Generalitat de Catalunya
Departament d'Educació
Institut Thos i Codina

Model relacional i normalització

Domini Públic (CC0)

Índex

1. Introducció **1**
Exemples d'aquest tutorial . . . 1

2. Model relacional **2**
2.1. Estructuració de les dades . . . 2
2.1.1. Domini . . . 3
2.1.2. Esquema i extensió . . . 4
2.1.3. Claus candidates, clau primària i claus alternatives . . . 5
2.1.4. Claus foranes . . . 6
2.1.5. Operacions amb relacions . . . 7
2.2. Regles d'integritat . . . 8
2.2.1. Unicitat de la clau primària . . . 8
2.2.2. Entitat de la clau primària . . . 9
2.2.3. Integritat referencial . . . 9
2.2.4. Integritat del domini . . . 11
2.3. Traducció del model Entitat-Relació al model relacional . . . 11
2.3.1. Entitats . . . 12
2.3.2. Interrelacions . . . 12
2.3.3. Entitats febles . . . 24
2.3.4. Generalització i especialització . . . 25
2.3.5. Entitats associatives . . . 26

3. Normalització **27**
3.1. La relació universal . . . 29
3.2. Dependències funcionals . . . 30
3.3. Primera forma normal (1FN) . . . 32
3.4. Preservació d'informació i dependències en la normalització . . . 33
3.5. Segona forma normal (2FN) . . . 33
3.6. Tercera forma normal (3FN) . . . 35
3.7. Forma normal de Boyce-Codd (FNBC) . . . 36
3.8. Quarta forma normal (4FN) . . . 37
3.9. Cinquena forma normal (5FN) . . . 39
3.10. Desnormalització . . . 42

4. Resum **43**
Elements del model relacional . . . 43
Regles d'integritat . . . 43
Traducció ER → model relacional . . . 43
Formes normals . . . 44

Índex de figures **45**

1. Introducció

Els **sistemes gestors de bases de dades (SGBD)** són grans aplicacions informàtiques pensades per facilitar la gestió de dades. Perquè aquesta gestió sigui eficient, cal que les dades tinguin un **disseny adequat**. Per això s'aconsella seguir mètodes de disseny com el disseny Entitat-Relació.

Partim del supòsit que ja tenim bases de dades ben dissenyades i que les hem d'implementar en un SGBD. Al llarg de la història de la informàtica han aparegut diversos models de sistemes gestors: **jeràrquics, en xarxa i relacionals**. La majoria dels SGBD actuals es basen en el **model relacional**, i per això ens hi centrem.

Aquest document té dues parts:

- **Model relacional:** com s'estructuren les dades i quines regles té associades aquest model. Inclou la **traducció del model ER al model relacional**, una tècnica imprescindible en el disseny lògic de BD.
- **Normalització:** un conjunt de conceptes (la *teoria de la normalització*) que permeten **detectar errors de disseny** i ofereixen mecanismes per corregir-los.

En principi, un model relacional derivat d'un model ER previ i ben elaborat hauria de ser correcte. Però no sempre és així: en l'explotació d'una BD es poden detectar problemes que obliguin a revisar el disseny relacional. A més, hi ha dissenyadors que dissenyen directament el model relacional, sense passar pel model ER, i això és molt freqüent en bases de dades complexes.

Conèixer bé el model relacional i la normalització és fonamental per a qualsevol persona que analitzi, dissenyi o administri bases de dades relacionals, o que implementi aplicacions que hi interactuïn.

Exemples d'aquest tutorial

Els exemples utilitzen dos escenaris, els mateixos que al tutorial del model Entitat-Relació:

Escenari	On es fa servir
Comercial de productes informàtics (`CLIENT`, `VENEDOR`, `PRODUCTE`, `CAMPANYA`, `BOTIGA`…)	La major part de les seccions
Connecta, una xarxa social de proximitat (`USUARI`, `PUBLICACIO`, `GRUP`, `POBLACIO`…)	Reaccions, seguiments, grups i desnormalització

NOTA

Cada exemple és **independent**: l'entitat `PRODUCTE` pot tenir atributs diferents d'una figura a una altra. Les dades de les taules són fictícies.

2. Model relacional

El **model relacional** és un model de dades basat en dues disciplines matemàtiques: la **lògica de predicats** i la **teoria de conjunts**.

Gràcies a aquest fonament teòric sòlid, els **SGBD relacionals (SGBDR)** són els que tenen una implantació més gran al mercat.

El model relacional va ser proposat per **Edgar Frank Codd** al seu treball *A Relational Model of Data for Large Shared Data Banks* (1970), tot i que no es va implementar comercialment fins al final de la dècada.

NOTA

Codd treballava per a IBM, però no va ser aquesta multinacional qui va creure primer en les possibilitats del model, sinó la competència, i en especial *Oracle*, una empresa que va néixer, justament, amb el nom de *Relational Software*.

2.1. Estructuració de les dades

El **model relacional** permet construir estructures de dades per representar les diferents informacions del món real que tinguin algun interès.

Les estructures de dades estan formades per **conjunts de relacions**.

Les **relacions** es poden concebre com a **representacions tabulars** de les dades.

Cal precisar els extrems següents:

- Tota relació ha de tenir un **nom** que la identifiqui unívocament dins de la base de dades.
- Cada **fila** és una **tupla** (també anomenat *registre*): conté les dades relacionades entre elles que ens interessa reflectir d'un objecte concret del món real.
- Cada **columna** conté, en cada cel·la, dades d'un mateix tipus, i s'anomena **atribut** o **camp**.
- Cada **cel·la** (intersecció entre fila i columna) pot emmagatzemar **un únic valor**.

NOTA

Tupla: seqüència finita d'objectes que comprèn les diferents associacions entre cada atribut de la relació i un valor concret, admissible dins del domini respectiu.

Exemple de relació. La taula 1 reflecteix l'estructura tabular de la relació `CLIENT`, que conté les dades personals dels clients d'una comercial de productes informàtics. Cada fila conté dades que pertanyen a un mateix client. La relació té un nom (`CLIENT`) i cada columna també (`DNI`, `Nom`, `Cognoms`, `Telefon`).

Taula 1. Exemple de relació

DNI	Nom	Cognoms	Telefon
38120457K	Marta	Soler Puig	937481250

45219876B	Pau	Ferrer Vidal	651203948
51873204T	Núria	Camps Roca	722915630

Tota base de dades relacional està formada per un **conjunt de relacions**.

Aquesta manera de visualitzar les bases de dades relacionals és molt entenedora, però cal aprofundir en algunes característiques addicionals per poder distingir clarament les relacions dels fitxers tradicionals.

2.1.1. Domini

Un **domini** és un **conjunt finit de valors indivisibles**.

Els atributs només poden prendre valors que estiguin inclosos dins del seu domini. Altrament, no són valors vàlids, i un SGBD relacional no n'ha de permetre l'emmagatzematge.

Exemples de dominis

- Atribut `Telefon` de `CLIENT`: si el definim de manera que només pugui emmagatzemar **nou caràcters numèrics**, el domini inclou totes les combinacions possibles (10^9, una magnitud gran però finita). Molts d'aquests valors no es correspondran mai amb telèfons reals (difícilment un operador assignarà el número `000000000`); per evitar-ho caldria restringir més el domini.
- Atribut `Cognoms`: conté els dos cognoms separats per un espai. Conceptualment, els usuaris distingeixen dos objectes (primer i segon cognom), però **un SGBD relacional considera el valor de manera atòmica**, sense cap estructuració interna.

Hi ha dos tipus de dominis:

Tipus	Definició	Exemples
Dominis predefinits	Tipus de dades que admet cada SGBD	Cadenes de caràcters, nombres enters, nombres decimals, dates…
Dominis definits pels usuaris	Restriccions addicionals sobre un domini predefinit, establertes pels dissenyadors i administradors	Vegeu l'exemple següent

Exemple de domini definit per l'usuari. En una relació `CAMPANYA` hi ha el camp `Descompte`, que guarda el percentatge de rebaixa. Podem restringir el seu domini perquè **no admeti valors inferiors a 0 ni superiors a 30**, ja que la política de l'empresa no permet descomptes més grans.

2.1.2. Esquema i extensió

Tota relació consta d'un **esquema** (també anomenat *intenció de la relació*) i de la seva **extensió**.

L'**esquema d'una relació** consisteix en un **nom** que la identifica unívocament dins de la BD i el **conjunt d'atributs** que conté.

És molt recomanable seguir una notació uniforme en tots els esquemes d'una mateixa base de dades:

- El nom de les relacions s'escriu en **MAJÚSCULES** i preferiblement en **singular**.
- El nom dels atributs comença amb **majúscula** i continua amb minúscules (`Nom`). Si són sigles, tot en majúscules (`DNI`). Als noms compostos, cada paraula comença amb majúscula (`DataNaixement`, `TelefonParticular`).

Exemple d'esquema:

```
CLIENT(DNI, Nom, Cognoms, Telefon)
```

L'**ordre** en què es mostren els atributs és **indiferent**, per definició del model relacional.

Els atributs d'una relació són **únics dins d'aquesta**: el seu nom no es pot repetir dins d'una mateixa relació. Ara bé, **diferents relacions sí que poden contenir atributs amb el mateix nom** (`Nom` a `CLIENT` i a USUARI).

Els dominis de diferents atributs d'una mateixa relació poden ser idèntics, malgrat que emmagatzemin propietats diferents (`TelefonFix`, `TelefonMobil` i `TelefonFeina` poden tenir el mateix domini).

L'**extensió d'una relació** consisteix en els **valors de les dades** emmagatzemades en totes les tuples que conté.

Exemple d'extensió. Per a l'esquema `CLIENT(DNI, Nom, Cognoms, Telefon)` de la taula 1, l'extensió és una llista amb tots els clients de la BD:

```
Client 1: 38120457K, Marta, Soler Puig, 937481250
Client 2: 45219876B, Pau, Ferrer Vidal, 651203948
Client 3: 51873204T, Núria, Camps Roca, 722915630
```

De vegades els atributs poden **no contenir cap valor**, és a dir, poden contenir **valors nuls**.

Exemple de valor nul. S'hi registra un quart client que no té telèfon. Les seves dades reflecteixen aquesta circumstància amb l'absència de valor a l'atribut `Telefon`. En les representacions tabulars (en el pla teòric) s'indica amb la paraula `NUL` o deixant la cel·la en blanc.

Taula 2. Exemple de relació amb valors nuls

DNI	Nom	Cognoms	Telefon
38120457K	Marta	Soler Puig	937481250
45219876B	Pau	Ferrer Vidal	651203948
51873204T	Núria	Camps Roca	722915630
29384710M	Jordi	Pons Alsina	NUL

El **grau d'una relació** depèn del nombre d'atributs que inclou el seu esquema.

Exemple: la relació `CLIENT(DNI, Nom, Cognoms, Telefon)` és de **grau 4**.

La **cardinalitat d'una relació** és expressada pel nombre de tuples que en formen l'extensió.

Exemple: a la taula 2, la cardinalitat de `CLIENT` és **4**.

2.1.3. Claus candidates, clau primària i claus alternatives

Perquè l'emmagatzematge sigui útil, ha de permetre **identificar les dades**. En les bases de dades relacionals, les tuples s'identifiquen mitjançant les anomenades *superclaus*.

Una **superclau** és un subconjunt dels atributs de l'esquema d'una relació tal que no és possible que hi hagi més d'una tupla a l'extensió amb la mateixa combinació de valors en els atributs que en formen part.

Una superclau pot contenir atributs innecessaris. El que habitualment interessa és treballar amb **superclaus mínimes**: cap subconjunt propi és capaç, per si sol, d'identificar les tuples.

Per definició, **cap superclau mínima no pot admetre valors nuls** en cap dels seus atributs. En una mateixa relació pot haver-hi més d'una superclau mínima.

S'anomenen **claus candidates** totes les superclaus mínimes d'una relació.

A l'hora d'implementar una BD, entre totes les claus candidates només se'n tria **una**.

La **clau primària** és la clau candidata que el dissenyador lògic de la BD tria per distingir unívocament cada tupla d'una relació de la resta.

Les claus candidates no triades com a clau primària són les **claus alternatives**.

Per diferenciar els atributs de la clau primària, se **subratllen**. En aquest tutorial, quan escrivim esquemes en text pla, els marquem amb dobles guions baixos: `CLIENT(__DNI__, Nom)`. Normalment, es col·loquen junts abans que la resta d'atributs, però és només una qüestió d'elegància: el model relacional **no es basa ni en l'ordre dels atributs ni en l'ordre de les tuples**.

NOTA

Amb molta freqüència és l'administrador de la BD qui tria la clau primària entre les claus candidates, tot acomplint tasques de disseny lògic.

Exemple de claus candidates, primària i alternatives. Imaginem que `CLIENT` té uns quants atributs més:

```
CLIENT(DNI, NumTargeta, Correu, Nom, Cognoms, Telefon)
```

`DNI`, `NumTargeta` (número de la targeta de fidelitat) i `Correu`, en ser personals i irrepetibles, ens podrien servir per identificar els clients: són **claus candidates**. El dissenyador tria, per exemple, `DNI` com a **clau primària**; `NumTargeta` i `Correu` passen a ser **claus alternatives**:

```
CLIENT(__DNI__, NumTargeta, Correu, Nom, Cognoms, Telefon)
```

Si no hi ha cap atribut capaç d'identificar les tuples per si sol, cal buscar un **subconjunt d'atributs** la combinació de valors dels quals no es pugui repetir. Si aquesta possibilitat no existeix, cal afegir a la relació un **atribut addicional** que faci d'identificador.

Per definició, el model relacional **no admet tuples repetides**: no permet dues tuples de la mateixa relació amb els mateixos valors en tots els atributs.

Les implementacions concretes dels SGBD sí que permeten tuples repetides, sempre que no s'estableixi cap clau primària a la relació. Aquesta permissivitat de vegades soluciona certes eventualitats, però **no hauria de ser la manera habitual** de treballar amb BD relacionals.

2.1.4. Claus foranes

Qualsevol BD relacional, per petita que sigui, conté normalment **més d'una relació**. Per reflectir els vincles entre objectes del món real, cal que les tuples de les diferents relacions es puguin **interrelacionar**. De vegades fins i tot cal relacionar les tuples d'una relació amb altres tuples **de la mateixa relació**.

