

Disseny lògic de bases de dades

Xavier Burgués Illa

PID_00220510



Els textos i imatges publicats en aquesta obra estan subjectes –llevat que s'indiqui el contrari– a una llicència de Reconeixement-NoComercial-SenseObraDerivada (BY-NC-ND) v.3.0 Espanya de Creative Commons. Podeu copiar-los, distribuir-los i transmetre'ls públicament sempre que en citeu l'autor i la font (FUOC. Fundació per a la Universitat Oberta de Catalunya), no en feu un ús comercial i no en feu obra derivada. La llicència completa es pot consultar a <http://creativecommons.org/licenses/by-nc-nd/3.0/es/legalcode.ca>

Índex

Introducció.....	5
Objectius.....	6
1. Introducció al disseny lògic.....	7
2. Reconsideració del model conceptual: paranys de disseny.....	8
2.1. Ventall	8
2.2. Tall	9
2.3. Pèrdua d'affiliació	11
2.4. Aritat dels tipus de relació	12
2.5. Semàntica dels tipus d'entitat	13
3. Disseny lògic: transformació del model conceptual al model relacional.....	15
3.1. Conceptes previs del model relacional	16
3.2. Impacte de l'ús dels valors nuls	18
3.3. Tipus d'entitat	20
3.3.1. Atributs multivaluats	21
3.4. Tipus de relació	22
3.4.1. Tipus de relacions binàries amb una multiplicitat 1	22
3.4.2. Tipus de relacions binàries reflexives	24
3.4.3. Tipus de relacions binàries de composició	25
3.4.4. Tipus de relacions <i>n</i> -àries	26
3.5. Tipus d'entitats associatives	28
3.6. Generalitzacions	30
3.7. Restriccions	32
3.7.1. Multiplicitats	34
3.7.2. Generalitzacions	36
3.7.3. Abraçades mortals	37
3.8. Reconsideracions	38
4. Normalització.....	41
4.1. Anomalies	41
4.2. Conceptes previs	44
4.3. Teoria de la normalització	48
4.3.1. Primera forma normal	49
4.3.2. Segona forma normal	50
4.3.3. Tercera forma normal	51
4.3.4. Forma normal de Boyce-Codd	52
4.3.5. Regles d'Armstrong	53

4.3.6.	Algorisme d'anàlisi	55
4.3.7.	Algorisme de síntesi	55
4.3.8.	Quarta forma normal	56
4.3.9.	Cinquena forma normal	59
4.4.	Pràctica de la normalització	62
Resum		64
Glossari		65
Bibliografia		66

Introducció

El procés de desenvolupament d'una base de dades per a un sistema d'informació és un procés seqüencial, format per un conjunt de diferents fases que, de manera progressiva, ens acosten al resultat final. Després de l'etapa de disseny conceptual, que permet obtenir una especificació que inclou l'esquema conceptual a partir dels requisits i una anàlisi acurada del domini de l'aplicació, hem de dur a terme el disseny lògic.

Començarem l'estudi d'aquest mòdul amb algunes reflexions sobre possibles errors que es poden haver comès en el disseny conceptual i que convé repassar abans de prendre'l com a punt de partida del disseny lògic.

A continuació, ens centrarem en l'etapa de disseny lògic i estudiarem les pautes per dur-la a terme. El disseny lògic té per objectiu obtenir un esquema lògic de la base de dades a partir de l'esquema conceptual que hem obtingut en la fase anterior. Es pot considerar, doncs, l'etapa que permet transformar un model conceptual en un model lògic de la futura base de dades.

En aquest material utilitzarem models conceptuais basats en el llenguatge UML i els transformarem en models lògics relacionals, anomenats simplement *models relacionals*.

En aquest mòdul també veurem la teoria de la normalització, que permet assegurar que l'esquema relacional satisfà una sèrie de condicions que garanteixen una millor qualitat de la base de dades.

Objectius

El contingut d'aquests materials didàctics us permetrà assolir els objectius següents:

1. Entendre el disseny lògic com a activitat de transformació.
2. Conèixer els paranys de disseny i identificar les situacions en què es poden produir.
3. Conèixer els efectes negatius que pot comportar l'existència de valors nuls.
4. Conèixer les alternatives de transformació dels elements del model conceptual en els diferents elements del model relacional.
5. Conèixer i aplicar els diferents mecanismes de definició de restriccions en el model relacional.
6. Conèixer les anomalies que es poden produir en un esquema no normalitzat.
7. Conèixer les formes normals fins a la cinquena, i ser capaços d'aplicar-les a un esquema donat.

1. Introducció al disseny lògic

El disseny lògic és una etapa intermèdia de les que componen el procés de desenvolupament d'una base de dades. Per tant, es parteix dels resultats d'una etapa prèvia (l'etapa del disseny conceptual) i se'n produeixen de nous que, al seu torn, serviran com a punt d'entrada d'una etapa posterior (el disseny físic).

Si parlem de desenvolupament de programari en general, l'etapa de disseny lògic parteix de les especificacions del sistema per a dissenyar una solució independent de la tecnologia que, després, es refinarà i s'implementarà en etapes posteriors del desenvolupament. Si ens centrem en la part de dades del disseny lògic, partirem de la part de l'especificació que correspon a la modelització conceptual del domini de l'aplicació, per a obtenir un esquema de la base de dades expressat en un llenguatge corresponent a algun model lògic de base de dades, però sense adoptar una versió concreta de cap sistema de gestió de base de dades (SGBD) ni entrar en detalls d'optimització o afinament de la base de dades, que es deixaran per a les etapes posteriors de desenvolupament.

Hi ha diverses opcions per al llenguatge de modelització conceptual en què estarà expressada la informació d'entrada al disseny lògic i per al model lògic que farem servir per a expressar la solució resultant d'aquest disseny. En aquest material utilitzarem models conceptuais basats en el llenguatge UML i els transformarem en models lògics relacionals.

Començarem l'estudi d'aquest mòdul amb el tractament dels anomenats *paranys de disseny*, que són errors que es poden haver comès en fer el disseny conceptual i dels quals cal assegurar-se que el model conceptual està lliure abans de prendre'l com a punt de partida per a iniciar el disseny lògic.

A continuació, ens centrarem en l'etapa de disseny lògic pròpiament dita. Analitzarem un diagrama de classes d'UML i veurem quines són les diferents alternatives de transformació de cadascun dels elements del diagrama en elements del model relacional. També estudiarem els avantatges i inconvenients de cada alternativa per tal de poder escollir de forma correcta, en cas de disposar de més d'una alternativa.

En aquest mòdul també veurem la teoria de la normalització. Entre altres aplicacions, aquesta teoria permet posar un segell de qualitat al disseny obtingut des del punt de vista de l'absència de redundància, facilitat de manteniment de la consistència i eficiència en les operacions d'actualització de la base de dades derivada del disseny lògic. A més, en el cas que el disseny no satisfaci les propietats requerides, la teoria de la normalització inclou procediments de correcció del disseny.

2. Reconsideració del model conceptual: paranys de disseny

En aquest apartat veurem alguns errors que es poden cometre durant la realització del disseny conceptual de la base de dades. Aquests errors estan relacionats amb conceptualitzacions errònies de la realitat que s'han incorporat en el model conceptual.

Precisament per la natura del disseny conceptual, que implica interpretar i modelitzar conceptes del món real, sorgeix la possibilitat de cometre errors depenent de si la interpretació que fem és del tot correcta o no. És important evitar aquests errors, ja que poden provocar greus problemes en fases posteriors del desenvolupament de la base de dades.

Reflexió

Hi ha diferents tipus de diagrames per a representar els models conceptuais, però en aquest text emprarem els diagrames de classes del llenguatge UML.

Alguns d'aquests errors són recurrents i segueixen uns patrons que ens permeten identificar-los; són els que anomenem **paranys de disseny**. Tot seguit en presentarem cinc, ja que si els coneixem ens serà més senzill detectar-los i evitar de caure-hi.

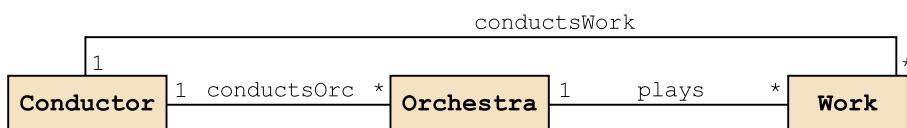
2.1. Ventall

El ventall és el primer parany que se'ns pot presentar, quan tenim tres tipus d'entitats relacionades per tres tipus de relacions binàries (1..*).

Diem que hem caigut en el **parany del ventall** quan un dels tipus de relació és derivat dels altres dos i, en no voler incorporar el derivat en l'esquema conceptual, ens equivoquem i n'eliminem un dels que no ho és.

Il·lustrem-ho amb un exemple. Imaginem que es tracta de modelitzar els enregistraments que fa una discogràfica de música clàssica. En la producció d'aquesta companyia, cada obra (*work*) és interpretada (*play*) per una orquestra (*orchestra*) un sol cop i cada orquestra és dirigida sempre per un director (*conductor*). La situació es pot modelitzar com s'indica en la figura 1.

Figura 1. Tres tipus de relacions binàries, un dels quals derivat



Aquest model, tot i que conté els elements (tipus d'entitat i tipus de relacions) que cal, no és del tot correcte, perquè hi falta una restricció: si una orquestra interpreta alguna obra i un director dirigeix aquella orquestra, el director dirigeix l'obra. És a dir, el tipus de relació *conductsWork* és derivat a partir dels altres dos.

Si decidim representar el domini amb l'esquema de la figura 2 haurem caigut en el parany, ja que haurem tret el tipus de relació *plays* en lloc d'eliminar *conductsWork*.

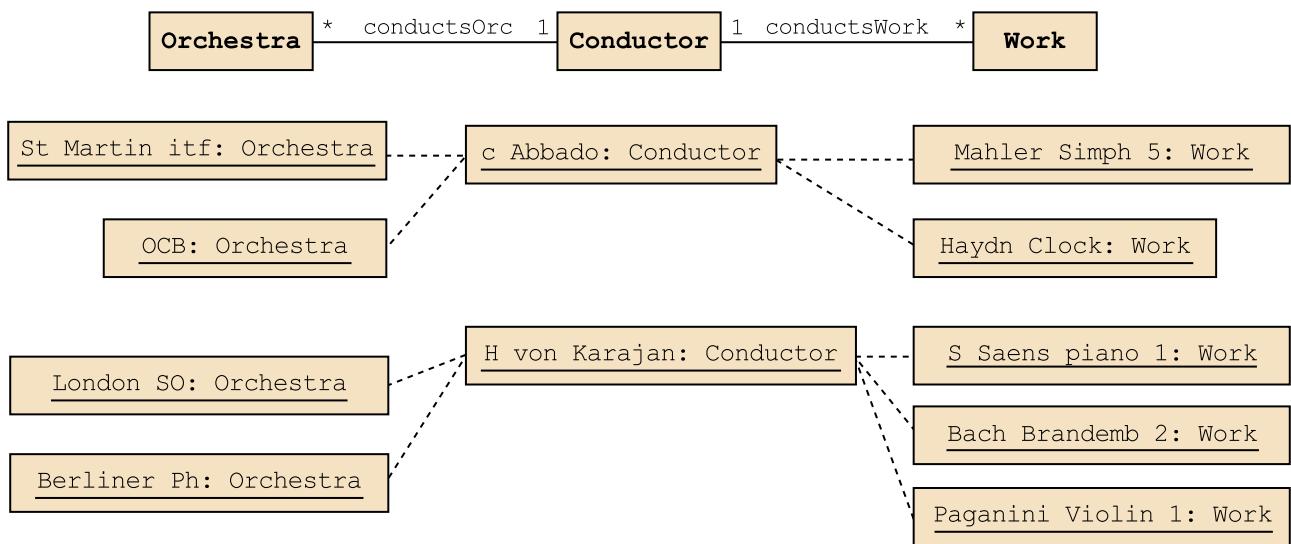
Reflexió

Quan hi ha elements derivats, podem decidir eliminar-los del diagrama de classes si pensem que d'aquesta manera queda més clar i l'element derivat no és d'interès per a la base de dades que dissenyem.

També pot passar que caiguem en el parany directament, sense ser conscients que incloem un tipus de relació derivat i, en canvi, no incloguem en el model el tipus de relació a partir del qual es pot derivar la nova informació.

En caure en el parany, deixem de representar informació rellevant. En l'exemple, no sabem quina orquestra ha interpretat cada obra. El nom *ventall* té l'origen en la forma que resulta de representar les instàncies i els lligams entre elles, com es mostra en la figura 2.

Figura 2. El parany del ventall



El model conceptual correcte, sense representar el tipus de relació derivat, es pot veure en la figura 3.

Figura 3. Esquema correcte de la figura 1



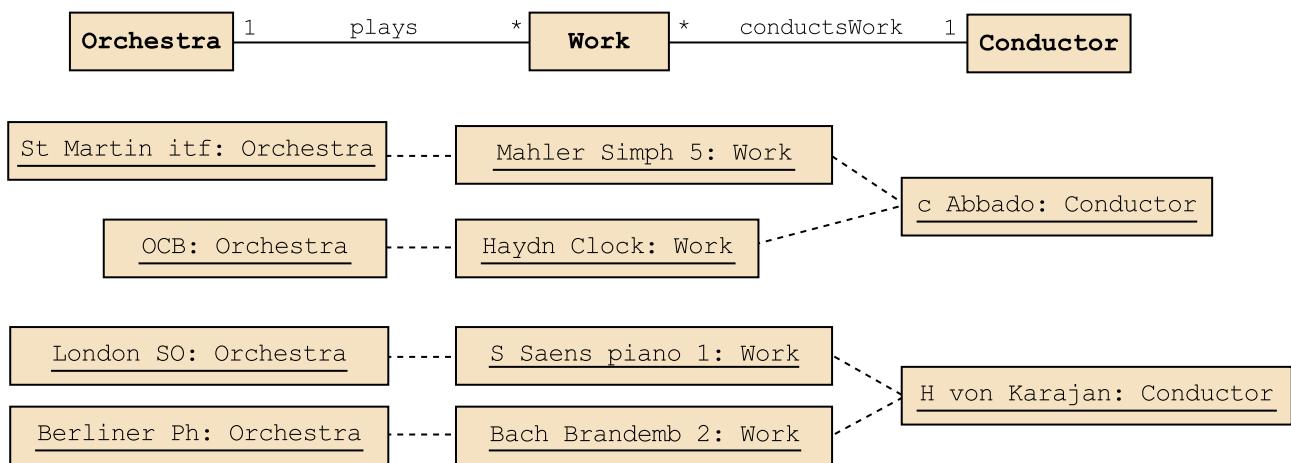
2.2. Tall

Aquest altre parany es presenta en la mateixa situació que el del ventall: quan tres tipus d'entitat estan relacionats per tres tipus de relacions binàries (**1..***), un dels quals és derivat dels altres dos.

Diem que hem caigut en el **parany del tall** quan, en no voler incorporar el tipus de relació derivat en l'esquema conceptual, ens equivoquem i n'eliminem un dels que no ho és. En aquest cas, la conseqüència és que la informació sobre la relació entre objectes de les entitats que no han quedat directament relacionades queda supeditada a l'existència d'algún objecte de la tercera entitat per a fer de pont.

Continuant amb l'exemple anterior, podem entendre més bé la problemàtica del parany de tall i entendrem per què aquest parany ens hauria dut al model conceptual de la figura 4.

Figura 4. El parany del tall



El model de la figura 4 torna a ser incorrecte, perquè recull un tipus de relació derivat (*conductsWork*) i s'oblida d'un que no ho és (*conductsOrc*).

Veient les instàncies que acompañen el model conceptual de la figura 4, no es veuen les conseqüències que té caure en aquest parany. Malgrat això, és molt fàcil veure que aquest model no és correcte si pensem:

- 1) com podem indicar qui és el director d'una orquestra que encara no ha interpretat cap obra, o
- 2) que si eliminem una obra, perdem dades del director de l'orquestra que la va interpretar.

D'aquí ve el nom d'aquest parany: eliminar una obra pot "tallar" la connexió entre una orquestra i el seu director.

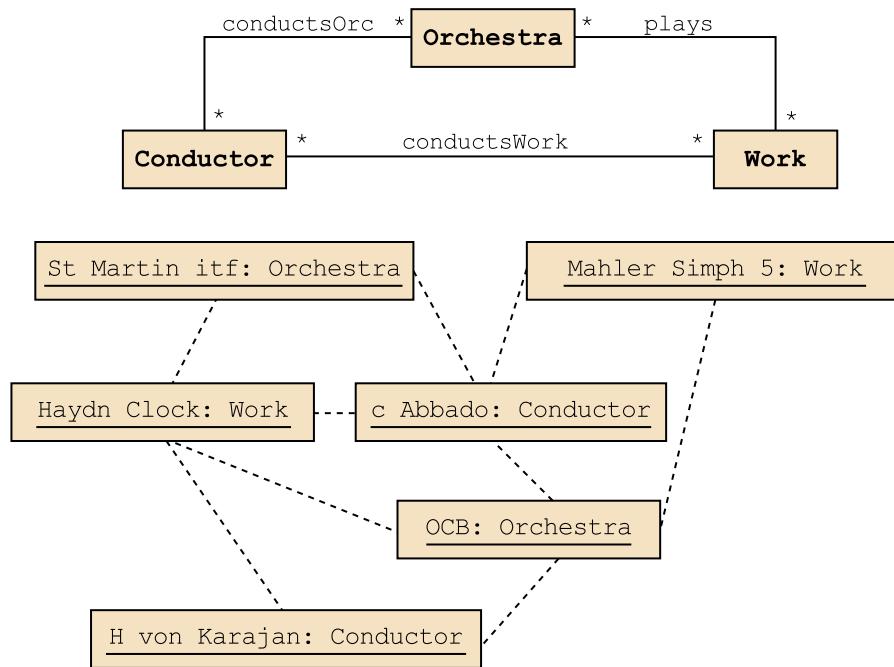
2.3. Pèrdua d'affiliació

El tercer parany que podem detectar és el que es coneix com a *pèrdua d'affiliació*. Es pot presentar quan hi ha un tipus de relació ternària i tres tipus de relacions binàries entre els tipus d'entitat que el constitueixen.

Diem que hem caigut en el **parany de pèrdua d'affiliació** quan en comptes de representar un tipus de relació ternària, representem els tipus de relacions binàries que se'n deriven.

Reprenem l'exemple dels casos anteriors, però ara suposem que una mateixa orquestra pot ser dirigida per diversos directors i una mateixa obra pot ser interpretada per diverses orquestres. Aquesta situació ens pot dur a un model com el de la figura 5.

