodiArticle* } és la clau primària de la relació *Subministraments*, estem segurs que cada valor de la parella {*CodiProv*, *CodiArticle*} determina de manera unívoca els valors dels atributs *Quantitat* i *CiutatProv*.

També seria dependència funcional:

$$\{ CodiProv \} \rightarrow \{ CiutatProv \}$$

En aquest cas també podem veure que cada cop que l'atribut *CodiProv* pren el mateix valor, el valor de l'atribut l'atribut *CiutatProv* es repeteix. Més concretament:

- Per a les tuples $t_1$ i $t_2$, que corresponen al proveïdor amb *CodiProv* = 1, el valor associat a l'atribut *CiutatProv* és sempre el mateix i indica que la ciutat del proveïdor és Reus.

- Per a les tuples $t_3$ i $t_4$, que corresponen al proveïdor amb *CodiProv* = 2, el valor associat a l'atribut *CiutatProv* es repeteix i indica que la ciutat del proveïdor és Vic.

- Per a la tupla $t_5$, que correspon al proveïdor amb *CodiProv* = 3, el valor associat a l'atribut *CiutatProv* indica que la ciutat del proveïdor és Vic.

El significat que té aquest fet en el món real és que un proveïdor està localitzat sempre a la mateixa ciutat.

En canvi, considerem el següent:

$$\{ \text{CodiArticle} \} \rightarrow \{ \text{Quantitat} \}$$

Això no seria dependència funcional, atès que no sempre que l'atribut *CodiArticle* pren el mateix valor, es repeteix el valor associat a l'atribut *Quantitat*. Per exemple, $t_1$ i $t_3$ tenen el mateix valor associat (*a1*) per l'atribut *CodiArticle*, però en cada cas el valor que pren l'atribut *Quantitat* és diferent (100 a $t_1$ i 200 a $t_3$, respectivament).

Les conclusions principals que podem extreure de la definició i l'exemple de les dependències funcionals són aquestes:

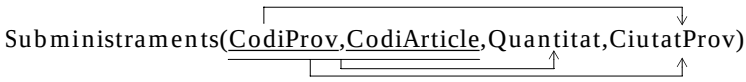
**a)** Una dependència funcional $\{ X \} \rightarrow \{ Y \}$ sobre una relació *R* no és més que una funció que s'estableix entre un conjunt d'originals $\{ X \}$ i un conjunt d'imatges $\{ Y \}$. És més, una dependència funcional caracteritza una propietat de totes les possibles instàncies d'una relació. Així, doncs, una dependència funcional representa una propietat d'intensió de la relació i, per tant, per a identificar-la cal analitzar detingudament el significat dels atributs que hi intervenen.

**b)** La clau primària d'una relació sempre determina funcionalment la resta d'atributs de la relació. Aquesta conclusió es pot estendre a totes les claus alternatives que la relació pugui tenir.

La segona conclusió és particularment important. A l'exemple, nosaltres hem deduït que el codi d'un proveïdor en determina funcionalment la ciutat. A més, aparentment, hem arribat a aquesta conclusió per examen de l'extensió de la relació *Subministraments*. Però això no és del tot exacte o, dit d'una altra manera, no és suficient. Examinar l'extensió d'una relació ens ajuda a descartar dependències funcionals (a l'exemple hem descartat que el codi dels articles determini el nombre d'unitats que un proveïdor és capaç de subministrar d'aquest article), però no ens permet d'assegurar rotundament que hi ha dependència funcional. Realment, hem d'anar més enllà, i preguntar-nos si el comportament que inferim a partir de l'examen de l'extensió es dóna o no en la realitat.

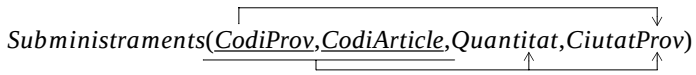
En resum, per a deduir dependències funcionals, no necessitem en absolut examinar l'extensió de la relació, atès que és una propietat d'intensió. En tot cas, examinar l'extensió ens pot ajudar a corroborar les dependències funcionals.

Les dependències funcionals d'una relació es poden representar gràficament. Si agafem de nou l'exemple anterior de la relació de *Subministraments*, podem representar les dependències funcionals que hem trobat de la manera següent:




És important que ens adonem que, per cada dependència funcional $\{ X \} \rightarrow \{ Y \}$, l'inici de la fletxa denota el conjunt $\{ X \}$ mentre que el final o la punta de la fletxa denota el conjunt $\{ Y \}$.

Atès que hi pot haver més d'una dependència funcional sobre el mateix conjunt d'originals $\{ X \}$, farem servir la notació simplificada següent:




Una dependència funcional $\{ X \} \rightarrow \{ Y \}$ és **dependència funcional completa** quan cap subconjunt propi de $\{ X \}$ determina funcionalment $\{ Y \}$.

Exemple de dependència funcional no completa

Si tornem de nou a la nostra relació d'exemple *Subministraments*, ens podem adonar fàcilment que la dependència funcional $\{ CodiProv, CodiArticle \} \rightarrow \{ CiutatProv \}$ no és dependència funcional completa, atès que $\{ CodiProv \} \rightarrow \{ CiutatProv \}$ i sabem que $\{ CodiProv \}$ és subconjunt propi de $\{ CodiProv, CodiArticle \}$.

Subconjunts propis i impropis

Donat un conjunt:
$A = \{ 1, 2, 3 \}$,
els conjunts següents són subconjunts propis de $A$:
$\varnothing$, $\{ 1 \}$, $\{ 2 \}$, $\{ 3 \}$, $\{ 1, 2 \}$, $\{ 1, 3 \}$, $\{ 2, 3 \}$
mentre que el conjunt:
$\{ 1, 2, 3 \}$
que és igual a $A$ és subconjunt impropi de $A$.

Quan fem el disseny conceptual d'una base de dades i a continuació trobem el disseny lògic corresponent, totes les dependències funcionals que hi pugui haver a les diferents relacions que formen part del disseny lògic han de ser dependències funcionals completes. Altrament, com veurem tot seguit, es produeixen anomalies de disseny que denoten que el disseny conceptual de la base de dades no és prou correcte.

## 2.3. Anomalies de disseny

Com a conseqüència d'un disseny dolent, podem tenir relacions que presenten un alt grau de redundància, és a dir, presenten repeticions que són evitables. Aquest fet en complica el manteniment, atès que es produeixen anomalies. A continuació analitzem aquestes anomalies sobre la nostra relació d'exemple *Subministraments*:

**a) Anomalies de modificació**: si un proveïdor canvia de ciutat, caldrà posar la nova ciutat del proveïdor a totes les tuples que facin referència al proveïdor en qüestió, si no volem que la base de dades sigui inconsistent. Per exemple, si el proveïdor amb *CodiProv* = 1 canvia de ciutat i passa a Salou, haurem de modificar dues tuples; si aquest proveïdor subministrés mil articles diferents, hauríem de modificar mil tuples, etc. La situació ideal fóra que només haguéssim de fer una modificació.

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1 | 100 | ~~Reus~~ Salou |
| 1 | a2 | 150 | ~~Reus~~ Salou |
| 2 | a1 | 200 | Vic |
| 2 | a2 | 250 | Vic |
| 3 | a2 | 100 | Vic |

Les anomalies de modificació obliguen a modificar totes les tuples que guarden un fet determinat, si aquest canvia.

**b) Anomalies d'esborrament**: si un proveïdor que només subministra un producte (per exemple, el proveïdor amb *CodiProv* = 3), deixés de subministrar-lo, caldrà esborrar-ne la tupla de la relació *Subministraments*. Com a

conseqüència, haurem perdut les dades personals del proveïdor en qüestió, en aquest cas, el codi de proveïdor i la ciutat:

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1 | 100 | Reus |
| 1 | a2 | 150 | Reus |
| 2 | a1 | 200 | Vic |
| 2 | a2 | 250 | Vic |
| ~~3~~ | ~~a2~~ | ~~100~~ | ~~Vic~~ |

A causa de les anomalies d'esborrament es poden perdre fets elementals sense voler.

**c) Anomalies d'inserció**: no podem emmagatzemar dades personals d'un proveïdor amb, per exemple, *CodiProv* = 4 i de la ciutat de Mollet, si no sabem els articles que subministra, atès que caldria afegir una tupla a la relació de *Subministraments* amb valor *null* per a l'atribut *CodiArticle*, i es violaria la regla d'integritat d'entitat de la clau primària:

Integritat d'entitat de la clau primària

Recordeu que la regla d'integritat d'entitat de la clau primària disposa que els atributs de la clau primària d'una relació no poden tenir valor *null*.

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1 | 100 | Reus |
| 1 | a2 | 150 | Reus |
| 2 | a1 | 200 | Vic |
| 2 | a2 | 250 | Vic |
| 3 | a2 | 100 | Vic |
| ~~4~~ | ~~NULL~~ | ~~NULL~~ | ~~Mollet~~ |

Les anomalies d'inserció consisteixen a no poder inserir fets elementals de manera independent.

L'origen de totes aquestes anomalies rau en el fet que la relació *Subministraments* descriu dos fets elementals del món real diferents: d'una banda, els articles que subministra cada proveïdor; i de l'altra, el proveïdor en si mateix. A més, aquests fets són completament independents entre si, atès que els articles concrets que subministra cada proveïdor, no guarda cap relació directa amb el fet que el proveïdor sigui, per exemple, d'una ciutat o d'una altra, i a l'inrevés. En tot cas, entre aquests dos fets hi ha una relació indirecta, perquè afecten un mateix individu del món real, és a dir, el mateix proveïdor.

En conclusió, tota relació que no representa un concepte (o fet elemental) únic del món real està subjecta a presentar redundàncies, anomalies de manteniment i inconsistències potencials. Aquest és el cas de la nostra relació *Subministraments*.

## 2.4. Teoria de la normalització

L'objectiu de la teoria de la normalització és formalitzar un conjunt d'idees simples que guien un bon disseny de bases de dades relacionals clàssiques, és a dir, bases de dades relacionals basades en la teoria de conjunts, la lògica i l'àlgebra relacional.

La **teoria de la normalització** es fonamenta en el principi bàsic següent: tota relació ha de descriure un concepte semàntic únic.

La teoria de la normalització ens permet de reconèixer els casos en els quals aquest principi no es compleix. El mecanisme que fa servir per a això són les **formes normals** (**FN**). Una relació està en una forma normal determinada, si satisfà un conjunt de restriccions determinades que són pròpies d'aquesta forma normal. La violació d'aquestes restriccions origina que la relació tingui les anomalies i les redundàncies que hem esmentat abans.

**Abreugem *forma normal* amb la sigla FN.**

Hi ha diverses formes normals, de les quals nosaltres estudiarem les sis més importants. Cada forma normal indica unes restriccions específiques que ha de complir una relació. Com més gran és el grau d'una forma normal, més restrictiva és aquesta forma normal i elimina més redundàncies que les formes normals de grau menor, atès que inclou les restriccions d'aquelles i agrega una restricció addicional.


Graus de les formes normals

Tal com mostra la figura, si una relació està en tercera forma normal (3FN), també estarà en segona forma normal (2FN) i també estarà en primera forma normal (1FN).

Per a indicar que una relació no està en 1FN, s'acostuma a fer servir l'acrònim $NF^2$ (*NF×NF, Non First Normal Form*).

Quan una relació no compleix una forma normal, és perquè viola la restricció associada a aquella forma normal. Per a aconseguir que es verifiqui la forma normal, caldrà evitar la condició que fa que es violi la restricció en qüestió. El procediment que aplicarem per a aconseguir que no es violi la restricció associada a la forma normal rep el nom de **normalització**.

Una altra característica de les formes normals és que són declaratives, és a dir, cada forma normal ens indica quines restriccions s'han de complir, però no ens descriu un procediment de com arribar-hi. Per tant, hi ha diverses maneres de normalitzar una relació. Nosaltres en farem servir una que condueix a una millor comprensió de les formes normals, de manera que anirem pas a pas. És a dir, analitzarem una a una cada forma normal, i resoldrem les condicions que fan que es violi una forma normal determinada. Atès que el procés de normalització no és únic, s'han proposat diferents algorismes per a normalitzar una relació.

La primera forma normal (1FN) es basa en el concepte de *valor atòmic*, mentre que la resta de formes normals es basen en el concepte de *dependència*. La segona forma normal (2FN), la tercera forma normal (3FN) i la forma normal de Boyce-Codd (FNBC) es basen en el concepte de *dependència funcional*. Per acabar, la quarta forma normal (4FN) i la cinquena forma normal (5FN) es basen en els conceptes de *dependència multivaluada* i *dependència de projecció-combinació*, respectivament.

### 2.4.1. Primera forma normal

**La primera forma normal va ser proposada per Codd l'any 1970.**

Una relació està en **primera forma normal** (**1FN**) si, i només si, cap atribut de la relació és en si mateix una relació, és a dir, si tot atribut de la relació és atòmic, no descomponible, no grup repetitiu.

La relació *Subministraments* que mostrem tot seguit no està en 1FN, atès que els atributs *CodiArticle* i *Quantitat* no són atòmics:

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1<br>a2 | 100<br>150 | Reus |
| 2 | a1<br>a2 | 200<br>250 | Vic |
| 3 | a2 | 100 | Vic |

Per a aconseguir de normalitzar una relació a 1FN, caldrà aplicar una operació que es coneix amb el nom d'*aplanar*. Com a conseqüència d'aquesta operació, la relació *Subministraments* quedaria de la manera següent:

| Subministraments | | | |
|---|---|---|---|
| **CodiProv** | **CodiArticle** | **Quantitat** | **CiutatProv** |
| 1 | a1 | 100 | Reus |
| 1 | a2 | 150 | Reus |
| 2 | a1 | 200 | Vic |
| 2 | a2 | 250 | Vic |
| 3 | a2 | 100 | Vic |

És important que ens adonem que quan s'aplana la relació *Subministraments* original, la clau primària de la relació canvia i passa a ser composta. També cal destacar que el model relacional sempre garanteix que les relacions estan en 1FN, atès que només hi ha una estructura per a representar les dades (la relació) i, a més, cada dada es representa uniformement mitjançant valors atòmics.

### 2.4.2. Segona forma normal

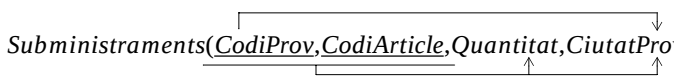
**La segona forma normal va ser proposada per Codd l'any 1970.**

Una relació està en **segona forma normal** (**2FN**) si, i només si, està en 1FN i tot atribut no clau depèn funcionalment en forma completa de la clau primària.

Hi ha una excepció: un atribut pot dependre funcionalment de part de la clau primària, si aquest atribut és part d'una clau alternativa.

És important de destacar que la 2FN es basa en el concepte de *dependència funcional completa* i, per tant, aquesta forma normal només es pot violar quan la clau primària d'una relació és composta. En resum, aquelles relacions amb clau primària simple, és a dir, amb clau primària formada per un únic atribut sempre estaran en 2FN.