Una **clau forana** és un atribut, o un conjunt d'atributs, de l'esquema d'una relació que serveix per **relacionar les seves tuples amb les tuples d'una altra relació** de la BD (o amb els d'ella mateixa, en alguns casos).

La clau forana ha de **referenciar la clau primària** de la relació amb la qual es vol relacionar. Les combinacions de valors d'una clau forana han d'existir a la clau primària referenciada, o bé han de ser **valors nuls**; altrament les referències serien errònies.

Característiques de les claus foranes:

- Tota clau forana ha de tenir el **mateix nombre d'atributs** que la clau primària a la qual fa referència.
- Entre els atributs de la clau forana i els de la clau primària s'ha de poder establir una **correspondència** (concretament, una bijecció).
- Els **dominis** dels atributs de la clau forana han de coincidir amb els de la clau primària respectiva (o, com a mínim, ser compatibles).

Una relació pot tenir **més d'una clau forana**, o cap. La clau primària d'una relació pot ser referenciada per una o més claus foranes, o per cap. I es pot donar el cas que un mateix atribut formi part tant de la clau primària com d'alguna de les seves claus foranes.

Exemple de claus foranes. La relació `CLIENT` (taula 3) incorpora dues claus foranes:

- `CodiBotiga` fa referència a la clau primària de la relació `BOTIGA` (formada per l'atribut `Codi`, taula 4), per indicar quina és la **botiga habitual** de cada client.
- `DNIPadri` fa referència a la clau primària **de la mateixa relació** (`DNI`), i serveix per indicar quin client el va convidar al programa de fidelitat («porta un amic»).

Jordi Pons Alsina encara no té botiga assignada ni cap padrí; per això la seva tupla conté **valors nuls** en els dos atributs de clau forana.

Taula 3. Relació amb claus foranes (`CLIENT`)

DNI	Nom	Cognoms	Telefon	DNIPadri	CodiBotiga
38120457K	Marta	Soler Puig	937481250	NUL	102
45219876B	Pau	Ferrer Vidal	651203948	38120457K	201
51873204T	Núria	Camps Roca	722915630	38120457K	201
29384710M	Jordi	Pons Alsina	NUL	NUL	NUL

Taula 4. Relació amb clau primària referenciada (`BOTIGA`)

Codi	Superficie
101	120
102	85
201	60

Notació de les claus foranes. S'afegeix a continuació de l'esquema, amb la paraula `ON`, el conjunt d'atributs de la clau forana entre claus `{ }`, la forma verbal `REFERENCIA` i la relació referenciada. Si hi ha més d'una clau forana, se separen per comes i l'última va precedida de la conjunció `i`.

```
CLIENT(__DNI__, Nom, Cognoms, Telefon, DNIPadri, CodiBotiga)
    ON {DNIPadri} REFERENCIA CLIENT i {CodiBotiga} REFERENCIA BOTIGA

BOTIGA(__Codi__, Superficie)
```

2.1.5. Operacions amb relacions

El model relacional permet realitzar una sèrie d'operacions amb les dades:

Operació	Descripció
Actualització	Canvia les tuples de les relacions. Pot ser de tres tipus: **inserció**, **esborrat** i **modificació**
Consulta	Obté dades de manera parametritzada, **sense alterar** les dades emmagatzemades

- **Inserció:** afegir una o més tuples noves a una relació.
- **Esborrat:** eliminar una o més tuples d'una relació.
- **Modificació:** canviar el valor d'un o més atributs d'una o més tuples.

Per fer-les cal conèixer l'estructura formada per les relacions: els esquemes i les interrelacions entre elles, mitjançant les claus foranes.

Exemples d'operacions sobre la taula 3:

Inserció d'un nou client:

```
<60318472R, Aina, Bosch Mora, 648302917, 38120457K, 102>
```

Esborrat d'un client donat d'alta:

```
<51873204T, Núria, Camps Roca, 722915630, 38120457K, 201>
```

Modificació: canviar el telèfon de Marta Soler Puig (937481250) per un altre (937995021), o assignar-ne un a Jordi Pons Alsina, que no en tenia (872614730 en lloc del valor nul).

Consulta: llista, ordenada alfabèticament pels cognoms, dels clients que tenen com a botiga habitual la 201:

```
<51873204T, Núria, Camps Roca, 722915630, 38120457K, 201>
<45219876B, Pau, Ferrer Vidal, 651203948, 38120457K, 201>
```

2.2. Regles d'integritat

Els valors que emmagatzemen les BD han de reflectir en tot moment, de manera correcta, la porció de la realitat que volem modelitzar.

Anomenem **integritat** la propietat de les dades que consisteix a representar correctament les situacions del món real que modelitzen.

Per tal que les dades siguin íntegres, cal garantir que siguin **correctes** i que estiguin **senceres**. Les condicions que han de complir es poden agrupar en dues tipologies:

Tipus	Descripció
Restriccions d'integritat de l'usuari	Condicions específiques de cada BD. Els SGBD han de permetre als administradors establir-les i han de garantir que es respecten
Regles d'integritat del model	Condicions de caràcter general que han de complir totes les BD relacionals. No cal definir-les en cada BD: són preestablertes

Exemple de restricció d'integritat de l'usuari. En donar d'alta un nou client, podríem exigir al sistema que validi, mitjançant l'algorisme corresponent, si la lletra introduïda del DNI es correspon amb les xifres, i que denegui la inserció en cas contrari.

Exemple de regla d'integritat del model. Com que `CLIENT` té `DNI` com a clau primària, el sistema validarà automàticament que no s'introdueixi un segon client amb el mateix DNI.

2.2.1. Unicitat de la clau primària

En cap moment no hi pot haver **dues o més tuples** amb la mateixa combinació de valors en els atributs de la clau primària d'una relació.

Els SGBD han de garantir aquesta regla en totes les insercions de noves tuples i en les modificacions que afectin algun atribut de la clau primària.

Exemple (relació `BOTIGA`, taula 4):

- No es pot inserir `<102, 70>`: la clau primària ja conté el valor 102.
- Per donar d'alta una altra botiga cal un valor de `Codi` no present, per exemple `<103, 70>`.
- Tampoc es pot modificar la clau de la tupla `<101, 120>` assignant-hi el valor 102.

2.2.2. Entitat de la clau primària

Cap atribut que formi part d'una clau primària pot contenir mai **valors nuls**.

Si s'admetés, les tuples amb nuls a la clau no es podrien distingir d'altres. Els SGBD han de garantir aquesta regla en les insercions i en les modificacions.

Exemple: a `BOTIGA` no es pot inserir `<NUL, 45>`. Si volem donar d'alta una botiga de 45 m^2, cal un codi no nul, per exemple `<202, 45>`. Tampoc es pot modificar la clau de la tupla `<101, 120>` assignant-hi el valor nul.

2.2.3. Integritat referencial

La **integritat referencial** implica que, per a qualsevol tupla, la combinació de valors de la clau forana ha de ser present a la clau primària referenciada, o bé ha d'estar constituïda **exclusivament per valors nuls** (si els atributs implicats admeten aquesta possibilitat).

Els SGBD han de fer les comprovacions pertinents de manera automàtica en dos tipus d'operacions amb relacions que tinguin claus foranes:

- **Insercions** de noves tuples.
- **Modificacions** que afectin atributs que formen part de qualsevol clau forana.

I també han de validar la correcció de dos tipus d'operacions amb relacions que tinguin la clau primària referenciada des d'alguna clau forana:

- **Esborraments** de tuples.
- **Modificacions** que afectin atributs que formen part de la clau primària.

Per garantir la integritat referencial en aquests dos últims casos es pot seguir una de tres polítiques: **restricció**, **actualització en cascada** i **anul·lació**.

Exemple de violació de la integritat referencial. La tupla de Jordi Pons Alsina té un valor nul a l'atribut `CodiBotiga`. Si el volguéssim actualitzar amb el valor 316, el sistema no ho hauria de deixar fer, perquè aquest valor no és present a la clau primària de cap tupla de `BOTIGA`.

Per veure les tres polítiques partim de les mateixes relacions `CLIENT` (taula 3) i `BOTIGA` (taula 4).

Política de restricció

Consisteix a **prohibir** l'operació d'actualització de què es tracti.

- En cas d'**esborrament**, no es permet eliminar una tupla si la seva clau primària està referenciada des d'alguna clau forana.
- En cas de **modificació**, no es permet alterar el valor de cap dels atributs de la clau primària d'una tupla si està referenciada des d'alguna clau forana.

Exemple: ni esborrar ni modificar la botiga 102 serà possible, perquè hi ha un client (Marta Soler Puig) que hi té la seva botiga habitual. En canvi, sí que es podria esborrar la botiga 101, perquè no està referenciada des de `CLIENT`.

Actualització en cascada

Consisteix a **permetre** l'operació, però fent al mateix temps una sèrie d'operacions compensatòries que **propaguin** els canvis a les tuples que referencien la tupla actualitzada.

- En cas d'**esborrament**, s'eliminen totes les tuples que facin referència a la tupla esborrada.
- En cas de **modificació**, els valors de la clau forana de les tuples que hi fan referència s'alteren perquè continuïn coincidint amb els nous valors de la clau primària.

Exemples:

- Si esborrem la tupla `<201, 60>` de `BOTIGA`, també s'esborraran les **dues tuples** de `CLIENT` que hi fan referència (Pau i Núria).
- Si modifiquem la tupla `<201, 60>` canviant el codi per 203, les dues tuples de `CLIENT` que hi fan referència actualitzaran el valor de `CodiBotiga` de 201 a 203, per mantenir la connexió correcta.

Política d'anul·lació

Consisteix a **permetre** l'operació, però posant **valors nuls** a totes les claus foranes de les tuples que facin referència la tupla actualitzada.

- En cas d'**esborrament**, els atributs de la clau forana de les tuples que hi feien referència passen a valer nul i no indiquen cap connexió.
- En cas de **modificació**, passa exactament el mateix.

Només es pot aplicar si els atributs de les claus foranes implicades **admeten valors nuls**.

Exemple: si esborrem la tupla `<102, 85>` de `BOTIGA` (o canviem el seu codi per 105), la tupla de `CLIENT` que hi feia referència perd la connexió:

```
<38120457K, Marta, Soler Puig, 937481250, NUL, NUL>
```

Selecció de la política

Serà el dissenyador de cada BD qui escollirà la política més adequada. Com a orientació:

Operació	Opció més habitual
Esborrament	**Restricció**
Modificació	**Actualització en cascada**

La política d'anul·lació és molt menys freqüent: es posa en pràctica quan es volen conservar certes dades, encara que hagin perdut la connexió que tenien abans, de vegades amb l'esperança de poder-la recuperar més endavant.

2.2.4. Integritat del domini

La regla d'**integritat del domini** implica que tots els valors **no nuls** que contenen els atributs han de pertànyer als **dominis** declarats per als atributs en qüestió.

Es compleix tant per als dominis predefinits com per als definits per l'usuari. A més, els **operadors** que es poden aplicar sobre els valors depenen dels dominis dels atributs que els emmagatzemen.

Exemple. Si a `BOTIGA(Codi, Superficie)` definim el domini de `Codi` com el dels nombres enters de 0 fins a 999, no podem inserir un valor que no pertanyi al domini, com ara `WEB` o `MAG`. Tampoc no podem consultar una botiga amb `Codi = 'WEB'`, perquè `'WEB'` és una cadena de caràcters.

2.3. Traducció del model Entitat-Relació al model relacional

Un cop coneguts els elements del model relacional, cal partir del model conceptual (el model ER) i fer el **disseny lògic** de la BD. En tots els exemples suposem que prèviament s'ha fet la fase de disseny conceptual i que n'ha resultat un diagrama de Chen.

NOTA

Fases del disseny de BD: 1) disseny conceptual, 2) disseny lògic, 3) disseny físic.

El model ER es basa en entitats i interrelacions. Aspectes generals de la traducció:

- Les **entitats** sempre donen lloc a **relacions** (a excepció de les entitats auxiliars de tipus `DATA`).
- Les interrelacions **binàries 1-1 o 1-N** originen **claus foranes** en relacions ja existents.
- Les interrelacions **binàries M-N** i totes les **n-àries** d'ordre superior a 2 sempre es transformen en **noves relacions**.

Convé seguir un cert ordre. Una bona pràctica:

1. Transformar les **entitats** del diagrama en relacions.
2. Transformar les entitats amb alguna **especificitat**: les febles, les associatives i les derivades de generalitzacions o especialitzacions.
3. Afegir a les relacions els atributs necessaris per formar les **claus foranes** de les interrelacions binàries 1-1 i 1-N.
4. Finalment, transformar les interrelacions binàries **M-N** i les **n-àries**.

D'aquesta manera evitem que hi hagi claus foranes que referenciïn relacions que encara no s'han descrit. Això fa més llegible el model i estalvia feina a l'hora d'escriure les instruccions SQL: cap SGBD no pot resoldre una referència a una taula que encara no ha estat creada.

2.3.1. Entitats

Cada entitat del model ER es transforma en una relació:

- Els atributs de l'entitat seran els atributs de la relació.
- La clau primària de l'entitat serà la clau primària de la relació.
- Si l'entitat intervé en alguna interrelació binària 1-1 o 1-N, pot ser necessari afegir-hi nous atributs que actuïn com a claus foranes.

Exemple: el diagrama de la figura 1 es tradueix així:

```
CLIENT(__DNI__, Nom, Cognoms)
```

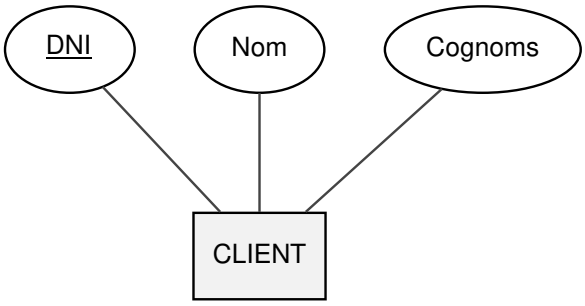

Figura 1: Figura 1. Entitat CLIENT amb els atributs DNI, Nom i Cognoms

2.3.2. Interrelacions

Un cop transformades les entitats, cal traduir les interrelacions. Hi influeixen la **connectivitat** i les **dependències d'existència**.