Figura 5. El parany de la pèrdua d'affiliació



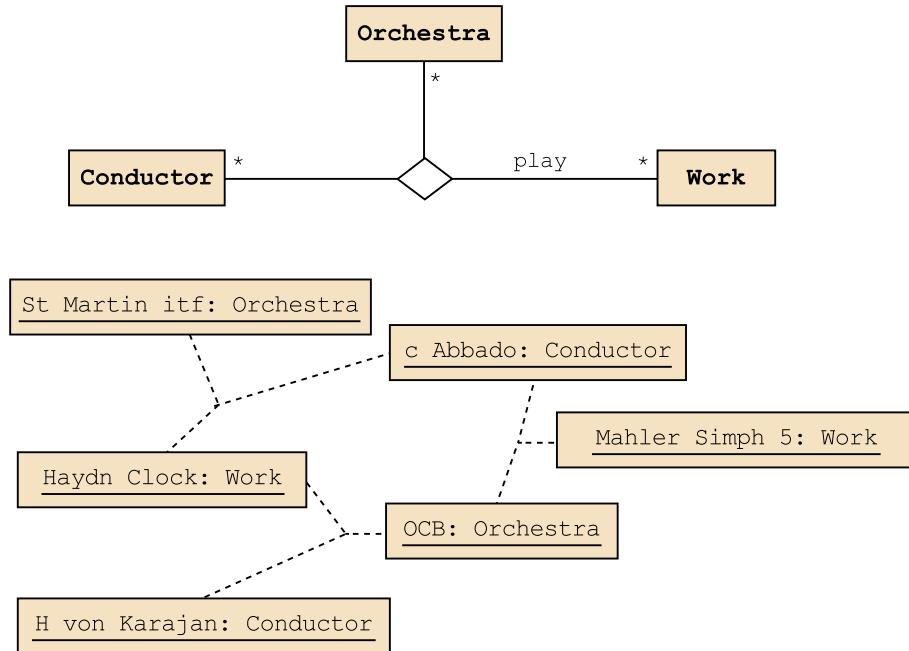
Si suposem que en el món real modelitzem el fet que C. Abbado ha dirigit l'orquestra de St. Martin in the Fields interpretant la Simfonia del Rellotge de Haydn, que H. von Karajan ha dirigit l'OCB interpretant la Simfonia del Rellotge i que C. Abbado ha dirigit l'OCB interpretant la cinquena simfonia de Mahler, les instàncies seran les que es veuen a la figura 5. Però a la vista de les instàncies, podríem pensar que Claudio Abbado ha dirigit l'OCB interpretant la Simfonia del Rellotge, cosa que no és certa. És a dir, l'ús de múltiples tipus de relacions binàries en lloc d'una de ternària pot provocar confusions. Hem perdut part de la informació que teníem, ja que sabem quins directors han dirigit una orquestra, però no quines de les obres interpretades per l'orquestra han estat dirigides per cada director.

Reflexió

Apliquem la mateixa denominació al cas en què pretenem representar aquesta situació amb dos dels tipus de relacions binàries. Per exemple, si només modelitzem els tipus de relació *conductsWork* i *plays*.

El model conceptual correcte que s'ajusta a la situació plantejada correspon a un model amb un tipus de relació ternària, tal com es descriu en la figura 6.

Figura 6. Esquema conceptual correcte



2.4. Aritat dels tipus de relació

En el cas anterior hem vist que el fet d'errar en el nombre de tipus d'entitat que participen en un tipus de relació pot ser una font d'errors en els esquemes conceptuais.

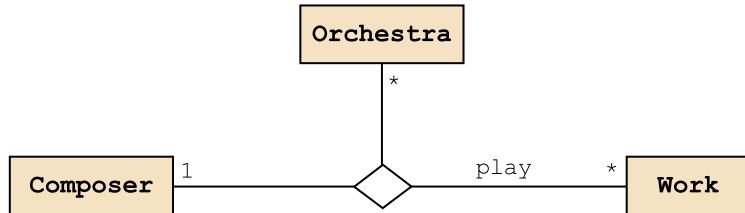
En aquest punt veurem que incorporar un tipus d'entitat a un tipus de relació que no li correspon també pot ser font d'errors en el disseny conceptual.

El **parany d'aritat dels tipus de relació** es pot presentar quan hi ha un tipus de relació de grau més gran que 2 que té alguna multiplicitat de valor 1. En aquest cas, pot ser que l'entitat que es troba en el costat amb multiplicitat igual a 1, en realitat no hagi de formar part del tipus de relació. Diem que hem caigut en aquest parany quan hem inclòs l'entitat de forma errònia.

Així, doncs, quan ens apareix una multiplicitat 1 en un tipus de relació ternària (o de grau més alt) hem de tenir molta cura i verificar si el grau del tipus de relació és tal com el plantegem o si en realitat es tracta de dos o més tipus de relació de grau més baix.

Pensem ara en un context en què es vol modelitzar el fet que les obres són interpretades per orquestres i que cada obra té un compositor, amb la restricció que cada obra té un únic compositor. Un possible model conceptual per a aquesta descripció es mostra en la figura 7.

Figura 7. El parany de l'aritat dels tipus de relació



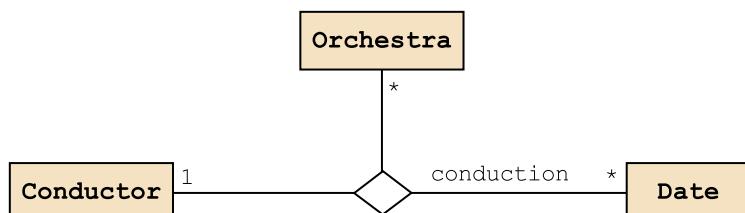
Aquest model és erroni perquè el compositor d'una obra no depèn de les orquestres que la interpreten. És a dir, hi ha una relació directa entre l'obra i el compositor i una altra entre l'orquestra i l'obra. El model correcte seria el que es mostra en la figura 8

Figura 8. L'esquema correcte



Això no vol dir que no hi pugui haver algun tipus de relació de grau més gran que 2 i amb alguna multiplicitat igual a 1. Com a exemple correcte de tipus de relació ternària on un dels tipus d'entitat participa amb multiplicitat igual a 1, considerem el de la figura 9, on donem per fet que en un moment donat una orquestra només té un director, tot i que permetem que un director es faci càrrec de més d'una orquestra simultàniament.

Figura 9. Un tipus de relació ternària correcte amb una multiplicitat igual a 1



2.5. Semàntica dels tipus d'entitat

El parany conegut com a *parany de semàntica del tipus d'entitat* és el darrer parany que presentem i que caldrà evitar. Tal com indica el seu nom, està relacionat amb la interpretació que es faci de la semàntica d'un tipus d'entitat.

Diem que caiem en el **parany de semàntica de tipus d'entitat** quan s'assignen a un tipus d'entitat atributs que no li corresponen, a causa d'una interpretació incorrecta del significat del tipus d'entitat.

Una manera de confirmar o descartar que un atribut es troba en un tipus d'entitat que li correspon és no perdre de vista la semàntica del tipus d'entitat i comprovar si l'atribut és coherent amb aquesta semàntica.

Suposem el model conceptual que es presenta en la figura 10, en la qual definim l'atribut *minutes* en el tipus d'entitat *Work*.

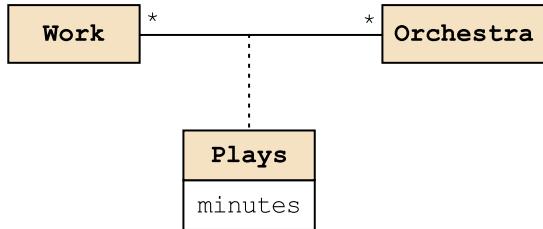
Figura 10. El parany de la semàntica dels tipus d'entitat



Si hi reflexionem una mica, però, podrem conoure que els minuts de durada no corresponen tant a l'obra, sinó a cadascuna de les interpretacions que se'n fan. Per tant, l'atribut *minutes* hauria de ser un atribut del tipus de relació *plays* i no pas del tipus d'entitat *Work*. Des del punt de vista semàntic, fixem-nos que una obra és un concepte abstracte que hem confós amb un concepte més físic, com és la interpretació de l'obra.

L'esquema conceptual correcte seria el que es presenta en la figura 11.

Figura 11. L'esquema correcte



3. Disseny lògic: transformació del model conceptual al model relacional

L'etapa de disseny lògic consisteix a obtenir un esquema lògic a partir de l'esquema conceptual generat en l'etapa anterior. L'esquema lògic depèn del tipus de base de dades que hagim escollit, però serà independent de la implementació concreta del sistema de gestió de bases de dades (SGBD¹).

Com ja hem dit, en aquest mòdul ens centrarem en el cas de convertir un esquema conceptual expressat en llenguatge UML en un esquema lògic per a un tipus de base de dades relacional. En aquests casos, el model lògic rep el nom de *model lògic relacional* o, simplement, *model relacional*.

Farem servir la paraula *relació* per a referir-nos a l'element bàsic del model relacional. Cal no confondre's, doncs, amb les relacions a què ens hem referit en tractar el disseny conceptual, que eren instàncies de tipus de relació. Val a dir que el context ens hauria d'ajudar a distingir ambdós conceptes de manera unívoca.

Actualment, existeixen eines CASE² que són capaces de fer aquest procés de traducció de manera automàtica. Ara bé, atès que hi ha situacions en què part d'un model es pot traduir de diverses formes, els usuaris d'aquestes eines han de tenir els coneixements necessaris per tal de fer la traducció manualment, perquè, com a usuaris d'una eina CASE, han de ser capaços de modificar el resultat de la traducció automàtica si l'eina no ha triat l'alternativa que s'ajusta millor a les condicions de cada cas concret. També cal tenir aquests coneixements per a poder decidir correctament si l'eina, en trobar-se amb alternatives que no pot resoldre, demana la intervenció de l'usuari.

Les eines CASE

Una eina CASE és la implementació d'un conjunt d'eines i de mètodes per al desenvolupament de programari a partir d'especificacions i definicions d'alt nivell. Un dels objectius principals és alliberar els dissenyadors de la base de dades dels processos rutinaris, amb possibilitat d'automatització, no solament per a aconseguir una producció més eficient, sinó també per a evitar alguns dels errors que es poden cometre en aquests processos de desenvolupament. En l'àrea de la modelització conceptual i les bases de dades, algunes de les eines més conegudes actualment són VisualParadigm, ArgoUML, Poseidon o MagicDraw. Algunes eines CASE poden efectuar enginyeria inversa, és a dir, poden ajudar a trobar l'esquema conceptual que correspon a una base de dades ja dissenyada i implementada.

A continuació, tot i que es pressuposa que coneixeu el model relacional i teniu coneixements bàsics de SQL, es fa esment dels conceptes més bàsics del model relacional i de com es representen en llenguatge SQL, en el cas dels elements que cal considerar per a fer la transformació de model conceptual al model lògic.

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

Reflexió

Atès que en aquest mòdul tractarem únicament el cas de models lògics per a bases de dades relacionals, quan parlem de *model lògic* interpretarem que es tracta d'un model relacional.

⁽²⁾CASE és la sigla de l'expressió anglesa *computer-aided software engineering*, 'enginyeria de programari assistida per ordinador'.

3.1. Conceptes previs del model relacional

El model relacional representa la informació sobre la base d'un conjunt de relacions. Una **relació** es defineix com un conjunt d'**atributs**, cadascun amb un **domini** concret (el domini és el conjunt de valors que es poden assignar a l'atribut), i un d'ells és la clau primària. A aquesta descripció se l'anomena **esquema d'una relació**.

Al conjunt de tots els esquemes de relació que descriuen la base de dades se l'anomena **esquema de la base de dades**.

Donat l'esquema de la relació, podem crear **tuples** que conformen l'extensió de la relació; cada tupla dóna un valor del domini corresponent a cada atribut. Per exemple, podem definir una relació *Obra* que tingui un esquema amb dos atributs: un atribut *nom*, el domini del qual són les cadenes de caràcters, i un atribut *anyComp*, el domini del qual són els números. Per exemple, l'extensió de la relació *Obra* pot contenir els tuples <"Concert per a piano núm 3 de Mozart", 1779> i <"Parsifal", 1857>.

El model també permet expressar restriccions que limiten els valors o combinacions de valors que poden prendre els atributs. Les més importants són:

- **Clau candidata.** Un atribut o grup d'atributs formen una clau candidata de la relació quan no hi pot haver dues tuples amb el mateix valor en aquell atribut o grup d'atributs. A més, no és possible assignar valor nul a aquests atributs.

Reflexió

Una clau candidata serveix, doncs, per a identificar cada-cuna de les tuples de forma unívoca dins d'una relació.

Exemple

En una relació d'orquesters on cada tupla conté les dades d'una orquestra, l'atribut que correspon al número d'identificació fiscal es pot declarar com a clau candidata.

- **Clau primària.** D'entre les claus candidates, se'n tria una que serà la que farem servir habitualment per a identificar de forma unívoca una tupla de la relació. En llenguatge SQL es fa servir la restricció *PRIMARY KEY* per a especificar la clau primària d'una relació.
- **Clau alternativa.** Cadascuna de les claus candidates que no és la clau primària rep el nom de *clau alternativa*. En llenguatge SQL es fa servir la clàusula *UNIQUE* per a especificar una clau alternativa en una relació.

Exemple

Seguint amb l'exemple anterior sobre la relació d'orquesters, si afegim un atribut amb el nom i cada orquestra té un nom diferent, aquest nom també pot ser clau candidata i, com que l'altra és la primària, aquesta seria una clau alternativa.

- **Clau forana.** Es pot especificar que un atribut o conjunt d'atributs d'una relació *R1* formen una clau forana que referencia una relació *R2* de l'esquema mitjançant una clau candidata de *R2*. Aquesta clau candidata

de $R2$ haurà d'estar formada per un conjunt d'atributs que es corresponen un a un amb els de la clau forana de $R1$. La declaració de clau forana implica que per a cada tupla t de $R1$, hi ha d'haver una tupla de $R2$ que té en els atributs de la clau candidata referenciada els mateixos valors que té la tupla t en els atributs de la clau forana. Alternativament, t pot tenir valors nuls en els atributs de la clau forana. En llenguatge SQL, la restricció *FOREIGN KEY* permet especificar la clau forana d'una relació.

Exemple

Si hem definit les relacions *Obra* (amb un atribut *nomObra* i un atribut *comp*) i *Composer* (amb un atribut *nomComp* -que és clau primària- i un atribut *anyNaix*) i diem que l'atribut *comp* d'*Obra* és clau forana que referencia *Composer* per mitjà de *nomComp*, per a cada obra que no tingui el valor nul a *comp*, ha d'existir un compositor amb aquest valor a l'atribut *nomComp*.

- **Valors nuls.** Diem que no admeten valors nuls tots aquells atributs que sempre han d'estar informats. En llenguatge SQL, un atribut d'aquest tipus s'especifica mitjançant la restricció *NOT NULL* aplicada a la columna en qüestió.
- **Comprovació d'una condició.** És una restricció que verifica que el valor d'un o més atributs satisfà una expressió booleana que s'especifica en la declaració de la restricció. En llenguatge SQL fem servir la restricció *CHECK* seguida de l'expressió que cal satisfer.

Exemple

Si la relació d'orquesters que hem fet servir té un atribut numèric anomenat *nombreMúsics* i volem que totes les orquestres presents en la relació tinguin 30 músics o més, establirem la restricció *Check(nombreMúsics >= 30)*.

Per claredat i concisió, expressarem un esquema del model relacional mitjançant una notació simplificada del llenguatge SQL estàndard.

L'estàndard SQL

L'estàndard SQL és un llenguatge que permet treballar amb bases de dades relacionals. És molt utilitzat i gairebé totes les aplicacions desenvolupades sobre bases de dades relacionals el fan servir. Els SGBD relacionals que s'escullen habitualment per a emmagatzemar la informació l'implementen, de vegades amb petites diferències. Conté sentències tant per a definir bases de dades com per a fer-hi actualitzacions i consultes. Ha tingut tres versions principals que han aparegut, respectivament, en els anys 1989, 1992 i 1999. El llenguatge continua evolucionant i s'hi incorporen nous elements en successives revisions, la darrera publicada l'any 2011.

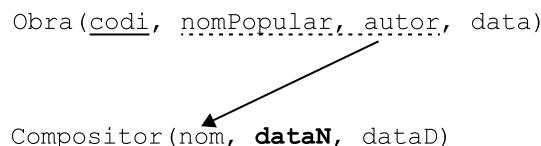
A continuació es defineix breument la notació utilitzada en aquests materials:

- Denotarem les relacions a partir del nom, seguit de la llista d'atributs entre parèntesis i separats per comes.
- Denotarem les claus primàries subratllant amb línia contínua els atributs que les formen.

- Denotarem les claus alternatives subratllant amb línia discontinua els atributs que les formen.
- Denotarem les claus foranes com fletxes que tenen l'origen en el conjunt d'atributs que les formen i destinació en el conjunt d'atributs que formen la clau referenciada. Aquesta notació és diferent de la notació utilitzada en altres textos sobre disseny de bases de dades, però és interessant, ja que mostra gràficament i de manera més entenedora la transformació de l'esquema conceptual en el model relacional. En cas de tenir un gran nombre de relacions, pot resultar confusa, perquè pot implicar un gran nombre de fletxes que es creuen entre si. En aquestes ocasions serà millor fer servir una notació textual que indiqui, en cada relació, quines claus foranes conté i quina relació referencia cada clau forana.
- Farem servir el tipus de lletra **negreta** en els noms d'atribut que volem declarar NOT NULL.

Expressió d'un model amb dues relacions

A continuació mostrem com expressar un model amb dues relacions, una d'obres i una de compositors, on les obres s'identifiquen mitjançant un codi de catalogació i contenen un nom popular que es pot repetir en compositors diferents, però no en obres d'un mateix compositor. Els compositors s'identifiquen pel seu nom i és obligatori assignar valor a l'atribut de data de naixement (*dataN*).



3.2. Impacte de l'ús dels valors nuls

Per començar, farem una breu reflexió sobre els valors nuls, que es van incorporar al model relacional des d'un primer moment per a poder expressar el fet que una dada és desconeguda o que no és aplicable.

És important conèixer l'impacte que pot tenir l'existència de **valors nuls**, perquè en fer la traducció de l'esquema conceptual a l'esquema relacional, poden sorgir atributs que els tipus d'entitat de l'esquema conceptual no tenien i que admeten valors nuls.