Per exemple, la relació de *Subministraments* que mostrem tot seguit no està en 2FN:




La idea que hi ha al darrera de la 2FN és molt simple. Si un atribut no clau (en el nostre exemple, *CiutatProv*) depèn funcionalment d'un subconjunt propi de la clau (*CodiProv*, a l'exemple), és perquè representa un fet d'aquest subconjunt propi i, per tant, tots dos (l'atribut no clau i el subconjunt propi de la clau) són un altre concepte semàntic (en el nostre cas, la ciutat de residència

del proveïdor). La redundància que hi ha en una relació que no està en 2FN és immediata: els valors de l'atribut no clau es repetiran per a tots els valors diferents de la part de la clau de la qual no depèn.

Per a aconseguir de passar una relació a 2FN, cal que evitem que hi hagi dependències funcionals no completes respecte de la clau. Per tant, tots els atributs que participen en la dependència funcional no completa s'hauran de projectar en una nova relació que correspon al concepte semàntic que representen, tal com mostrem a continuació:

*Subministraments*(*CodiProv*,*CodiArticle*,*Quantitat*)

*CiutatsProveïdors*(*CodiProv*,*Ciutat*)

D'aquesta manera, els diferents conceptes semàntics es descriuen en diferents relacions seguint el principi bàsic de la normalització.

### 2.4.3. Tercera forma normal

Una relació està en **tercera forma normal** (**3FN**) si, i només si, està en 2FN i cap atribut no clau depèn funcionalment de cap altre conjunt d'atributs no clau.

**La tercera forma normal va ser proposada per Codd l'any 1970.**

L'excepció aplicada a la 2FN es propaga també a la tercera forma normal.

Considerem la relació *Clients* que mostrem a continuació:

*Clients*(*CodiProv*,*Carrer*,*Numero*,*Ciutat*,*Provincia*)

Aquesta relació està en segona forma normal, però no en tercera forma normal, atès que hi ha una dependència funcional { *Ciutat* } → { *Provincia* } entre dos atributs no clau i, a més, cap dels dos no forma part ni és clau alternativa de la relació i, per tant, l'excepció no s'aplica. De nou, el fet que l'atribut *Provincia* depengui funcionalment de l'atribut *Ciutat* és independent del client en si mateix. Les redundàncies i anomalies que presenta aquesta relació són immediates.

Redundàncies associades a 3FN

Per cada client que estigui localitzat a la ciutat de Mataró, tindrem repetit el fet que aquesta ciutat pertany a la província de Barcelona.

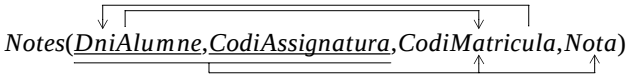
Per a normalitzar una relació que viola la 3FN, caldrà evitar la dependència funcional entre atributs no clau. Per tant, els atributs que participen en la dependència funcional, s'hauran de projectar en una nova relació que correspon al concepte semàntic que representen, tal com mostrem a continuació:

*Clients*(*CodiProv*,*Carrer*,*Numero*,*Ciutat*)

*Ciutats*(*Ciutat*,*Província*)

### 2.4.4. Forma normal de Boyce-Codd

Analitzem ara la relació *Notes* següent:




La relació *Notes* verifica la 1FN (recordem que el model relacional garanteix sempre per defecte aquesta forma normal). Aquesta relació també està en 2FN, malgrat que hi ha una dependència funcional no completa de part de la clau cap a un atribut no clau ({ *DniAlumne* } → { *CodiMatricula* }). Atès que aquesta dependència involucra un atribut no clau (*CodiMatricula*) que forma part d'una clau alternativa de la relació (*CodiAssignatura*,*CodiMatricula*), s'aplica l'excepció. Per acabar, la relació també està en 3FN encara que hi ha una dependència funcional entre atributs no clau { *CodiMatricula* } → { *DniAlumne* }. De nou, atès que involucra atributs que formen part de claus alternatives, s'aplica l'excepció.

Malgrat això, tal com podem veure a continuació, la relació *Notes* presenta redundàncies i anomalies:

| Notes | | | |
|---|---|---|---|
| **DniAlumne** | **CodiAssignatura** | **CodiMatricula** | **Nota** |
| 234567654 | 05.001 | 123 | A |
| 234567654 | 04.002 | 123 | B |
| 45678323 | 05.002 | ~~215~~<br>312 | B |
| 45678323 | 05.001 | ~~215~~<br>312 | B |
| 45678323 | 04.002 | ~~215~~<br>312 | C |

**a)** Redundàncies: per cada assignatura diferent en què estigui matriculat un alumne, es repetirà el seu codi de matrícula.

**b)** Anomalies de modificació: si volem modificar, per exemple, el codi de matrícula de l'estudiant amb DNI 45678323 perquè sigui 312 en comptes de 215, haurem de modificar tres tuples. Com ja sabem, idealment només en voldríem modificar una.

L'origen d'aquestes anomalies és històric i es deu a un error d'omissió. Quan Codd va enunciar la 2FN i la 3FN l'any 1970, no va considerar la possibilitat que en una relació hi pogués haver diverses claus candidates que fossin com-

postes, ni tampoc no va considerar la possibilitat que entre aquestes hi pogués haver encavalcaments. Per això, l'any 1974, Boyce i Codd van proposar la forma normal de Boyce-Codd que soluciona les limitacions de la 2FN i la 3FN. De fet, sovint es normalitza una relació directament a la forma normal de Boyce-Codd sense passar prèviament per la 2FN i la 3FN.

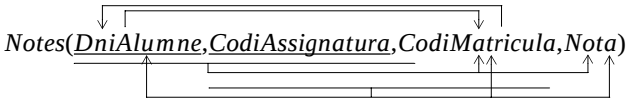
Una relació està en **forma normal de Boyce-Codd** (**FNBC**) si, i només si, està en 1FN, i si tots els determinants són clau candidata de la relació.

A la definició apareix un concepte nou: el concepte de *determinant*.

Donada una dependència funcional $\{ X \} \rightarrow \{ Y \}$, un **determinant** és el conjunt $\{ X \}$.

Per tant, si ho expressem d'una manera informal, els determinants en una dependència funcional són els orígens de fletxa.

Si tornem a examinar la relació *Notes*, i busquem totes les dependències funcionals que hi ha en la relació, arribem a la situació següent:




La taula següent mostra tots els determinants (primera columna) i totes les claus candidates de la relació *Notes* (segona columna):

| Notes | |
|---|---|
| **Determinants** | **Claus candidates** |
| DniAlumne | DniAlumne, CodiAssignatura |
| CodiMatricula | CodiAssignatura, CodiMatricula |
| DniAlumne, CodiAssignatura | |
| CodiAssignatura, CodiMatricula | |

Atès que no tots els determinants són clau candidata de la relació *Notes*, arribem a la conclusió que *Notes* no està en FNBC. De nou, per a normalitzar la relació, caldrà evitar la condició que causa la violació de la restricció. Per tant, haurem d'evitar les dependències funcionals $\{ DniAlumne \} \rightarrow \{ CodiMatricula \}$ i $\{ CodiMatricula \} \rightarrow \{ DniAlumne \}$ de la relació *Notes*.

Això s'aconsegueix si es projecten els atributs involucrats en una nova relació que correspon al concepte semàntic que representen. Ara bé, en aquest cas, i a diferència dels casos anteriors, tenim diverses alternatives:

**a)**

*Notes*(*DniAlumne*,*CodiAssignatura*,*Nota*)

*Alumnes*(*DniAlumne*,*CodiMatricula*)

**b)**

*Notes*(*CodiMatricula*,*CodiAssignatura*,*Nota*)

*Alumnes*(*CodiMatricula*,*DniAlumne*)

**c)**

*Notes*(*DniAlumne*,*CodiAssignatura*,*Nota*)

*Alumnes*(*CodiMatricula*,*DniAlumne*)

**d)**

*Notes*(*CodiMatricula*,*CodiAssignatura*,*Nota*)

*Alumnes*(*DniAlumne*,*CodiMatricula*)

Les diferents possibilitats de projecció d'atributs entre la relació *Notes* i la nova relació *Alumnes* s'obtenen com a resultat de combinar les claus candidates de la relació original *Notes*. Com a dissenyadors, som lliures de triar qualsevol de les opcions presentades. Ara bé, les més naturals són les opcions **a** o **b**, atès que les opcions **c** i **d** són mixtes. Entre les opcions **a** i **b**, des del punt de vista de la Secretaria del Centre, possiblement l'opció **b** seria més coherent*.

*** Per exemple, pensem que hi ha alumnes d'altres països que no tenen DNI.**

Observem que, independentment de l'opció triada, les relacions *Notes* i *Alumnes* estan en FNBC, atès que tots els determinants són claus candidates de la relació. Això és particularment important per a la nova relació *Alumnes*.

### 2.4.5. Conclusions sobre dependències funcionals en les formes normals

Abans de progressar en el temari, resumirem les conclusions principals que podem extreure fins a la forma normal de Boyce-Codd. Aquestes conclusions són les que presentem a continuació:

**a)** Sempre és possible normalitzar fins a la forma normal de Boyce-Codd.

**b)** El procés de normalització no és únic.

**c)** Donat un model lògic inicial d'una base de dades, si hi apliquem un procés de normalització, el model lògic final obtingut equival sempre al model lògic inicial. És a dir, el procés de normalització preserva la semàntica (o significat) del model lògic inicial.

**d)** Com a conseqüència de la normalització, el model lògic final és millor que el model lògic inicial atès que:

- elimina redundàncies i anomalies.
- separa fets semànticament diferents.

Per acabar amb les formes normals bàsiques, completarem el concepte de *dependència funcional*, que constitueix el fonament de les formes normals que hem vist fins ara.

Donada una relació $R$ amb esquema $R(A_1,A_2,...,A_n)$, anomenem $L$ el conjunt de dependències funcionals associat a $R$. A partir de $L$ podem obtenir totes les dependències funcionals que se'n deriven mitjançant l'aplicació d'un conjunt de regles que es coneixen amb el nom d'*axiomes d'Armstrong*.

Els **axiomes d'Armstrong** són aquests:

**1) Reflexivitat**:

$\forall X, \{ X \} \rightarrow \{ X \}$, dependència funcional trivial.

**2) Augmentativitat**:

$\forall X,Y$ si $\{ X \} \rightarrow \{ Y \} \Rightarrow \{ X, Z \} \rightarrow \{ Y \}$

$\{ X \} \rightarrow \{ Y \}$ dependència funcional mínima

**3) Distributivitat**:

$\forall X,Y,Z$ si $\{ X \} \rightarrow \{ Y,Z \} \Rightarrow \{ X \} \rightarrow \{ Y \} \wedge \{ X \} \rightarrow \{ Z \}$

**4) Additivitat**:

$$\forall X,Y,Z \text{ si } \{ X \} \rightarrow \{ Y \} \wedge \{ X \} \rightarrow \{ Z \} \Rightarrow \{ X \} \rightarrow \{ Y, Z \}$$

**5) Transitivitat**:

$$\forall X,Y,Z, \text{ si } \{ X \} \rightarrow \{ Y \} \wedge \{ Y \} \rightarrow \{ Z \} \Rightarrow \{ X \} \rightarrow \{ Z \}$$

**6) Pseudotransitivitat**:

$$\forall X,Y,Z,W, \text{ si } \{ X \} \rightarrow \{ Y \} \wedge \{ Y, Z \} \rightarrow \{ W \} \Rightarrow \{ X, Z \} \rightarrow \{ W \}$$

Malgrat que a la literatura les regles prèvies es coneixen com a axiomes, des d'un punt de vista estrictament matemàtic no totes ho són. Més concretament, podem combinar les regles prèvies de diferents maneres i, segons la combinació que es triï, alguns d'aquests axiomes, efectivament, resulten ser axiomes i la resta, teoremes.

Com a resultat de l'aplicació exhaustiva dels axiomes d'Armstrong sobre un conjunt de dependències funcionals $L$, obtenim el que es coneix com a **clausura** o **tancament** de $L$. La clausura de $L$ es representa com $L^+$.

Donats un conjunt d'atributs i un conjunt $L$ de dependències funcionals entre aquests atributs, la clausura $L^+$ d'aquest conjunt ens permet de:

**a)** veure si una dependència funcional determinada és certa o no en una empresa o organització.

**b)** calcular el conjunt de claus candidates de les diferents relacions que componen el model lògic d'una base de dades, atès que la finalitat última fins a arribar a la forma normal de Boyce-Codd és aconseguir que en una relació $R$, $\{ X \}$ sigui clau candidata de $R$ per a qualsevol dependència funcional $\{ X \} \rightarrow \{ Y \}$.

**c)** determinar si dos models lògics aparentment diferents són equivalents o no. Més concretament, donats dos models lògics diferents i donats $L_1$ i $L_2$ els dos conjunts de dependències funcionals associats a cadascun dels models lògics, si les clausures $L_1^+$ i $L_2^+$ són idèntiques, aleshores podem afirmar que els dos models lògics representen semànticament la mateixa empresa o organització.

Per acabar, és important de destacar que la cardinalitat de $L^+$ és molt més gran que la cardinalitat de $L$ ($\text{Card}(L^+) >> \text{Card}(L)$). En altres paraules, a $L$ hi ha les dependències funcionals que podem deduir a partir de la intensió de la relació, mentre que a $L^+$ hi ha totes aquestes dependències funcionals, més tot un conjunt de dependències funcionals que podem considerar redundants i que es dedueixen a partir de l'aplicació dels axiomes d'Armstrong.

**Diferents combinacions de les regles prèvies**

Podem fer les combinacions de regles prèvies següents:

**a)** Triem **1**, **2** i **5** com a axiomes; aleshores **3**, **4** i **6** són teoremes.

**b)** Elegim **1**, **4** i **5** com a axiomes; aleshores **2**, **3** i **6** són teoremes.

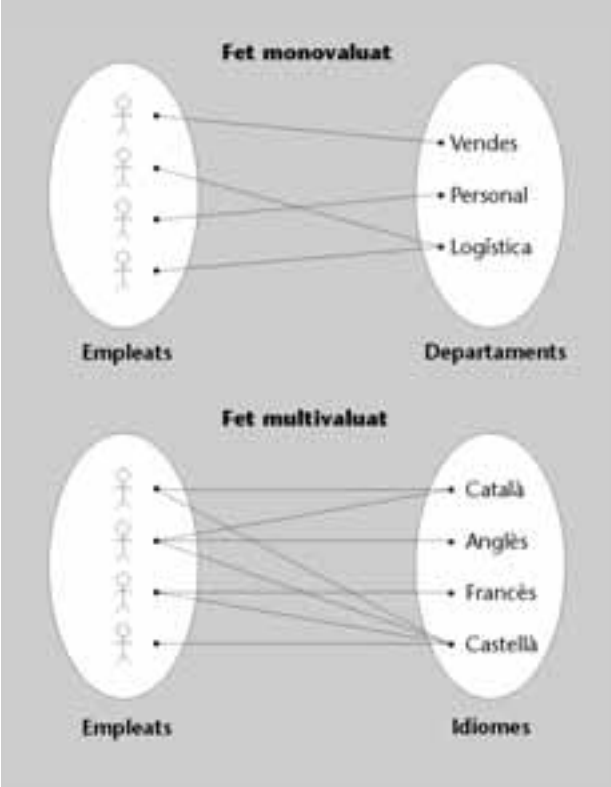
**c)** Escollim **1**, **2** i **6** com a axiomes; aleshores **3**, **4** i **5** són teoremes.