1. Binàries

a. Connectivitat 1-1 i dependències d'existència

Cal afegir a **qualsevol** de les dues relacions una **clau forana** que faci referència a l'altra. Si una de les entitats és **obligatòria** en la interrelació (totes les seves instàncies hi participen) i l'altra és **opcional**, la clau forana va a la relació derivada de l'entitat **obligatòria**: així s'evita emmagatzemar valors nuls i s'estalvia espai. Els atributs de la interrelació (si n'hi ha) acompanyen la clau forana.

Exemple (figura 2). `VENEDOR` **Coordina** `DEPARTAMENT`: cada departament té un venedor coordinador, però la majoria de venedors no coordinen cap departament. `DEPARTAMENT` és, per tant, l'entitat obligatòria. Hi ha dues possibilitats:

```
VENEDOR(__DNI__, Nom, Cognoms)

DEPARTAMENT(__Codi__, Descripcio, DNIVenedor)
    ON {DNIVenedor} REFERENCIA VENEDOR
```

O bé:

```
DEPARTAMENT(__Codi__, Descripcio)

VENEDOR(__DNI__, Nom, Cognoms, CodiDepartament)
    ON {CodiDepartament} REFERENCIA DEPARTAMENT i CodiDepartament
 ↪  ADMET VALORS NULS
```

L'opció més correcta és la **primera**: si posem la clau forana a `VENEDOR`, valdria nul en molts casos i ocuparia un espai d'emmagatzematge innecessari. A més, com que la connectivitat és 1-1, el valor de `DNIVenedor` no es pot repetir entre departaments.

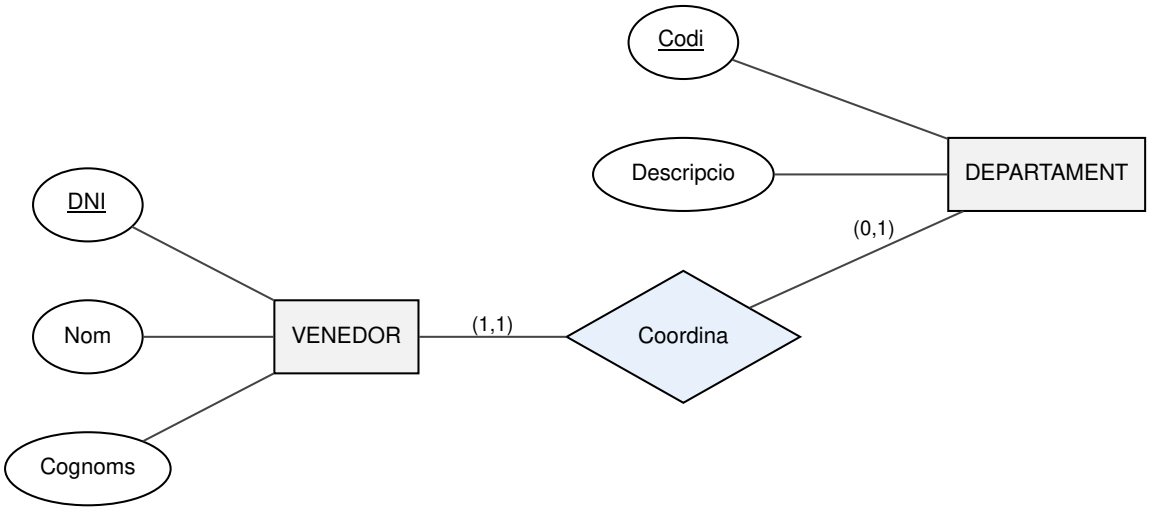

Figura 2: Figura 2. Interrelació binària amb connectivitat 1-1

b. Connectivitat 1-N

Cal afegir una clau forana a la relació que resulta de l'entitat ubicada al **costat N** de la interrelació, que faci referència a l'altra relació. Si es col·loqués a l'altra relació, l'atribut hauria de ser **multivalent** per poder representar totes les connexions, i això no està permès al model relacional. Els atributs de la interrelació acompanyen la clau forana.

Exemple (figura 3). `DEPARTAMENT` **Treballa** `VENEDOR` (1-N). La clau forana s'afegeix a la relació derivada de l'entitat del costat N (`VENEDOR`):

```
DEPARTAMENT(__Codi__, Descripcio)

VENEDOR(__DNI__, Nom, Cognoms, CodiDepartament)
    ON {CodiDepartament} REFERENCIA DEPARTAMENT i CodiDepartament
 ↪  ADMET VALORS NULS
```

L'entitat del costat 1 (`DEPARTAMENT`) és opcional a la interrelació `Treballa`: això vol dir que `VENEDOR` admetrà nuls a la clau forana, perquè hi pot haver venedors encara no assignats a cap departament. A diferència de les 1-1, aquí no es poden evitar aquests nuls, perquè la clau forana ha d'anar necessàriament a la relació del costat N.

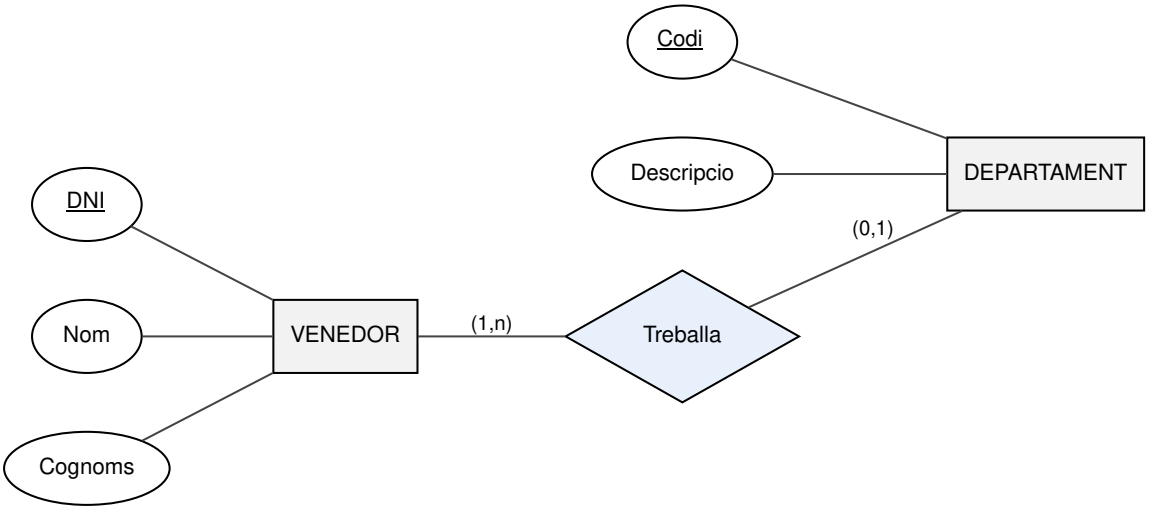

Figura 3: Figura 3. Interrelació binària amb connectivitat 1-N

c. Connectivitat M-N

Cada interrelació M-N es transforma en una **nova relació**:

- La seva **clau primària** estarà formada pels atributs de les claus primàries de les dues entitats interrelacionades.
- Els atributs de la interrelació (si n'hi ha) es converteixen en atributs de la nova relació.

Exemple (figura 4), a la xarxa social Connecta: un usuari pot reaccionar a moltes publicacions i una publicació pot rebre reaccions de molts usuaris.

```
USUARI(__Alias__, Nom)

PUBLICACIO(__Codi__, Text)

REACCIO(__AliasUsuari__, __CodiPublicacio__, Data)
    ON {AliasUsuari} REFERENCIA USUARI i {CodiPublicacio} REFERENCIA
 ↪  PUBLICACIO
```

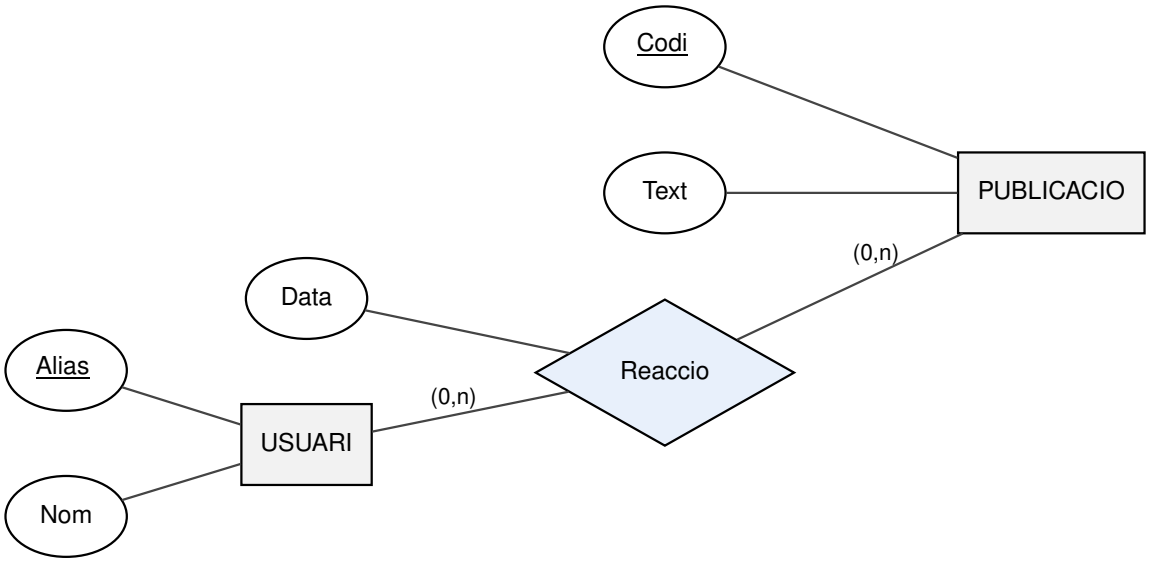

Figura 4: Figura 4. Interrelació binària amb connectivitat M-N

2. Ternàries

Tota interrelació ternària es transforma en una **nova relació**, que tindrà com a atributs les claus primàries de les tres entitats interrelacionades, més els atributs propis de la interrelació. La **composició de la clau primària** depèn de la connectivitat.

a. Connectivitat M-N-P

La clau primària està formada per **tots** els atributs de les claus primàries de les tres entitats. Si no fos així, la clau hauria de repetir algunes combinacions de valors per modelitzar totes les possibilitats, i això no està permès.

Exemple (figura 5). `Compra` entre `CLIENT`, `PRODUCTE` i `CAMPANYA`:

```
CLIENT(__DNI__, Nom, Cognoms)

PRODUCTE(__Codi__, Descripcio)

CAMPANYA(__Codi__)

COMPRA(__DNIClient__, __CodiProducte__, __CodiCampanya__, Quantitat)
    ON {DNIClient} REFERENCIA CLIENT, {CodiProducte} REFERENCIA
 ↪  PRODUCTE
    i {CodiCampanya} REFERENCIA CAMPANYA
```

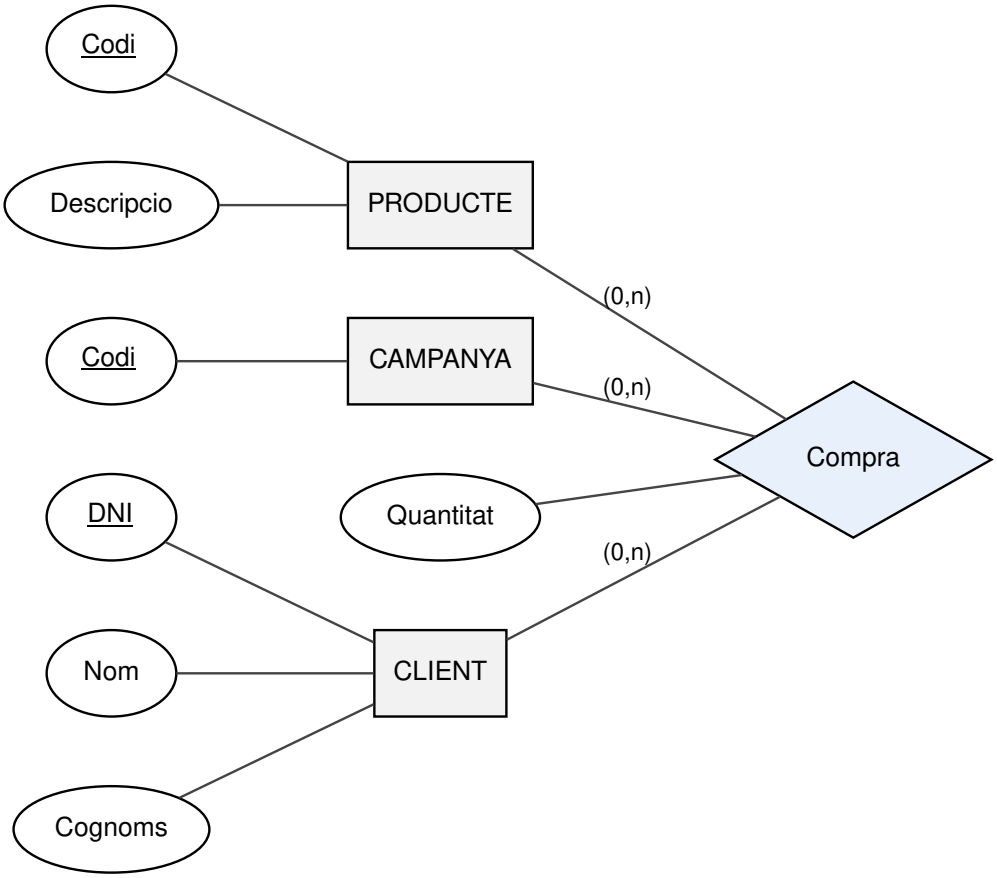

Figura 5: Figura 5. Interrelació ternària amb connectivitat M-N-P

b. Connectivitat 1-M-N

La clau primària està composta per tots els atributs de les claus primàries de les **dues entitats que són als dos costats N** de la interrelació.