Un mal ús dels valors nuls pot causar problemes, bàsicament de dos tipus: en l'eficiència i en la construcció de consultes que manipulen atributs que contenen valors nuls.

Reflexió

Recordem que els valors nuls constitueixen un mecanisme que soluciona el problema de representació de dades desconegudes o que no es poden aplicar. En general, no podem evitar l'existència de valors nuls.

El problema d'eficiència es deriva de l'accés a files que contenen columnes amb valors nuls, que es podrien haver evitat amb un disseny diferent de la base de dades. L'exemple següent ho il·lustra:

```
Obra(codi, autor, nom, nTenors, nBaixos,
      nContralts, nSopranos, nBarítons)
```

```
SELECT * FROM Obra WHERE nTenors >= 1
```

La relació *Obra* ens diu, per a cada obra, quants tenors, baixos, contralts, sopranos i barítons calen per a interpretar-la (les columnes amb un nom que comença per *n* tenen aquesta informació). Si considerem que hi haurà obres sense intervenció de veu, aquestes obres tindran valors nuls en les columnes que comencen per *n*, ja que aquesta informació no es pot aplicar a obres instrumentals. La consulta de l'exemple, que pretén localitzar les obres on hi ha un o més tenors, haurà d'accedir a totes les obres instrumentals, i lògicament no retornarà cap d'aquestes tuples com a resultat de la consulta. Podem dissenyar un model relacional que eviti l'ús de valors nuls i també l'accés innecessari a tuples. Per exemple, es pot fer separant les obres en dues relacions: una d'obres amb intervenció de veus i una altra d'obres sense intervenció de veus. Amb aquesta idea obtenim el model relacional següent:

```
ObraInstr(codi, autor, nom)
ObraCoral(codi, autor, nom, nTenors, nBaixos,
           nContralts, nSopranos, nBarítons)
```

Amb aquest model relacional, ara farem la consulta d'aquesta manera:

```
SELECT * FROM ObraCoral WHERE nTenors >= 1
```

La diferència d'accessos entre una alternativa i l'altra dependrà de la proporció d'obres corals respecte al nombre total d'obres. Com menys obres corals (i, per tant, més tuples amb valors nuls en la primera alternativa), més diferència hi haurà.

Així, doncs, quan hi ha columnes per a les quals una proporció gran de les files pot tenir valor nul, cal analitzar si les consultes poden ser penalitzades i, si escau, cal canviar el disseny per a eliminar la presència de valors nuls.

Pel que fa a la correcció de les consultes que involucren atributs que poden prendre valor nul, hem d'estar atents per a assegurar-nos que la consulta retorna el resultat correcte, tant en el cas d'existir tuples amb valor nul com en el cas contrari. Considerem el cas següent, on tant la relació de directors com la de compositors té una columna que indica de quin país és el director

o compositor. Si volem obtenir els directors que són d'un país on no hi ha nascut cap compositor, podríem pensar en les dues consultes que es mostren a continuació:

```
Director(nom, país)
Compositor(nom, anyNaix, anyDef, país)
```

```
SELECT * FROM Director
WHERE pais NOT IN (SELECT pais FROM Compositor)

SELECT * FROM Director d
WHERE NOT EXISTS
(SELECT * FROM Compositor c WHERE d.pais = c.pais)
```

Les dues consultes retornen el mateix resultat sempre que la columna *pais* no tingui valors nuls. Fixeu-vos que si un director té el valor nul a la columna *pais*, la primera consulta no l'incorpora en el resultat i en canvi la segona sí.

Aquest és només un exemple per a il·lustrar l'impacte que pot tenir l'existència de valors nuls en les consultes. Caldrà, doncs, tenir present aquest fet en totes les consultes que involucren els valors nuls en la condició de selecció, atributs d'agrupació (*GROUP BY*), funcions d'agregació (*MAX*, *SUM*, ...), etc.

Després dels preliminars, passem a veure com podem transformar cadascun dels elements del model conceptual, expressats en llenguatge UML, al model lògic relacional.

3.3. Tipus d'entitat

El primer element del model conceptual que transformarem serà el concepte de *tipus d'entitat*.

El **tipus d'entitat** de l'esquema conceptual es transforma, en general, en una relació del model relacional. Cada **atribut** del tipus d'entitat esdevindrà una columna de la relació.

Suposem el tipus d'entitat de la figura 12:

Figura 12. Un tipus d'entitat

Composer
name <P>: String
residCity: String
bYear: Integer
dYear: Integer

Veiem que aquest tipus d'entitat té els atributs nom (*name*), que és clau primària, ciutat de residència (*residCity*), que és una cadena de caràcters, any de naixement (*bYear*) i any de defunció (*dYear*), que són enters. Aquest tipus d'entitat es representa en el model lògic com la relació que es mostra a continuació:

```
Composer(name, residCity, bYear, dYear)
```

3.3.1. Atributs multivaluats

La transformació dels atributs multivaluats requereix una anàlisi més detallada. El model relacional no suporta directament aquesta possibilitat, però hi ha dues maneres de representar atributs multivaluats:

- 1) **Per columnes.** La representació per columnes consisteix a definir en l'esquema relacional tantes columnes com el nombre màxim de valors que pugui prendre l'atribut multivaluat de l'esquema conceptual. Aquesta representació requereix conèixer el nombre màxim de valors, informació que no sempre està definida.
- 2) **Per files.** Aquesta segona representació consisteix a representar cada valor d'un atribut multivaluat com una fila o tupla d'una nova relació en l'esquema relacional.

Exemple d'atribut multivaluat

Suposem, per exemple, que el tipus d'entitat *Composer* ara té, en comptes de l'atribut *residCity*, un atribut *residCities* multivaluat, tal com es mostra en la figura 13.

Figura 13. Un tipus d'entitat amb un atribut multivaluat

Composer
name <P>: String residCities: String [*] bYear: Integer dYear: Integer

Suposant que cada compositor pot tenir com a màxim cinc ciutats de residència, podem representar aquest atribut multivaluat mitjançant dos models lògics possibles:

- 1) Per columnes:

```
Composer(name, bYear, dYear, city1, city2,  
city3, city4, city5)
```

- 2) Per files:

```
Composer(name, bYear, dYear)
```

↑
CitiesComp(comp, n, city)

Fixem-nos que, com hem dit, per a fer la representació per columnes ens cal saber el nombre màxim de ciutats en què pot haver viscut un compositor, i que per a cada compositor deixarem amb valor nul els atributs de ciutat que no calguin. Si coneixem aquest

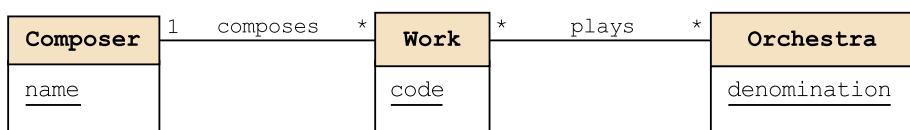
nombre màxim i la majoria de compositors deixen pocs o cap valor nul, aquesta pot ser una bona representació. Si no és el cas, probablement serà millor la representació per files. També cal tenir present que en aquesta segona representació caldrà efectuar operacions de combinació (*Join*) per a recuperar les ciutats d'un compositor, cosa que no caldrà fer si s'utilitza la primera representació.

3.4. Tipus de relació

Per a fer la transformació dels tipus de relacions al model relacional cal fixar-se en l'aritat, o grau, i en les multiplicitats, o connectivitat. Quant al grau, distingirem entre grau 2 o més i pel que fa a la connectivitat distingirem si el mínim és 0 o més i si el màxim és 1 o més. També estudiarem els casos particulars de tipus de relació reflexives i de composicions.

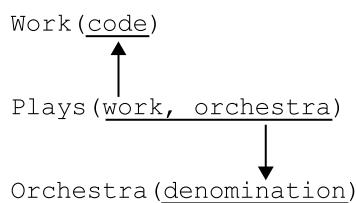
Tot tipus de relació es pot representar amb una nova relació, que té com a clau primària la concatenació de claus primàries de les relacions que representen els tipus d'entitat que participen en el tipus de relació. A més, aquesta nova relació tindrà una clau forana per cada tipus d'entitat relacionat.

Figura 14. Un esquema amb tipus de relacions binàries



Exemple de tipus de relació

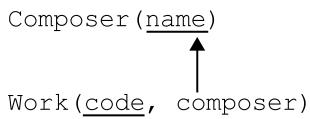
La relació "interpreta" (*plays*) de la figura 14 es pot transformar al model relacional de la manera següent:



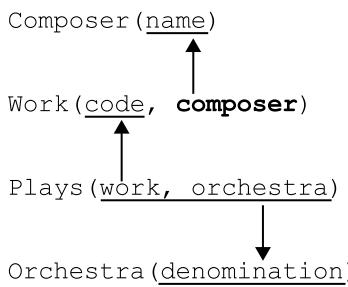
3.4.1. Tipus de relacions binàries amb una multiplicitat 1

Els **tipus de relacions binàries 1..1 o 1..*** es poden representar mitjançant una clau forana en l'extrem oposat al que té la multiplicitat màxima igual a 1.

Així, el tipus de relació *composes* de la figura 14 esdevé:



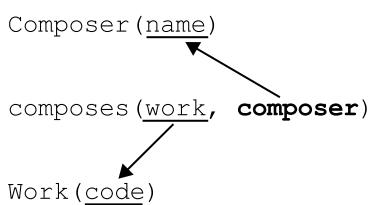
Si, a més, tenim en compte que tal com especifica l'esquema conceptual, tota obra té un compositor, aleshores caldrà afegir la restricció *NOT NULL* sobre la columna *composer*. Per tant, el model conceptual de la figura 14 es transforma en el següent model lògic:



Aturem-nos i reflexionem una mica sobre aquesta representació, en el cas que la multiplicitat mínima del tipus de relació *composes* fos igual a 0 en lloc d'igual a 1. Si suposem que una obra pot no tenir associat cap compositor, per exemple, perquè es desconeix o consta com a anònima, aleshores la clau forana podria prendre valors nuls.

Si volem evitar l'existència de valors nuls, podem optar per utilitzar la representació dels tipus de relació mitjançant una relació.

En el nostre exemple, si representem *composes* per mitjà d'una relació, obtindrem:



En el cas particular d'un tipus de relació binària 1..1 entre dos tipus d'entitat *E1* i *E2* que es transformen en les relacions *R1* i *R2* del model lògic, podrem representar el tipus de relació com a relació, com a clau forana de *R1* que referencia *R2* o com a clau forana de *R2* que referencia *R1*.

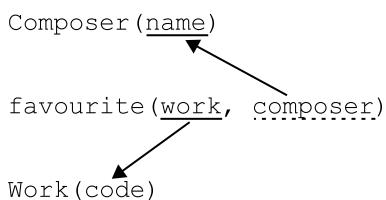
En cas que les multiplicitats mínimes siguin igual a 0, la representació per clau forana haurà d'admetre valors nuls, tal com hem explicat. Si optem per la representació per relació, cal tenir present que ara la relació no té una clau composta sinó dues claus candidates.

Per exemple, si volem representar el fet que un compositor pot tenir una obra preferida (*favourite*) i que una obra pot ser la preferida d'un compositor, aleshores tenim l'esquema conceptual de la figura 15.

Figura 15. Un tipus de relació amb multiplicitats mínimes igual a 0



Si optem per una transformació per relació, obtenim el model lògic següent:

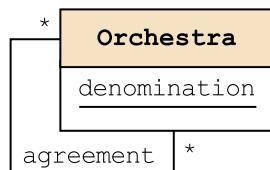


3.4.2. Tipus de relacions binàries reflexives

Un altre cas particular que hem d'estudiar és el dels tipus de relacions binàries reflexives. Es poden representar igual que en el cas general: mitjançant claus foranes o mitjançant relacions.

Com a exemple de transformació de tipus de relació reflexiva, considerem l'esquema de la figura 16, que modelitza els convenis que poden existir entre orquestres.

Figura 16. Un tipus de relació reflexiva



El model relacional corresponent és:

Orchestra (denomination)



En el cas que el tipus de relació tingui la multiplicitat màxima més gran que 1, en tots dos extrems s'ha de representar per relació i en aquest cas, cal analitzar si és simètrica.

En cas afirmatiu, podem optar per estalviar espai guardant la meitat de la informació i deixant implícita l'altra meitat, que es pot deduir per simetria. També cal garantir que per a cada parell (o_1, o_2) de l'extensió hi hagi el parell (o_2, o_1) :

- bé físicament, amb assercions que ho garanteixin,
- bé virtualment, guardant només la meitat dels parells i amb assercions que evitin l'existència de parells simètrics i una vista definida sobre la taula. Aquesta vista ha de reconstruir l'extensió sincera a partir de la meitat que tenim emmagatzemada.

Recordeu

Una vista es declara donant-li un nom i definint-la com una consulta sobre una o més taules. Una vegada s'ha declarat, s'hi poden fer consultes com si fos una taula.

3.4.3. Tipus de relacions binàries de composició

Les composicions són, en el fons, un cas especial de tipus de relació binària que, a més, es transformen al model relacional de la mateixa manera. Per això les tractem en aquest subapartat com un altre cas de tipus de relació binària.

Les composicions estableixen una relació de dependència d'existència entre dos tipus d'entitat. Normalment, aquesta dependència fa que la identificació del component sigui només vàlida en el conjunt de components d'un mateix compostat.

Aquest **tipus de relació de composició** es representa com a clau forana en la relació que representa el tipus d'entitat dependent. A més, la clau primària d'aquest tipus d'entitat està formada per l'identificador del tipus d'entitat concatenat amb la clau forana.

Imaginem que volem representar el conjunt d'obres de cada compositor. Cada obra s'identifica per l'atribut *opus* (un número que ordena cronològicament les obres d'un compositor), però diferents obres de diferents compositors poden tenir un mateix opus. En la figura 17 es representa l'esquema conceptual corresponent.

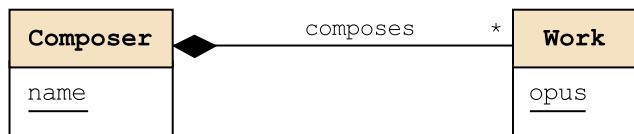
Tipus de relació binària

Un tipus de relació binària és simètric si per a tota relació del tipus (a, b) també existeix la relació (b, a) . Per exemple, el tipus de relació *germans* entre dues entitats de tipus *Persona* és simètric.

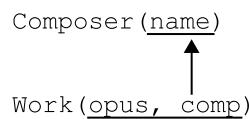
Vegeu també

El mecanisme d'assercions es presenta en el subapartat 3.7 d'aquest mòdul.

Figura 17. Exemple de relació binària de composició



D'acord amb la regla de transformació dels tipus de relació de composició vista anteriorment, obtenim el model lògic següent:

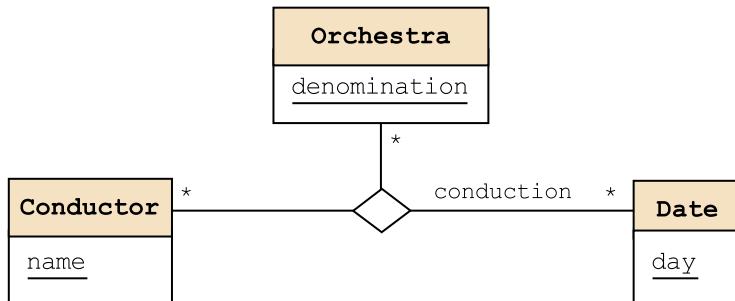


3.4.4. Tipus de relacions n-àries

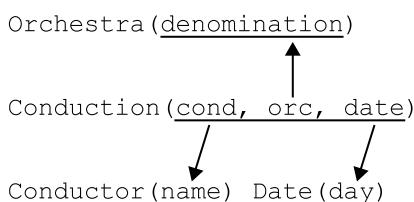
Els **tipus de relació d'aritat més gran que 2** es representen per relacions. El que cal tenir en compte són les multiplicitats, ja que si hi ha alguna connectivitat màxima igual a 1, apareixen claus alternatives.

Suposem que volem registrar les dades històriques referents a les direccions d'orquestra amb el corresponent director, tal com es mostra en la figura 18.

Figura 18. Un tipus de relació ternària



La figura 18 representa el cas senzill, sense cap multiplicitat limitada a 1. El model relacional corresponent és el següent:



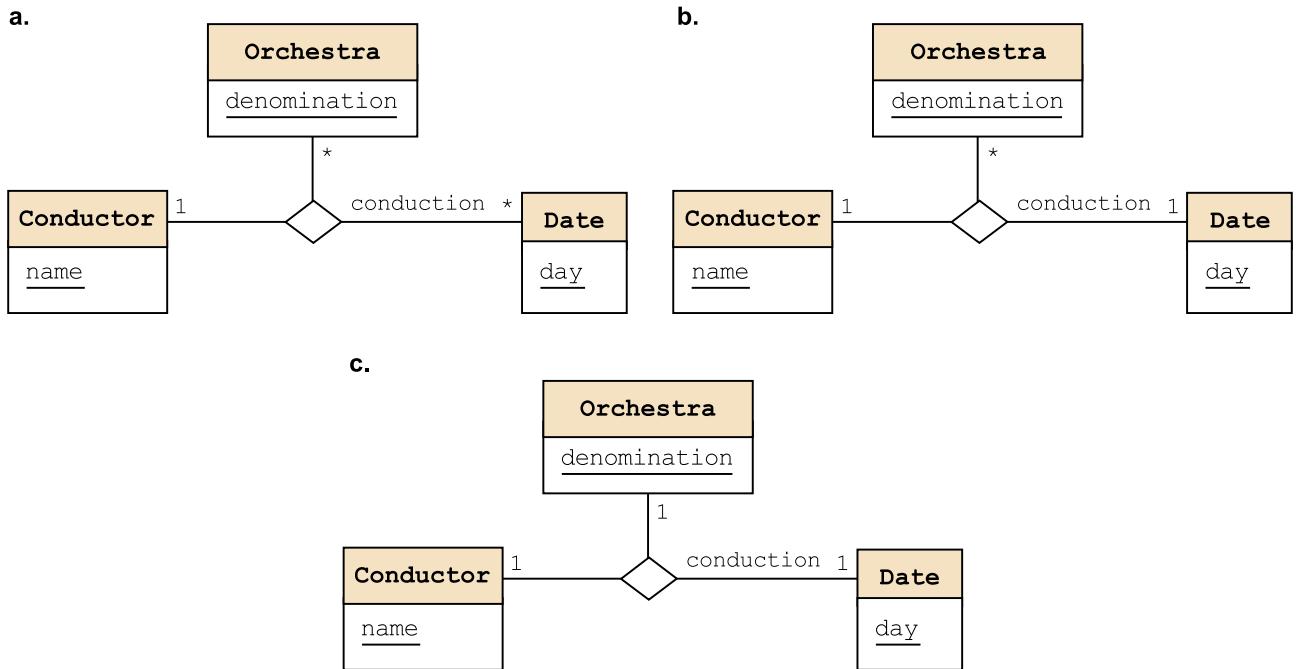
Si l'extrem d'un tipus d'entitat participant té connectivitat màxima igual a 1, indica que, fixades les altres entitats, ja queda determinada la primera. Per tant, l'atribut corresponent a aquesta entitat no forma part de la clau de la re-

lació en què es transforma el tipus de relació *n*-ària. Si hi ha més d'un participant amb multiplicitat màxima igual a 1, podem formar diverses claus segons quin atribut sigui descartat per tal de formar la clau.