Recordeu que un axioma és una propietat de la matemàtica que no es demostra.

### 2.4.6. Quarta forma normal

Mentre que les formes normals que acabem de veure es relacionen amb fets monovaluats, la quarta i la cinquena forma normal estan relacionades amb fets multivaluats.

La figura següent presenta dos exemples, un d'un fet monovaluat i un altre, d'un fet multivaluat, que mostren, respectivament, que cada empleat treballa en un únic departament i que un empleat pot parlar més d'un idioma.




A continuació considerem la relació *ConeixProgr* següent:

| ConeixProgr | | |
|---|---|---|
| **DNI** | **LlengProg** | **Idiomes** |
| 74567000 | Cobol<br>Pascal | Anglès<br>Català |
| 56432999 | C++<br>Java | Català<br>Castellà |
| 23450988 | Java | Anglès<br>Castellà<br>Francès |

La intensió d'aquesta relació és mostrar els llenguatges de programació que dominen un grup de programadors, i registrar els diferents idiomes que parlen.

Si analitzem aquesta relació, ràpidament veiem que no està en 1FN, atès que tant l'atribut *LlengProg* com l'atribut *Idioma* no són atòmics. Si aplanem la relació amb l'objectiu de normalitzar-la, obtenim la relació de *ConeixProgr* següent:

| ConeixProgr | | |
|---|---|---|
| **DNI** | **LlengProg** | **Idiomes** |
| 74567000 | Cobol | Anglès |
| 74567000 | Cobol | Català |
| 74567000 | Pascal | Anglès |
| 74567000 | Pascal | Català |
| 56432999 | C++ | Català |
| 56432999 | C++ | Castellà |
| 56432999 | Java | Català |
| 56432999 | Java | Castellà |
| 23450988 | Java | Anglès |
| 23450988 | Java | Castellà |
| 23450988 | Java | Francès |

**Redundàncies?**

Si examinem la nova relació *ConeixProgr*, veiem que verifica tant la 2FN, com la 3FN i la FNBC. Malgrat això, Fagin es va preguntar l'any 1977 si en una relació d'aquest tipus hi ha redundàncies o no n'hi ha, és a dir, si ha repeticions evitables o no n'hi ha. La resposta és afirmativa.

Si, per exemple, el programador amb DNI 74567000 aprengués un nou llenguatge de programació, com ara C++, hauríem de registrar aquest fet en la relació *ConeixProgr* i caldria afegir dues tuples a la relació *ConeixProgr* pel fet d'haver de creuar el llenguatge de programació nou amb els diferents idiomes que el programador parla. Si a continuació volguéssim registrar que el mateix programador també parla alemany, hauríem d'inserir tres tuples, com a resultat de combinar l'idioma nou amb els tres llenguatges de programació que l'empleat domina. La figura següent mostra la nova extensió de la relació *ConeixProgr*:

| ConeixProgr | | |
|---|---|---|
| **DNI** | **LlengProg** | **Idiomes** |
| 74567000 | Cobol | Anglès |
| 74567000 | Cobol | Català |
| 74567000 | Pascal | Anglès |
| 74567000 | Pascal | Català |
| 56432999 | C++ | Català |
| 56432999 | C++ | Castellà |
| 56432999 | Java | Català |
| 56432999 | Java | Castellà |

| ConeixProgr | | |
|---|---|---|
| **DNI** | **LlengProg** | **Idiomes** |
| 23450988 | Java | Anglès |
| 23450988 | Java | Castellà |
| 23450988 | Java | Francès |
| 74567000 | C++ | Anglès |
| 74567000 | C++ | Català |
| 74567000 | Cobol | Alemany |
| 74567000 | Pascal | Alemany |
| 74567000 | C++ | Alemany |

**Tuples afegides**

L'origen de les anomalies de la relació *ConeixProgr** rau en el fet que la relació recull dos fets multivaluats: d'una banda, els llenguatges de programació que l'usuari coneix i, de l'altra, els diferents idiomes que un programador domina. I, a més, aquests dos fets són independents entre si, és a dir, el fet que un programador conegui un llenguatge de programació o un altre no determina en absolut els idiomes que aquest mateix programador pugui parlar.

*** La relació *ConeixProgr* també presenta anomalies d'esborrament i modificació.**

Les redundàncies que s'originen en una relació que presenta fets multivaluats independents són immediates, atès que caldrà aplicar una política de manteniment de la relació basada en el producte cartesià amb l'objectiu que hi hagi totes les combinacions possibles i així mantenir la intensió de la relació. Respecte de la relació *ConeixProgr*, aquesta política implica que caldrà tenir registrades totes les parelles possibles de llenguatge de programació i idioma per cada programador.

La quarta forma normal identifica les anomalies que acabem de veure i indica com es poden solucionar en termes del concepte de dependència multivaluada independent.

Sigui $R$ una relació amb esquema $R(A_1,A_2,...,A_n)$ i siguin $\{ X \}$, $\{ Y \}$, $\{ Z \}$ atributs de $R$. Aleshores, la **dependència multivaluada independent** $\{ X \} \twoheadrightarrow \{ Y \}$ existeix si, i només si, el conjunt de valors possibles de $\{ Y \}$ per a un parell $\{ X, Z \}$ depèn únicament del valor de $\{ X \}$ i és independent del valor de $\{ Z \}$. Si existeix $\{ X \} \twoheadrightarrow \{ Y \}$, llavors també existirà la dependència multivaluada independent $\{ X \} \twoheadrightarrow \{ Z \}$.

Les conclusions principals que podem extreure de la definició prèvia són aquestes:

**a)** Una relació només pot tenir dependències multivaluades independents si la relació té clau composta formada, com a mínim, per tres atributs.

**b)** En una relació, les dependències multivaluades independents sempre es donen per parelles, és a dir, si $\{ X \} \twoheadrightarrow \{ Y \}$ independentment de $\{ Z \}$, llavors també $\{ X \} \twoheadrightarrow \{ Z \}$ independentment de $\{ Y \}$.

**c)** Si $R$ és una relació amb esquema $R(\underline{X},\underline{Y},\underline{Z})$ que conté les dependències multivaluades independents $\{ X \} \twoheadrightarrow \{ Y \}$ i $\{ X \} \twoheadrightarrow \{ Z \}$, i si en $R$ hi ha les tuples $< x, y, z >$ i $< x, y', z' >$, per simetria també hauran d'existir les tuples $< x, y, z' >$ i $< x, y', z >$.

**d)** Una dependència funcional $\{ X \} \rightarrow \{ Y \}$ és un cas particular de dependència multivaluada independent on el conjunt de valors que pren l'atribut dependent $\{ Y \}$ es restringeix a un únic valor.

De manera similar a les dependències funcionals, podem representar les dependències multivaluades independents de manera gràfica. Així, podem representar les dependències multivaluades independents corresponents a l'exemple de la relació *ConeixProgr* de la manera següent:

*ConeixProgr*(*DNI*,*LlengProg*,*Idiomes*)

A continuació indiquem les condicions que ha de complir una relació perquè estigui en quarta forma normal.

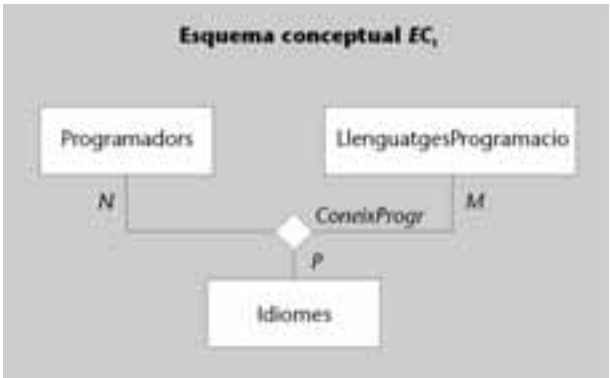
> Una relació està en **quarta forma normal** (**4FN**) si, i només si, està en FNBC i no té dependències multivaluades independents.

Per a aconseguir normalitzar una relació a 4FN, caldrà evitar les dependències multivaluades independents. Per tant, els atributs que participen en cada dependència multivaluada independent s'hauran de projectar en relacions diferents, tal com mostrem a continuació:

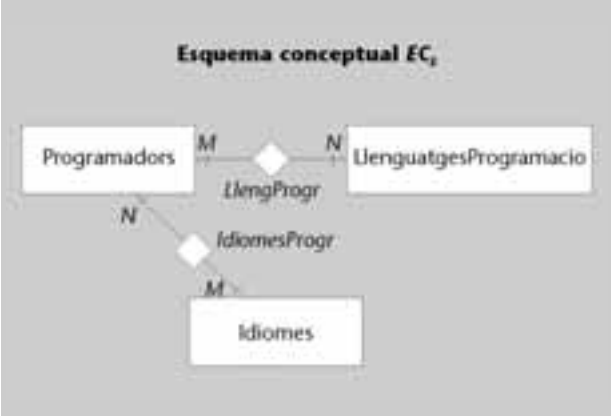
*LlengProgr*(*DNI*,*LlengProg*)

*IdiomesProgr*(*DNI*,*Idiomes*)

Fixem-nos que la relació *ConeixProgr* inicial s'ha obtingut com una interrelació ternària a partir de tres entitats. Aquestes serien una entitat *Programadors*, una entitat *Idiomes*, i una tercera entitat *LlenguatgesProgramacio*, tal com mostra l'esquema conceptual $EC_1$ següent:



Com a resultat d'haver normalitzat, hem separat la relació *ConeixProgr* original en dues. Si ara intentem esbrinar de quin esquema conceptual ($EC_2$) poden haver sorgit aquestes dues relacions, arribarem a la conclusió següent:




El primer esquema conceptual ($EC_1$) mostra precisament les anomalies que identifica la 4FN. El dissenyador de l'$EC_1$ ha elaborat un disseny dolent, atès que ha modelat com un únic fet ternari allò que en realitat són dos fets binaris independents. En canvi, el dissenyador del segon esquema conceptual ($EC_2$), ha distingit clarament els fets binaris independents. Com a conseqüència, el seu esquema no presenta anomalies.

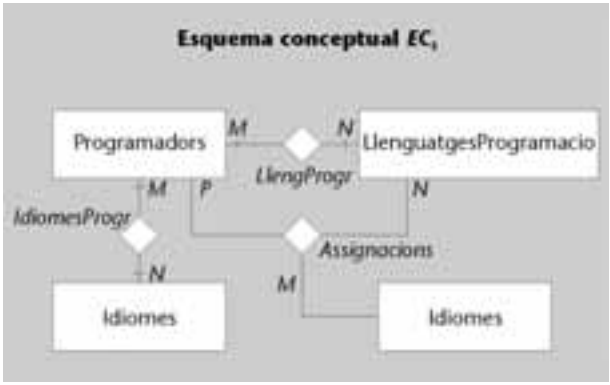
De totes maneres, no hem de cometre mai l'error de pensar que una interrelació ternària sempre representa dos fets binaris independents i que, per tant, sempre la podem substituir per dues interrelacions binàries. Al contrari, si això succeeix, ha estat com a resultat d'un disseny dolent que en cap cas s'ha de donar per vàlid.

Imaginem ara que en l'esquema conceptual $EC_2$ volem considerar també els aspectes següents:

**a)** Una entitat *Projectes* que recull dades sobre els diferents projectes que es desenvolupen a l'empresa. Concretament, es desenvolupen tres projectes identificats com PR001, PR002 i PR003. A més, sabem que el projecte PR001 es desenvolupa exclusivament en C++, mentre la resta de projectes es desenvolupen amb Java i C++.

**b)** Les assignacions dels programadors als diferents projectes. Caldrà considerar que la participació de cada programador als diferents projectes només serà possible si el programador coneix algun dels llenguatges que utilitzen en el desenvolupament del projecte. Per tant, també es vol registrar per a cada assignació, quin o quins llenguatges de programació fa servir cada programador dins el projecte.

Com a resultat d'afegir els elements previs, obtenim un esquema conceptual $EC_3$ nou que mostrem tot seguit:




Si elaborem el model lògic associat a l'esquema conceptual $EC_3$, obtenim un conjunt de relacions d'entre les quals destaquem la relació *Assignacions*:

*Assignacions*(*DNI*,*Projecte*,*LlengProg*)

Aquesta relació també incorpora una clau primària composta per tres atributs però, en aquest cas, la relació representa un fet ternari únic. La intensió de la relació és representar en quins projectes participen els diferents programadors i quin llenguatge de programació fan servir a cada assignació.

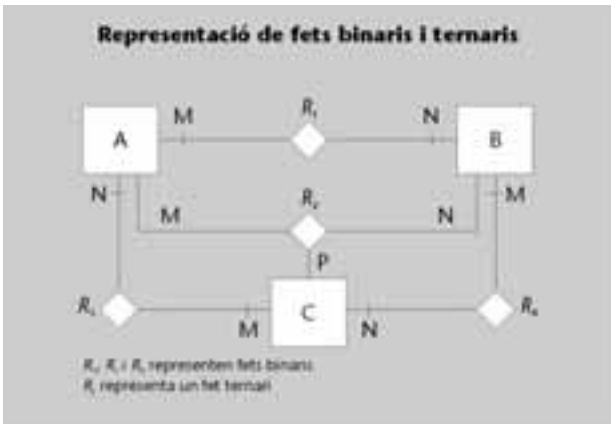
A més, també tenim fets multivaluats perquè, donat un valor concret de l'atribut *DNI*, podem inferir els valors que pot prendre l'atribut *LlengProg* i el conjunt de valors possibles de l'atribut *Projecte*. Però a diferència dels casos anteriors, aquests fets multivaluats són dependents entre si, atès que el llenguatge de programació que s'utilitza en cada projecte depèn del projecte concret que considerem (C++ a PR001, i Java i C++ a PR002 i PR003). En altres paraules, si poguéssim examinar l'extensió de la relació *Assignacions* veuríem que, donat un DNI concret, no trobem totes les parelles possibles de valors per a llenguatge de programació i projecte, tal com mostra la taula següent:

| Assignacions | | |
|---|---|---|
| **DNI** | **Projecte** | **LlengProg** |
| 56432999 | PR001 | C++ |
| 74567000 | PR001 | C++ |
| 74567000 | PR002 | C++ |
| 56432999 | PR002 | Java |
| 23450988 | PR002 | Java |
| 56432999 | PR003 | C++ |
| 56432999 | PR003 | Java |

Per tant, les dependències multivaluades que hi ha en una relació poden ser dependents o independents. Només quan les dependències multivaluades siguin independents, la relació violarà la 4FN. En cas que les dependències multivaluades siguin dependents, la relació presenta repeticions, però aquestes repeticions (en principi) són inevitables i, en conseqüència, no són redundàncies.