Exemple (figura 6). `Bescanvi`: en cada campanya, un client només pot bescanviar un val de descompte d'**un** producte. Per tant, no cal incorporar la clau de PRODUCTE a la clau de la nova relació:

```
CLIENT(__DNI__, Nom, Cognoms)

PRODUCTE(__Codi__, Descripcio)

CAMPANYA(__Codi__)

BESCANVI(__DNIClient__, CodiProducte, __CodiCampanya__, DataBescanvi)
    ON {DNIClient} REFERENCIA CLIENT, {CodiProducte} REFERENCIA
↪   PRODUCTE
    i {CodiCampanya} REFERENCIA CAMPANYA
```

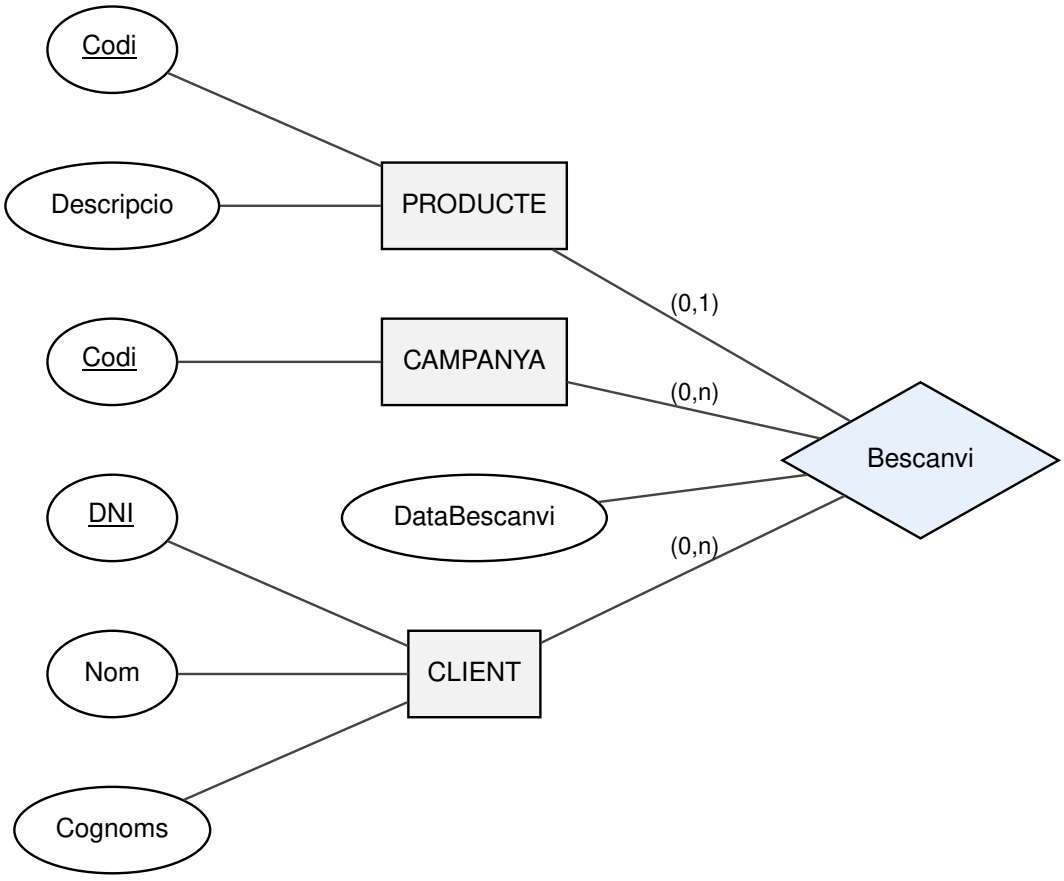

Figura 6: Figura 6. Interrelació ternària amb connectivitat 1-M-N

c. Connectivitat 1-1-N

La clau primària està composta pels atributs de la clau primària de l'entitat del **costat N**, més els de la clau primària de **qualsevol** de les altres dues entitats (les connectades amb cardinalitat 1). Així, la nova relació té **dues claus candidates**, i l'elecció d'una com a clau primària queda al criteri del dissenyador lògic.

Exemple (figura 7). `Responsable`: en una campanya, un venedor és responsable com a molt d'una categoria, i una categoria té com a molt un venedor responsable. Es pot traduir de dues maneres:

```
VENEDOR(__DNI__, Nom, Cognoms)

CATEGORIA(__Codi__, Descripcio)

CAMPANYA(__Codi__)

RESPONSABILITAT(__CodiCampanya__, __DNIVenedor__, CodiCategoria,
 ↪  Objectiu)
    ON {DNIVenedor} REFERENCIA VENEDOR, {CodiCategoria} REFERENCIA
 ↪  CATEGORIA
    i {CodiCampanya} REFERENCIA CAMPANYA
```

O bé:

```
RESPONSABILITAT(__CodiCampanya__, __CodiCategoria__, DNIVenedor,
 ↪  Objectiu)
    ON {DNIVenedor} REFERENCIA VENEDOR, {CodiCategoria} REFERENCIA
 ↪  CATEGORIA
    i {CodiCampanya} REFERENCIA CAMPANYA
```

Hem canviat el nom de la relació derivada de la interrelació per convertir el verb original en un **substantiu**, que normalment és més adequat per designar relacions.

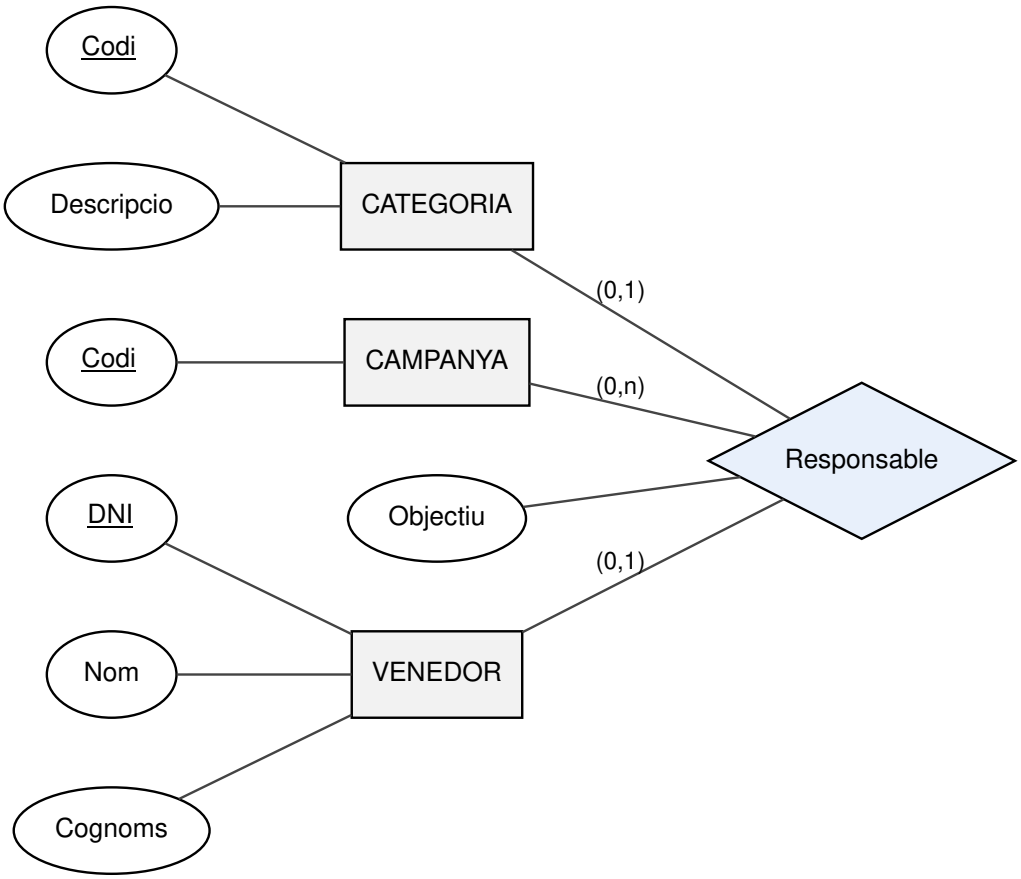

Figura 7: Figura 7. Interrelació ternària amb connectivitat 1-1-N

d. Connectivitat 1-1-1

La clau primària està composta pels atributs de la clau primària de **dues** entitats qualssevol, ja que totes tres estan connectades amb cardinalitat 1. Per tant, la nova relació té **tres claus candidates**.

Exemple (figura 8). Ara un venedor només pot ser responsable d'**una** categoria durant **una sola** campanya, al llarg de la seva vida a l'empresa (per afavorir la rotació). Es pot traduir de tres maneres, que només difereixen en la clau primària:

```
RESPONSABILITAT(__CodiCampanya__, __DNIVenedor__, CodiCategoria,
 ↪  Objectiu)
RESPONSABILITAT(__CodiCampanya__, __CodiCategoria__, DNIVenedor,
 ↪  Objectiu)
RESPONSABILITAT(__CodiCategoria__, __DNIVenedor__, CodiCampanya,
 ↪  Objectiu)
```

(En tots tres casos, amb les tres claus foranes corresponents.)

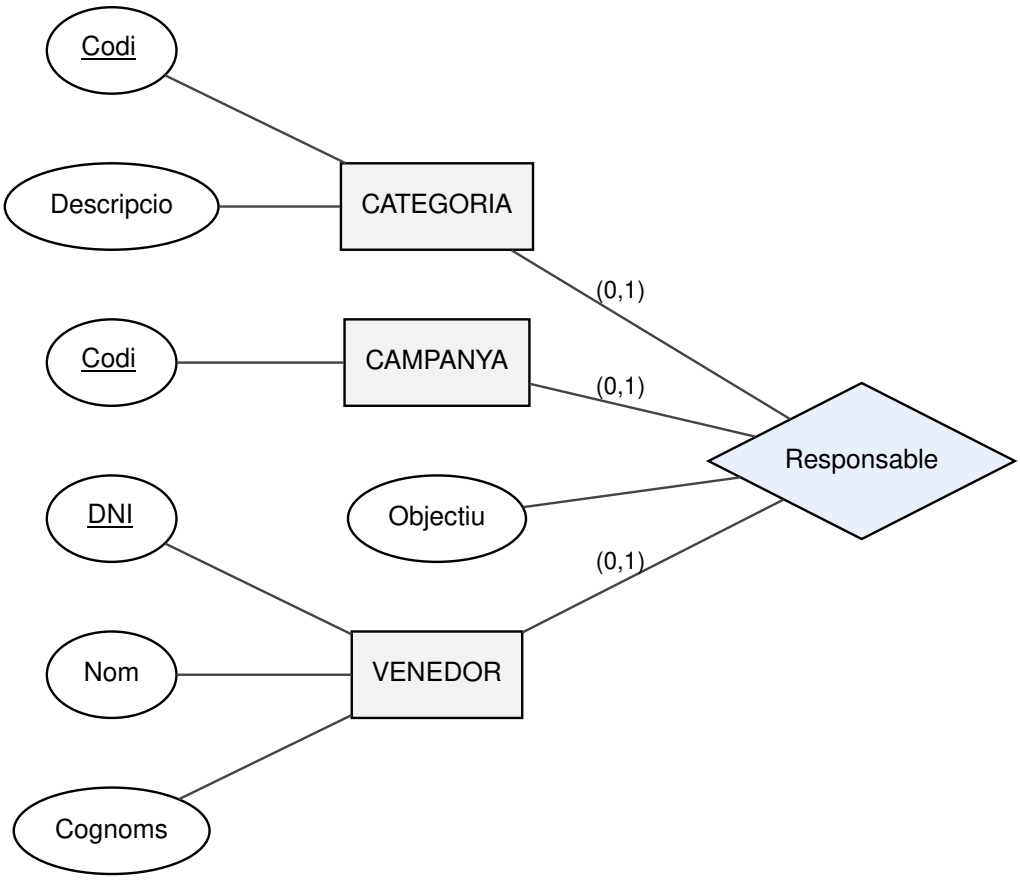

Figura 8: Figura 8. Interrelació ternària amb connectivitat 1-1-1

3. n-àries

Cada interrelació n-ària es transforma en una **nova relació** amb les claus primàries de **totes** les entitats relacionades, més els atributs propis de la interrelació. La clau primària depèn de la connectivitat:

Connectivitat	Clau primària
Totes les entitats amb cardinalitat **N**	Tots els atributs de les claus primàries de les *n* entitats (com a la M-N-P)
Una o més entitats amb cardinalitat **1**	Les claus primàries de totes les entitats **excepte una** (*n-1*). L'entitat que no hi aporta la clau ha d'estar forçosament connectada amb un 1

4. Recursives

Les interrelacions recursives es comporten com la resta:

- Les binàries 1-1 i 1-N donen lloc a una **clau forana**.
- Les binàries M-N i les n-àries originen una **nova relació**.

a. Binàries 1-1 o 1-N

Cal afegir a la relació de l'entitat una clau forana que faci referència a **la seva pròpia clau primària**. Els atributs de la clau forana **no poden tenir el mateix nom** que els de la clau primària als quals fan referència, perquè tots dos són a la mateixa relació.

Exemple (figura 9). `Apadrina`: un client pot apadrinar molts clients nous, però cada client té com a molt un padrí.

```
CLIENT(__DNI__, Nom, Cognoms, DNIPadri)
    ON {DNIPadri} REFERENCIA CLIENT
```

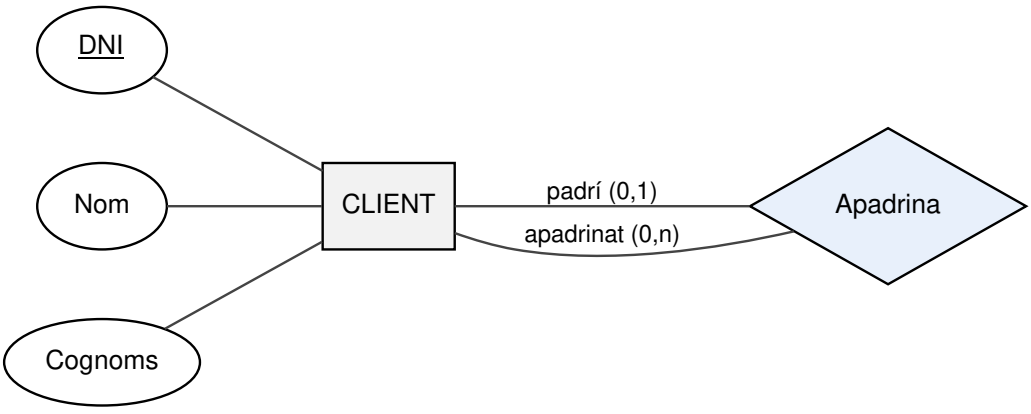

Figura 9: Figura 9. Interrelació recursiva binària amb connectivitat 1-N

b. Binàries M-N

S'origina una nova relació amb **dos cops** els atributs de la clau primària de l'entitat, perquè cal modelitzar que l'única entitat que intervé s'interrelaciona amb ella mateixa. Cal **modificar els noms** d'aquests atributs perquè no coincideixin.