A la figura 19 podem veure diferents casos en el mateix context, on anem limitant la multiplicitat màxima dels extrems del tipus de relació *conduction*.

Observem l'impacte que té en la relació *conduction* i com afecta la definició de les claus candidates.

Figura 19. Tipus de relacions ternàries amb multiplicitats mínimes igual a 1



L'impacte que té això en la relació *conduction* afecta la definició de les claus candidates d'aquesta relació. Considerant els esquemes de la figura 19, obtenim:

a) una clau candidata, però formada només per dos atributs. L'altre s'ha de declarar *NOT NULL*:

```
conduction(cond, orc, date)
```

b) dues claus candidates:

```
conduction(cond, orc, date)
.....
```

c) tres claus candidates:

```
conduction(cond, orc, date)
-----
```

De manera similar, es representen els tipus de relació d'aritat major que 3, tot i que són poc freqüents.

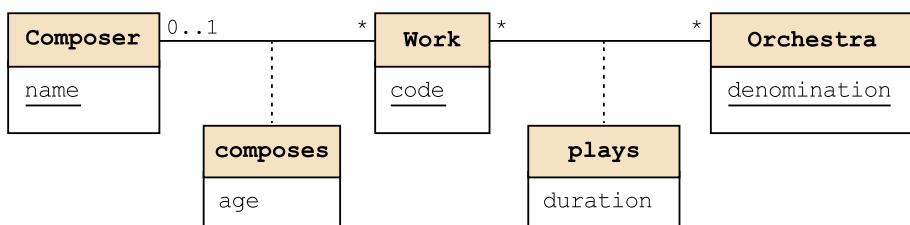
3.5. Tipus d'entitats associatives

Les entitats associatives del model relacional també s'han de transformar al model lògic, i per això cal tenir cura d'on situar els atributs d'aquest tipus d'entitat, ja que conceptualment pertanyen al tipus de relació.

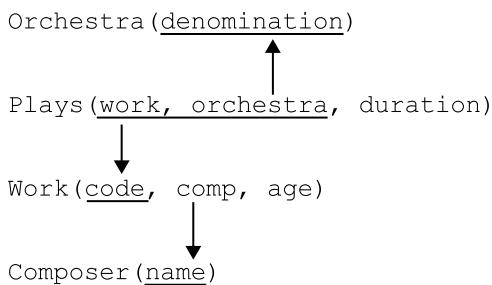
En un **tipus d'entitat associativa**, els atributs es poden representar de dues maneres, dependent de si el tipus de relació s'ha representat com una relació o com una clau forana. En el primer cas, els atributs del tipus de relació esdevindran columnes de la relació i, en el segon cas, esdevindran columnes de la clau forana de la relació.

Vegem-ho amb un exemple. Suposem el model conceptual de la figura 20.

Figura 20. Tipus d'entitats associatives



Atès que *plays* té una multiplicitat $*..*$, es transforma mitjançant una relació. En canvi, com que *composes* té una multiplicitat $0..1$, es pot transformar mitjançant una clau forana. Tenint en compte això, la traducció al model relacional de l'esquema conceptual és la següent:



Hem de parar atenció, un cop més, als valors nuls. L'atribut *comp* pot ser nul perquè el tipus de relació té multiplicitat $0..1$ i, per tant, l'atribut *age* també ho pot ser.

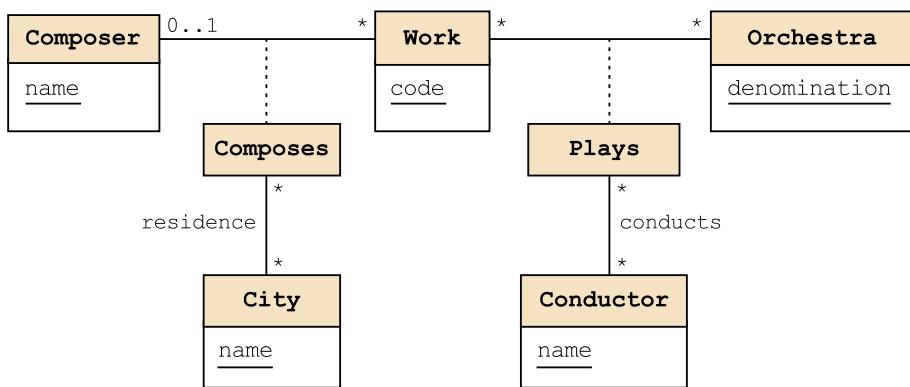
Cal tenir en compte que si les claus foranes que representen un tipus de relació poden ser nul·les, tots els atributs que tingui el tipus de relació també ho poden ser.

Un tipus d'entitat associativa pot participar en tipus de relació. Igual que la resta de tipus d'entitats participants d'un tipus de relació, també els tipus d'entitat associativa hauran de ser referenciats per alguna clau forana si participen en tipus de relacions.

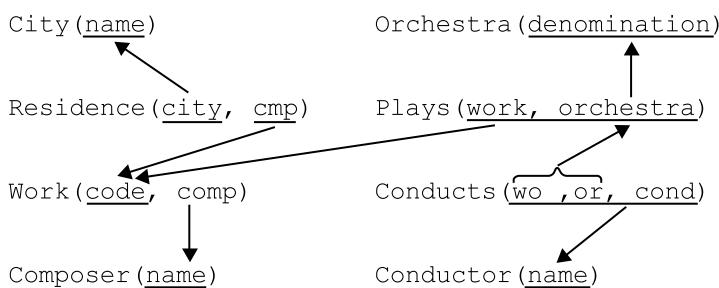
Si el tipus d'entitat associativa s'ha representat amb una relació, la clau primària serà composta i les claus foranes que la referencien també ho hauran de ser. En canvi, si s'ha representat com a clau forana en una relació R , les claus foranes que hagin de referenciar el tipus d'entitat associativa hauran de referenciar la relació R .

Per a il·lustrar aquesta casuística, fixem-nos en el model conceptual de la figura 21. Aquest model relaciona cada obra amb el seu compositor i les orquestres que l'han interpretada. A més, es recull en quines ciutats residia el compositor durant la composició de l'obra i quins directors dirigien l'orquestra en les interpretacions de l'obra.

Figura 21. Exemple de tipus d'entitats associatives que participen en altres tipus de relacions



D'acord amb la regla de transformació dels tipus d'entitat associativa comentada anteriorment, obtenim el model relacional següent:



Fixem-nos que el tipus d'entitat associativa *Plays* s'ha transformat en la relació *Plays*. Per això, en l'esquema relacional, *Conducts* referencia *Plays*. En canvi, el tipus d'entitat associativa *Composes* s'ha transformat en la clau forana *comp* de la relació *Work*. Per això *residence* referencia *Work*, que és on ha quedat la clau forana *comp*, que és la transformació de *Composes*.

3.6. Generalitzacions

La darrera estructura del model conceptual que veurem com es transforma al model relacional és la generalització/especialització, simplement, generalització.

Hi ha tres formes de transformar una generalització al model relacional:

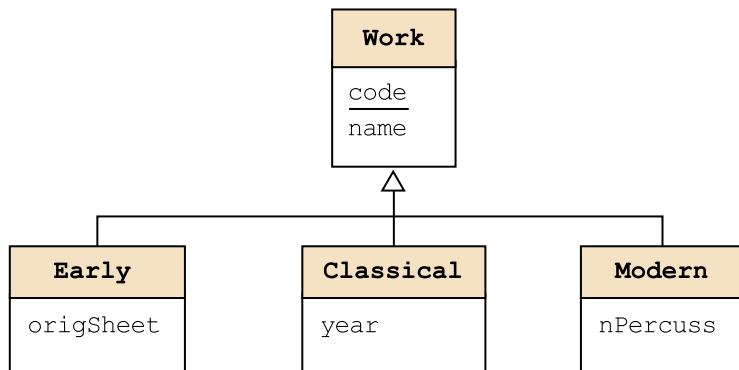
1) Una única relació, les columnes de la qual són la unió dels atributs de tots els tipus d'entitat (superclasse i subclasses).

2) Una relació per a cada classe, però cap relació per a la superclasse. Cada relació tindrà les columnes que corresponen als atributs de la seva classe i, a més, els de la superclasse.

3) Una relació per a cada tipus d'entitat. Cada relació contindrà les columnes dels atributs corresponents al seu tipus d'entitat. En el cas de les relacions que representen les subclasses, a més, hi tindrem l'identificador amb una restricció de clau forana que referenciarà la relació pare o superclasse.

Considerem l'esquema conceptual de la figura 22, que representa diferents tipus d'obra. Hi ha les obres de música antiga (*early*) de les quals volem informació sobre la partitura original (*original sheet*), les de música clàssica (*classical*) i les de música moderna (*modern*).

Figura 22. Un cas de generalització/especialització



A continuació vegem com es transforma segons les diferents opcions:

Recordeu

Tota generalització/especialització presenta una superclasse (o tipus d'entitat genèrica) i un conjunt de subclasses (o tipus d'entitats específiques). A més, pot ser que aquestes subclasses siguin disjunes o encavalcades, d'una banda, i que l'especialització sigui total o parcial, d'una altra.

1.

`Work(code, name, origSheet, year, nPercuss)`

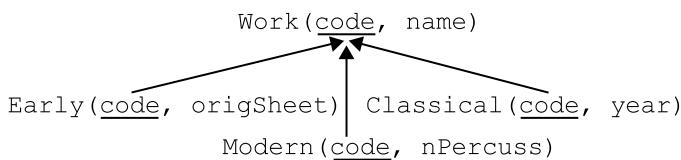
2.

`Early(code, name, origSheet)`

`Classical(code, name, year)`

`Modern(code, name, nPercuss)`

3.



Per tal d'escollar entre les tres representacions possibles d'una generalització, hem de considerar una sèrie de circumstàncies:

1. Una primera qüestió que cal tenir en compte és quines de les possibles estructures poden emmagatzemar tota la informació de totes les possibles instàncies.
2. Un altre criteri important és el nombre de valors nuls generats en cada estructura.
3. També cal dir que algunes de les opcions poden comportar redundància, quan la generalització és encavalcada. Per exemple, en la segona opció, una instància que pertanyi a dues subclasses repetirà dues vegades els valors dels atributs provinents de la superclass.
4. Finalment, també en funció de la consulta que vulguem fer, hem de considerar qüestions de rendiment. En l'exemple anterior, quan s'intenten combinar les relacions més específiques amb la relació genèrica o pare, l'opció 3 genera operacions de combinació (*join*) entre les relacions per a poder obtenir tots els resultats, mentre que l'opció 2 genera operacions d'unió (*union*) entre les diferents relacions per a poder accedir a tots els resultats.

Comentem alguns exemples per tal d'il·lustrar aquests criteris:

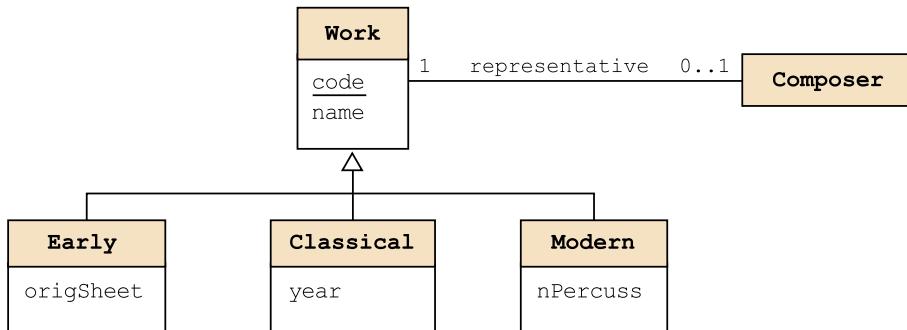
- a) Si escollim l'opció 2 i es tracta d'una generalització parcial, no disposarem de lloc on emmagatzemar les instàncies que no pertanyen a cap subclass.
- b) Si triem l'opció 1, cada tupla tindrà valors nuls en les columnes que no corresponen a la subclass de la instància que es consideri. Si la generalització és total i encavalcada, tota instància serà d'una o més subclasses i la proporció de nuls potser serà admissible. En l'altre extrem, una generalització parcial i disjunta donarà lloc a una gran proporció de nuls.

- c) Si triem l'opció 3, en el cas d'una generalització encavalcada, els atributs comuns de la superclasse es repetiran, per a cada instància de la relació, tantes vegades com el nombre de subclasses a què pertany la instància.

Cal escollir l'alternativa que permeti emmagatzemar totes les instàncies i que no generi valors nuls (o en generi un nombre mínim) i no generi redundància, excepte quan hi hagi raons de rendiment que exigeixin una opció diferent.

A continuació, podem veure un altre exemple en què la superclasse forma part d'un tipus de relació. La figura 23 mostra un model en el qual es defineix que cada compositor té una obra representativa i que les obres poden ser de diferents tipus.

Figura 23. Un cas de generalització/especialització en què la superclasse participa en un tipus de relació



La regla de transformació de generalitzacions s'ha de considerar com una guia general, però, a més, s'ha de tenir en compte la resta de l'esquema conceptual en què participa la generalització. Per exemple, si hi ha referències d'altres relacions (sorgides de la traducció d'una altra part de l'esquema), algunes de les opcions deixen de ser vàlides. Per exemple, en l'esquema de la figura 23, si decidim convertir el tipus de relació *representative* en una clau forana que referencia *Work*, l'opció 2 no és vàlida, perquè necessitem una relació per a la classe genèrica.

3.7. Restriccions

Finalment ens manca analitzar la manera en què les restriccions, que formen part del model conceptual, es transformen al model lògic relacional.

És important que les **restriccions** del model conceptual es conservin en el model lògic. Algunes d'aquestes restriccions són estructurals del diagrama de classes, d'altres són restriccions textuais que el dissenyador ha indicat en el model conceptual.

En aquest subapartat ens centrem en les restriccions estructurals del diagrama de classes. Algunes restriccions queden incorporades en el model relacional si s'han aplicat de manera correcta les transformacions que hem vist fins ara. Hi ha, però, altres restriccions que caldrà afegir al model lògic relacional de forma més explícita.

Les restriccions estructurals son de tres tipus:

1) Restriccions d'identitat. Corresponden a identificadors dels tipus d'entitat. Ja hem comentat que s'incorporen com a claus primàries o alternatives en el disseny lògic.

2) Multiplicitat dels tipus de relació. Cal garantir la connectivitat entre les instàncies de relacions, respectant els màxims i mínims d'aquestes multiplicits. Ja hem tingut en compte algunes d'aquestes restriccions, però en queden d'altres que encara no hem tractat.

3) Tipologia de les generalitzacions. S'ha de procurar que quedin reflectides les restriccions de pertinença a superclasses i a les subclasses tenint en compte la seva tipologia (declaració de disjunt/encavalcat i total/parcial).

Els mecanismes de què disposem per a mantenir i controlar les restriccions són els següents:

- Restriccions admeses en la creació de taules: *PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK*. Són mecanismes automàtics i de baixa complexitat que són fàcils de definir i de mantenir.
- Assercions. L'estàndard SQL incorpora aquest mecanisme de definició de restriccions basat en condicions que s'especifiquen fent servir una construcció que pot involucrar instruccions *SELECT*. Per això són molt potents i permeten expressar condicions que no poden recollir les restriccions del subapartat anterior, que només poden accedir a informació d'una tupla d'una relació. Es tracta d'un mecanisme amb les característiques positives dels anteriors (són automàtiques, de baixa complexitat, fàcils de definir i de mantenir) però que malauradament cap SGBD no implementa avui dia.
- Procediments emmagatzemats. Són procediments que poden contenir sentències SQL (consulta, actualització i definició) combinades amb estructures de control clàssiques de programació (iteracions, alternatives, etc.). Es poden definir uns procediments d'accés a les taules que efectuaran els controls necessaris per a fer el manteniment de restriccions.
- Disparadors³. Són similars a procediments emmagatzemats que s'associen a operacions sobre taules i que s'executen automàticament a partir de determinades accions sobre les dades (inserció, modificació o eliminació de dades). És un mecanisme automàtic, però requereix un esforç considera-

⁽³⁾En anglès, *triggers*.

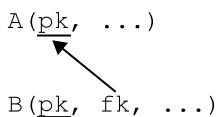
ble definir-los i mantenir-los. Podem, doncs, dissenyar disparadors que s'executin quan es facin actualitzacions en les dades de taules en les quals calgui comprovar que no es violen les restriccions.

- Precondicions. Es pot considerar la possibilitat de delegar el control d'algunes restriccions a les aplicacions i alliberar-ne la base de dades, la qual actuarà sota el supòsit que les operacions que se li demanen satisfan les condicions de correctesa que calgui.
- Control extern. Fins i tot es pot dur la idea anterior més enllà i confiar que determinades restriccions es controlin de manera externa a l'aplicació, habitualment per part d'algun mecanisme o procés automàtic que valida les dades.

En aquest mòdul ja hem tractat les restriccions d'identitat. A continuació veurem com podem tractar els altres dos tipus de restriccions estructurals i els problemes amb què ens podem trobar.

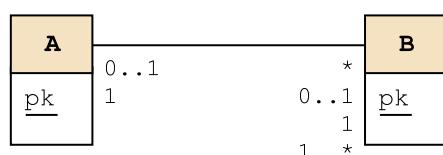
3.7.1. Multiplicitats

Pel que fa a les multiplicitats, caldrà analitzar els diferents tipus de relació. Comencem pels tipus de relacions binàries que es representen mitjançant alguna clau forana, és a dir, tipus de relacions entre dues relacions *A* i *B* de manera que una d'elles té una o més columnes que referencien les tuples de l'altra relació.



En la figura 24 es mostren les possibles multiplicitats d'un tipus de relació binària.