Per acabar, destaquem que a partir de la relació *Assignacions* no podem determinar el conjunt complet de llenguatges de programació que un programador coneix. Aquest fet ("llenguatges coneguts per cada programador") es recull a la interrelació *LlengProgr* de l'$EC_3$ que, en el model lògic, donarà lloc a la relació *LlengProgr*.

En resum, donat un esquema amb tres entitats (com a mínim), podem estar interessats no solament a reflectir fets ternaris entre aquestes tres entitats, sinó també a reflectir fets binaris tal com mostra la figura següent:




Fets binaris i ternaris en l'esquema conceptual

En l'esquema conceptual, els fets ternaris donen lloc a interrelacions ternàries, mentre que els fets binaris donen lloc a interrelacions binàries.

### 2.4.7. Cinquena forma normal

Hi ha relacions en què, malgrat que presenten redundàncies en el seu contingut, no es pot aplicar el procés de reducció a dues relacions que hem vist fins ara, és a dir, aquestes relacions no es poden descompondre en dues relacions sense que hi hagi pèrdua d'informació. Analitzem, per exemple, la relació següent:

| ProfAssigCentre | | |
|---|---|---|
| **Professor** | **Assignatura** | **Centre** |
| Pons | BD | FIB |
| Pons | Programació | FIB |
| García | BD | FIB |
| García | Àlgebra | FIB |
| García | Àlgebra | FME |
| Puig | Àlgebra | FIB |

La intensió de la relació *ProfAssigCentre* és oferir informació sobre quins professors imparteixen cada assignatura, quines assignatures s'imparteixen i en quins centres s'imparteix cada assignatura.

Si ens hi fixem, aquesta relació presenta redundàncies. Per exemple, el fet que un professor imparteixi una assignatura apareix tantes vegades com el nombre de centres diferents en què el professor imparteix l'assignatura. Un altre exemple: el fet que en un centre s'imparteix una assignatura concreta apareix per cada professor que imparteix l'assignatura al centre.

Malgrat això, la relació està en 4FN atès que, donat un professor, les assignatures que aquest imparteix i tots els centres on treballa, en l'extensió de la relació *ProfAssigCentre* no existeixen totes les possibles combinacions de professor, assignatura i centre. Si la relació no estigués en 4FN, a la seva extensió hi hauria d'haver, per exemple, la tupla < García, BD, FME> .

També és important que ens adonem que les redundàncies de la relació *ProfAssigCentre* no es poden solucionar amb un procés de descomposició de la relació en dues tal com mostrem a continuació, en què explorem totes les opcions de descomposició possibles:

**a)** Amb la descomposició següent no podem saber quins professors estan assignats a cada centre:

| ProfAssig | |
|---|---|
| **Professor** | **Assignatura** |
| Pons | BD |
| Pons | Programació |
| García | BD |
| García | Àlgebra |
| Puig | Àlgebra |

| AssigCentre | |
|---|---|
| **Assignatura** | **Centre** |
| BD | FIB |
| Programació | FIB |
| Àlgebra | FIB |
| Àlgebra | FME |

**b)** Amb aquesta altra descomposició no podem saber quines assignatures s'imparteixen a cada centre:

| ProfAssig | |
|---|---|
| **Professor** | **Assignatura** |
| Pons | BD |
| Pons | Programació |
| García | BD |
| García | Àlgebra |
| Puig | Àlgebra |

| ProfCentre | |
|---|---|
| **Professor** | **Centre** |
| Pons | FIB |
| García | FIB |
| García | FME |
| Puig | FIB |

**c)** Amb la descomposició que presentem a continuació no podem saber quines són les assignatures que imparteix cada professor:

| AssigCentre | |
|---|---|
| **Assignatura** | **Centre** |
| BD | FIB |
| Programació | FIB |
| Àlgebra | FIB |
| Àlgebra | FME |

| ProfCentre | |
|---|---|
| **Professor** | **Centre** |
| Pons | FIB |
| García | FIB |
| García | FME |
| Puig | FIB |

Les raons per les quals les descomposicions anteriors perden informació rauen en el fet que, malgrat que la relació original *ProfAssigCentre* presenta dependències basades en fets multivaluats, aquests fets no són independents entre si, sinó dependents. Donat un professor i una assignatura, els centres on el professor imparteix l'assignatura resulten perfectament determinats, i també, donat un professor i un centre, es determinen les assignatures que el professor imparteix en el centre. Per acabar, donat un centre i una assignatura, també podem saber els diferents professors que imparteixen l'assignatura al centre.

La diferència entre aquest exemple i el de la relació *ConeixProgr* és que en el primer, donat un programador, els llenguatges de programació i els idiomes que aquest coneixia restaven perfectament determinats, independentment els uns dels altres.

Vegeu l'exemple de la relació *ConeixProgr* en el subapartat 2.4.6 d'aquest mòdul.

En qualsevol cas, si la relació *ProfAssigCentre* presenta redundàncies, hi ha d'haver una llei de simetria que ens permeti de trobar una descomposició de la relació sense pèrdua d'informació. Sobre el nostre exemple, la llei de simetria és la següent:

**Si** un professor imparteix una assignatura **i**
l'assignatura s'imparteix en un centre **i**
en aquest centre hi treballa el professor
**llavors**
el professor imparteix l'assignatura al centre

Fixem-nos que la mateixa llei de simetria ens permet de descompondre la relació original en tres relacions noves que corresponen als tres fets multivaluats que hem detectat:

| ProfAssig | |
|---|---|
| **Professor** | **Assignatura** |
| Pons | BD |
| Pons | Programació |
| García | BD |
| García | Àlgebra |
| Puig | Àlgebra |

| AssigCentre | |
|---|---|
| **Assignatura** | **Centre** |
| BD | FIB |
| Programació | FIB |
| Àlgebra | FIB |
| Àlgebra | FME |

| ProfCentre | |
|---|---|
| **Professor** | **Centre** |
| Pons | FIB |
| García | FIB |
| García | FME |
| Puig | FIB |

Quan una relació es pot descompondre en més de dues sense pèrdua d'informació, es diu que la relació no està en cinquena forma normal.

Els fets multivaluats als quals es pot reduir la relació original són conseqüència d'una llei de simetria i constitueixen un nou tipus de dependència denominada **dependència de projecció-combinació**. El nom d'aquest tipus de dependència té l'origen en el fet que la relació original es pot reconstruir a partir de la combinació de les noves relacions en les quals la relació s'ha descompost.

En el nostre exemple podem comprovar que, per a recuperar la relació original *ProfAssigCentre*, n'hi ha prou de fer la combinació natural de les tres relacions noves que hem derivat, tal com mostrem a continuació:

$$ProfAssigCentre = (ProfAssig * AssigCentre) * ProfCentre$$

La combinació natural entre *ProfAssig* i *AssigCentre* es fa segons l'atribut *Assignatura*, mentre que l'altra combinació natural es fa segons els atributs *Professor* i *Centre*.

Les operacions de combinació i de combinació natural s'estudien en l'assignatura *Bases de dades I*.

Ara ja estem en condicions de definir de manera més precisa les propietats que ha de complir una relació perquè estigui en cinquena forma normal.

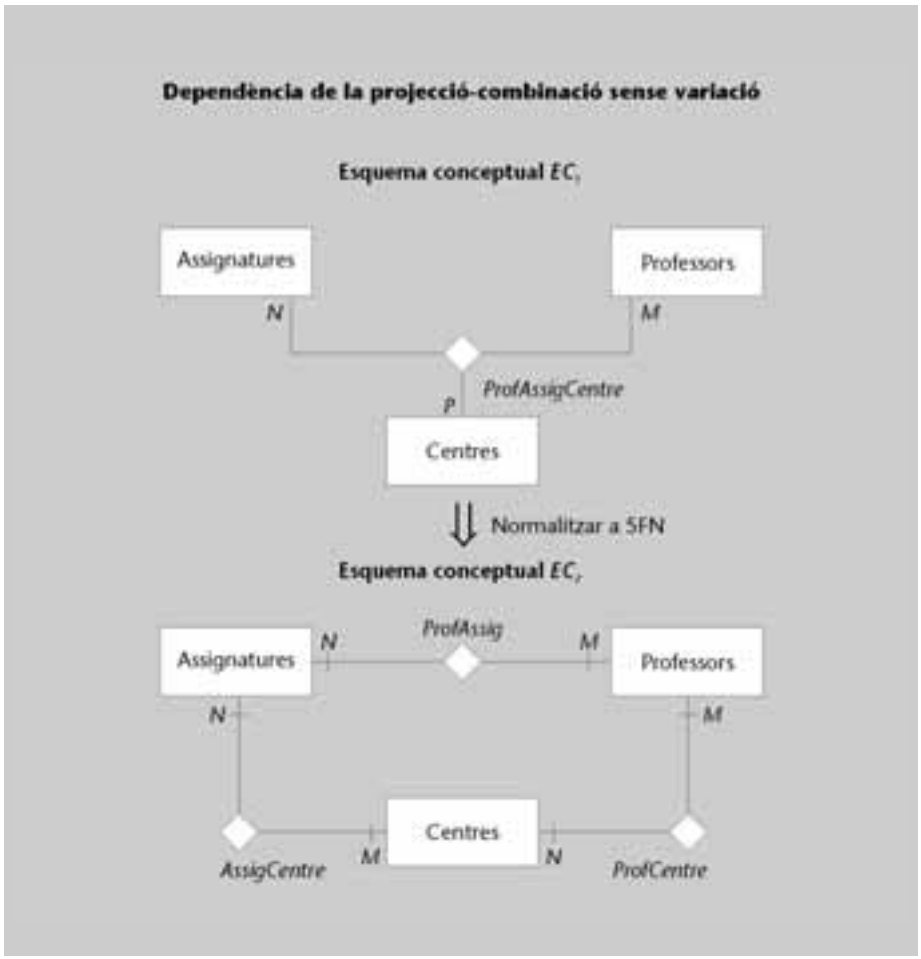
**La cinquena forma normal va ser proposada per Fagin l'any 1979.**

> Una relació està en **cinquena forma normal** (**5FN**) si, i només si, està en 4FN i no té dependències de projecció-combinació sense variació d'informació.

A partir de la definició de cinquena forma normal podem deduir que és possible tenir dependències de projecció-combinació sense variació d'informació i amb variació d'informació. Analitzem cada cas amb una mica més de detall:

**a) Dependències de projecció-combinació sense variació**: la relació no està en 5FN i, per tant, presenta redundàncies. Hi ha d'haver una llei de simetria i la relació original es pot descompondre en relacions noves. A més, sempre podrem reconstruir la relació original de manera exacta, és a dir, sense variació en la informació que s'hi conté, mitjançant la combinació de les noves relacions obtingudes com a conseqüència de la descomposició.

Aquest és el cas de la relació *ProfAssigCentre*. Si n'intentem esbrinar l'origen, ens adonarem que s'ha obtingut com una interrelació ternària a partir de tres entitats. Aquestes entitats serien *Professors*, *Assignatures* i *Centres*, tal com mostra l'esquema conceptual $EC_1$ de la figura següent. Per la seva banda, a la mateixa figura, $EC_2$ mostra l'esquema conceptual resultat d'haver aplicat el procés de descomposició de la relació original.




**a) Dependències de projecció-combinació amb variació**: en aquest cas, un cop descomposta la relació original en relacions noves, ja no es pot reconstruir. La relació original podia tenir repeticions, però aquestes són inevitables i, per tant, no són redundàncies. La relació original representava un únic fet del món real i ja estava en 5FN.

Per a il·lustrar aquesta situació modifiquem el nostre exemple, n'eliminem la tupla < García, Àlgebra, FIB > de la relació *ProfAssigCentre* de manera que obtenim una altra relació que anomenem *ProfAssigCentre*', tal com mostrem a continuació:

| ProfAssigCentre' | | |
|---|---|---|
| **Professor** | **Assignatura** | **Centre** |
| Pons | BD | FIB |
| Pons | Programació | FIB |
| García | BD | FIB |
| García | Àlgebra | FME |
| Puig | Àlgebra | FIB |

Si tornem a descompondre la relació original en tres noves relacions *ProfAssig*', *AssigCentre*' i *ProfCentre*' obtenim el següent:

| ProfAssig' | |
|---|---|
| **Professor** | **Assignatura** |
| Pons | BD |
| Pons | Programació |
| García | BD |
| García | Àlgebra |
| Puig | Àlgebra |

| AssigCentre' | |
|---|---|
| **Assignatura** | **Centre** |
| BD | FIB |
| Programació | FIB |
| Àlgebra | FME |
| Àlgebra | FIB |

| ProfCentre' | |
|---|---|
| **Professor** | **Centre** |
| Pons | FIB |
| García | FIB |
| García | FME |
| Puig | FIB |

Si a ara intentem reconstruir la relació original mitjançant l'aplicació d'operacions de combinació, obtenim el resultat següent:

*ProfAssigCentre*' = (*ProfAssig*' * *AssigCentre*') * *ProfCentre*'

| ProfAssig' * AssigCentre' | | |
|---|---|---|
| **Professor** | **Assignatura** | **Centre** |
| Pons | BD | FIB |
| Pons | Programació | FIB |
| García | BD | FIB |
| García | Àlgebra | FME |
| García | Àlgebra | FIB |
| Puig | Àlgebra | FME |
| Puig | Àlgebra | FIB |

| (ProfAssig' * AssigCentre') * ProfCentre' | | |
|---|---|---|
| **Professor** | **Assignatura** | **Centre** |
| Pons | BD | FIB |
| Pons | Programació | FIB |
| García | BD | FIB |
| García | Àlgebra | FME |
| García | Àlgebra | FIB → Tupla afegida |
| Puig | Àlgebra | FIB |

Com podem veure, la reconstrucció no ha estat possible. La relació obtinguda ((*ProfAssig*' * *AssigCentre*') * *ProfCentre*') mitjançant l'aplicació d'operacions de combinació natural incorpora una tupla que no hi era en la relació original de *ProfAssigCentre*', precisament la tupla < García, Àlgebra, FIB >. En aquest cas, en l'intent de reconstrucció, hem afegit una tupla i, per tant, hi ha hagut variació d'informació.