Exemple (figura 10), a "Connecta": un usuari pot seguir molts usuaris i ser seguit per molts.

```
USUARI(__Alias__, Nom)

SEGUEIX(__AliasSeguidor__, __AliasSeguit__, DataInici)
    ON {AliasSeguidor} REFERENCIA USUARI i {AliasSeguit} REFERENCIA
 ↪  USUARI
```

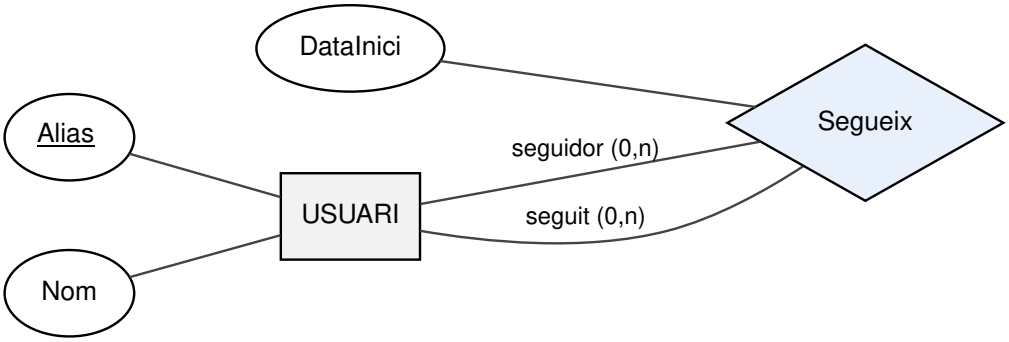

Figura 10: Figura 10. Interrelació recursiva binària amb connectivitat M-N

c. n-àries

S'origina una nova relació. Si la connexió de totes les entitats és amb cardinalitat N, la clau primària són tots els atributs de les claus primàries. Si alguna és amb cardinalitat 1, la clau primària són totes les claus excepte una (*n-1*), i l'entitat que no hi aporta la clau ha d'estar connectada amb un 1.

Exemple (figura 11). `Recomana` entre dos `CLIENT` i un `PRODUCTE` (1:M:N):

```
CLIENT(__DNI__, Nom, Cognoms)

PRODUCTE(__Codi__, Descripcio)

RECOMANA(__DNIClient__, __CodiProducte__, DNIRecomanador)
    ON {DNIClient} REFERENCIA CLIENT, {CodiProducte} REFERENCIA
 ↪  PRODUCTE
    i {DNIRecomanador} REFERENCIA CLIENT
```

Hem incorporat a la clau primària els atributs de les dues entitats connectades amb cardinalitat N (`PRODUCTE` i `CLIENT`), però des de la posició dels clients que **no** són recomanadors. Així es modelitza que cada client té **un** recomanador per producte, i que el recomanador d'un producte representa una pluralitat de clients.

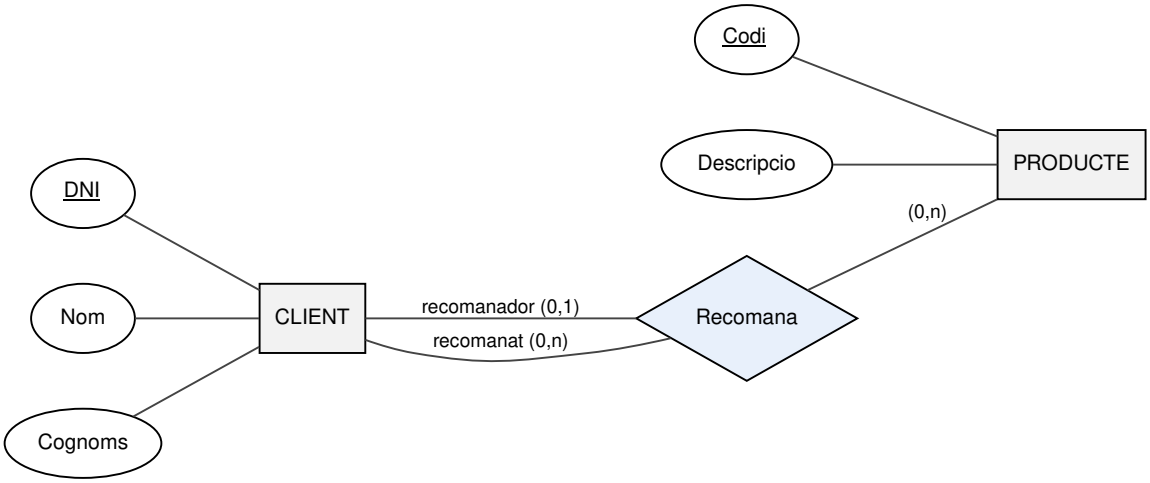

Figura 11: Figura 11. Interrelació recursiva n-ària

2.3.3. Entitats febles

Les entitats febles estan sempre al **costat N** d'una interrelació 1-N que serveix per completar la seva identificació. La relació derivada de l'entitat feble ha d'**incorporar a la seva clau primària** els atributs de la clau primària de l'entitat de la qual depèn. Aquests atributs són, alhora, una **clau forana** que hi fa referència.

Exemple (figura 12), a "Connecta": el nom d'un grup es repeteix entre poblacions («Amics del Poble»). `GRUP` és entitat feble de `POBLACIO`:

```
POBLACIO(__CodiPoblacio__)

GRUP(__CodiPoblacio__, __NomGrup__)
    ON {CodiPoblacio} REFERENCIA POBLACIO
```

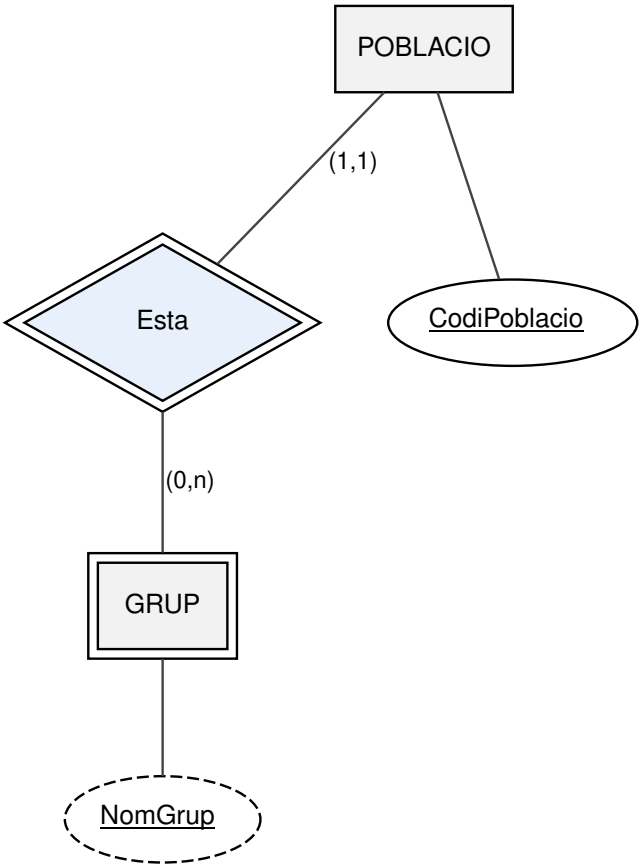

Figura 12: Figura 12. Entitat feble GRUP identificada a través de POBLACIO

2.3.4. Generalització i especialització

Tant l'entitat **superclasse** com les entitats **subclasse** es transformen en noves relacions:

- La relació derivada de la superclasse **hereta** d'aquesta la clau primària, i emmagatzema els atributs comuns a tota l'especialització o generalització.
- Les relacions derivades de les subclasses tenen com a clau primària **la clau de la superclasse**, que alhora actua com a **clau forana** en referenciar la relació derivada de la superclasse.

Exemple (figura 13). Un encadenament de generalitzacions i especialitzacions:

```
PERSONA(__DNI__, Nom, Cognoms, Telefon)

VENEDOR(__DNI__, Sou)
    ON {DNI} REFERENCIA PERSONA

CLIENT(__DNI__)
    ON {DNI} REFERENCIA PERSONA

VENEDOR_BOTIGA(__DNI__, Botiga, Torn)
    ON {DNI} REFERENCIA VENEDOR

VENEDOR_ONLINE(__DNI__, Plataforma, ComissioOnline)
    ON {DNI} REFERENCIA VENEDOR
```

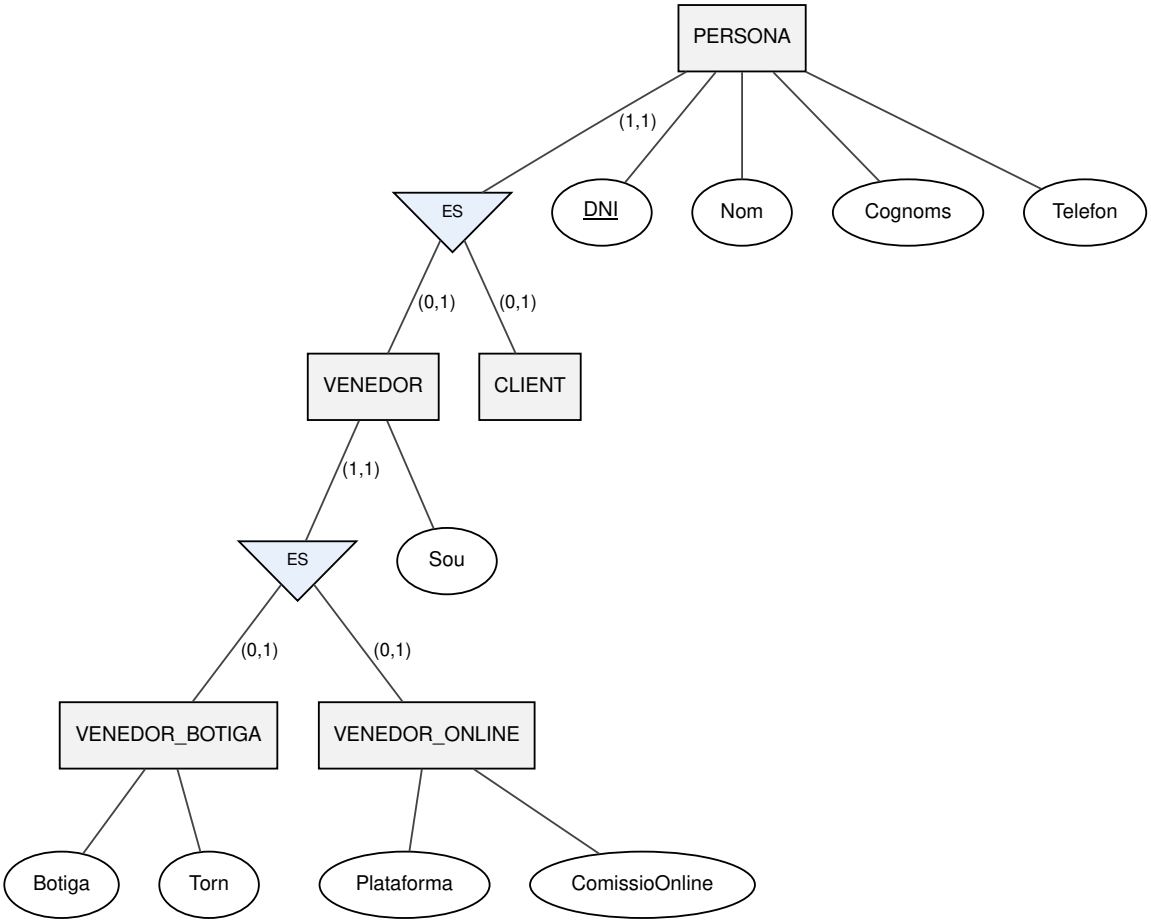

Figura 13: Figura 13. Generalització i especialització

2.3.5. Entitats associatives

Les entitats associatives es basen en una interrelació entre entitats. La traducció d'aquesta interrelació al model relacional **equival a la traducció de l'entitat associativa**.

Exemple (figura 14). L'agregació `INTERESSA` (formada per `CLIENT`, `Interessa` i `CATEGORIA`) es relaciona amb `VENEDOR` mitjançant `Assessora`:

```
CLIENT(__DNI__, Nom, Cognoms)

CATEGORIA(__Codi__, Descripcio)

VENEDOR(__DNI__, Nom, Cognoms)

INTERESSA(__DNIClient__, __CodiCategoria__, DNIVenedor)
    ON {DNIClient} REFERENCIA CLIENT, {CodiCategoria} REFERENCIA
 ↪  CATEGORIA
    i {DNIVenedor} REFERENCIA VENEDOR
```

Fixa't que la relació `INTERESSA`, derivada de l'entitat associativa, incorpora una **clau forana**

que fa referència a VENEDOR, perquè l'entitat associativa és al costat N d'una interrelació binària amb VENEDOR. Aquesta clau forana **admet nuls**: hi ha clients interessats en una categoria que no tenen venedor assessor.

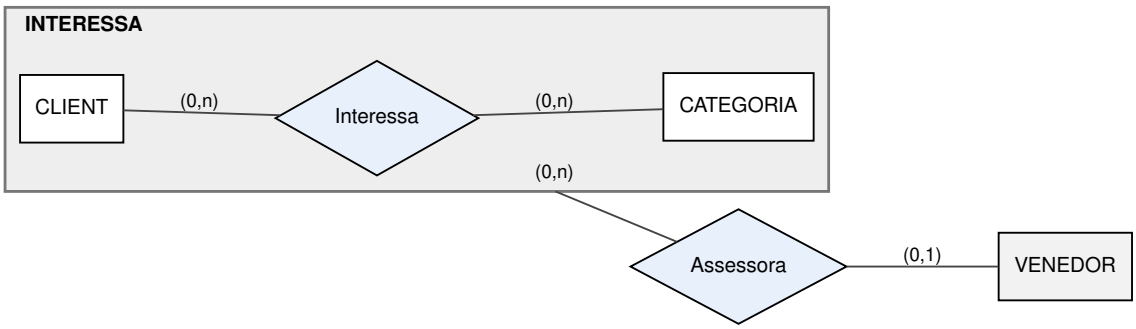

Figura 14: Figura 14. Entitat associativa INTERESSA

3. Normalització

El disseny d'una base de dades pot ser una tasca extremadament complexa. Hi ha diferents metodologies per trobar l'esquema relacional que representi millor la realitat que es vol modelitzar.