Figura 24. Conjunt de possibles multiplicitats d'un tipus de relació binària que es pot transformar en clau forana



Segons la multiplicitat de l'esquerra (costat de l'entitat *A*), la clau forana admetrà valors nuls o no, com ja hem discutit. La multiplicitat de la dreta (costat de l'entitat *B*), però, ens pot obligar a afegir alguna restricció que abans no hem tingut en compte. Segons el valor d'aquesta multiplicitat cal tenir en compte:

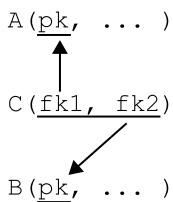
- Si és *, no cal afegir res més.

Vegeu també

Vegeu les multiplicitats i els diferents tipus de relació en el subapartat 3.4 d'aquest mòdul didàctic.

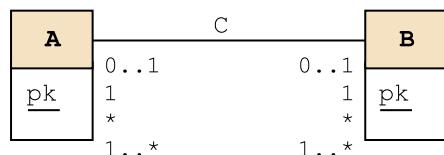
- Si és 0..1, hem de declarar que fk és *UNIQUE*, assegurant així que no hi pot haver més d'un element de B associat amb el mateix element de A .
- Si és 1, hem d'assegurar que tota tupla de A és referenciada per una tupla de B . Podem expressar de diverses maneres que tota fila de A és referenciada per una fila de B . Per exemple, com una asserció o una clau forana de A cap a B . Aquesta clau forana seria l'atribut pk de A que referenciaria fk de B , que haurem definit com a *UNIQUE*. A més, caldrà controlar que si la tupla b de B referencia la tupla a de A , la tupla a referenciï la tupla b . Aquesta comprovació es podria fer, per exemple, emprant un disparador.
- Si és 1..*, hem d'assegurar, mitjançant una asserció, que tota fila de A és referenciada per una o més files de B .

A continuació analitzem un altre tipus de relació binària.



Les multiplicitats possibles d'aquest tipus de relació són les que es mostren en la figura 25.

Figura 25. Possibles multiplicitats d'un tipus de relació binària que es pot transformar en relació



En funció del nombre màxim de les multiplicitats dels tipus d'entitats A i B en el tipus de relació C (si és igual a 1 o més gran que 1) $fk1$ i $fk2$ seran claus per elles mateixes o no:

- en el cas en què tots dos màxims siguin iguals a 1, tindrem dues claus alternatives.
- en el cas en què un màxim sigui igual a 1 però l'altre sigui més gran que 1, hi haurà una sola clau formada per una sola columna.
- en el cas en què tots dos màxims siguin més grans que 1, hi haurà una clau formada per les dues columnes.

Pensem ara en les multiplicitats mínimes, que es tractaran cadascuna de forma independent: si el mínim és igual a 0, no cal afegir més controls. Si la multiplicitat a l'esquerra de la figura 25 és igual a 1, hem d'assegurar que totes les tuples de la relació *B* apareguin una i només una vegada en la relació *C*. I si la multiplicitat és 1..* haurem d'assegurar que apareguin una o més vegades. Caldrà establir els controls simètrics si el mínim de la dreta de la figura 25 és igual a 1.

Per acabar l'estudi de restriccions provinents de tipus de relació, veurem ara els tipus de relació d'aritat superior a 2. Com ja hem explicitat, en funció del màxim de les multiplicitats (si és igual a 1 o més gran que 1), la clau de la relació del tipus de relació estarà formada per totes les claus foranes o només per una part. També pot passar, com ja hem vist, que apareguin claus alternatives.

Reflexió

Ens trobem, doncs, situacions anàlogues a les que hem tractat en el cas de representació per clau forana i que solucionem, també, de manera anàloga.

Vegeu també

Vegeu els tipus de relacions *n*-àries en el subapartat 3.4.4 d'aquest mòdul didàctic.

Pel que fa a les **multiplicitats mínimes**, en el cas d'aritat més gran que 2 és habitual que siguin 0. Si es dóna el cas de multiplicitat més gran que 0, caldrà un asserció que comprovi que es donen efectivament les relacions indicades per la multiplicitat.

Per exemple, en la transformació de l'esquema conceptual de la figura 19a, l'asserció haurà de comprovar que, per a tot parell d'orquestra i data, hi ha un director relacionat.

3.7.2. Generalitzacions

En aquesta secció ens ocuparem de les restriccions corresponents a les característiques de les generalitzacions. Segons quina sigui l'opció escollida de les tres presentades, la que s'utilitzi per transformar la generalització del model conceptual al model lògic, haurem d'inserir un tipus de restriccions o un altre:

Vegeu també

Vegeu les generalitzacions en el subapartat 3.6 d'aquest mòdul didàctic.

a) Una única relació per a tots els tipus d'entitat de la jerarquia. És útil en casos molt particulars de generalitzacions encavalcades i parcials, com ja s'ha comentat. En funció de com s'indiqui a quines classes o subclasses pot pertànyer una instància en el model conceptual, serà necessari dissenyar alguna asserció per a comprovar que cada instància pertany a alguna classe.

b) Una relació per a cada tipus d'entitat específica. S'ajusta bé als casos en què la jerarquia és disjunta i total. En aquest cas, serà necessari controlar que una mateixa instància no pertany a més d'una classe. Aquesta comprovació es pot fer mitjançant l'ús de disparadors incorporats en la inserció i la modificació.

c) Una relació per a l'entitat genèrica. És l'opció més flexible i permet representar qualsevol combinació disjunta/encavalcada i total/parcial. En aquest cas serà necessari afegir els controls de disjunta (com acabem d'explicar) i/o total (que podem implementar amb una asserció) segons escaigui.

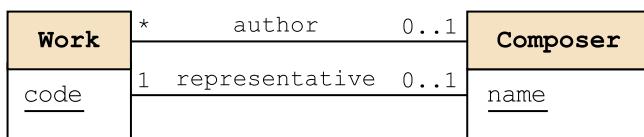
3.7.3. Abraçades mortals

Finalitzarem aquest subapartat de restriccions comentant un problema que pot aparèixer en el procés d'incorporació de les restriccions al model relacional o, també, durant el pas de l'esquema conceptual al model relacional en determinades situacions. Ens referim a les abraçades mortals, referències cícliques entre relacions del model.

Quan en un esquema relacional es genera un cicle de claus foranes diem que s'ha produït una **abraçada mortal de definició**. Això vol dir que no podem definir directament les taules corresponents perquè cadascuna necessita que se n'hagi definit prèviament alguna altra. La solució d'aquest problema passa per definir primer una taula sense clau forana mitjançant una sentència *CREATE TABLE*. A continuació, es defineixen altres taules que requerien la primera i, finalment, s'incorpora la clau forana a la primera taula amb una sentència *ALTER TABLE*.

Suposem el cas en què volem modelitzar una situació en la qual una obra sigui representativa d'un compositor i que un compositor tingui o no una obra de referència. El model conceptual per a aquesta situació seria el que es descriu en la figura 26.

Figura 26. Un tipus de relació binària que es pot transformar en dues claus foranes cícliques



Aquest model conceptual es pot transformar en el model relacional següent:

Work (code, repr)
 Composer (name)

La clau forana de *Work* a *Composer* representa el tipus de relació *representative*. Hi afegim la clau forana de *Composer* a *Work* juntament amb la restricció *UNIQUE* per a garantir la multiplicitat igual a 1. En un primer moment, ens trobem que no podem definir aquestes relacions perquè totes dues necessiten

que l'altra ja estigui definida prèviament. Per a solucionar-ho, crearem primer una de les taules sense clau forana, després l'altra taula i, finalment, incorporarem la clau forana a la primera taula.

Un cop superada la dificultat de la creació de taules amb referències cícliques, ens podem trobar amb una situació similar a l'hora d'inserir files en les taules buides: no podem inserir una obra si no hi ha abans el compositor, però tampoc no podem inserir el compositor fins que no tinguem la seva obra representativa.

Quan en un esquema relacional no podem inserir les tuples que donarien lloc a un contingut que no viola cap restricció, perquè hi ha un cicle de claus foranes que impedeix inserir les tuples d'una en una sense violar la integritat referencial, diem que s'ha produït una **abraçada mortal de càrrega**.

Una primera solució de l'abraçada mortal de càrrega consisteix a eliminar les restriccions que la provoquen o delegar-les a algun altre nivell. Podem, però, mantenir el control d'aquestes restriccions en l'àmbit de l'SGBD diferint-ne la comprovació.

Els SGBD, per mitjà del mecanisme de transaccions, ens ofereixen la possibilitat de diferir la comprovació de restriccions en comptes de fer la comprovació immediatament després de cada operació elemental. D'aquesta manera, després del primer *INSERT* s'estarà violant la restricció, però el segon *INSERT* ens durà a un nou estat en el qual se satisfà la restricció. Entremig de les dues sentències d'inserció de files, l'SGBD impedirà els accessos a aquestes taules que són, provisionalment, inconsistents. Una vegada hagin estat completades les dues sentències d'inserció l'SGBD farà la comprovació de restriccions i tornarà a permetre l'accés a les taules.

3.8. Reconsideracions

Un cop acabada la transformació d'un esquema conceptual al model relacional, hi ha alguns detalls que cal afinar en l'esquema relacional obtingut.

En aquest subapartat presentem dues situacions que poden ajudar a ajustar l'esquema lògic.

La primera d'aquestes situacions es presenta quan hi ha la possibilitat d'eliminar alguna relació que pot semblar innecessària. Quan ens trobem amb relacions que consisteixen exclusivament en la clau primària i que, a més, són referenciades des d'una altra relació, ens podem preguntar si és millor eliminar aquestes relacions i quedar-nos només amb la relació que les referencia, de manera que s'eviti repetir els mateixos valors tant a la clau forana referencia-

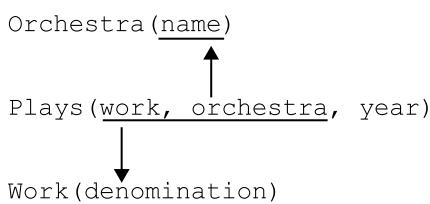
dora com a la clau primària referenciada. La resposta és, generalment, negativa, perquè l'existència d'aquestes relacions ens permet comprovar la integritat referencial.

Considerem el cas de la figura 27 que representa obres, orquestres i quines obres ha interpretat cada orquestra.

Figura 27. Un tipus de relació binària



A partir del model conceptual de la figura 27 obtenim l'esquema lògic següent:



Si *Work* i *Orchestra* no tenen més atributs que l'identificador, ens podem preguntar si és millor eliminar aquestes relacions i quedar-nos només amb la relació *Plays*, on ja apareixen aquests identificadors. Hem de valorar, però, que *Orchestra* i *Work* ens permeten estar segurs que sempre que apareix una obra o una orquestra es tracta d'una obra o orquestra que existeix i que sempre ho fa amb el mateix nom o denominació.

Si ens trobem en una situació (que és poc freqüent) en què la integritat referencial ens és garantida per algun altre sistema (per exemple, no cal una taula de dates perquè l'SGBD ja no permet l'existència de valors de data incorrectes), sí que podrem eliminar la taula.

Una altra opció de simplificació consisteix a fusionar dos tipus d'entitat connectats per un tipus de relació 1..1 en un únic tipus d'entitat. La decisió sobre si és millor fusionar o no la prendrem en funció de l'espai ocupat i del temps requerit per a les operacions més freqüents (si és més freqüent consultar atributs d'un sol tipus d'entitat o, al contrari, és més freqüent consultar informació que inclou atributs dels dos tipus d'entitats). La fusió consisteix a representar l'esquema conceptual, no amb dues relacions lligades amb claus foranes, sinó amb una única relació que inclogui els atributs de tots dos tipus d'entitat.

En el cas en què ens decidim a fer la fusió, la relació resultant tindrà dues claus alternatives i haurem de triar quina ha de ser la primària. El criteri de tria més adequat sol ser el de mínima freqüència de canvi.

Fixem-nos en el cas de la figura 28, en què es representen orquestres, directors i qui dirigeix cada orquestra.

Figura 28. Dos tipus d'entitat susceptibles de ser transformats en una única relació



Si fusionem els dos tipus d'entitat, la relació resultant tindrà dues claus alternatives (la de *Conductor* i la d'*Orchestra*). En aquest cas, segurament és millor que la clau primària sigui l'identificador d'orquestra, que pràcticament no canvia mai; en canvi, el director d'una orquestra sí que pot canviar de manera més freqüent.

4. Normalització

Durant el procés de disseny s'han pres diferents decisions que determinen un disseny concret, entre les diverses alternatives possibles, per a resoldre una mateixa necessitat. Per exemple, un mateix concepte del món real pot donar lloc a esquemes conceptuais lleugerament diferents o la transformació del model conceptual al model lògic es pot dur a terme amb diferents alternatives o matisos.

En els apartats anteriors hem vist alguns criteris per a triar o descartar determinades opcions (per exemple, afavorir determinades operacions per sobre d'altres o evitar l'existència de valors nuls). En aquest apartat, veurem les condicions de normalització, que són les condicions que garanteixen que la base de dades està dissenyada de tal manera que no es barregin conceptes diferents en una mateixa relació. Aquesta característica és positiva, perquè facilita la comprensió del disseny i evita redundàncies innecessàries.

En primer lloc, veurem les anomalies que es poden produir quan una base de dades no està normalitzada. Aquestes anomalies impliquen ineficiència i complexitat en el manteniment de la coherència de les dades. En segon lloc, després d'un repàs previ de conceptes del model relacional i de l'àlgebra de conjunts, presentarem la teoria de normalització i veurem com la podem aplicar als esquemes lògics. Veurem que aquesta aplicació elimina les anomalies de la base de dades.

4.1. Anomalies

Un disseny lògic no normalitzat té com a conseqüències negatives la manca de separació de conceptes i l'existència de redundàncies que ens aboquen a l'aparició de les anomenades **anomalies d'actualització**. Es diu que hi ha anomalies quan cal actualitzar moltes tuples per a reflectir un canvi elemental que, amb un disseny normalitzat, implicaria un volum molt menor de tuples.

Aquestes anomalies poden aparèixer en la inserció, la supressió o la modificació de tuples.

Anomalies d'inserció

Quan resulta impossible inserir informacions elementals de manera independent en una base de dades, es diu que es produeix una anomalia d'inserció.

Exemple

Imaginem que volem emmagatzemar informació de les obres que tenim disponibles a la nostra discoteca i dels compositors que en són els autors. Amb aquest objectiu creem la relació següent:

Compositions (work, composer, yearComp, bCentury, digitDegree)

De les obres es registra l'any de composició (en l'atribut *yearComp*) i dels compositors, el segle en què van néixer (*bCentury*) i el percentatge de digitalització que hem assolit de les seves obres (*digitDegree*). En un moment determinat, la relació podria tenir l'extensió que representa la figura 29.

Figura 29. Extensió d'una relació amb redundàncies i anomalies

Compositions				
<u>work</u>	composer	yearComp	bCentury	digitDegree
Symphony 9	Mahler	1923	19	70
Symphony 5	Mahler	1918	19	70
Parsifal	Wagner	1857	19	42
Conc Piano 3	Mozart	1779	18	42

Si volem afegir un nou compositor, anomenat *Motsalvatge*, del qual tenim el nom i el segle de naixement però encara no en coneixem cap obra, no podem fer-ho. Hauríem d'inserir la tupla:

NULL	Montsalvatge	NULL	20	NULL
------	--------------	------	----	------

però no ho podem fer perquè l'atribut *work* és la clau primària i, en conseqüència, no pot tenir valor nul. Ens trobem, doncs, que no podem inserir informació d'un compositor independentment de les seves obres.

Anomalies de supressió

Quan es dóna una pèrdua d'informació involuntària d'un fet elemental per la supressió d'un altre fet elemental, es diu que es produeix una anomalia de supressió.

Exemple

Suposem ara que volem suprimir el tercer concert per a piano de Mozart. Després de la supressió, la nostra relació tindrà l'extensió que mostra la figura 30.

Figura 30. Extensió de la relació després d'una supressió

Compositions				
<u>work</u>	composer	yearComp	bCentury	digitDegree
Symphony 9	Mahler	1923	19	70
Symphony 5	Mahler	1918	19	70
Parsifal	Wagner	1857	19	42

Observeu que, de manera involuntària, hem perdut la informació que teníem del compositor Mozart.

Anomalies de modificació

Quan es presenta la necessitat de modificar diverses (potencialment moltes) tuples per a reflectir el canvi d'un sol fet elemental, es diu que es produeix una anomalia de modificació.

Exemple

Si ara volem anotar que el grau de digitalització de les obres de *Mahler* ha passat a ser del 73%, haurem d'actualitzar la relació tal com es mostra en la figura 31.

Figura 31. Extensió de la relació després d'una modificació

Compositions				
<u>work</u>	composer	yearComp	bCentury	digitDegree
Symphony 9	Mahler	1923	1897	70 73
Symphony 5	Mahler	1918	1897	70 73
Parsifal	Wagner	1857	1813	42

Fixeu-vos que haurem de modificar tantes tuples com obres de *Mahler* tinguem en la relació.

L'origen de les anomalies d'inserció, eliminació i actualització que acauen de veure és la barreja de dos conceptes en una mateixa relació, la qual cosa, a més, implica redundància. La solució no és altra que la separació de conceptes, cosa que s'aconsegueix dividint la relació original en dues noves relacions.

La solució del cas que hem fet servir d'exemple seria la separació d'obres i compositors, que donaria lloc al model normalitzat següent:

```
Work (wName, composer, yearComp)
      ↓
Composer (cName, bCentury, digitDegree)
```

L'extensió d'aquestes noves relacions és la que es presenta en la figura 32.

Figura 32. Extensió de dues relacions normalitzades

Work			Composer		
wName	composer	yearComp	cName	bCentury	digitDegree
Symphony 9	Mahler	1923	Mahler	19	70
Symphony 5	Mahler	1918	Wagner	19	42
Parsifal	Wagner	1857	Mozart	18	42
Conc Piano 3	Mozart	1779			

Ara podem efectuar les operacions d'inserció, supressió i modificació que hem fet servir d'exemple sense que es produueixi cap de les anomalies.

La teoria de la normalització ens permetrà detectar si un disseny pot provocar anomalies com les que hem descrit i, a més, ens permetrà obtenir un nou disseny amb aquestes problemàtiques resoltes.