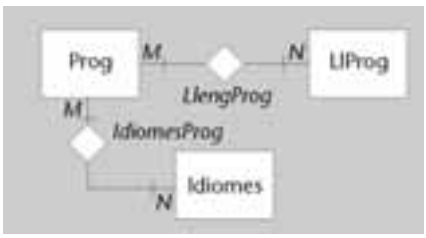
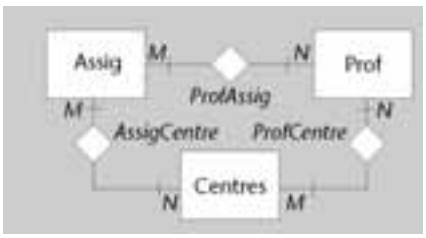
En conseqüència, la relació *ProfAssigCentre*' ja estava en 5FN. A més, si ens hi fixem, la llei de simetria a *ProfAssigCentre* ha deixat de ser vàlida per a la relació *ProfAssigCentre*'. Ja no és cert que quan un professor imparteix una assignatura i aquesta s'imparteix al centre, el professor imparteix l'assignatura al centre: el professor García imparteix àlgebra, àlgebra s'imparteix a la FIB, però no és cert que el professor García imparteixi àlgebra a la FIB. En resum, la relació *ProfAs-*

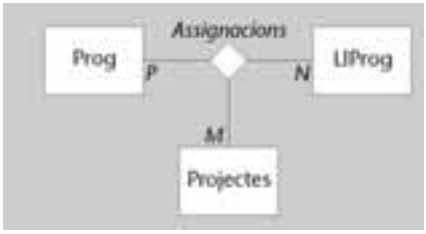
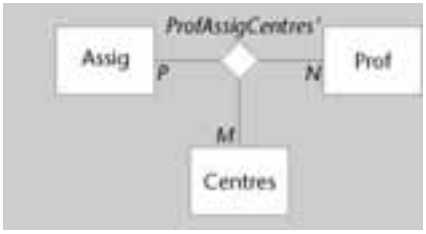
*sigCentre*' descriu un únic fet ternari entre les entitats *Professors*, *Assignatures* i *Centres* tal com es mostra a la figura següent:


Per acabar, només resta aclarir que mentre les dependències funcionals i les dependències multivaluades independents són relativament fàcils d'identificar perquè tenen una interpretació directa en el món real, les dependències de projecció-combinació, que no tenen un significat intuïtiu obvi, poden resultar difícils de descobrir. Per tant, el procés per a determinar quan una relació està en 4FN, però no està en 5FN, no és directe. Afortunadament, aquestes relacions es poden considerar com a casos patològics que rarament apareixen a la pràctica.

### 2.4.8. Conclusions sobre les formes normals basades en fets multivaluats

La taula següent resumeix els objectius fonamentals de les formes normals 4FN i 5FN:

| 4FN | 5FN |
|---|---|
| Una relació està en 4FN si, i només si, està en FNBC i no té dependències multivaluades independents. | Una relació està en 5FN si, i només si, està en 4FN i no té dependències de projecció-combinació sense variació d'informació. |
| **1)** Dependències multivaluades independents:<br>• La relació no està en 4FN.<br>• La relació presenta redundàncies que apareixen com a conseqüència d'aplicar una política basada en el producte cartesià.<br>• La relació no representa un únic fet del món real. Cal normalitzar! Els diferents fets es representen amb interrelacions per "fora":<br>  | **1)** Dependències de projecció-combinació sense variació:<br>• La relació no està en 5FN.<br>• La relació presenta redundàncies que apareixen com a conseqüència d'una llei de simetria.<br>• La relació no representa un únic fet del món real. Cal normalitzar! Els diferents fets es representen amb interrelacions per "fora":<br>  |

| 4FN | 5FN |
|---|---|
| **2)** Dependències multivaluades dependents:<br>• La relació ja està en 4FN.<br>• La relació presenta repeticions, però són inevitables* atès que no s'aplica una política basada en el producte cartesià.<br>• La relació representa un únic fet* del món real, que es representa com una interrelació per "dins":<br>  | **2)** Dependències de projecció-combinació amb variació:<br>• La relació ja està en 5FN.<br>• La relació presenta repeticions, però són inevitables, atès que no hi ha una llei de simetria.<br>• La relació representa un únic fet del món real, que es representa com una interrelació per "dins":<br>  |

* La veracitat d'aquestes afirmacions queden supeditades al fet que la relació estigui en 5FN

Si analitzem detingudament el contingut de la taula, podem extreure les conclusions següents:

**a)** Una relació només pot violar les formes normals 4FN o 5FN si s'ha obtingut com a conseqüència d'una interrelació entre entitats. És més, aquesta interrelació ha de ser, com a mínim, ternària. En altres paraules, una relació només pot violar les formes normals 4FN o 5FN si la seva clau primària està formada, com a mínim, per tres atributs.

**b)** Atès que una relació només pot estar en 5FN si també està en 4FN, les dependències de projecció-combinació són un cas particular de les dependències multivaluades dependents.

La forma normal 5FN és l'objectiu últim a assolir. Una relació que està en la forma normal 5FN pot presentar repeticions en el seu contingut, però aquestes són inevitables i, per tant, no són redundàncies.

## 2.5. Aplicació de la teoria de la normalització al disseny de bases de dades relacionals clàssiques

El disseny de bases de dades relacionals comença sempre amb la captura i abstracció dels requisits de dades dels usuaris de l'organització que encarreguen el disseny de la base de dades. Això implica que el dissenyador ha de representar amb la màxima exactitud possible els objectes (o entitats) de món real que hi intervenen, les propietats (o atributs), les interrelacions entre entitats, i també les regles d'integritat que s'han de complir. Tanmateix, és important distingir entre un disseny d'una base de dades partint de zero i un disseny d'una base de dades que parteixi de dissenys parcials ja existents.

En el segon cas, els dissenys parcials s'integren en un disseny únic i nou per a l'organització. La utilització d'aquests dissenys és freqüent en organitzacions que volen centralitzar diferents sistemes d'informació que fins ara funcionaven de manera independent i també en el disseny de grans bases de dades; aleshores, en el disseny de la base de dades intervenen diferents equips que fan dissenys parcials amb l'objectiu d'obtenir un disseny global. En ambdós casos és necessària una etapa d'integració de vistes per a obtenir un esquema conceptual global.

En canvi, en el primer cas, el disseny de la base de dades es fa a partir de zero i es pot tractar teòricament des de dos punts de vista extrems:

**1)** Es pot partir d'una única relació, també coneguda com a **relació universal**, que conté tots els atributs d'interès per a una organització concreta, i també totes les dependències funcionals entre aquests atributs. A partir d'aquests elements, caldrà aplicar de manera reiterada un procés de normalització amb l'objectiu d'obtenir un esquema relacional (o conjunt d'esquemes de relacions) que caracteritzarà la base de dades de l'organització.

**2)** Es parteix d'un conjunt d'atributs d'interès per a una determinada organització i de les respectives dependències funcionals, i després es construeix un esquema relacional que caracteritza la base de dades de l'organització.

El primer enfocament sobre el disseny teòric de la base de dades dóna lloc als mètodes d'anàlisi o descomposició, també coneguts com a **mètodes descendents***, mentre que el segon dóna lloc als mètodes de síntesi o composició, també coneguts com a **mètodes ascendents****.

*** En anglès, *top-down*.**
**** En anglès, *bottom-up*.**

Tots dos enfocaments fan servir mètodes que podríem denominar *purs*. Tanmateix, a la pràctica, s'utilitzen mètodes mixtos, que combinen anàlisi i síntesi.

En els **mètodes mixtos**, el dissenyador haurà d'elaborar el disseny conceptual de la base de dades mitjançant un model semàntic de dades.

Després de l'elaboració del model conceptual mitjançant els mètodes mixtos, caldrà traduir el model obtingut a un disseny equivalent, però ara expressat en el model relacional. Com a resultat d'aquest procés, obtenim el que es coneix com a *disseny lògic de la base de dades*. Per acabar, caldrà implementar el disseny lògic obtingut sobre l'SGBD relacional concret que empra l'organització i així donar lloc al disseny físic de la base de dades.

Per què és més natural la utilització dels mètodes mixtos? La resposta és senzilla. Quan un dissenyador detecta un atribut, també detecta a quina o quines

entitats pertany l'atribut en qüestió. O a l'inrevés: quan es detecta una entitat, també es poden detectar fàcilment una bona part dels seus atributs. Això significa que implícitament s'identifiquen algunes de les relacions que formen part del disseny lògic de la base de dades, precisament aquelles que corresponen a entitats en el disseny conceptual.

Explicitar les interrelacions entre les entitats és una tasca que està contemplada en qualsevol model semàntic de dades. En el disseny lògic posterior, aquestes interrelacions donaran lloc a relacions noves o a atributs nous que han de complir unes regles d'integritat determinades.

Així com la normalització sempre és necessària en els dissenys elaborats amb els mètodes d'anàlisi **1** (basats en la relació universal), també és convenient d'aplicar-la al final del procés dels mètodes mixtos per al disseny de bases de dades relacionals clàssiques. D'aquesta manera eliminarem redundàncies i evitarem les anomalies de disseny que hem estudiat a l'inici de la teoria de la normalització. Malgrat això, convé tenir present que el seguiment correcte de qualsevol metodologia de disseny condueix a un disseny lògic compost per relacions que verifiquen, com a mínim, la tercera forma normal.

En definitiva, la normalització resulta útil com a la comprovació d'aspectes semàntics difícils, però poques vegades és realment necessària, atès que un dissenyador mínimament experimentat és capaç de distingir fets semànticament independents i, per tant, ja té en compte el principi bàsic en el qual es fonamenta la teoria de la normalització.

En casos molt particulars pot interessar que una relació concreta no verifiqui una forma normal determinada. Aquesta tècnica es coneix en el disseny físic de bases de dades com a **desnormalització.** Pensem que la normalització minimitza l'existència de redundàncies i prevé les anomalies de disseny d'una relació per a millorar la integritat de les dades, però es paga el preu d'una degradació en el rendiment perquè les dades que abans estaven en una única relació, després de la normalització estan disperses en diferents relacions.

La normalització, doncs, tendeix a penalitzar la recuperació o consulta de les dades perquè exigeix operacions de combinació per a recuperar dades que originalment estaven en una única relació. Quan els atributs d'una relació que representen un altre concepte semàntic no s'actualitzen amb freqüència, potser resulta convenient de fer servir la desnormalització.

En el mòdul "Reconsideració dels models conceptual i lògic" s'estudien els avantatges i inconvenients de la desnormalització.

Per acabar, destaquem que el nostre estudi s'ha centrat en el disseny i la normalització de bases de dades relacionals clàssiques, és a dir, en bases de dades relacionals que es fonamenten en la teoria de conjunts, la lògica i l'àlgebra relacional. La nova generació d'SGBD relacionals incorpora el paradigma de l'orientació a l'objecte i dóna lloc a les bases de dades relacional amb objectes.

Les bases de dades relacional amb objectes incorporen relacions que no estan en primera formal normal ($NF^2$) i, per tant, la teoria de la normalització que hem estudiat en aquest mòdul no és vàlida. Això no vol dir que no hi hagi d'haver una teoria de la normalització per les bases de dades relacional amb objectes, sinó que aquesta teoria de la normalització ha de ser ben diferent. En qualsevol cas, l'objectiu bàsic continua essent el mateix: evitar redundàncies i eliminar anomalies de disseny. 

# Resum

En aquest mòdul hem estudiat disseny de bases de dades relacionals clàssiques. A la primera part hem vist que, des d'un punt de vista didàctic, podem estructurar el disseny d'una base de dades en diferents etapes, si bé l'objectiu cabal és l'obtenció del disseny conceptual, el disseny lògic i el disseny físic de la base de dades.

El disseny conceptual modelitza les necessitats de dades de l'empresa que encarrega el disseny de la base de dades. Per a l'elaboració, caldrà fer servir un model semàntic de dades; en el nostre cas el model triat ha estat l'*Unified Modeling Language* (UML), encara que també hem comentat altres models semàntics de dades. Un cop obtingut el disseny conceptual de la base de dades, cal transformar-lo a un model de bases de dades relacional genèric que, en el nostre cas, és el model relacional clàssic; el resultat d'aquesta transformació constitueix el disseny lògic de la base de dades.

Finalment, atès que del model relacional clàssic s'han fet nombroses implementacions, hem estudiat que cal acomodar el disseny lògic obtingut a l'SGBD concret de què l'empresa que encarrega el disseny de la base de dades disposa. Aquesta acomodació dóna lloc al disseny físic de la base de dades.

A la segona part del mòdul hem estudiat que el disseny d'una base de dades ha de complir uns requisits mínims de qualitat. Hem vist que, com a conseqüència d'un disseny conceptual dolent, podem obtenir un disseny lògic de la base de dades amb redundàncies i anomalies de disseny. La teoria de la normalització ens permet de detectar aquestes redundàncies i anomalies, mitjançant el concepte de *forma normal*. Atès que les formes normals són declaratives, hem mostrat diferents alternatives per a evitar les condicions que fan que es violi una forma normal determinada.

També hem estudiat diferents alternatives per a dur a terme el disseny d'una base de dades, i hem vist el paper que fa la teoria de la normalització en alguna d'aquestes alternatives.

## Activitats

**1.** En les assignatures *Bases de dades I* i *Bases de dades II* heu fet el disseny conceptual i el disseny lògic de diferents bases de dades. Agafeu aquests dissenys lògics i verifiqueu en quina forma normal estan les diferents relacions que els componen.

**2.** En cas que detecteu relacions que no estan en una determinada forma normal, procediu a la seva normalització. Un cop tingueu el disseny lògic normalitzat, intenteu de trobar el disseny conceptual que hauria donat lloc a aquest disseny lògic nou, i raoneu per què us havíeu equivocat.

## Exercicis d'autoavaluació

**1.** Considereu la relació *Comandes* amb l'esquema següent:

*Comandes*(*CodiComanda*,*CodiProducte*,*Quantitat*,*CodiClient*,*DataComanda*)

La intensió d'aquesta relació és representar dades referents als productes que els clients demanen en cadascuna de les seves comandes. Es demana:
**a)** En quina forma normal està la relació? Per què?
**b)** Normalitzeu-la tant com es pugui.

**2.** Considereu la relació d'*Horaris* amb l'esquema següent:

*Horaris*(*CodiProf*,*CodiAssig*,*Dia*,*Hora*,*Aula*,*NomAssig*)

La seva intensió és representar dades respecte als horaris de les assignatures que s'imparteixen en un centre universitari presencial. Es demana:
**a)** En quina forma normal està la relació? Per què?
**b)** Normalitzeu-la tant com es pugui.

**3.** Considereu la relació universal $R(A,B,C,D,E,F,G)$ i les dependències funcionals següents:

$$\{ A, B \} \rightarrow \{ C \}$$
$$\{ A \} \rightarrow \{ E \}$$
$$\{ A, B \} \rightarrow \{ D \}$$
$$\{ A \} \rightarrow \{ F \}$$
$$\{ F \} \rightarrow \{ G \}$$

Normalitzeu $R$ fins almenys la forma normal de Boyce-Codd.