El camí aconsellable és: primer efectuar el **diagrama Entitat-Relació** i després **traduir-lo al model relacional**. Si el diagrama ER és correcte, l'esquema relacional serà correcte. Però no sempre és així i ens trobem dissenys fets directament sobre l'esquema relacional. Causes:

- El model ER és posterior al relacional, i hi ha BD formulades directament en terminologia relacional.
- Hi ha dissenyadors que «no volen perdre el temps» amb un model ER.
- De vegades es modifica la BD per noves necessitats, i el canvi es fa directament sobre l'esquema relacional en lloc de fer-lo sobre el model ER i traslladar-lo després.

Cal donar un suport absolut a l'ús del model ER **anteriorment** al model relacional. Si no hi ha disseny ER previ, hi ha més possibilitats de tenir una BD relacional mal dissenyada.

La **teoria de la normalització** és un mètode que permet assegurar si un disseny relacional (tant si prové de la traducció d'un diagrama ER com si s'ha fet directament en el model relacional) és més o menys correcte.

Els mals dissenys poden originar:

- **Repetició** de la informació.
- **Impossibilitat de representar** certa informació, amb **anomalies** en les insercions, en les modificacions i en els esborraments.

Un bon disseny ha d'aconseguir:

- Emmagatzemar tota la informació necessària amb el **mínim d'informació redundant**.

- Mantenir el **mínim de lligams** entre les relacions per facilitar-ne la utilització.
- Millorar la **consultabilitat** de les dades.
- **Minimitzar els problemes d'actualització** (altes, baixes i modificacions) que poden sorgir en haver d'actualitzar simultàniament dades de diferents relacions.

Exemple de disseny relacional inadequat. Considerem el disseny de la taula 5, per enregistrar la informació dels **venedors** amb els **clients** que assessoren i la **valoració** que cada client ha fet dels productes. La clau primària està formada per totes les columnes excepte `Valoracio`.

Taula 5. Exemple de disseny relacional inadequat

DniVenedor	NomVenedor	DniClient	NomClient	Edat	Producte	Valoracio
11.111.111	Albert Serra	55.555.555	Pau Ferrer	34	Portàtil X15	8
11.111.111	Albert Serra	66.666.666	Núria Camps	28	Portàtil X15	9
11.111.111	Albert Serra	55.555.555	Pau Ferrer	34	Ratolí M3	7
11.111.111	Albert Serra	66.666.666	Núria Camps	28	Ratolí M3	6
22.222.222	Berta Coll	55.555.555	Pau Ferrer	34	Monitor 27”	9
22.222.222	Berta Coll	66.666.666	Núria Camps	28	Monitor 27”	8
22.222.222	Berta Coll	55.555.555	Pau Ferrer	34	Teclat K2	5
22.222.222	Berta Coll	66.666.666	Núria Camps	28	Teclat K2	6

Ràpidament hi veiem els problemes següents:

- Hi ha **informació repetida**, fet que pot provocar inconsistències. Si cal modificar qualsevol dels valors dels camps que formen la clau primària (`DniVenedor`, `NomVenedor`, `DniClient`, `NomClient`, `Edat`, `Producte`), el canvi s'ha d'efectuar en **totes** les files on apareix aquest valor.
- **No hi pot haver valors nuls** en les columnes de la clau primària. Així, si no coneixem l'edat d'un client, tenim un greu problema.
- Si cal emmagatzemar més informació dels venedors o dels clients, caldrà afegir més columnes i repetir la informació a cada fila en què aparegui el venedor o el client.
- **Consultar** la informació pot esdevenir feixuc, atesa la gran quantitat d'informació diferent que conté.

El mètode que proposa la teoria de la normalització consisteix a **avaluar el disseny de totes les relacions** per veure en quin grau de normalitat es troba cadascuna i decidir si cal refinar-lo.

La teoria de la normalització defineix les **formes normals** com a indicadors per avaluar el grau de normalitat de les relacions. Una relació està en una forma normal determinada quan satisfà un conjunt determinat de condicions.

Hi ha diferents graus de normalitat, que compleixen la **relació d'inclusió** de la figura 15: a mesura que augmenta la forma normal, la relació ha de complir un conjunt de condicions més restrictiu i, a més, continua verificant les condicions de les formes normals de nivell inferior.

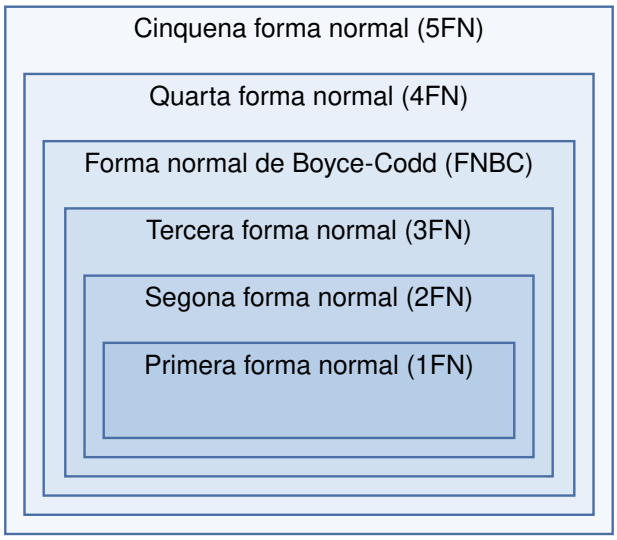

Figura 15: Figura 15. Relació d'inclusió entre les diverses formes normals

L'objectiu hauria de ser aconseguir un esquema relacional en què totes les relacions tinguin el grau màxim de normalitat (la **cinquena forma normal**, 5FN).

El **procés de normalització** per aconseguir que una relació que es troba en una forma normal X passi a estar en una forma normal Y superior consisteix sempre en la **descomposició** o subdivisió de la relació original en dues o més relacions que verifiquin el nivell de forma normal Y.

El procés de normalització **augmenta el nombre de relacions** de la BD. Això comporta una disminució de redundàncies i d'anomalies d'actualització, però **penalitza les consultes**, perquè caldrà anar a cercar la informació en moltes taules relacionades. Cal trobar un equilibri: de vegades pot ser convenient renunciar al nivell màxim de normalització i permetre certa redundància per alleugerir el cost de les consultes. Això és la **desnormalització**.

L'objectiu final és conèixer les condicions que han de complir les relacions per assolir cada forma normal, i el procés per dividir-les en noves relacions. Per això cal conèixer els conceptes de **relació universal** i **dependència funcional**.

3.1. La relació universal

En fer directament el disseny relacional, el dissenyador té un conjunt de conceptes que tradueix en atributs, els quals, pel seu significat, agruparà en una o més relacions.

Anomenem **relació universal** la relació consistent en l'agrupament dels atributs corresponents a **tots** els conceptes que constitueixen una base de dades relacional.

Imaginem que es vol dissenyar una BD per al control de les **comandes de compra a proveïdors** de la comercial de productes informàtics. Hi volem incloure: número i data de la comanda; codi, descripció, quantitat i preu pactat per cada article sol·licitat; data prevista de lliurament; nom (`NomProv`) i país (`PaisProv`) del proveïdor; i moneda en què es pacta la comanda. La relació universal és la de la taula 6.

Taula 6. Relació universal per a una gestió de comandes de compra

Num	DataComa	Article	Descripcio	Qtat	Preu	DataPrevis	NomProv	PaisProv	Moneda
3001	04-11-2026	LT-15	Portàtil 15" i5	10	620	12-11-2026	NovaTech	Taiwan	USD
3001	04-11-2026	MS-03	Ratolí sense fil	10	9	12-11-2026	NovaTech	Taiwan	USD
3002	06-11-2026	LT-15	Portàtil 15" i5	4	655	18-11-2026	DataPlus	Alemanya	EUR
3002	06-11-2026	MN-27	Monitor 27"	8	190	18-11-2026	DataPlus	Alemanya	EUR
3003	09-11-2026	MS-03	Ratolí sense fil	50	8	20-11-2026	NovaTech	Taiwan	USD

Hi ha molta redundància i poca organització. El disseny relacional basat en la relació universal acostuma a ser del tot incorrecte, i cal aplicar un procés de normalització per dividir-la en relacions que assoleixin graus de normalitat millors.

NOTA

Molt poques vegades es parteix de la relació universal. L'experiència dels dissenyadors fa que, d'entrada, ja es pensi en relacions que assoleixen un cert grau de normalitat.

3.2. Dependències funcionals

Les definicions de les formes normals es basen en el concepte de **dependència funcional**.

Donats dos atributs (o conjunts d'atributs) **A** i **B** d'una relació R, diem que **B depèn funcionalment de A** si per cada valor d'A existeix **un, i només un**, valor de B associat amb ell. També diem que **A implica B**. Ho simbolitzem per `A -> B`.

Les dependències funcionals estableixen lligams entre atributs d'una mateixa relació. A la relació universal de la taula 6 diem que, entre d'altres, la data de comanda, la data prevista, el nom i el país del proveïdor i la moneda depenen funcionalment del número de comanda:

```
Num -> DataComanda
Num -> DataPrevista
Num -> NomProv
Num -> PaisProv
Num -> Moneda
```

En tractar-se de diferents atributs que depenen d'un mateix atribut, escrivim:

```
Num -> DataComanda, DataPrevista, NomProv, PaisProv, Moneda
```

Donats dos atributs (o conjunts d'atributs) A i B d'una relació R, diem que B té una **dependència funcional completa o total** de A si B depèn funcionalment de A, però **no depèn funcionalment de cap subconjunt d'A**.

És convenient representar les dependències funcionals d'una relació mitjançant un **esquema de dependències funcionals** (figura 16).

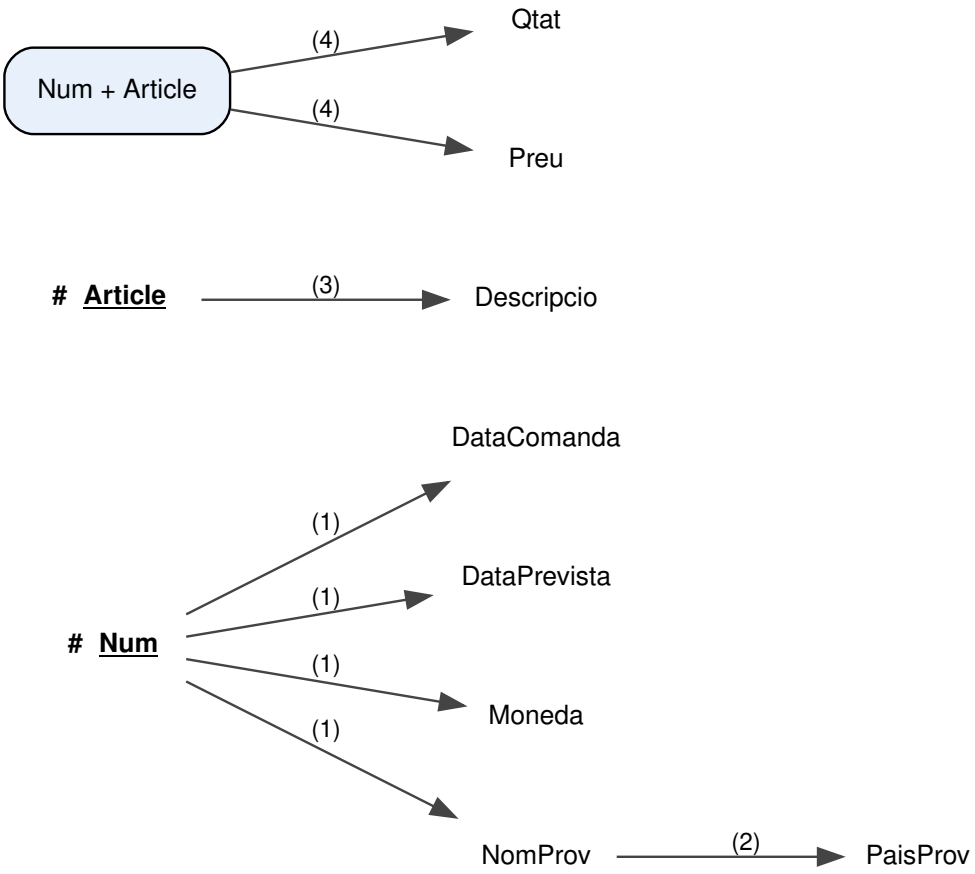

Figura 16: Figura 16. Exemple d'esquema de dependències funcionals

Hem marcat amb # els atributs que són clau d'alguna de les entitats que formen la relació: `Article` identifica l'article i `Num` identifica la comanda. La parella (`Num`, `Article`) identifica la quantitat i el preu dels articles demanats a la comanda.

En aquest esquema es veuen les dependències següents:

Núm.	Dependència	Explicació
(1)	`Num -> DataComanda, DataPrevista, Moneda, NomProv`	Depenen funcionalment de `Num`
(2)	`NomProv -> PaisProv`	`PaisProv` depèn funcionalment de `NomProv` (suposem que cada proveïdor té un únic país)
(3)	`Article -> Descripcio`	`Descripcio` depèn funcionalment d'`Article`
(4)	`Num, Article -> Qtat, Preu`	`Qtat` i `Preu` depenen funcionalment de la parella

Les dependències (1), (2) i (3) són **totals**, perquè la part esquerra (l'**implicador**) està formada per un únic atribut i, per tant, és impossible que la part dreta (l'**implicat**) depengui d'un subconjunt de l'implicador. La (4) també és total, ja que `Qtat` i `Preu` depenen de la parella i no de cap subconjunt.

Observa que en una mateixa comanda (`Num`) no és possible tenir diverses vegades el mateix article. Si calgués, caldria un altre atribut per identificar l'article dins la comanda, com un `NumeroDeLinia`.

Donat un atribut o conjunt d'atributs A d'una relació, diem que **A és un determinant** de la relació si hi ha algun altre atribut o conjunt d'atributs B que té dependència funcional total d'A.