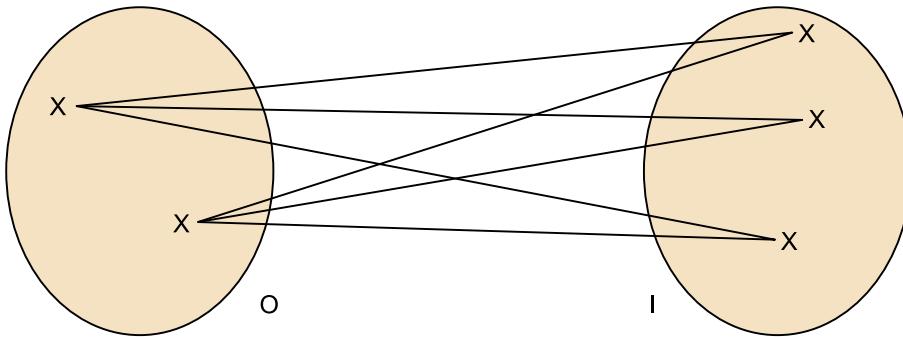
4.2. Conceptes previs

La teoria de la normalització es basa en el concepte de dependència funcional, el qual, al seu torn, es defineix en termes de l'àlgebra relacional i de conceptes de la teoria de conjunts. En aquest subapartat es presenten aquests elements que permetran abordar la teoria de la normalització.

Per començar, repassarem alguns conceptes d'**àlgebra de conjunts**. Aquests conceptes ens serviran després per a raonar sobre els valors que prenen els atributs de les relacions. Concretament, volem definir el producte cartesià, la correspondència i la funció:

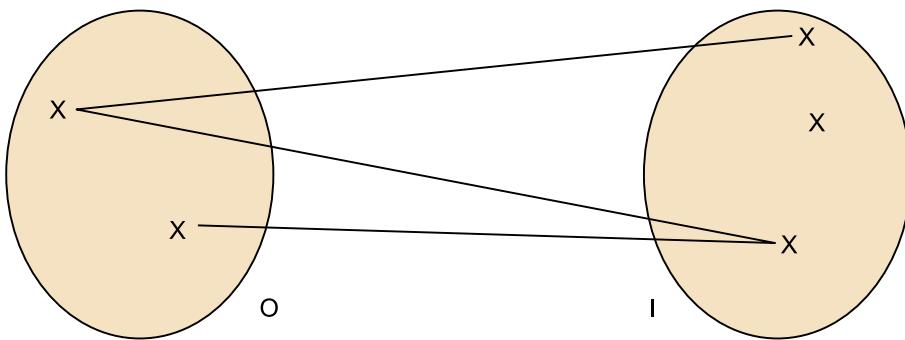
- Donats dos conjunts O i I , el **producte cartesià** $O \times I$ és el conjunt de tots els parells ordenats (o, i) tals que $o \in O$ i $i \in I$. Ho podem representar gràficament com es veu en la figura 33. El conjunt O rep el nom de *conjunt origen* i el conjunt I s'anomena *conjunt imatge*.

Figura 33. El producte cartesià



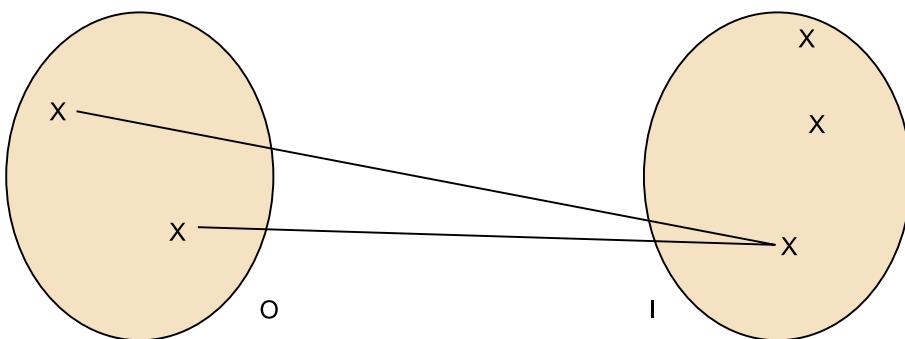
- Un subconjunt qualsevol del producte cartesià és una **correspondència**. La figura 34 mostra una correspondència entre els mateixos conjunts de l'exemple anterior.

Figura 34. Una correspondència



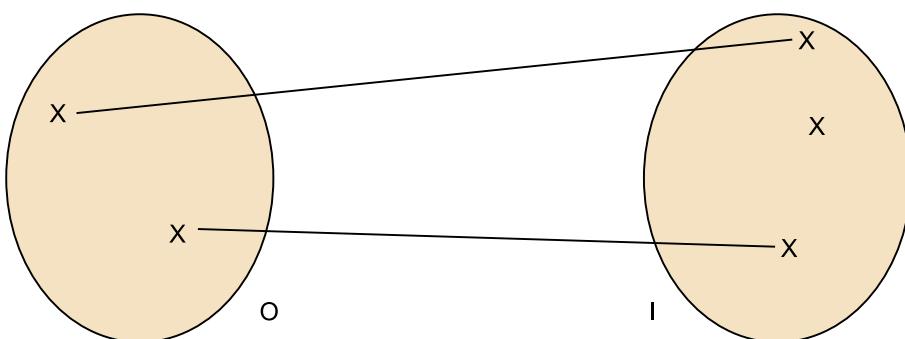
- Algunes correspondències són, a més, **funcions**: aquelles en què cada element del conjunt origen està relacionat amb un, i només un, element del conjunt imatge. La figura 35 és un exemple de funció entre els conjunts O i I de les figures anteriors.

Figura 35. Una funció



- Diem que una funció és **injectiva** quan cada element del conjunt imatge està relacionat, com a molt, amb un element del conjunt origen. En la figura 36 podem veure un exemple de funció injectiva.

Figura 36. Una funció injectiva



Com ja hem vist, les relacions s'ajusten a un esquema que en defineix els atributs i tenen una extensió formada per tuples que donen valors als atributs.

Vegeu també

Vegeu els conceptes previs del model relacional en el subapartat 3.1 d'aquest mòdul didàctic.

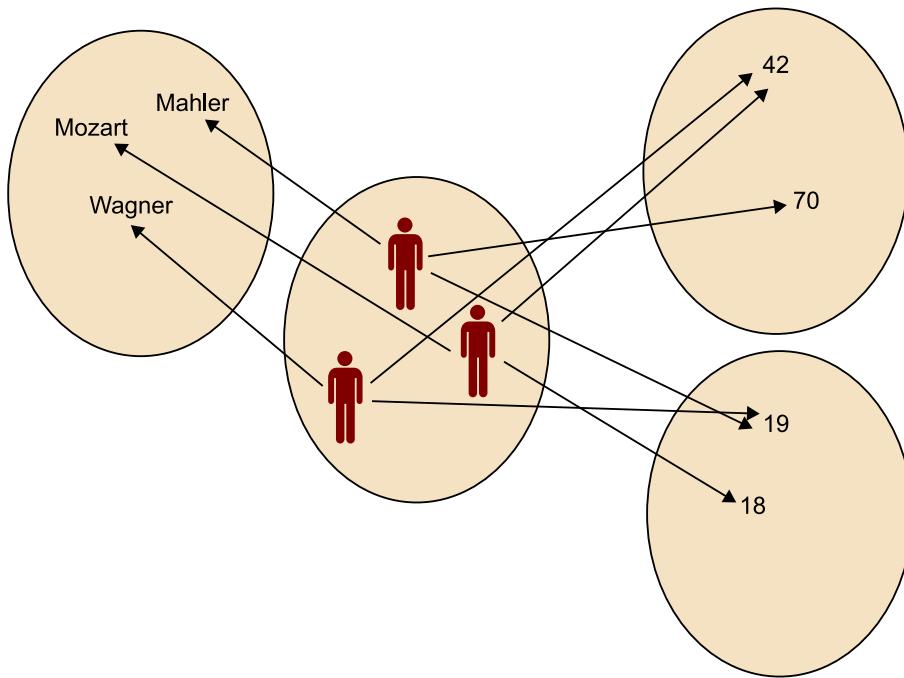
Cada tupla de l'extensió d'una relació té la informació d'una entitat del món real. Podem fer servir el concepte de *funció* per a representar els valors que pren un atribut en cada tupla i, per tant, per a cada entitat.

Per exemple, la relació *Composer* de la figura 32 té tres atributs: *cName*, *Century* i *digitDegree*. El domini del primer són les cadenes de caràcters i el dels altres dos, els nombres.

Els valors que pren un atribut en l'extensió d'una relació es poden considerar una funció que té com a origen el conjunt de les entitats del món real i com a imatge el domini de l'atribut.

La figura 37 és la visualització, mitjançant conjunts i funcions, dels atributs de la relació de la figura 32.

Figura 37. La visió com a funcions dels atributs d'una relació



Podem observar que la funció corresponent a *cName* és injectiva; fixem-nos que aquesta és una propietat que, per definició, compliran totes les funcions que corresponguin a atributs identificadors (observeu que ja havíem indicat que aquest atribut és clau de la relació).

Les **dependències funcionals** s'estableixen entre conjunts d'atributs d'una relació i les podem considerar un tipus més de restriccions que ha de satisfer l'extensió de la relació .

Dependències funcionals

Donada una relació $R(A_1, A_2, \dots, A_n)$, podem declarar $X \rightarrow Y$ com a dependència funcional si $X, Y \subset \{A_1, A_2, \dots, A_n\}$. Per a satisfer aquesta restricció, l'extensió de la relació ha de ser de tal manera que X determini de forma única el valor de Y ; és a dir, que si dues tuples tenen els mateixos valors en els atributs de X , també tinguin els mateixos valors en els atributs de Y . En aquest cas diem que Y depèn funcionalment de X o, alternativament, que X determina funcionalment Y (i, per això, X s'anomena *determinant de la dependència*). És a dir, generalitzant el concepte de *funció*, podem dir que hi ha una funció amb origen X i imatge Y .

Fixem-nos que un cas particular de dependències funcionals són les claus d'una relació: en aquest cas, els atributs que formen la clau determinen tots els altres. Això és conseqüència de la no-repetició de valors de la clau: com que només hi pot haver una tupla amb un valor concret de la clau, totes les tuples amb aquell valor (és a dir, com a màxim una) tenen el mateix valor per a tots els altres atributs de la relació.

Exemple de dependència funcional

Per acabar de presentar aquest concepte, l'il·lustrem amb un exemple. Analitzem la relació següent per a identificar-hi dependències funcionals:

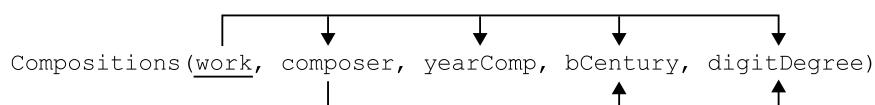
Compositions (work, composer, yearComp, bCentury, digitDegree)

Hi trobem les dependències funcionals següents:

- $\{\text{work}\} \rightarrow \{\text{composer}, \text{yearComp}, \text{bCentury}, \text{digitDegree}\}$. Aquesta dependència correspon a la clau primària.
- $\{\text{composer}\} \rightarrow \{\text{bCentury}, \text{digitDegree}\}$. Sabem que el compositor determina el segle en què va néixer i també el grau de digitalització que hem assolit de les seves obres. Podem comprovar que l'extensió que hem fet servir d'exemple és correcta respecte a aquesta dependència: Mahler, el compositor que es repeteix, apareix ambdues vegades amb els mateixos valors per als atributs *bCentury* i *digitDegree*.

En canvi, $\{\text{composer}\} \rightarrow \{\text{yearComp}\}$ és fals, perquè el compositor no determina l'any de composició de les obres; un mateix compositor pot haver compost diverses obres en anys diferents. Per això, l'extensió de la relació pot contenir una tupla amb Mahler i 1923 i una altra amb Mahler i 1918.

Per a denotar les dependències que hi ha en una relació, farem servir una notació a base de fletxes, tal com mostra l'exemple següent:



Per acabar, notem que pot haver-hi dependències en què podem prescindir d'alguns dels atributs del determinant.

Diem que una dependència funcional $X \rightarrow Y$ és completa quan no hi ha cap altra dependència $X' \rightarrow Y$, essent X' un subconjunt propi de X .

Per exemple, podem afirmar que $\{work, composer\} \rightarrow \{yearComp\}$, però en aquesta dependència podem prescindir de *composer* perquè $\{work\} \rightarrow \{yearComp\}$ també és veritat. La primera de les dependències anteriors no és completa, perquè hi ha la segona i es verifica que $\{work\}$ és un subconjunt propi de $\{work, composer\}$.

Reflexió

Observeu que si en una dependència funcional $X \rightarrow Y$, el conjunt X només té un atribut, la dependència funcional segur que és completa.

4.3. Teoria de la normalització

L'objectiu de la teoria de la normalització és fixar unes condicions que ens garanteixin la separació de conceptes i l'absència de redundància per tal d'evitar les anomalies d'actualització. Aquestes condicions es basen en gran mesura en el concepte de dependència funcional plena que acabem de presentar en el subapartat anterior.

Les anomalies presents en una relació tenen l'origen en dependències existents entre els atributs de la relació. La **teoria de la normalització** defineix una sèrie de nivells, anomenats **formes normals**, que eliminan progressivament determinades dependències que són causants de diferents anomalies. Aquestes formes normals són inclusives; és a dir, si una relació compleix les condicions d'un determinat nivell, també compleix les condicions de tots els nivells anteriors. Com més alt és el grau de normalització, més redundàncies s'eliminen i, per tant, menys anomalies es poden produir.

La teoria de la normalització dóna les bases per a poder modificar una relació que no està en una determinada forma normal amb l'objectiu que hi estigui. Aquest procés de modificació s'anomena *normalització*. Tal com anirem veient, una mateixa relació no normalitzada es pot modificar de diverses maneres fins a convertir-se en una relació normalitzada; és a dir, el procés de normalització pot tenir resultats diversos.

Estudiarem sis formes normals:

- la primera (1FN) es defineix en termes de l'atomicitat dels atributs,
- les tres següents (2FN, 3FN i BCNF), en termes de dependències funcionals,
- la penúltima (4FN) es basa en dependències multivaluades i
- la darrera (5FN), en la dependència de projecció-combinació.

Aquests dos darrers tipus de dependències es presentaran en el moment de definir les formes normals corresponents.

En un primer bloc tractarem les quatre primeres formes normals i després presentarem un parell d'algorismes capaços de normalitzar a aquest nivell. Finalment, veurem les dues darreres formes normals.

4.3.1. Primera forma normal

Una relació està en **primera forma normal (1FN)** si, i només si, cap atribut de la relació és ell mateix una relació, ni descomponible ni amb multiplicitat de valors. Els atributs, doncs, han de ser atòmics.

Per a il·lustrar aquest concepte, fixem-nos en la figura 38, on tenim una relació que permet recollir informació dels compositors.

Figura 38. Una relació amb atributs no atòmics

Composers			
<u>composer</u>	works	yearComp	bCentury
Mahler	Symphony 9	1923	19
	Symphony 5	1918	
Wagner	Parsifal	1857	19

Aquesta relació no està en 1FN perquè hi ha dos atributs, *works* i *yearComp*, que no són atòmics. Per a normalitzar una relació a la primera forma normal, hem d'aplanar els atributs que no són atòmics.

A partir de la relació de la figura 38, obtenim la relació de la figura 39.

Aplanar un atribut no atòmic

Aplanar un atribut no atòmic d'una relació consisteix a substituir cada tupla per tantes com repetitions hi hagi de l'atribut no atòmic.

Figura 39. La relació després d'aplanar els atributs

Composers			
composer	work	yearComp	bCentury
Mahler	Symphony 5	1918	19
	Symphony 9	1923	19
Wagner	Parsifal	1857	19

Cal observar que la clau primària de la relació normalitzada canvia: hem de buscar la nova clau a partir de la superclau formada per la composició de la clau que teníem abans (en el nostre cas, *composer*) amb la clau de la subrelació que formaven els atributs no atòmics (en el nostre cas, *work* i *yearComp* que té com a clau *work*).

En el nostre exemple, doncs, la superclau és $\{composer, work\}$ però com que $\{work\} \rightarrow \{composer\}$, la clau primària està formada exclusivament per l'atribut *work*.

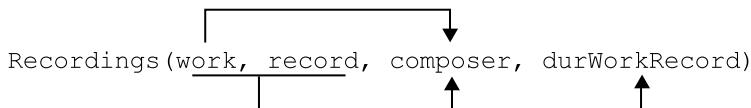
Cal tenir present que el model relacional, i els SGBD relacionals, ja garateixen aquesta primera forma normal en qualsevol relació que hi puguem definir.

4.3.2. Segona forma normal

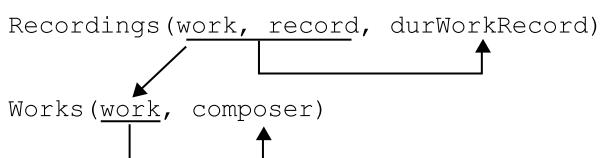
Una relació està en **segona forma normal** (2FN) si, i només si, està en primera forma normal i tot atribut que no forma part d'una clau candidata depèn completament de totes les claus candidates de la relació.

Fixeu-vos que tota relació en primera forma normal que tingui les claus candidates formades per un sol atribut, està automàticament en segona forma normal: per definició de clau, tot atribut depèn de les claus candidates i, en el cas de ser d'un sol atribut, segur que la dependència és completa.

Com a exemple d'aquesta problemàtica, considerem la relació següent, que emmagatzema quines obres hi ha a cadascun dels enregistraments (discs, cintes, arxius MP3, ...) que tenim. Una mateixa obra (*work*) pot estar repetida en diversos enregistraments (*recordings*) i un enregistrament pot contenir diverses obres. També es vol guardar el compositor de cada obra (*composer*) i quans minuts dura cada obra (*durWorkRecord*) en un enregistrament determinat.



Aquesta relació, que té com a clau primària $\{work, record\}$, no està en segona forma normal perquè *composer* no depèn completament de la clau, atès que existeix la dependència $\{work\} \rightarrow \{composer\}$. L'exemple ens permet veure amb claredat quin és l'objectiu de la segona forma normal: impedir que es barregin dos fets elementals que comparteixen part de la clau (els enregistraments de les obres que tenim, d'una banda, i els compositors de les obres, d'una altra) en una mateixa relació. D'aquesta manera, evitarem redundàncies com la de tenir replicat el compositor d'una obra tantes vegades com el nombre d'enregistraments que tenim que contenen l'obra. Per a normalitzar una relació a 2FN hem de separar els fets barrejats que violen la condició de 2FN en dues relacions. En l'exemple que estem veient, cal transformar la relació *Recordings* en dues noves relacions:

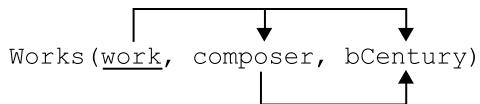


Com es pot observar, les dues relacions resultants s'han de lligar amb una clau forana per mitjà de la part de la clau compartida pels dos fets. Cal no confondre les fletxes que representen dependències amb la que representa aquesta clau forana.