**4.** Una empresa dedicada al transport internacional de mercaderies vol registrar dades dels viatges que fan els seus empleats. Per cada viatge que es fa s'utilitza un únic camió, on hi viatgen un conductor o més segons la distància del viatge. Addicionalment, cada viatge té com a destinació una única ciutat del continent. Es vol registrar les dietes dels conductors en cada viatge. Per això s'ha creat una relació *Dietes* amb l'esquema següent:

*Dietes*(*CodiViatge*,*CodiConductor*,
*MatriculaCamio*,*ImportDieta*,*CiutatDestinacio*,*PaisDestinacio*)

Es demana:
**a)** En quina forma normal està la relació? Per què?
**b)** Normalitzeu-la tant com es pugui.

**5.** Justifiqueu les respostes a les preguntes següents i poseu-hi exemples:
**a)** Si en una relació $R$ amb esquema $R(X,Y,Z,U)$ es compleix que l'atribut $X$ depèn funcionalment en forma completa dels atributs $Y$ i $Z$ (és a dir, hi ha dependència funcional $\{ Y, Z \} \rightarrow \{ X \}$), es pot deduir que $\{ Y, Z \}$ és clau candidata de $R$?
**b)** Si $\{ Y,Z \}$ és la clau primària de la relació $R$ del punt anterior, podem deduir d'aquest fet que $X$ depèn funcionalment en forma completa de $\{ Y, Z \}$?

# Solucionari

**1.** Per a solucionar aquest exercici cal que busquem les dependències funcionals de la relació *Comandes*. Aquestes dependències són:

{ *CodiComanda*,*CodiProducte* } → { *Quantitat* }
{ *CodiComanda*,*CodiProducte* } → { *CodiClient* }
{ *CodiComanda*,*CodiProducte* } → { *DataComanda* }

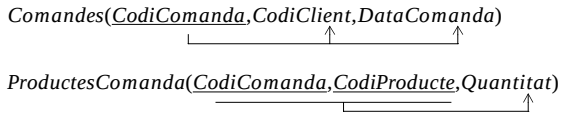
Aquestes dependències funcionals es deriven del fet que { *CodiComanda*,*CodiProducte* } és la clau primària de la relació *Comandes*. Addicionalment, atès que cada comanda és d'un únic client i es tramita en una data determinada, arribem a la conclusió que també hi ha les dependències funcionals següents:

{ *CodiComanda* } → { *CodiClient* }
{ *CodiComanda* } → { *DataComanda* }

**a)** Per tant, en la relació *Comandes* hi ha atributs que no depenen de totalment de la clau primària i, a més, aquests atributs no són clau alternativa de la relació ni formen part de cap clau alternativa de la relació. En conseqüència, la relació *Comandes* està en 1FN.
Per a normalitzar la relació *Comandes* a 2FN caldrà evitar les dependències funcionals no completes respecte de la clau, tal com mostrem a continuació:

*Comandes*(*CodiComanda*,*CodiClient*,*DataComanda*)
*ProductesComanda*(*CodiComanda*,*CodiProducte*,*Quantitat*)

Si representem de manera gràfica el conjunt de dependències funcionals de cada relació, obtenim:




Fixeu-vos que, com a conseqüència de la normalització, hem separat els dos conceptes semàntics que hi havia representats en la relació *Comandes* original:
- Les comandes en si mateixes
- Els productes que es demanen a cada comanda

Si examinem les dues relacions obtingudes (*Comandes* o *ProductesComanda*) arribem ràpidament a la conclusió que totes dues estan en 5FN.

**2.** Per a solucionar aquest exercici cal que busquem les dependències funcionals de la relació *Horaris*. Aquestes dependències són:

{ *CodiProf*,*CodiAssig*,*Dia*,*Hora* } → { *Aula* }
{ *CodiProf*,*CodiAssig*,*Dia*,*Hora* } → { *NomAssig* }

Aquestes dependències funcionals són conseqüència del fet que { *CodiProf*,*CodiAssig*,*Dia*,*Hora* } és la clau primària de la relació *Horaris*. A més, també hi ha les dependències funcionals següents:

{ *CodiAssig* } → { *NomAssig* }
{ *NomAssig* } → { *CodiAssig* }

En la relació hi ha un atribut que no depèn totalment de la clau primària:

{ *CodiAssig* } → { *NomAssig* }

Aquest fet podria violar la 2FN. Però aquest atribut forma part d'una clau alternativa de la relació *Horaris*. Efectivament, la relació *Horaris* té la clau alternativa següent:

{ *CodiProf*,*NomAssig*,*Dia*,*Hora* }

Per tant, s'aplica l'excepció associada a la 2FN i podem afirmar que com a mínim la relació està en 2FN.
Si tornem a examinar el conjunt de dependències funcionals, ens adonem que hi ha un atribut no clau que determina funcionalment un altre atribut no clau, la dependència funcional:

{ *NomAssig* } → { *CodiAssig* }

Aquest fet podria violar la 3FN i, atès que involucra un atribut que forma part d'una clau alternativa, s'aplica l'excepció. En conseqüència, la relació d'*Horaris* també verifica la 3FN.

Per a saber si la relació *Horaris* està en la FNBC, cal que refinem el nostre conjunt de dependències funcionals. Atès que { *CodiProf,NomAssig,Dia,Hora* } és clau alternativa de la relació d'*Horaris*, estem segurs que han d'existir les dependències funcionals següents:

{ *CodiProf,NomAssig,Dia,Hora* } → { *Aula* }
{ *CodiProf,NomAssig,Dia,Hora* } → { *CodiAssig* }

Si ara comparem tots els determinants obtinguts i les claus candidates de la relació *Horaris*, veiem que no tots els determinants són claus candidates, i per tant, la relació no està en FNBC. A continuació mostrem els determinants i les claus candidates de la relació *Horaris*:

| Horaris | |
|---|---|
| **Determinants** | **Claus candidates** |
| CodiProf,CodiAssig,Dia,Hora | CodiProf,CodiAssig,Dia,Hora |
| CodiProf,NomAssig,Dia,Hora | CodiProf,NomAssig,Dia,Hora |
| CodiAssig | |
| NomAssig | |

En conclusió, la relació *Horaris* està en 3FN.
Per a aconseguir normalitzar la relació *Horaris* a la FNBC, caldrà evitar les dependències funcionals següents:

{ *CodiAssig* } → { *NomAssig* }
{ NomAssig } → { CodiAssig }

Això es pot aconseguir de diferents maneres. A continuació mostrem totes les possibilitats:

**a)**

*Horaris*(*CodiProf*,*CodiAssig*,*Dia*,*Hora*,*Aula*)

*Assignatures*(*CodiAssig*,*NomAssig*)

**b)**

*Horaris*(*CodiProf*,*CodiAssig*,*Dia*,*Hora*,*Aula*)

*Assignatures*(*NomAssig*,*CodiAssig*)

**c)**

*Horaris*(*CodiProf*,*NomAssig*,*Dia*,*Hora*,*Aula*)

*Assignatures*(*NomAssig*,*CodiAssig*)

**d)**

*Horaris*(*CodiProf*,*NomAssig*,*Dia*,*Hora*,*Aula*)

*Assignatures*(*CodiAssig*,*NomAssig*)

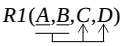
Fixeu-vos que, com a conseqüència de la normalització, hem separat els dos conceptes semàntics que hi havia representats en la relació original *Horaris*:
- Els horaris en si mateixos
- Les assignatures impartides

**3.** Perquè una relació estigui en la FNBC cal aconseguir que tots els determinants siguin claus candidates de la relació. Si partim d'aquest principi i apliquem els axiomes d'Armstrong, ens adonem que cal descompondre la relació original *R* en tres relacions que anomenarem *R1*, *R2* i *R3*:

**a)** Relació *R1*:

(1) $\{ A, B \} \rightarrow \{ C \}$
(2) $\{ A, B \} \rightarrow \{ D \}$
(3) $\{ A, B \} \rightarrow \{ C, D \}$ aquesta dependència funcional s'obté com a conseqüència d'aplicar l'axioma 4 d'Armstrong (additivitat) sobre (1) i (2).

Per tant la relació *R1* té l'esquema següent:



**b)** Relació *R2*:

(1) $\{ A \} \rightarrow \{ E \}$
(2) $\{ A \} \rightarrow \{ F \}$
(3) $\{ A \} \rightarrow \{ E, F \}$ aquesta dependència funcional s'obté com a conseqüència d'aplicar l'axioma 4 d'Armstrong (additivitat) sobre (1) i (2).

Per tant la relació R2 té l'esquema següent:

**c)** Relació *R3*:

(1) $\{ F \} \rightarrow \{ G \}$

Per tant, la relació *R3* té l'esquema següent:

Tenint en compte el conjunt de dependències funcionals subministrat, aquesta és l'única descomposició possible de *R* en relacions que verifiquen la FNBC. Per exemple, la descomposició següent no hauria estat vàlida:

*R1*(*A*,*B*,*C*,*D*)
*R2*(*A*,*E*,*F*,*G*)

L'esquema de *R2* s'hauria obtingut a partir de:

(1) $\{ A \} \rightarrow \{ E \}$
(2) $\{ A \} \rightarrow \{ F \}$
(3) $\{ F \} \rightarrow \{ G \}$
(4) $\{ A \} \rightarrow \{ G \}$, aquesta dependència funcional s'obté aplicant l'axioma 5 d'Armstrong (transitivitat)
(5) $\{ A \} \rightarrow \{ E, F, G \}$, aquesta dependència funcional s'obté aplicant l'axioma 4 d'Armstrong (additivitat)

Si representem de manera gràfica les dependències funcionals de *R2* obtenim:

La relació *R2* no pot estar en la FNBC perquè no està en 3FN, atès que *F* no és clau alternativa ni forma part de cap clau alternativa de *R2*. Si això no fos cert i *F* fos clau alternativa de *R2*, llavors al conjunt inicial de dependències funcionals hi hauria d'haver hagut les dependències:

$\{ F \} \rightarrow \{ A \}$ i $\{ F \} \rightarrow \{ E \}$.

**4.** Per a solucionar aquest exercici cal que busquem les dependències funcionals de la relació *Dietes*. Aquestes dependències són:

$\{$ *CodiViatge*,*CodiConductor* $\} \rightarrow \{$ *MatriculaCamio* $\}$

{ *CodiViatge*,*CodiConductor* } → { *ImportDieta* }
{ *CodiViatge*,*CodiConductor* } → { *CiutatDestinacio* }
{ *CodiViatge*,*CodiConductor* } → { *PaisDestinacio* }

Aquestes dependències funcionals són conseqüència del fet que { *CodiViatge*,*CodiConductor* } és la clau primària de la relació *Dietes*.

Atès que en cada viatge només es fa servir un camió i la destinació és única, també hi ha les dependències funcionals següents:

{ *CodiViatge* } → { *MatriculaCamio* }
{ *CodiViatge* } → { *CiutatDestinacio* }
{ *CodiViatge* } → { *PaisDestinacio* }

Per acabar, la darrera dependència funcional que trobem és aquesta:

{ *CiutatDestinacio* } → { *PaisDestinacio* }

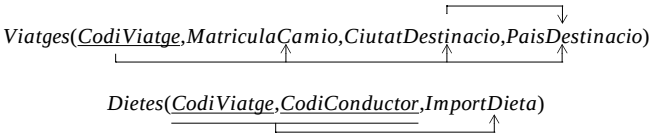
La veracitat d'aquesta dependència funcional roman supeditada al fet que no es facin viatges a ciutats que, tinguin el mateix nom, però estiguin en països diferents.

A partir d'aquestes dependències funcionals podem dir el següent:

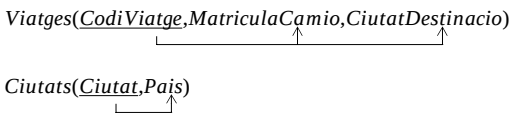
**a)** Atès que en la relació *Dietes* hi ha atributs que no depenen totalment de la clau primària i, a més, aquests atributs no són clau alternativa ni formen part de claus alternatives de la relació de *Dietes*, podem afirmar que la relació *Dietes* està en 1FN.

**b)** Hi ha diferents alternatives per a normalitzar la relació. A continuació en mostrem dues:

- Normalitzem la relació *Dietes* original a 2FN. Cal descompondre la relació original en dues:

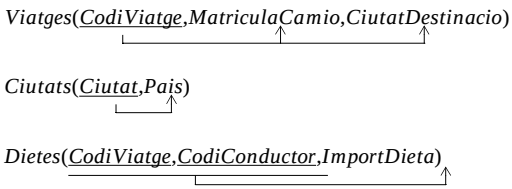



La nova relació *Dietes* ja està en 5FN; en canvi, la relació *Viatges* viola la 3FN, atès que hi ha dependències funcionals entre atributs que no són clau ni formen part de claus alternatives. Per a normalitzar la relació *Viatges*, cal descompondre-la en dues relacions, tal com mostrem tot seguit:




Ara, tant la relació de *Viatges* com la relació de *Ciutats* estan en 5FN.

- En aquest cas, i després d'examinar detingudament la relació *Dietes*, ens adonem que a la relació es representen tres fets diferents del món real: els viatges, les dietes dels diferents viatges i les ciutats. Per a aconseguir separar aquests fets, normalitzem directament fins a la FNBC, que ens garanteix que qualsevol determinant a una dependència funcional ha de ser una clau candidata de la relació on es dóna la dependència funcional. Com a conseqüència, obtenim tres relacions: la relació *Viatges*, la relació *Ciutats* i, finalment, la relació *Dietes*:




**5.** A continuació raonem les respostes a les preguntes formulades:

**a)** Del fet que l'atribut *X* depengui funcionalment de forma completa dels atributs { *Y*, *Z* }, no podem deduir sempre que { *Y*, *Z* } sigui clau candidata de la relació. La veracitat de l'afirmació depèn de si la relació *R* està, o no, en FNBC. Només quan la relació està en la FNBC es verifica que tots els determinants d'una dependència funcional són claus candidates. A con-

tinuació mostrem un exemple en què un atribut depèn funcionalment de manera completa d'un conjunt d'atributs, i aquest conjunt d'atributs no és clau candidata de la relació:

*EmpleatsBanc*(*CodiEmpleat*,*NomEmpleat*,
*SouEmpleat*,*CodiSucursal*,*CiutatSucursal*; *AdreçaSucursal*)

La intensió de la relació *EmpleatsBanc* és mostrar dades respecte d'un conjunt d'empleats que treballen en una entitat bancària. De cada empleat se'n guarda el codi (que és la clau primària de la relació), el nom, el sou que cobra, la sucursal del banc on treballa i l'adreça de la sucursal. Suposem que les diferents sucursals s'identifiquen per un codi de sucursal que és diferent per a cada sucursal del banc dins una mateixa ciutat, però que pot coincidir en sucursals que estan en diferents ciutats. Per tant, cada sucursal queda identificada pels atributs { *CodiSucursal*, *CiutatSucursal* }.