En el cas anterior, `Num`, `Article`, `NomProv` i la parella (`Num`, `Article`) són **determinants** de la relació.

Donats A, B i C atributs o conjunts d'atributs d'una relació, diem que **C depèn transitivament d'A** a través de B si B depèn funcionalment d'A, C depèn funcionalment de B, i A no depèn funcionalment de B.

A l'exemple, `PaisProv` depèn transitivament de `Num` a través de `NomProv`.

3.3. Primera forma normal (1FN)

En aplicar el procés de normalització a una relació, es comença comprovant si està en primera forma normal i, si no és així, es fan les modificacions oportunes.

Una relació està en **primera forma normal (1FN)** si cap atribut pot contenir **valors no atòmics** (indivisibles).

La relació de la taula 7 **no està en 1FN**: té atributs que contenen més d'un valor. Hi ha tres files (comandes 3001, 3002 i 3003) i alguns atributs (`Article`, `Descripcio`, `Qtat` i `Preu`) tenen diversos valors per a algunes files.

Taula 7. Relació amb atributs multivalor (no està en 1FN)

Num	DataComa	Article	Descripcio	Qtat	Preu	DataPrevis	NomProv	PaisProv	Moneda
3001	04-11-2026	LT-15, MS-03	Portàtil 15" i5, Ratolí sense fil	10, 10	620, 9	12-11-2026	NovaTech	Taiwan	USD
3002	06-11-2026	LT-15, MN-27	Portàtil 15" i5, Monitor 27"	4, 8	655, 190	18-11-2026	DataPlus	Alemanya	EUR
3003	09-11-2026	MS-03	Ratolí sense fil	50	8	20-11-2026	NovaTech	Taiwan	USD

El procés per assolir una 1FN és afegir **tantes files com sigui necessari** per a cadascun dels valors del camp o camps que tinguin valors no atòmics.

Així obtenim la relació en 1FN, que és la de la taula 6.

De fet, la restricció que persegueix la 1FN forma part de la definició del model relacional i, per tant, **tota relació, per definició, ha d'estar en 1FN**. Aquesta forma normal és redundant amb la definició del model relacional i no caldria considerar-la; es manté per assegurar que les relacions dissenyades tenen un punt de partida correcte.

En general, les relacions en 1FN poden tenir molta informació redundant. Això no ens ha de preocupar: la solució rau en les formes normals de nivell superior.

3.4. Preservació d'informació i dependències en la normalització

En el procés de normalització d'una relació R s'apliquen processos de **descomposició** per aconseguir relacions R1, R2, …, Rn que verifiquin un nivell de normalització superior. La descomposició consisteix a efectuar **projeccions** de R sobre atributs que verifiquin certes condicions.

NOTA

Natural join (R1 * R2): combinació de totes les dades de la primera relació (R1) amb totes les de la segona (R2), sempre que els valors de les dades de les columnes anomenades de forma idèntica a les dues relacions coincideixin.

Cal garantir que la descomposició **preservi la informació**: que el *natural join* R1 * R2 * … * Rn proporcioni exactament la mateixa informació que la relació R original, tant en **intenció** (quantitat d'atributs) com en **extensió** (quantitat de files).

De la mateixa manera, caldria garantir la **conservació de les dependències**: el conjunt de dependències associades a R ha de ser equivalent al conjunt de dependències de R1, R2, …, Rn.

La conservació de les dependències **no es pot garantir** en tots els processos de normalització.

3.5. Segona forma normal (2FN)

La segona forma normal persegueix l'eliminació dels problemes motivats per la presència de **dependències funcionals no totals** dels atributs que no formen part de la clau primària respecte a la clau primària.

Una relació està en **segona forma normal (2FN)** si està en 1FN i **tot atribut que no pertany a la clau té dependència funcional total de la clau**.

Considerem la relació següent en 1FN, ideada per a la gestió de comandes de compra:

```
R (__Num__, DataComanda, __Article__, Descripcio, Qtat, Preu,
   DataPrevista, NomProv, PaisProv, Moneda)
```

La clau primària és la parella (`Num`, `Article`) i, a l'esquema de dependències funcionals (figura 16), hi ha atributs fora de la clau que **no tenen dependència funcional completa de la clau**: les dependències (1), (2) i (3) ens presenten atributs que no tenen dependència total de la parella (`Num`, `Article`).

El procés per assolir una 2FN és **dividir la relació** (conservant la informació i les dependències) en tantes relacions com sigui necessari de manera que cada relació verifiqui que els seus atributs no-clau tenen dependència funcional total de la clau.

L'esquema de dependències ajuda a veure les relacions que han d'aparèixer. De les dependències (1), (3) i (4) obtenim les relacions en 2FN:

```
COMANDA (__Num__, DataComanda, DataPrevista, Moneda, NomProv,
 ↪  PaisProv)
```

```
ARTICLE (__Article__, Descripcio)
DETALL  (__Num__, __Article__, Qtat, Preu)
          comanda  article
```

(A `DETALL`, `Num` fa referència a `COMANDA` i `Article` a `ARTICLE`.)

La informació de la taula 6 passa a estar repartida en tres taules (8, 9 i 10).

Taula 8. Relació en 2FN que emmagatzema les comandes (`COMANDA`)

Num	DataComanda	DataPrevista	NomProv	PaisProv	Moneda
3001	04-11-2026	12-11-2026	NovaTech	Taiwan	USD
3002	06-11-2026	18-11-2026	DataPlus	Alemanya	EUR
3003	09-11-2026	20-11-2026	NovaTech	Taiwan	USD

Taula 9. Relació en 2FN pels articles (`ARTICLE`)

Article	Descripcio
LT-15	Portàtil 15" i5
MS-03	Ratolí sense fil
MN-27	Monitor 27"

Taula 10. Relació en 2FN pel detall de comanda (`DETALL`)

Num	Article	Qtat	Preu
3001	LT-15	10	620
3001	MS-03	10	9
3002	LT-15	4	655
3002	MN-27	8	190
3003	MS-03	50	8

Sovint, en fer un disseny ja obtindrem relacions que estan en 2FN i, fins i tot, en formes normals de nivell superior. Amb el nou disseny s'ha eliminat molta redundància i, per tant, es redueixen els problemes d'actualització i consulta. Però **no desapareixen tots**.

3.6. Tercera forma normal (3FN)

Considerem les relacions en 2FN ideades per a la gestió de comandes:

```
COMANDA (__Num__, DataComanda, DataPrevista, Moneda, NomProv,
 ↪  PaisProv)
ARTICLE (__Article__, Descripcio)
DETALL  (__Num__, __Article__, Qtat, Preu)
```

El país del proveïdor apareix a cada comanda. Si partim de la base que el país on resideix el proveïdor és únic, encara hi ha informació redundant.

La tercera forma normal persegueix l'eliminació dels problemes motivats per la presència de **dependències transitives** dels atributs que no formen part de la clau primària respecte a la clau primària.

Una relació està en **tercera forma normal (3FN)** si està en 2FN i **cap atribut que no pertany a la clau depèn transitivament de la clau**.

`ARTICLE` i `DETALL` ja estan en 3FN, però a `COMANDA` l'atribut `PaisProv` depèn transitivament de `Num` a través de `NomProv`. Per tant, aquesta relació **no està en 3FN**.

El procés per assolir una 3FN és **dividir la relació** (conservant la informació i les dependències) en noves relacions més simples, de manera que cada relació verifiqui que cap dels seus atributs no-clau depèn transitivament de la clau.

En el nostre cas, obtenim les relacions en 3FN:

```
PROVEIDOR (__CodProv__, NomProv, PaisProv)
COMANDA   (__Num__, DataComanda, DataPrevista, Moneda, CodProv)
                                                       proveïdor
```

En trencar la relació inicial `COMANDA`, ha semblat oportú considerar un nou atribut (`CodProv`) que identifiqui millor la nova relació `PROVEIDOR`. Això no és imprescindible ni sempre serà convenient. Podríem haver considerat el trencament següent:

```
PROVEIDOR (__NomProv__, PaisProv)
COMANDA   (__Num__, DataComanda, DataPrevista, Moneda, NomProv)
                                                       proveïdor
```

Ara bé, per a aquesta darrera possibilitat hem escollit el nom del proveïdor com a clau primària, i l'experiència ens aconsella definir **un codi** que permeti identificar-los de manera més clara que la que proporciona el seu nom.

Amb el disseny que incorpora `CodProv`, la taula 8 passa a ser les taules 11 i 12.

Taula 11. Relació en 3FN que emmagatzema les comandes (`COMANDA`)

Num	DataComanda	DataPrevista	CodProv	Moneda
3001	04-11-2026	12-11-2026	NVT	USD
3002	06-11-2026	18-11-2026	DPL	EUR
3003	09-11-2026	20-11-2026	NVT	USD

Taula 12. Relació en 3FN pels proveïdors (`PROVEIDOR`)

CodProv	NomProv	PaisProv
NVT	NovaTech	Taiwan
DPL	DataPlus	Alemanya

3.7. Forma normal de Boyce-Codd (FNBC)

Considerem els **magatzems** de la comercial. Cada magatzem està dedicat a guardar **una família de productes** (portàtils, monitors, perifèrics…) i cada proveïdor pot deixar mercaderia en consigna en magatzems diferents segons la família (un proveïdor de portàtils i perifèrics deixa cada família en un magatzem diferent). Tota la mercaderia d'una mateixa família d'un proveïdor es troba concentrada en un **prestatge** dins del mateix magatzem, per facilitar-ne la recollida.

Per saber quina família hi ha a cada magatzem es dissenya aquesta relació:

```
DIPOSIT (__Proveidor__, __Familia__, Magatzem, Prestatge)
```

La taula 13 exemplifica la situació. Fixa't que es troba en **3FN**:

- Tots els atributs no pertanyents a la clau (`Magatzem` i `Prestatge`) tenen dependència funcional total de la clau (2FN).
- Cap atribut no pertanyent a la clau no depèn transitivament de la clau: no hi ha cap atribut que pugui servir de pont per a la transitivitat.

Taula 13. Relació en 3FN (`DIPOSIT`)

Proveidor	Familia	Magatzem	Prestatge
NovaTech	Portàtils	1	A3-R2
NovaTech	Perifèrics	3	C1-R5
DataPlus	Portàtils	1	A1-R4
DataPlus	Monitors	2	B2-R1
LogiPart	Perifèrics	3	C4-R2
LogiPart	Monitors	2	B1-R3

Efectivament, `Magatzem` i `Prestatge` depenen del `Proveidor` i de la `Familia`: un proveïdor pot tenir diversos magatzems dedicats a famílies diferents, però tota la mercaderia semblant d'un proveïdor és en un magatzem concret. Tot i estar en 3FN, aquesta relació presenta **anomalies**:

- Si en un moment donat un magatzem no té mercaderia de cap proveïdor, **es perd la informació** de la família que correspon al magatzem.
- Si canvia la família assignada a un magatzem, cal modificar tantes files com proveïdors amb dipòsits en aquell magatzem hi hagi.

Una relació està en la **forma normal de Boyce-Codd (FNBC)** si està en 2FN i **tots els seus determinants són claus candidates**.

La relació `DIPOSIT` **no està en FNBC**: l'atribut `Magatzem` és un determinant (`Familia` té dependència funcional total de `Magatzem`) i, en canvi, `Magatzem` no és clau candidata:

```
Magatzem -> Familia
```

Tota relació en FNBC està en 3FN, però **no a l'inrevés**, com acabem de comprovar.

El procés per assolir una FNBC és **apartar de la relació** els atributs que depenen dels determinants que no són claus candidates, i formar noves relacions que recullin els atributs apartats i que preservin la informació inicial.

Apartarem l'atribut `Familia` i obtindrem les relacions en FNBC:

```
DIPOSIT          (__Proveidor__, __Magatzem__, Prestatge)
MAGATZEM_FAMILIA (__Magatzem__, Familia)
```

En els trencaments per assolir 2FN i 3FN es preserva sempre la informació **i** les dependències funcionals. En el pas a FNBC també es pot mantenir la informació, però **no sempre** es poden mantenir les dependències.

La relació inicial `DIPOSIT` contenia les dependències funcionals:

```
Proveidor, Familia -> Magatzem, Prestatge
Magatzem -> Familia
```

I, a les relacions finals, s'ha perdut la dependència que indicava que el magatzem i el prestatge depenien de la parella `Proveidor` i `Familia`:

```
DIPOSIT:          Proveidor, Magatzem -> Prestatge
MAGATZEM_FAMILIA: Magatzem -> Familia
```

La dependència s'ha perdut perquè en el model relacional les dependències funcionals es defineixen només entre atributs d'**una mateixa relació**. El concepte d'integritat referencial intenta superar aquesta limitació.

A causa de la pèrdua de dependències funcionals, que no sempre es produeix, sovint **no es normalitza a FNBC** i es treballa amb relacions en 3FN.

3.8. Quarta forma normal (4FN)

Considerem la relació `PREFERENCIA` (taula 14), que emmagatzema les **marques preferides** de cada client i els **canals de comunicació** pels quals vol rebre ofertes.

```
PREFERENCIA (__Dni__, __Marca__, __Canal__)
```

La relació recull la possibilitat que un client tingui diverses marques preferides i admeti diversos canals.

Taula 14. Relació en FNBC amb redundància (`PREFERENCIA`)

Dni	Marca	Canal
11.111.111	HP	Correu
11.111.111	Lenovo	Correu
11.111.111	HP	SMS

11.111.111	Lenovo	SMS
22.222.222	Asus	WhatsApp
22.222.222	Dell	WhatsApp
22.222.222	HP	WhatsApp
22.222.222	Asus	Correu
22.222.222	Dell	Correu
22.222.222	HP	Correu
33.333.333	Lenovo	Correu
33.333.333	Lenovo	WhatsApp

Aquesta relació es troba en FNBC i, així i tot, hi ha **redundància**, provocada per un nou concepte: les **dependències multivalents**.

Donats A i B atributs o conjunts d'atributs d'una relació, diem que B té una **dependència multivalent** de A si un valor d'A pot determinar un **conjunt de valors** de B. Ho simbolitzem amb la notació `A ->> B`.

Les dependències multivalents no són dependències funcionals. En canvi, una dependència funcional es pot arribar a considerar una dependència multivalent en què per cada valor de l'implicant hi ha un únic valor de l'implicat.