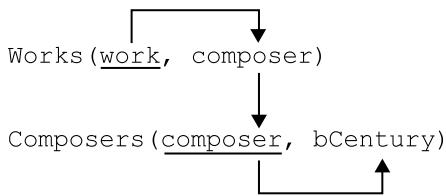
4.3.3. Tercera forma normal

Una relació està en **tercera forma normal** (3FN) si, i només si, està en segona forma normal i cap atribut que no forma part d'una clau candidata depèn d'un conjunt d'atributs que en conté algun que no forma part d'una clau candidata.

Com a exemple, prenem la relació *Works* de l'exemple anterior i hi afegim el segle de naixement dels compositors (*bCentury*); obtenim:



Aquesta relació, la clau de la qual és *work*, està en 2FN però no està en 3FN perquè hi ha la dependència $\{\text{composer}\} \rightarrow \{\text{bCentury}\}$ i cap d'aquests atributs forma part de cap clau candidata. Amb la tercera forma normal evitem, doncs, que es barregin fets (qui és el compositor d'una obra i quan va néixer un compositor, en l'exemple) encara que comparteixin atributs que són clau en un fet (*composer*, en l'exemple) i no ho són en l'altre. Així evitarem redundàncies com la repetició del segle de naixement d'un compositor tantes vegades com obres del compositor apareguin en la relació. Per a normalitzar a 3FN, com abans, hem de separar el fet que correspon a la dependència que viola la condició de 3FN. Si ho fem amb l'exemple anterior, obtindrem:



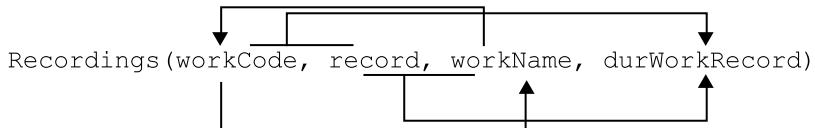
4.3.4. Forma normal de Boyce-Codd

Una relació està en la **forma normal de Boyce-Codd (FNBC)** si, i només si, està en 3FN i els determinants de totes les dependències que presenta la relació en són claus candidates.

Per a veure'n un exemple, podem fixar-nos en la següent relació, que està en 3FN:

La forma normal de Boyce-Codd

Aquesta forma normal es va haver de definir (ho van fer Boyce i Codd el 1974) per a corregir mancances de la 3FN que, quan Codd la va enunciar el 1970, pensava que era suficient per a evitar les anomalies d'actualització.



Es tracta d'una relació similar a la de l'exemple anterior, a la qual hem afegit una nova manera d'identificar les obres. Ara ho podem fer amb un nom i amb un codi. La relació presenta redundància perquè repetirem tant el nom com el codi de cada obra tantes vegades com enregistraments hi hagi de l'obra. Quan es va definir la 3FN no es va preveure una situació com aquesta, en què hi ha dues claus candidates compostes, encavalcades i amb dependències entre parts d'aquestes claus.

Com hem fet constar gràficament, hi ha quatre dependències en aquesta relació:

- $\{workCode\} \rightarrow \{workName\}$
- $\{workName\} \rightarrow \{workCode\}$
- $\{workCode, record\} \rightarrow \{durWorkRecord\}$
- $\{workName, record\} \rightarrow \{durWorkRecord\}$

Els determinants de les dues darreres són claus candidates de la relació, però els determinants de les dues primeres, no. Per tant, la relació no està en FNBC. La figura 40 mostra una possible extensió de la relació.

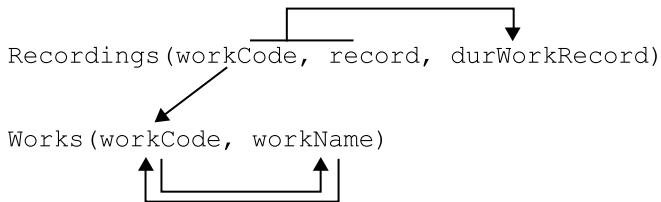
Figura 40. Exemple de relació que compleix la 3FN però no la FNBC

Recordings			
workCode	record	workName	durWorkRecord
MahS5	DeutscheGrammophon001	Symphony 5	52
MahS9	Decca036	Symphony 9	43
MahS5	Naxos201	Symphony 5	55
MahS5	Sony187	Symphony 5	51

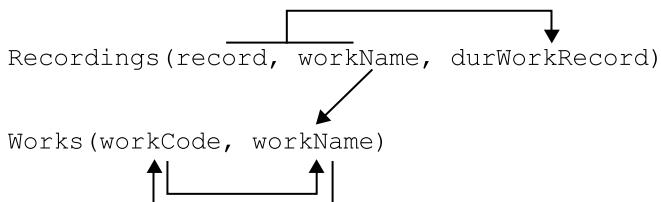
Fixeu-vos que, efectivament, es poden produir anomalies. Per exemple, si volem canviar el nom de la *Sinfonia 5*, haurem de modificar tres tuples.

Per a normalitzar a la FNBC hem d'eliminar les dependències, el determinant de les quals no constitueix una clau candidata. Com que n'hi ha dues, ho podem fer de dues maneres:

1) Eliminant $\{workCode\} \rightarrow \{workName\}$



2) Eliminant $\{workName\} \rightarrow \{workCode\}$



Ens decidirem per una opció o l'altra en funció de si preferim accedir més aviat per codi o per nom de l'obra. Independentment de l'opció seleccionada, caldrà escollir quina és la clau primària de la relació *Works*, que té dues claus candidates. El més coherent serà triar com a clau primària l'atribut que és referenciat des de la relació *Recordings* (*workCode* si hem triat l'opció 1 i *workName* si hem triat l'opció 2).

4.3.5. Regles d'Armstrong

Tal com ja hem comentat, un cop vistes les quatre primeres formes normals, presentarem dos algorismes capaços de normalitzar un esquema relacional fins a FNBC. Aquests algorismes es basen en una sèrie de propietats que tenen les dependències funcionals, que ens permeten deduir-ne de noves a partir d'unes que ja coneixem. Posem, per exemple, la relació que hem fet servir per a presentar les dependències funcionals:

Compositions (work, composer, yearComp, bCentury, digitDegree)

Abans hem dit directament que *work* n'és la clau, però també podríem haver-ho deduït així:

- $\{work\} \rightarrow \{composer, yearComp\}$ ho podem afirmar pel coneixement que tenim del domini sobre el qual estem fent el disseny.
- Igualment, com a coneixedors del domini sabem que $\{composer\} \rightarrow \{bCentury, digitDegree\}$.

- I ara podem raonar que si *work* determina *composer*, *work* també determina els atributs determinats per *composer*, i deduïm, doncs, que *work* determina tots els altres atributs.

Les **regles de deducció d'Armstrong**, que permeten fer aquest i altres raonaments, són les que s'enumeren a continuació:

- Reflexivitat: $X \rightarrow X$
- Augmentativitat: Si $X \rightarrow Y$, aleshores $X \cup Z \rightarrow Y$
- Distributivitat: Si $X \rightarrow Y \cup Z$, aleshores $X \rightarrow Y$ i $X \rightarrow Z$
- Additivitat: Si $X \rightarrow Y$ i $X \rightarrow Z$, aleshores $X \rightarrow Y \cup Z$
- Transitivitat: Si $X \rightarrow Y$ i $Y \rightarrow Z$, aleshores $X \rightarrow Z$
- Pseudotransitivitat: Si $X \rightarrow Y$ i $Y \cup Z \rightarrow W$, aleshores $X \cup Z \rightarrow W$

Reflexió

Cal aclarir que aquest no és un conjunt mínim de regles, ja que algunes es poden deduir a partir de les altres. En tot cas, tampoc hi ha un conjunt mínim únic, sinó que podem triar diversos conjunts com a axiomes i demostrar les altres regles a partir d'aquests axiomes.

La **clausura transitiva** d'un conjunt de dependències D és el conjunt que s'obté si s'apliquen repetidament i de forma exhaustiva (fins que ja no es poden deduir dependències noves) les regles d'Armstrong. La clausura transitiva de D , que denotem amb D^+ , conté totes les dependències que són conseqüència de D i només aquestes.

Per això, la clausura d'un conjunt de dependències ens serveix per a:

- Confirmar o descartar una dependència que sospitem que es verifica.
- Trobar totes les claus candidates de les relacions. Si volem normalitzar fins a FNBC aquesta informació és imprescindible.
- Confirmar o descartar que dos esquemes lògics són equivalents. A partir de les dependències conegeudes de D_1 i D_2 , es pot dir que D_1 i D_2 són equivalents si es verifica que $D_1^+ = D_2^+$.

Reflexió

La normalització fins a FNBC és el grau de normalització mínim que es requereix si no volem que la nostra base de dades resulti afectada per anomalies.

Prenent com a base algunes o totes aquestes regles, s'han definit algorismes de normalització.

A continuació presentem dos algorismes que es poden veure com una aproximació a la generació automàtica d'un esquema lògic a partir dels atributs i les dependències d'un domini, que sorgeixen d'una activitat prèvia equivalent al disseny conceptual de la BD. El primer algorisme fa un procés descendent, de descomposició d'una relació potencialment molt gran que representa tota la informació del domini, mentre que el segon està plantejat des d'un punt de vista ascendent, d'agrupació de petites informacions elementals.

4.3.6. Algorisme d'anàlisi

La idea de l'algorisme d'anàlisi és partir d'una única relació, anomenada *relació universal*, que conté tots els atributs identificats. Fent servir les dependències que també s'han d'haver identificat prèviament, l'algorisme va partint la relació universal repetidament fins a obtenir un conjunt de relacions normalitzat. A cada pas de la partició es comprova si hi ha alguna relació que no està en la FNBC. Si no se'n troba cap, ja tenim un esquema normalitzat; altrament, es tria una de les relacions que no estan en la FNBC i es divideix en dues fent servir la dependència (o una qualsevol, si n'hi ha diverses) que viola la FNBC en aquella relació. Aquest procés es va repetint fins que arribem a la normalització. De fet, el procés és el que seguim intuïtivament quan normalitzem un esquema, amb la diferència que habitualment no partim d'una relació universal sinó d'un conjunt de relacions resultants d'un disseny previ.

4.3.7. Algorisme de síntesi

En aquest altre algorisme, el procés és ascendent. Es parteix de múltiples relacions petites i l'algorisme les va fusionant per a acabar amb un conjunt de relacions normalitzat i tan compacte com sigui possible. Aquestes relacions inicials provenen del que s'anomena *recobriment mínim de les dependències funcionals*, que es defineix com un conjunt de dependències més simple a partir del qual es poden deduir les dependències inicials. El recobriment mínim es genera en tres fases:

- 1) Desdoblar les dependències de manera que només hi hagi un atribut a la part dreta.
- 2) Simplificar els determinants eliminant-hi atributs superflus. Podrem eliminar els atributs que són determinats per altres atributs del mateix determinant en virtut de les dependències.
- 3) Eliminar dependències redundants (dependències que es poden deduir a partir d'altres dependències).

Cada una de les dependències del recobriment mínim dóna lloc a una de les múltiples relacions inicials; i aquestes relacions es van fusionant, agrupant les que comparteixen una clau candidata.

Exemple d'obtenció d'un algorisme de síntesi

Agafem, per exemple, les dependències següents:

- $\{codiObra\} \rightarrow \{nomObra\}$, $\{nomObra\} \rightarrow \{codiObra\}$
- $\{codiObra, nomObra\} \rightarrow \{compositor, segleNaix\}$
- $\{compositor\} \rightarrow \{segleNaix\}$
- $\{codiObra, enreg\} \rightarrow \{duradaObraEnreg\}$

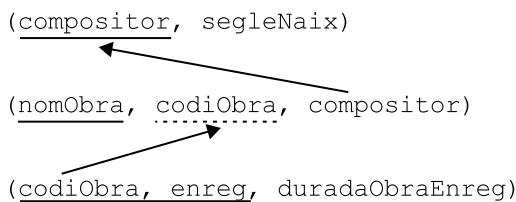
N'obtenim un recobriment mínim seguint els passos que hem descrit:

1) Hem de desdoblar $\{codiObra, nomObra\} \rightarrow \{compositor, segleNaix\}$ en $\{codiObra, nomObra\} \rightarrow \{compositor\}$ i $\{codiObra, nomObra\} \rightarrow \{segleNaix\}$.

2) Els determinants d'aquestes dues relacions que hem obtingut es poden simplificar. El resultat, entre altres possibilitats que tenim, és $\{codiObra\} \rightarrow \{compositor\}$ i $\{codiObra\} \rightarrow \{segleNaix\}$

3) Ara podem detectar que tenim una dependència redundant: $\{codiObra\} \rightarrow \{segleNaix\}$ es pot deduir a partir de $\{codiObra\} \rightarrow \{compositor\}$ i $\{compositor\} \rightarrow \{segleNaix\}$. El recobriment mínim que obtenim un cop fet aquest tercer pas està format per aquestes dependències: $\{codiObra\} \rightarrow \{nomObra\}$, $\{nomObra\} \rightarrow \{codiObra\}$, $\{codiObra\} \rightarrow \{compositor\}$, $\{compositor\} \rightarrow \{segleNaix\}$ i $\{codiObra, enreg\} \rightarrow \{duradaObraEnreg\}$.

Les dependències del recobriment donen lloc a aquestes relacions inicials: ($codiObra, nomObra$), ($nomObra, codiObra$), ($codiObra, compositor$), ($compositor, segleNaix$) i ($codiObra, enreg, duradaObraEnreg$). Les tres primeres relacions comparteixen una clau candidata: $codiObra$. Es fusionaran i l'esquema final obtingut per l'algorisme és:



4.3.8. Quarta forma normal

La condició que defineix aquesta forma normal no es basa en dependències funcionals sobre fets monovaluats, sinó que es basa en dependències sobre fets multivaluats que, si no es traslladen correctament a l'esquema lògic, fan aparèixer anomalies en relacions que estan en la FNBC.

Si provem de transformar un atribut multivaluat de l'esquema conceptual en un atribut no atòmic en un esquema relacional pensant que ja ho normalitzarem posteriorment, podem obtenir una relació en la FNBC que, no obstant això, presenta anomalies perquè no està en 4FN.

Vegem tot plegat amb un exemple: volem saber, per a cada director, quines orquestres ha dirigit i quines obres ha dirigit. Es tracta de dos fets multivaluats (un director pot haver dirigit moltes orquestres i pot haver dirigit moltes obres). Si ens imaginem una relació que admet atributs no atòmics, podem pensar en una situació com la que descriu la figura 41.

Figura 41. Una relació amb atributs no atòmics

Conductors		
conductor	works	orchestras
C. Abbado	Symphony 9 Symphony 5	OCB London SO
H. von Karajan	Symphony 9 Conc Piano 1	Berliner Ph OCB

Reflexió

Els fets multivaluats apareixen amb naturalitat en un esquema conceptual com a tipus de relació amb multiplicitat *, però no es poden representar com a atributs multivaluats en l'esquema lògic perquè el model relacional no admet atributs no atòmics.

Com que sabem que això ho podem imaginar però no ho podem definir en un SGBD relacional, aplanem la relació per a obtenir la relació que descriu la figura 42. Aquesta relació no té cap dependència, a part de la dependència trivial reflexiva $\{conductor, works, orchestras\} \rightarrow \{conductor, works, orchestras\}$ i similars.

Figura 42. La relació després d'aplanar els atributs

Conductions		
<u>conductor</u>	<u>works</u>	<u>orchestras</u>
C. Abbado	Symphony 9	OCB
C. Abbado	Symphony 9	London SO
C. Abbado	Symphony 5	OCB
C. Abbado	Symphony 5	London SO
H. von Karajan	Symphony 9	Berliner Ph
H. von Karajan	Symphony 9	OCB
H. von Karajan	Conc Piano 1	Berliner Ph
H. von Karajan	Conc Piano 1	OCB

La relació està, doncs, en la FNBC. És clar, però, que amaga redundàncies que provoquen anomalies. Per exemple, si volem registrar que *Claudio Abbado* ha dirigit la *Filharmònica de Berlín*, haurem d'afegir dues tuples:

C. Abbado	Symphony 9	Berliner Ph
C. Abbado	Symphony 5	Berliner Ph

El problema, com en els casos presentats en les formes normals anteriors, és que en una única relació estem barrejant dos fets: les orquestres dirigides i les obres dirigides. La condició que revela aquesta barreja de fets és el que anomenem *dependència multivaluada independent*, que definim formalment a continuació.

Siguin X i Y subconjunts dels atributs d'una relació amb el conjunt d'atributs R . La **dependència multivaluada independent** de Y respecte de X , que denotem per $X \twoheadrightarrow Y$, es verifica si per a tot parell de tuples de la relació que tenen el mateix valor en X i diferent en Y , hi ha dues tuples com aquestes que intercanvien els valors de $R - X - Y$. És a dir, si existeixen $\langle x, y_1, z_1 \rangle$ i $\langle x, y_2, z_2 \rangle$, també han d'existir $\langle x, y_1, z_2 \rangle$ i $\langle x, y_2, z_1 \rangle$ (essent x valors possibles per a X , y_1 i y_2 valors possibles per a Y i z_1 i z_2 valors possibles per a $R - X - Y$). Quan es verifica $X \twoheadrightarrow Y$ també es verifica $X \twoheadrightarrow R - Y$.

Efectivament, doncs, tenim les dependències $\{director\} \rightarrow\rightarrow \{obres\}$ i $\{director\} \rightarrow\rightarrow \{orquesters\}$ en la relació *Direccions*. Per exemple, per a les tuples $\langle C. Abbado, Simfonia 9, London SO \rangle$ i $\langle C. Abbado, Simfonia 5, OCB \rangle$ existeixen les tuples $\langle C. Abbado, Simfonia 9, OCB \rangle$ i $\langle C. Abbado, Simfonia 5, London SO \rangle$.

Ara ja estem en condicions de definir la quarta forma normal.

Reflexió

Fixem-nos que les dependències funcionals són un cas particular de les multivaluades en què un mateix valor de X només pot aparèixer amb un sol valor de Y . És a dir, si $\{X\} \rightarrow \{Y\}$ aleshores $\{X\} \rightarrow\rightarrow \{Y\}$.

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

Reflexió

Fixeu-vos que per la definició de dependència multivaluada independent, una relació amb només un o dos atributs no pot violar la condició de 4FN.