Entre les dependències funcionals que hi ha en la relació *EmpleatsBanc*, voldríem destacar la dependència funcional següent:

$$\{ CodiSucursal, CiutatSucursal \} \rightarrow \{ AdreçaSucursal \}$$

L'atribut *AdreçaSucursal* depèn funcionalment de manera completa dels atributs *CodiSucursal* i *CiutatSucursal*, i no és cert que { *CodiSucursal*,*CiutatSucursal* } sigui clau candidata de la relació *EmpleatsBanc*.

**b)** De nou, la veracitat de l'afirmació depèn de si la relació *R* està normalitzada o no ho està, més concretament, si la relació *R* verifica la 2FN o no ho fa. Per tant, en el cas general, del fet que una relació tingui clau primària composta, no podem deduir que tot atribut de la relació depèn completament de la clau primària. Com a exemple mostrem la relació de *Comandes* proposada a l'exercici 1:

*Comandes*(*CodiComanda*,*CodiProducte*,
*Quantitat*,*CodiClient*,*DataComanda*)

Aquesta relació presenta atributs que no depenen funcionalment de forma completa de la clau primària com, per exemple, els atributs *CodiClient* i *DataComanda*.

## Glossari

1FN
**Vegeu** *primera forma normal*

2FN
**Vegeu** *segona forma normal*

3FN
**Vegeu** *tercera forma normal*

4FN
**Vegeu** *quarta forma normal*

5FN
**Vegeu** *cinquena forma normal*

cinquena forma normal
Una relació està en cinquena forma normal si, i només si, està en quarta forma normal i no té dependències de projecció-combinació sense variació d'informació.
**sigla:** 5FN

dependència funcional
Restricció que s'aplica sobre els atributs d'una relació *R*, denotada com $\{ X \} \rightarrow \{ Y \}$, que garanteix que donat un valor de { *X* }, aquest determina de manera unívoca el valor de { *Y* }.

dependència funcional completa
Dependència quan cap subconjunt propi de { *X* } determina { *Y* }, donada una dependència funcional $\{ X \} \rightarrow \{ Y \}$ en una relació *R*.

FNBC
**Vegeu** *forma normal de Boyce-Codd*

forma normal de Boyce-Codd
Una relació està en forma normal de Boyce-Codd si, i només si, està en primera forma normal, i si tots els determinants són clau candidata de la relació.
**sigla:** FNBC

normalització
Teoria que explica quines condicions s'han de complir perquè una relació no tingui redundàncies ni anomalies de disseny.

primera forma normal
Una relació està en primera forma normal si, i només si, cap atribut de la relació és en si mateix una relació, és a dir, si tot atribut de la relació és atòmic, no descomponible, no grup repetitiu.
**sigla:** 1FN

quarta forma normal
Una relació està en quarta forma normal si, i només si, està en forma normal de Boyce-Codd i no té dependències multivaluades independents.
**sigla:** 4FN

redundàncies
Repeticions d'informació evitables.

segona forma normal
Una relació està en segona forma normal si, i només si, està en primera forma normal i tot atribut no clau depèn funcionalment en forma completa de la clau primària.
**sigla:** 2FN

tercera forma normal
Una relació està en tercera forma normal si, i només si, està en segona forma normal i cap atribut no clau depèn funcionalment de cap altre conjunt d'atributs no clau.
**sigla:** 3FN

## Bibliografia

Elmasri, R.; Navathe, S.B. (1997). *Sistemas de bases de datos. Conceptos fundamentales* (2a. ed.). Madrid: Addison-Wesley Iberoamericana.

Kent, W. (1983). "A Simple Guide to Five Normal Forms in Relational Database Theory". *Communications of the ACM* (vol. 26, febrer, núm. 2).

Maier, D. (1983). *The theory of Relational Databases.* Rockville: Pitman.

Ullman, J.D. (1988). *Principles of Database and Knowledge-Base Systems* (2 vol.). Computer Science Press.

# Annex

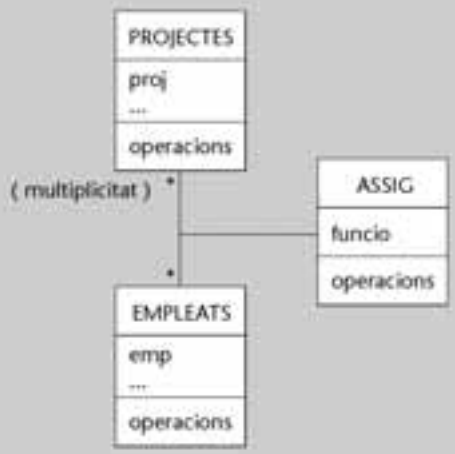
Jaume Sistac (gener 2001)

| Esquema general de transformació del model conceptual al model lògic relacional clàssic | | | | |
|---|---|---|---|---|
| **Model conceptual** | | | **Model lògic** | |
| Model UML | | | Model relacional clàssic | |
| Classes (Entitats) | | | Taules SQL (si interessen atributs) | |
| Associacions (interrelacions) | Binàries | M : N | Taules | 2 |
| | | 1 : N | Claus foranes o taules | 3 |
| | | 1 : 1 | Claus foranes o taules | 4 |
| | Reflexives (recursives) | M : N | Taules | 5 |
| | | 1 : N | Claus foranes o taules | |
| | | 1 : 1 | Claus foranes o taules | |
| | N-ÀRIES | | Taules | 6 |
| Generalització / especialització | | | Taules per a entitats supertipus<br>i<br>Taules per a cada entitat supertipus | 9 |
| Classes associatives | | | Atributs d'associacions M : N | 2 |
| | | | classes associatives | 8 |
| Agregació composta | | | agregació composta | 7 |

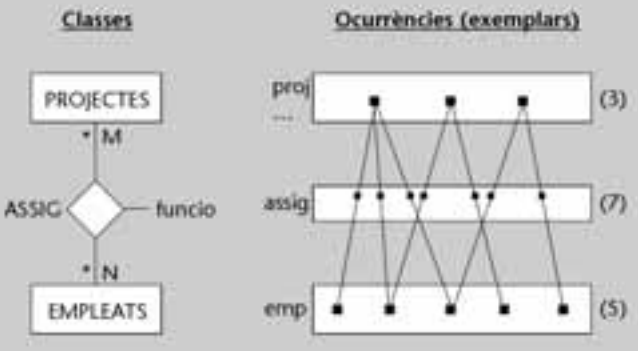
2

## ASSOCIACIÓ BINÀRIA M : N

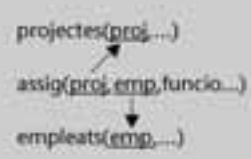
**UML**

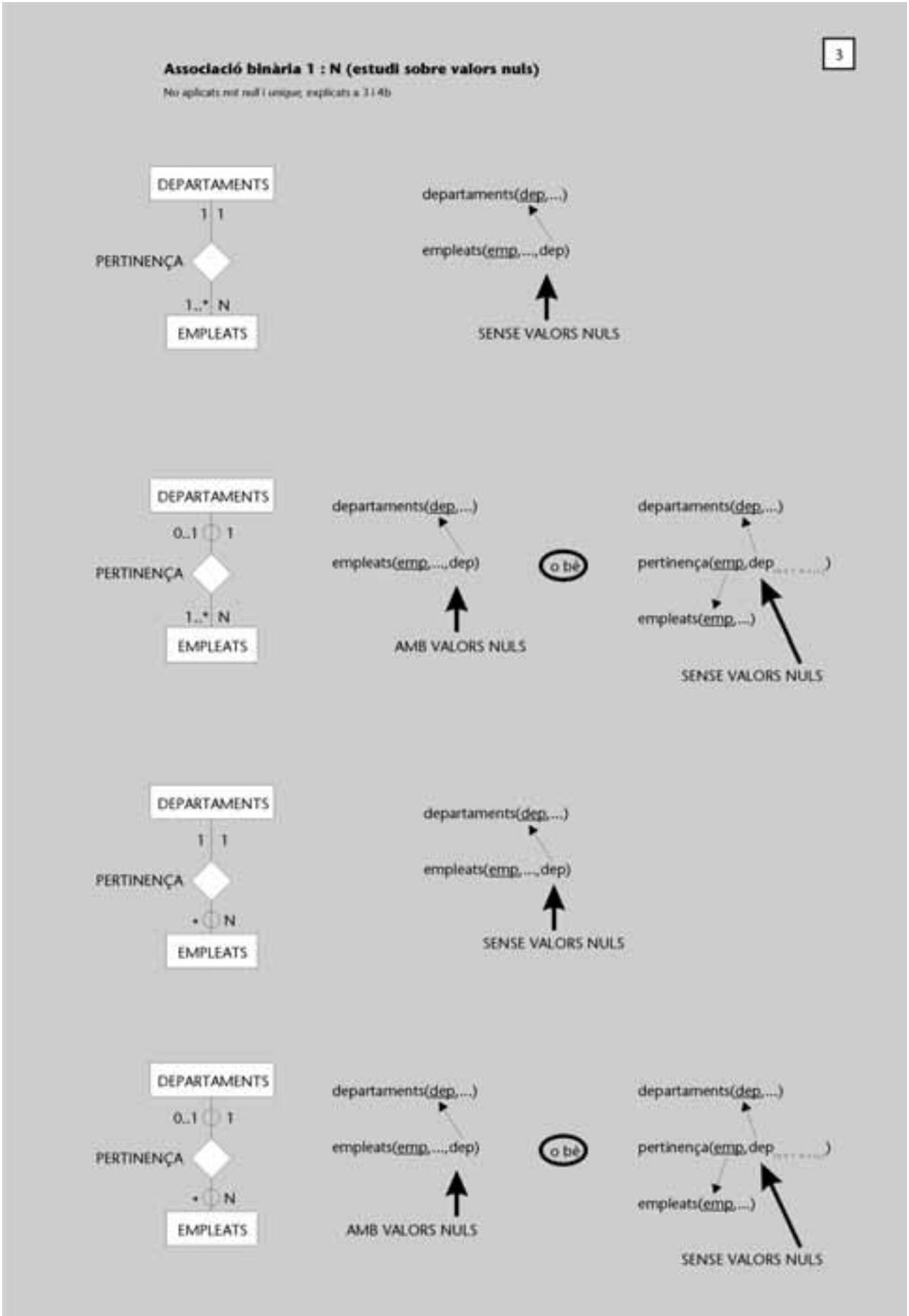



**Alternativa gràfica**




**Model lògic relacional**



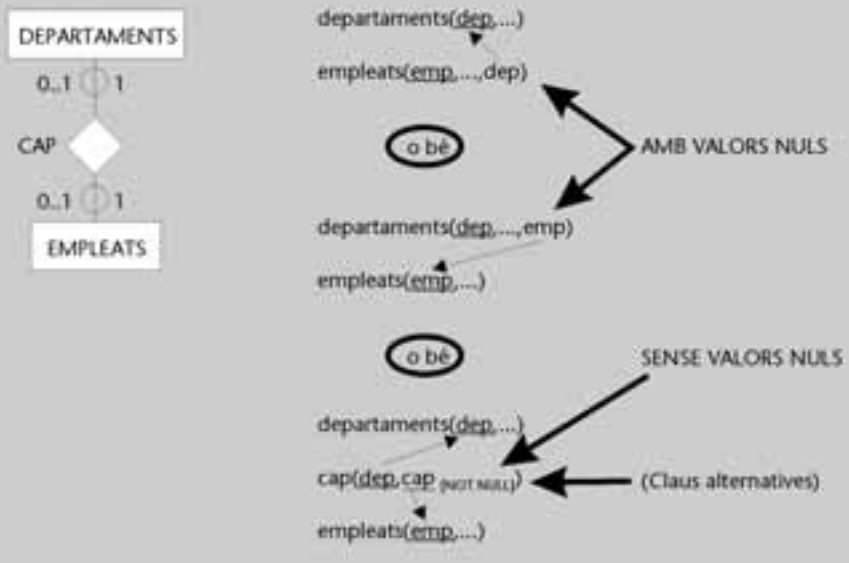

3
Associació binària 1 : N (estudi sobre valors nuls)
No aplicats mit null i unique; explicats a 3 i 4b
DEPARTAMENTS
1 1
PERTINENÇA
1..* N
EMPLEATS
departaments(dep,...)
empleats(emp,...,dep)
SENSE VALORS NULS
DEPARTAMENTS
0..1 1
PERTINENÇA
1..* N
EMPLEATS
departaments(dep,...)
empleats(emp,...,dep)
AMB VALORS NULS
o bé
departaments(dep,...)
pertinença(emp,dep,...)
empleats(emp,...)
SENSE VALORS NULS
DEPARTAMENTS
1 1
PERTINENÇA
* N
EMPLEATS
departaments(dep,...)
empleats(emp,...,dep)
SENSE VALORS NULS
DEPARTAMENTS
0..1 1
PERTINENÇA
* N
EMPLEATS
departaments(dep,...)
empleats(emp,...,dep)
AMB VALORS NULS
o bé
departaments(dep,...)
pertinença(emp,dep,...)
empleats(emp,...)
SENSE VALORS NULS

4

## ASSOCIACIÓ BINÀRIA 1 : 1 (ESTUDI SOBRE VALORS NULS)

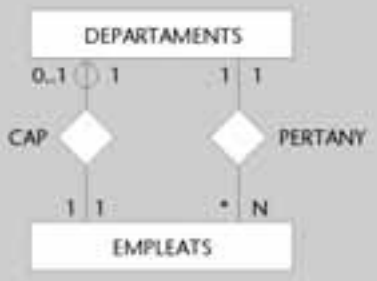
No aplicats *not null* i *unique*, explicats a 3 i 4b



3 i 4 b

## EXPRESSIÓ DE RESTRICCIONS (*CONSTRAINTS*) EN LES ASSOCIACIONS 1 : 1 i 1 : N





**A. Restriccions ja reflectides en el model**

1. Un empleat pertany a **un i sols un** departament
   Solució: l'atribut *dep* de la taula *empleats* ha d'estar definit com a *foreign key* i *not null*

2. a. Cada departament
   b. té **un i sols un** cap
   Solució: a. Programes amb SQL-Host o CLI o PSM (per als SGBD actuals, encara que l'estàndard permetria expressar la restricció amb *checks* o *assertions*)
   b. L'atribut *cap* de la taula *cap* és *foreign key* i *not null*

3. un empleat és cap d'un o cap departament (no de dos)
   Solució: L'atribut *cap* de la taula *cap* ha de ser *unique*