Els problemes provocats per les dependències funcionals han causat la definició de la 1FN, 2FN, 3FN i FNBC. La redundància provocada per les dependències multivalents ens porta a definir la **4FN**.

Una relació es troba en **quarta forma normal (4FN)** si està en 3FN i **l'implicant de tota dependència multivalent és una clau candidata**.

Quan té lloc una dependència multivalent `A ->> B`, també existeix la dependència multivalent `A ->> X - (A U B)`, on X indica el conjunt de tots els atributs de la relació. És a dir, les dependències multivalents es presenten **per parelles**.

En el nostre cas es verifica el següent:

```
Dni ->> Marca
Dni ->> Canal
```

Per assolir la 4FN a partir d'una relació R(A, B, C) que té una dependència multivalent A -» B, cal **descompondre** la relació R en dues relacions R1(A, B) i R2(A, C).

Taula 15. Relació en 4FN (`MARCA_PREFERIDA`)

Dni	Marca
11.111.111	HP
11.111.111	Lenovo
22.222.222	Asus
22.222.222	Dell
22.222.222	HP
33.333.333	Lenovo

Taula 16. Relació en 4FN (`CANAL_CONTACTE`)

Dni	Canal
11.111.111	Correu
11.111.111	SMS
22.222.222	WhatsApp
22.222.222	Correu
33.333.333	Correu
33.333.333	WhatsApp

En aquest cas, obtenim les relacions següents:

```
MARCA_PREFERIDA (__Dni__, __Marca__)
CANAL_CONTACTE  (__Dni__, __Canal__)
```

Sovint, la descomposició causada per les dependències multivalents s'efectua **abans** de les descomposicions per assolir 2FN, 3FN i FNBC. En aquesta situació, a les relacions obtingudes cal fer les comprovacions per aconseguir que estiguin en 2FN, 3FN i FNBC.

3.9. Cinquena forma normal (5FN)

Considerem la relació `VENEDOR` (taula 17), dissenyada per gestionar els venedors d'una cadena de botigues. Cada venedor està autoritzat a vendre unes determinades **especialitats** que pot posar en pràctica en **qualsevol** de les botigues de l'empresa. Així mateix, cada venedor pot exercir, a més de la venda, diferents **tasques** (assessorament, caixa, instal·lació…) en diverses botigues.

```
VENEDOR (__CodiVen__, __Botiga__, __Especialitat__, __Tasca__)
```

Taula 17. Relació en FNBC amb dependències multivalents (`VENEDOR`)

CodiVen	Botiga	Especialitat	Tasca
V1	Mataró	Portàtils	Assessor
V1	Barcelona	Portàtils	Assessor
V1	Barcelona	Xarxes	Instal·lador
V2	Mataró	Impressores	Responsable
V2	Barcelona	Components	Assessor

Aquesta relació és FNBC i s'hi aprecia una espècie de dependència multivalent, la qual **no es pot solucionar per la via de la descomposició**. És molt fàcil pensar en una descomposició en les tres relacions següents (taules 18, 19 i 20):

```
BOTIGA_DE_VENEDOR       (__CodiVen__, __Botiga__)
ESPECIALITAT_DE_VENEDOR (__CodiVen__, __Especialitat__)
TASCA_DE_VENEDOR        (__CodiVen__, __Tasca__)
```

Taula 18. Relació en 4FN (`BOTIGA_DE_VENEDOR`)

CodiVen	Botiga
V1	Mataró
V1	Barcelona
V2	Mataró
V2	Barcelona

Taula 19. Relació en 4FN (ESPECIALITAT_DE_VENEDOR)

CodiVen	Especialitat
V1	Portàtils
V1	Xarxes
V2	Impressores
V2	Components

Taula 20. Relació en 4FN (TASCA_DE_VENEDOR)

CodiVen	Tasca
V1	Assessor
V1	Instal·lador
V2	Responsable
V2	Assessor

Aquesta descomposició és **errònia**: si apliquem el *natural join* de les tres relacions no obtenim la relació inicial (taula 17), sinó una relació (taula 21) amb moltes més instàncies (16 files en lloc de 5).

Taula 21. Relació obtinguda del *natural join* de les relacions de les taules 18, 19 i 20

CodiVen	Botiga	Especialitat	Tasca
V1	Mataró	Portàtils	Assessor
V1	Mataró	Portàtils	Instal·lador
V1	Mataró	Xarxes	Assessor
V1	Mataró	Xarxes	Instal·lador
V1	Barcelona	Portàtils	Assessor
V1	Barcelona	Portàtils	Instal·lador
V1	Barcelona	Xarxes	Assessor
V1	Barcelona	Xarxes	Instal·lador
V2	Mataró	Impressores	Responsable
V2	Mataró	Impressores	Assessor
V2	Mataró	Components	Responsable
V2	Mataró	Components	Assessor
V2	Barcelona	Impressores	Responsable
V2	Barcelona	Impressores	Assessor
V2	Barcelona	Components	Responsable
V2	Barcelona	Components	Assessor

Queda clar que el mètode utilitzat no és correcte, i això és degut al fet que, en aquesta situació, hi ha el que s'anomena **dependències mútues** entre els atributs de la relació. Les dependències mútues provoquen que la descomposició de la relació en altres relacions (projeccions de l'original) no verifiqui que el seu *natural join* coincideix amb la relació original.

Direm que una relació R descomposta en relacions R1, R2, …, Rn satisfà una **dependència de reunió**, també anomenada **dependència de projecció-join**, respecte a R1, R2, …, Rn únicament si R és igual al *natural join* de R1, R2, …, Rn. La notarem com a `DR*(R1, R2, …, Rn)`.

Ens cal trobar una dependència de reunió per a la relació `VENEDOR`, és a dir, trobar una descomposició tal que el seu *natural join* recuperi la relació original. Fixem-nos en la descomposició següent (taules 22, 23 i 24):

```
VENEDOR (__CodiVen__, Botiga, Especialitat, Tasca)
PBE = VENEDOR [__CodiVen__, Botiga, Especialitat]
PBT = VENEDOR [__CodiVen__, Botiga, Tasca]
PET = VENEDOR [__CodiVen__, Especialitat, Tasca]
```

On PBE, PBT i PET són **seccions (projeccions)** de la taula original.

Taula 22. Relació en 4FN (PBE)

CodiVen	Botiga	Especialitat
V1	Mataró	Portàtils
V1	Barcelona	Portàtils
V1	Barcelona	Xarxes
V2	Mataró	Impressores
V2	Barcelona	Components

Taula 23. Relació en 4FN (PBT)

CodiVen	Botiga	Tasca
V1	Mataró	Assessor
V1	Barcelona	Assessor
V1	Barcelona	Instal·lador
V2	Mataró	Responsable
V2	Barcelona	Assessor

Taula 24. Relació en 4FN (PET)

CodiVen	Especialitat	Tasca
V1	Portàtils	Assessor
V1	Xarxes	Instal·lador
V2	Impressores	Responsable
V2	Components	Assessor

En aquesta situació veiem que si efectuem el *natural join* de les tres relacions PBE, PBT i PET obtenim la **relació original**.

Direm que una relació està en **cinquena forma normal (5FN)**, també anomenada **forma normal projecció-join (FNPJ)**, si està en 4FN i **tota dependència de reunió és conseqüència de claus candidates**.

Per tant, la relació VENEDOR del nostre exemple **no es troba en 5FN**, ja que hem trobat la dependència de reunió `VENEDOR*(PBE, PBT, PET)` en què les relacions PBE, PBT i PET no estan constituïdes per claus candidates de VENEDOR.

Una relació 4FN que no sigui 5FN a causa d'una dependència de reunió es pot descompondre, **sense pèrdua d'informació**, en les relacions sobre les quals es defineix la dependència de reunió, les quals estan en 5FN.

Així, en el nostre exemple, la relació `VENEDOR` desapareixeria per donar pas a les tres relacions `PBE`, `PBT` i `PET` en què es basa la dependència de reunió trobada:

```
PBE (__CodiVen__, __Botiga__, __Especialitat__)
PBT (__CodiVen__, __Botiga__, __Tasca__)
PET (__CodiVen__, __Especialitat__, __Tasca__)
```

3.10. Desnormalització

Pot semblar estrany plantejar-se la desnormalització d'una base de dades, després d'argumentar substancialment la importància que té disposar de bases de dades normalitzades, però en alguns casos una desnormalització controlada pot ser molt útil i, fins i tot, desitjable.

La **desnormalització** és la introducció de **redundàncies de forma controlada** en una base de dades, per fer més eficients alguns processos que, altrament, farien que globalment el rendiment del sistema resultés poc bo.

Habitualment, aquests tipus de modificacions es fan **després** d'haver observat certes ineficiències en algunes operacions habituals sobre la BD.

Algunes situacions en què pot ser útil:

- **Consultes intensives amb referències a descripcions.** Pensem en la xarxa social Connecta: cada `GRUP` guarda el `CodiPoblacio` de la població on es crea, i el nom de la població és a la taula `POBLACIO`. Cada cop que es mostri el nom de la població a la llista de grups caldria fer un `join` de les dues taules. Si es manté el **nom de la població** a la mateixa taula `GRUP`, s'evita aquesta operació, que resulta costosa.
- **Afegir camps calculables.** Guardar a `PUBLICACIO` un camp `NombreReaccions` (el recompte de files de `REACCIO` d'aquella publicació) evita haver de comptar les reaccions cada cop que es mostra la publicació. De la mateixa manera, a la comercial, guardar el `Total` d'una factura en lloc de calcular-lo a partir de les línies de factura.

La introducció de redundàncies que desnormalitzen la BD va lligada a la implementació de **mecanismes** que intenten solucionar els problemes que podrien generar. La forma més típica de controlar els canvis per evitar inconsistències en aquests casos és la programació de **triggers**. Així, es pot programar un *trigger* perquè, en inserir o eliminar una `REACCIO`, es recalculi i s'actualitzi `NombreReaccions` de la `PUBLICACIO` corresponent, per tenir aquest camp calculat sempre consistent.

NOTA

Trigger: procediment de BD que s'executa automàticament quan s'esdevenen situacions donades. Per exemple, inserció d'una nova dada en un camp donat, un nou registre en una taula, modificació d'una dada en una taula, etc.

4. Resum

Elements del model relacional

Concepte	Definició breu
Relació	Taula amb un nom únic
Tupla	Fila d'una relació
Atribut	Columna d'una relació
Domini	Conjunt finit de valors indivisibles que pot prendre un atribut
Esquema	Nom + conjunt d'atributs (intenció)
Extensió	Valors de les tuples en un moment donat
Grau	Nombre d'atributs
Cardinalitat	Nombre de tuples
Superclau mínima / clau candidata	Conjunt mínim d'atributs que identifica les tuples
Clau primària	Clau candidata triada; no admet nuls
Clau alternativa	Clau candidata no triada
Clau forana	Atributs que referencien la clau primària d'una altra relació (o de la mateixa)

Regles d'integritat

Regla	Què garanteix
Unicitat de la clau primària	No hi ha dues tuples amb la mateixa clau primària
Entitat de la clau primària	La clau primària no conté mai nuls
Integritat referencial	Tota clau forana apunta a una clau primària existent, o és nul·la
Integritat del domini	Tot valor no nul pertany al domini de l'atribut

Política (esborrament / modificació de la clau referenciada)	Efecte
Restricció	Es prohibeix l'operació
Actualització en cascada	Es propaga el canvi les tuples que hi fan referència
Anul·lació	Les claus foranes de les tuples relacionades passen a nul

Traducció ER → model relacional

Element ER	Resultat
Entitat	Relació (mateixos atributs i clau primària)
Interrelació binària **1-1**	Clau forana a la relació de l'entitat obligatòria
Interrelació binària **1-N**	Clau forana a la relació del costat N
Interrelació binària **M-N**	Nova relació; clau = les dues claus primàries
Interrelació **ternària o n-ària**	Nova relació; la clau depèn de la connectivitat (totes N: totes les claus; amb 1: totes menys una)
Interrelació **recursiva**	Igual que les altres, amb atributs de clau forana de nom diferent
Entitat **feble**	La clau de la forta passa a la clau primària de la feble (i és clau forana)
Generalització / especialització	Una relació per a la superclasse i una per a cada subclasse; clau de la superclasse com a clau primària i forana

Entitat **associativa**	Es tradueix com la interrelació que agrega; clau forana si és al costat N d'una altra interrelació

Formes normals

Forma normal	Condició	Elimina
1FN	Tots els atributs són atòmics	Valors multivalor
2FN	1FN + tot atribut no-clau té dependència funcional **total** de la clau	Dependències parcials
3FN	2FN + cap atribut no-clau depèn **transitivament** de la clau	Dependències transitives
FNBC	Tots els determinants són claus candidates	Determinants que no són clau candidata (pot perdre dependències)
4FN	L'implicant de tota dependència **multivalent** és clau candidata	Dependències multivalents
5FN	Tota dependència de **reunió** és conseqüència de claus candidates	Dependències de reunió

Índex de figures

1 Figura 1. Entitat CLIENT amb els atributs DNI, Nom i Cognoms 12
2 Figura 2. Interrelació binària amb connectivitat 1-1 13
3 Figura 3. Interrelació binària amb connectivitat 1-N 14
4 Figura 4. Interrelació binària amb connectivitat M-N 15
5 Figura 5. Interrelació ternària amb connectivitat M-N-P 16
6 Figura 6. Interrelació ternària amb connectivitat 1-M-N 17
7 Figura 7. Interrelació ternària amb connectivitat 1-1-N 19
8 Figura 8. Interrelació ternària amb connectivitat 1-1-1 20
9 Figura 9. Interrelació recursiva binària amb connectivitat 1-N 21
10 Figura 10. Interrelació recursiva binària amb connectivitat M-N 22
11 Figura 11. Interrelació recursiva n-ària 23
12 Figura 12. Entitat feble GRUP identificada a través de POBLACIO 24
13 Figura 13. Generalització i especialització 26
14 Figura 14. Entitat associativa INTERESSA 27
15 Figura 15. Relació d'inclusió entre les diverses formes normals 29
16 Figura 16. Exemple d'esquema de dependències funcionals 31

Versions d'aquest document

- HTML
- PDF
- ODT
- MD

Domini Públic (CC0)