La relació *Conductions* no està en 4FN perquè, com ja hem vist, presenta dependències multivaluades independents degudes a la barreja de fets en una mateixa relació. Per això, la solució torna a consistir en la separació dels dos fets en dues relacions. La relació de la figura 42, una vegada normalitzada, es converteix en les que es presenten en la figura 43.

Figura 43. Dues relacions normalitzades

ConductsWork		ConductsOrc	
<u>conductor</u>	<u>work</u>	<u>conductor</u>	<u>orchestra</u>
C. Abbado	Symphony 9	C. Abbado	OCB
C. Abbado	Symphony 5	C. Abbado	London SO
H. von Karajan	Symphony 9	H. von Karajan	OCB
H. von Karajan	Conc Piano 1	H. von Karajan	Berliner Ph

Si recapitulem el procés que ens ha dut fins aquí, recordarem que tot ha començat amb uns atributs multivaluats que hem representat en una sola relació.

Si haguéssim fet una traducció sistemàtica de l'esquema conceptual hauríem obtingut directament l'esquema lògic correcte.

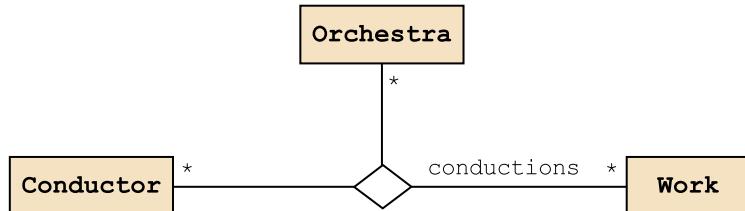
L'esquema conceptual de partida es mostra en la figura 44.

Figura 44. Els tipus de relacions binàries d'on provenen les relacions



La traducció dels tipus de relació *conductsWork* i *conductsOrc* duu directament a les relacions normalitzades de la figura 43. La relació no normalitzada de la figura 42 podria ser el resultat de traduir un esquema conceptual diferent, que presentem en la figura 45.

Figura 45. El tipus de relació ternària d'on prové la relació



Però aleshores no existirien les dependències multivaluades perquè, ara, el que volem és registrar un únic fet ternari: quines obres ha dirigit cada director i amb quina orquestra ha dirigit cada obra. En aquest cas, la relació *Conductions*, en no presentar les dependències multivaluades, estaria en 4FN. I ara podria tenir una extensió com la que es mostra en la figura 46, que abans no era vàlida d'acord amb les dependències.

Figura 46. L'extensió de la relació corresponent al tipus de relació ternària

Conductions		
<u>conductor</u>	<u>work</u>	<u>orchestra</u>
C. Abbado	Symphony 9	OCB
C. Abbado	Symphony 5	London SO
H. von Karajan	Symphony 9	Berliner Ph
H. von Karajan	Conc Piano 1	OCB

Aquesta relació no presenta anomalies, perquè ara no té sentit dir que volem afegir, per exemple, que Claudio Abbado ha dirigit la Filharmònica de Berlín. Ara el fet bàsic és que Claudio Abbado ha dirigit la Filharmònica de Berlín interpretant una obra concreta.

4.3.9. Cinquena forma normal

Hi ha encara un darrer cas de taules que barregen fets i que la 4FN no detecta. Prenem el cas de la figura 46 i suposem que hi ha una llei de simetria en el domini que afirma que si un director ha dirigit una obra i aquesta obra l'ha interpretada una orquestra i el director ha dirigit aquesta orquestra, aleshores el director ha dirigit l'obra amb aquesta orquestra. En aquest cas, l'extensió de la figura 46 no compliria aquesta llei; hi faltaría la tupla

H. von Karajan	Symphony 9	OCB
----------------	------------	-----

Això és així perquè *H. von Karajan* ha dirigit la *Sinfonia 9* (segons la tercera tupla), l'*OCB* ha interpretat la *Sinfonia 9* (segons la primera tupla) i *H. von Karajan* ha dirigit l'*OCB* (segons la quarta tupla). D'altra banda, afegir aquesta nova tupla no fa que la relació deixi d'estar en 4FN.

En casos així, la relació torna a presentar anomalies. Per exemple, si un director passa a dirigir una nova obra, per la llei de simetria hem d'afegir tantes tuples com orquestres que han interpretat l'obra hagi dirigit el director. Si provem d'eliminar les anomalies descomponent la relació en dues, tenim tres opcions de descomposició. Fixem-nos que cap de les opcions és vàlida:

- a) La descomposició següent no és vàlida perquè no permet saber quines orquestres han dirigit els directors (podríem pensar que Claudio Abbado ha dirigit la Filharmònica de Berlín).

CondWork		WorkOrc	
<u>conductor</u>	<u>work</u>	<u>work</u>	<u>orchestra</u>
C. Abbado	Symphony 9	Symphony 9	OCB
C. Abbado	Symphony 5	Symphony 5	London SO
H. von Karajan	Symphony 9	Symphony 9	Berliner Ph
H. von Karajan	Conc Piano 1	Conc Piano 1	OCB

- b) La descomposició següent no és vàlida perquè no permet saber quines obres ha interpretat cada orquestra (podríem pensar que la Simfònica de Londres ha interpretat la simfonía 9).

CondWork		CondOrc	
<u>conductor</u>	<u>work</u>	<u>conductor</u>	<u>orchestra</u>
C. Abbado	Symphony 9	C. Abbado	OCB
C. Abbado	Symphony 5	C. Abbado	London SO
H. von Karajan	Symphony 9	H. von Karajan	Berliner Ph
H. von Karajan	Conc Piano 1	H. von Karajan	OCB

- c) La descomposició següent no és vàlida perquè no permet saber quines obres han dirigit els directors (podríem pensar que Claudio Abbado ha dirigit el concert per a piano 1).

WorkOrc		CondOrc	
<u>work</u>	<u>orchestra</u>	<u>conductor</u>	<u>orchestra</u>
Symphony 9	OCB	C. Abbado	OCB
Symphony 5	London SO	C. Abbado	London SO
Symphony 9	Berliner Ph	H. von Karajan	Berliner Ph
Conc Piano 1	OCB	H. von Karajan	OCB

Observem, doncs, que no podem representar tres fets amb dues relacions; però què passa si prenem les tres relacions *CondWork*, *WorkOrc* i *CondOrc*?

Si descomponem la relació original en aquestes tres altres relacions, tenim la mateixa informació, perquè podem obtenir la relació original fent la combinació natural de les altres tres. Es pot demostrar que això és així per a qual-

sevol extensió que compleix la llei de simetria. Aquesta propietat es coneix amb el nom de **dependència de projecció-combinació** i és en el que es basa la definició de cinquena forma normal:

Una relació està en **cinquena forma normal (5FN)** si, i només si, està en 4FN i no presenta dependències de projecció-combinació.

Així, per a saber si una relació que està en 4FN també està en 5FN hem de projectar i combinar les projeccions i, seguidament, comprovar si obtenim la mateixa relació original. En cas contrari, podem assegurar que no hi ha dependència de projecció-combinació i que la relació està en 5FN. Altrament, hem d'estudiar si existeix una llei de simetria que comporti que, per a tota extensió de la relació original, el procés de projecció i combinació ens tornarà a donar la relació original o si això només passa en alguns casos.

Reflexió

A la vista de la definició, podem afirmar que una relació amb només un o dos atributs no pot violar la condició de 5FN.

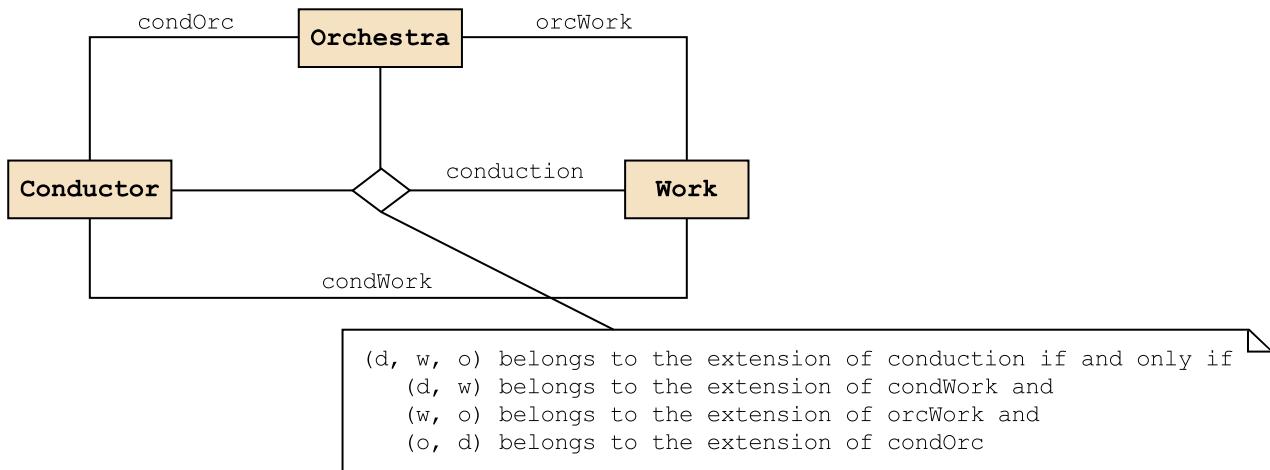
La relació *Conductions*, en el moment d'incorporar la llei de simetria, no està en 5FN. Per això, partint de l'extensió de la figura 46 i afegint-hi la tupla *<H. von Karajan, Simfonia 9, OCB>* que ja hem dit que faltava per complir aquesta llei, si projectem i combinem, tornem a l'extensió de partida. En canvi, si ara suposem que no hi ha llei de simetria i que l'extensió de *Conductions* és la de la figura 46, fent el procés de projecció i combinació no obtindríem l'extensió de partida, sinó que obtindríem la tupla addicional *<H. von Karajan, Simfonia 9, OCB>* i podríem dir immediatament que la relació està en 5FN.

Si reprenem el cas des del començament veurem que, si partim de l'esquema conceptual correcte i el traduïm de forma correcta al model relacional, obtindrem directament l'esquema lògic normalitzat.

Quan no hi ha la llei de dependència, estem davant d'un tipus de relació ternària (la que es mostra en la figura 45) que, per tant, esdevé una sola relació que està normalitzada. Quan afegim la llei de simetria, el que estem introduint són tres tipus de relacions binàries i una de ternària que es deriva de les tres binàries com es mostra en la figura 47.

A partir d'aquest esquema conceptual, el que hem de fer és traduir, no el tipus de relació derivat (que donaria lloc a la relació que no està en 5FN), sinó les tres binàries (que donen lloc a les tres relacions que sí que estan en 5FN).

Figura 47. Els tipus de relacions binàries d'on provenen les relacions normalitzades i el tipus de relació ternària derivada



4.4. Pràctica de la normalització

El procés de normalització té, en general, efectes beneficiosos per a la base de dades i ho fa preservant la semàntica de l'esquema de partida. El procés sempre és factible i existeixen algorismes que normalitzen fins a la FNBC. Un mateix esquema de partida es pot normalitzar, en general, de més d'una manera (tant si ho fem de forma manual com si fem servir algun algorisme) i és feina del dissenyador triar l'alternativa que més interessa en cada cas.

Els efectes beneficiosos d'aplicar les formes normals són l'eliminació de redundàncies i anomalies i la clarificació de l'esquema, separant els fets diferents que potser es troben barrejats en l'esquema inicial.

El moment i l'oportunitat de l'aplicació de la normalització poden ser diversos en funció del context en què es realitza el disseny de la base de dades. Si ens trobem en una situació de desenvolupament a partir d'un sistema preexistent (o d'una part), pot ser interessant aplicar la normalització abans de començar si no la tenim garantida. Si ens trobem modificant un esquema lògic del qual no tenim la documentació corresponent a l'esquema conceptual, la normalització *a posteriori* és l'única manera de tenir la certesa que el resultat està normalitzat. En canvi, si partim de zero i disposem d'un esquema conceptual correcte, una traducció ben feta al model relacional ens assegura un esquema lògic normalitzat. Tot i així, com que l'esquema conceptual de partida pot ser incorrecte i en el procés de traducció podem cometre errors, és molt recomanable comprovar si el resultat final està normalitzat. Per exemple, si hem introduït en l'esquema un tipus de relació ternària en comptes de dues o tres de binàries, pot ser que això hagi provocat l'aparició d'alguna relació que no està en 4FN o 5FN.

La normalització té com a conseqüència no desitjada la penalització de la recuperació de la base de dades. Com que descompon les relacions, separa atributs que en un esquema no normalitzat estarien en la mateixa relació. Això implica la necessitat d'executar més operacions de combinació per a obtenir informació que en l'esquema no normalitzat obtindríem consultant una única relació.

En general, però, podem assumir aquest desavantatge a canvi d'eliminar les anomalies i facilitar la integritat de la base de dades. Hi ha, però, casos o parts de les bases de dades en què les actualitzacions són poc freqüents. En aquests casos, l'equilibri entre avantatges i inconvenients pot fer-nos decantar per l'opció de no normalitzar.

La **desnormalització** és el procés de desfer la normalització agrupant dades lògicament independents o afegint redundància a la base de dades amb l'objectiu de fer més eficients les consultes.

Cal tenir present que aquest procés penalitza les actualitzacions de la base de dades i requereix un disseny molt acurat de les restriccions per tal de garantir que la base de dades es manté consistent. Es requereix, per tant, una anàlisi acurada d'avantatges i inconvenients de la desnormalització en cada cas. Un cas en què la desnormalització surt més a compte és el de bases de dades dedicades només a consultar dades històriques agregades per diferents conceptes.

Una opció a mig camí de la desnormalització és la definició de vistes materialitzades, addicionals al disseny normalitzat, que ajudin a accelerar les consultes més freqüents o costoses.

Resum

En aquest mòdul hem estudiat el procés de disseny lògic com una de les etapes del desenvolupament d'una base de dades per a un sistema d'informació. Aquest procés parteix de l'esquema conceptual que hem obtingut en la fase d'especificació i dóna com a resultat l'esquema lògic de la base de dades.

Hem començat l'estudi identificant els paranyos de disseny, possibles errors que es poden haver comès en el disseny conceptual que convé repassar abans de prendre'l com a punt de partida del disseny lògic.

L'etapa de disseny lògic l'hem tractat en el cas particular de transformació de models conceptuais basats en el llenguatge UML a models lògics relacionals. Hem analitzat cadascun dels elements de l'esquema conceptual (tipus d'entitats, tipus de relacions, generalitzacions i tipus d'entitats associatives) i hem donat pautes de transformació per a cadascun. Hem destriat cada element en els diversos casos en què es pot dividir i hem donat la solució per a cadascun, sense deixar de banda les restriccions. En alguns casos hem presentat diverses alternatives de transformació i hem fet èmfasi en el fet que moltes vegades la millor alternativa serà la que eviti la presència de valors nuls.

Finalment, hem estudiat la teoria de la normalització, que permet assegurar que l'esquema relacional satisfà una sèrie de condicions que garanteixen una millor qualitat de la base de dades. Hem identificat les anomalies que es poden produir en una base de dades no normalitzada i hem definit un seguit de formes normals mitjançant condicions que, si són satisfetes per la base de dades, ens garanteixen l'absència d'anomalies. Hem acabat fent una reflexió sobre els avantatges i els inconvenients de la normalització i fent una breu introducció al concepte de *desnormalització*.

Glossari

abraçada mortal de càrrega *f* Impossibilitat d'inserir tuples en cap taula d'un conjunt de taules perquè les claus foranes que contenen formen un cicle.

abraçada mortal de definició *f* Impossibilitat de definir un conjunt de taules perquè les claus foranes que contenen formen un cicle.

anomalia d'actualització *f* Necessitat d'actualitzar moltes tuples per a reflectir un canvi elemental.

clausura transitiva d'un conjunt de dependències *f* Conjunt de totes les dependències que es poden deduir aplicant reiteradament les regles d'Armstrong a partir del conjunt inicial.

dependència funcional *f* En una relació, diem que un conjunt d'atributs Y depèn d'un conjunt d'atributs X ($X \rightarrow Y$) si sempre que dues tuples tenen els mateixos valors en els atributs de X , també tenen els mateixos valors en els atributs de Y .

dependència funcional completa *f* $X \rightarrow Y$ és completa si no existeix cap X' subconjunt propi de X tal que $X' \rightarrow Y$.

dependència multivaluada independent *f* Diem que hi ha una dependència multivaluada independent de Y respecte a X , que denotem per $X \rightarrow\rightarrow Y$, si es verifica que per a tot parell de tuples de la relació que tenen el mateix valor en X i diferent en Y , hi ha dues tuples com aquestes que intercanvien els valors de $R - X - Y$.

dependència projecció-combinació *f* Diem que una relació amb tres atributs presenta una dependència de projecció-combinació si, per a qualsevol extensió correcta de la relació, es verifica que com a resultat de descompondre-la en tres altres relacions de dos atributs (fent les corresponents projeccions) i combinar aquestes tres relacions a continuació, obtenim de bell nou la relació original.

desnormalització *f* Procés consistent a desfer la normalització agrupant dades lògicament independents o afegint redundància a la base de dades, amb l'objectiu de fer més eficients les consultes.

determinant *m* Conjunt d'atributs X d'una dependència funcional $X \rightarrow Y$.

disseny lògic *m* Procés de transformació d'un esquema conceptual en un esquema lògic de base de dades.

forma normal *f* Diem que una relació està en una determinada forma normal si satisfa les condicions fixades per aquella forma normal. Les formes normals són inclusives: la condició d'una forma normal de nivell superior implica la condició de cadascun dels nivells inferiors i, per tant, si una relació està en una forma normal també està en totes les formes normals de nivell inferior.

normalització *f* Procés pel qual, a partir d'un conjunt de relacions, s'obté un conjunt de relacions equivalent que satisfa la condició de la forma normal desitjada.

parany de disseny *m* Patró de l'esquema conceptual que pot induir a cometre errades en la interpretació del món real.

recobriment mínim *m* Conjunt més simple d'un altre conjunt a partir del qual es poden deduir les dependències del conjunt original.

regles d'Armstrong *fpl* Conjunt de regles de deducció que permeten demostrar si una dependència és conseqüència d'unes altres.

restricció *f* Condició que limita les extensions vàlides d'una relació.

Bibliografia

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

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

Gulutzan, P.; Pelzer, T. (1999). *SQL-99 Complete, really*. R&D Books.

Ramakrishnan, Raghu; Gehrke, Johannes (2003). *Database management systems* (3a ed.). Boston: McGraw-Hill Higher Education.

Sciore, E. (2008). *Database Design and Implementation*. Wiley.

Unified Modeling Language: Superstructure (versió 2.0, OMG doc. formal/05-07-04). Accessible en línia des del web www.omg.org.