**B. Restriccions no reflectides en el model que es podrien imposar**

4. Un departament té **com a màxim 8** empleats
   Solució: Programes amb SQL-Host o CLI o PSM

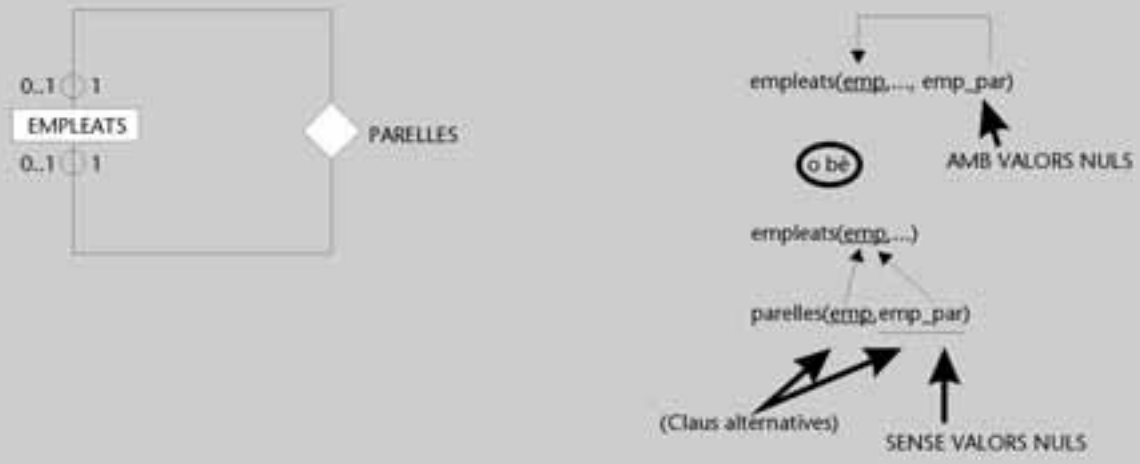
5. Un cap sempre **pertany al seu propi** departament
   Solució: Programes amb SQL-Host o CLI o PSM

5

## ASSOCIACIÓ REFLEXIVA M : N

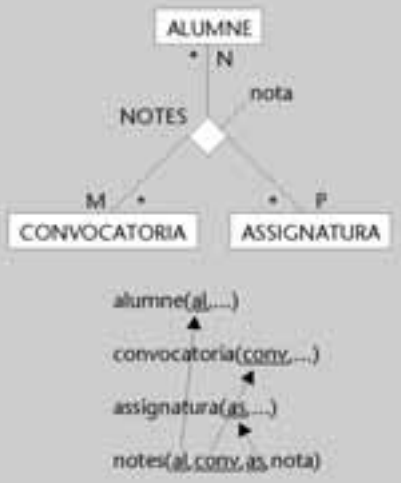

## ASSOCIACIÓ REFLEXIVA 1 : N


## ASSOCIACIÓ REFLEXIVA 1 : 1

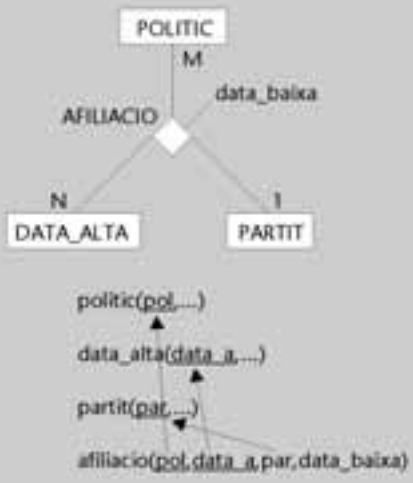


6a

**ASSOCIACIÓ TERNÀRIA M : N : P**



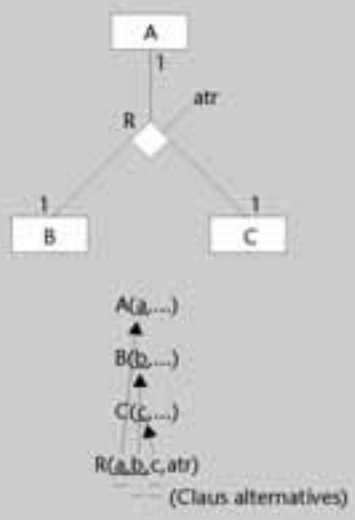

**ASSOCIACIÓ TERNÀRIA M : N : 1**

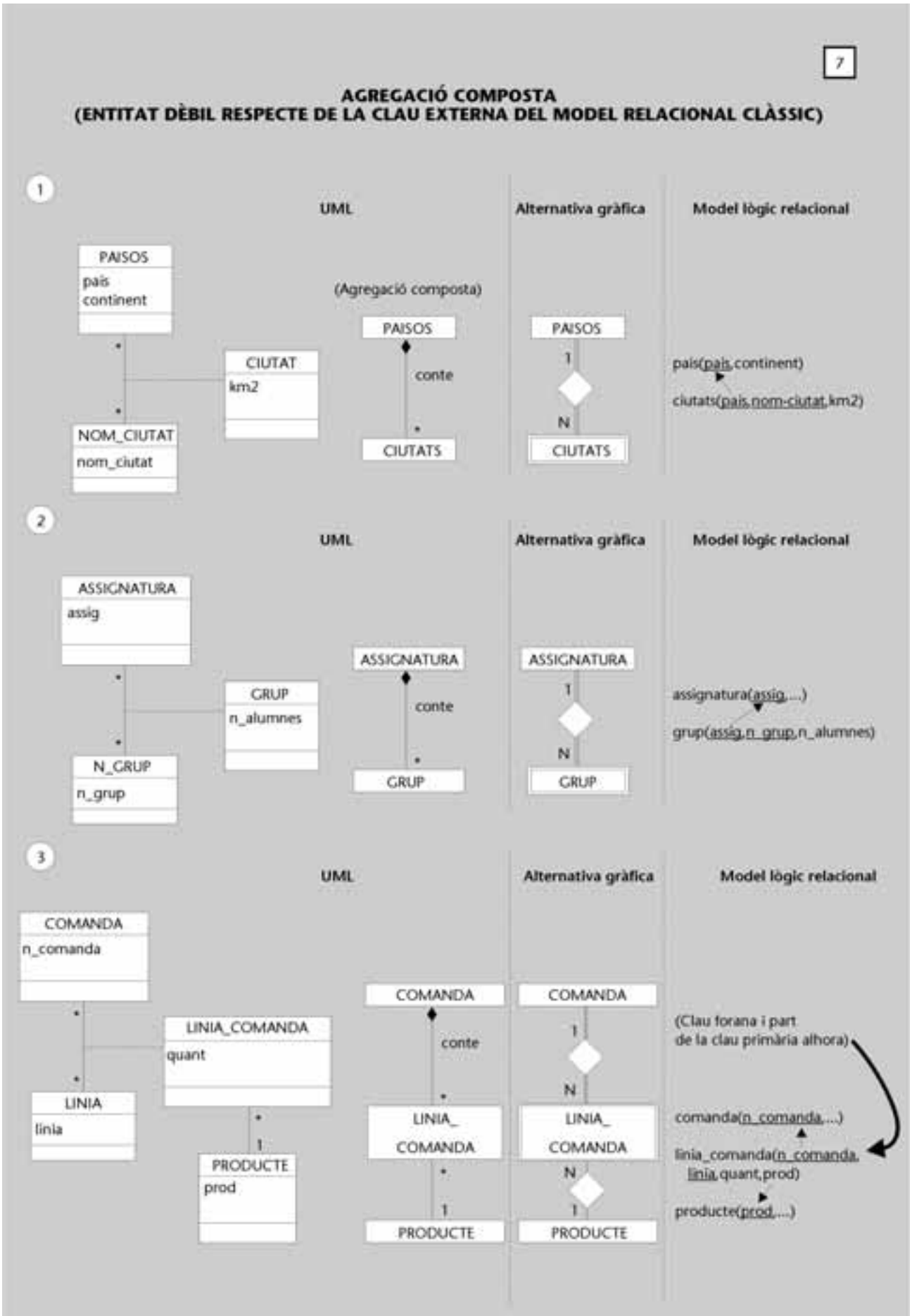


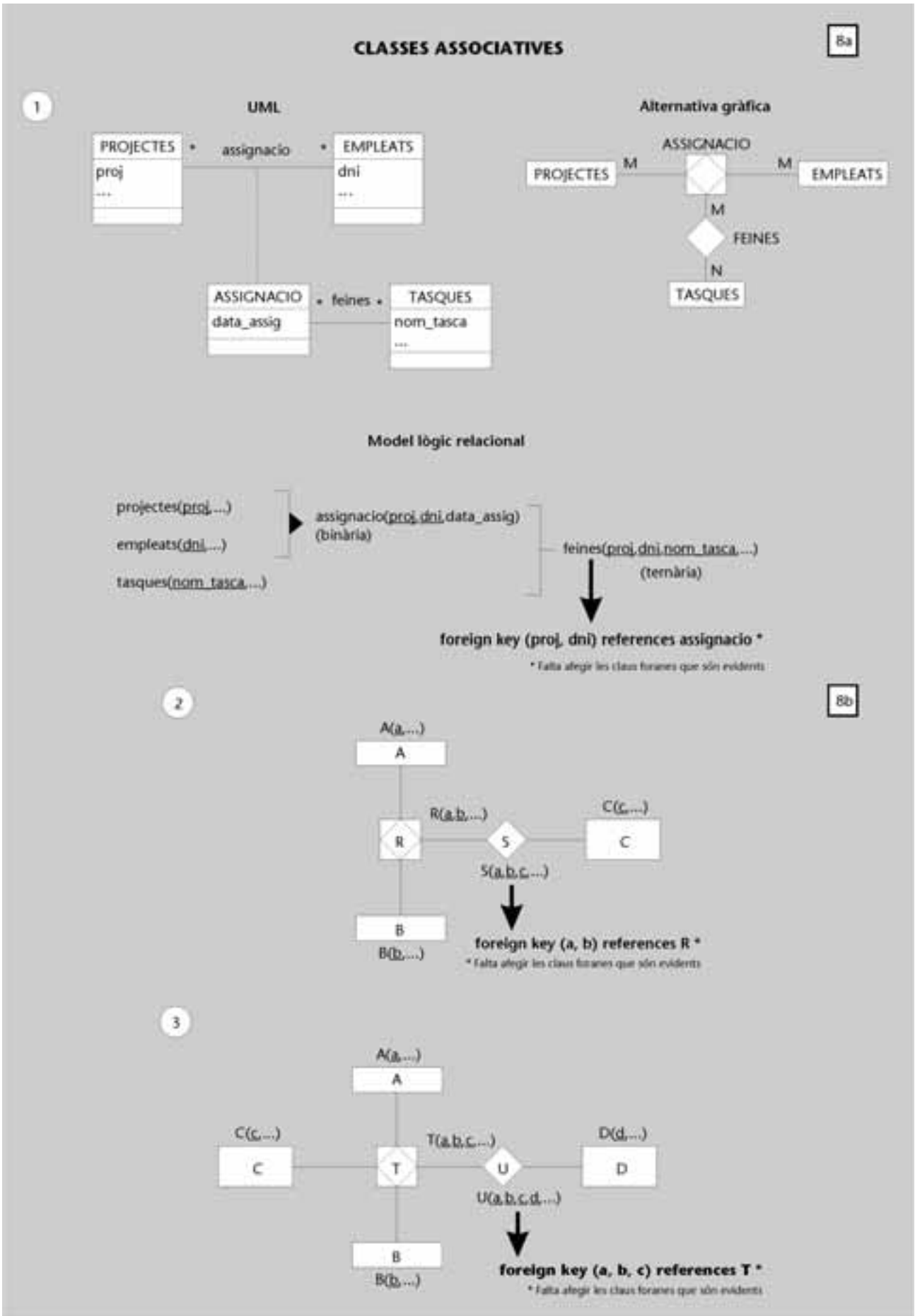

6b

**ASSOCIACIÓ TERNÀRIA M : 1 : 1 (??)**


**ASSOCIACIÓ TERNÀRIA 1 : 1 : 1 (??)**




7
AGREGACIÓ COMPOSTA
(ENTITAT DÈBIL RESPECTE DE LA CLAU EXTERNA DEL MODEL RELACIONAL CLÀSSIC)
1
UML
Alternativa gràfica
Model lògic relacional
PAISOS
país
continent
CIUTAT
km2
NOM_CIUTAT
nom_ciutat
(Agregació composta)
PAISOS
conte
CIUTATS
PAISOS
1
N
CIUTATS
pais(pais,continent)
ciutats(pais,nom-ciutat,km2)
2
UML
Alternativa gràfica
Model lògic relacional
ASSIGNATURA
assig
GRUP
n_alumnes
N_GRUP
n_grup
ASSIGNATURA
conte
GRUP
ASSIGNATURA
1
N
GRUP
assignatura(assig,...)
grup(assig,n_grup,n_alumnes)
3
UML
Alternativa gràfica
Model lògic relacional
COMANDA
n_comanda
LINIA_COMANDA
quant
LINIA
linia
PRODUCTE
prod
COMANDA
conte
LINIA_
COMANDA
PRODUCTE
COMANDA
1
N
LINIA_
COMANDA
N
1
PRODUCTE
(Clau forana i part
de la clau primària alhora)
comanda(n_comanda,...)
linia_comanda(n_comanda,
linia,quant,prod)
producte(prod,...)


CLASSES ASSOCIATIVES
8a
1
UML
PROJECTES
proj
...
assignacio
EMPLEATS
dni
...
ASSIGNACIO
data_assig
feines
TASQUES
norm_tasca
...
Alternativa gràfica
ASSIGNACIO
PROJECTES
M
M
EMPLEATS
M
FEINES
N
TASQUES
Model lògic relacional
projectes(proj,...)
empleats(dni,...)
tasques(norm_tasca,...)
assignacio(proj,dni,data_assig)
(binària)
feines(proj,dni,norm_tasca,...)
(ternària)
foreign key (proj, dni) references assignacio *
* Falta afegir les claus foranes que són evidents
8b
2
A(a,...)
A
R(a,b,...)
C(c,...)
R
S
C
S(a,b,c,...)
B
B(b,...)
foreign key (a, b) references R *
* Falta afegir les claus foranes que són evidents
3
A(a,...)
A
C(c,...)
T(a,b,c,...)
D(d,...)
C
T
U
D
U(a,b,c,d,...)
B
B(b,...)
foreign key (a, b, c) references T *
* Falta afegir les claus foranes que són evidents

9

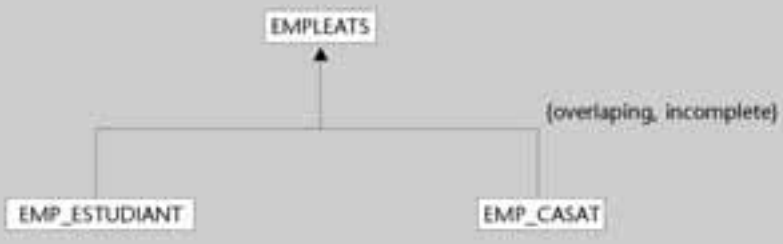
## GENERALITZACIÓ / ESPECIALITZACIÓ


empleats(emp,tipus_treball,atr_comuns)

directius(emp,atr_especifics)

administratius(emp,atr_especifics)

tecnics(emp,atr_especifics)




empleats(emp,atr_comuns)

emp_estudiant(emp,atr_especifics)

emp_casat(emp,atr_especifics)