Disseny lògic de bases de dades

Xavier Burgués Illa

3. NORMALITZACIÓ

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

En les seccions anteriors hem vist alguns criteris per triar o descartar determinades opcions (per exemple, afavorir determinades operacions en front d'altres o evitar l'existència de valors nuls). En aquest apartat veurem les condicions de normalització. Aquestes condicions garanteixen que la base de dades està dissenyada de tal manera que no es barregen 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, previ repàs de conceptes del model relacional i de l'àlgebra de conjunts, presentarem la teoria de normalització i anirem veient com la podem aplicar als esquemes lògics. Veurem com aquesta aplicació elimina les anomalies de la base de dades.

3.1 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 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.

Imaginem, per exemple, 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(<u>work</u>, composer, yearComp, bCentury, digitDegree)

De les obres es registra l'any de composició (a 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 (d*igitDegree*). En un moment determinat, la relació podria tenir l'extensió que representa la figura 3.1.1.

Compositions				
work	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

Fig. 3.1.1. Extensió d'una relació amb redundàncies i anomalies

Si volem afegir un nou compositor, anomenat *Montsalvatge*, 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.

Per 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 3.1.2.

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

Fig. 3.1.2. Extensió de la relació després d'una supressió

Observeu que, de forma 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 reflectir el canvi d'un sol fet elemental es diu que es produeix una anomalia de modificació.

Per 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 com es mostra en la figura 3.1.3.

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

Fig. 3.1.3 Extensió de la relació després d'una modificació

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

L'origen de les anomalies de inserció, eliminació i actualització que acabem 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 dóna 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 3.1.4.

Work		
wName	composer	yearComp
Symphony 9	Mahler	1923
Symphony 5	Mahler	1918
Parsifal	Wagner	1857
Conc Piano 3	Mozart	1779

Composer		
cName	bCentury	digitDegree
Mahler	19	70
Wagner	19	42
Mozart	18	42

Fig. 3.1.4 Extensió de dues relacions normalitzades

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

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

Donada una relació $R(A_1, A_2, \ldots, A_n)$, podem declarar $X \rightarrow Y$ com a dependència funcional si $X, Y \subset \{A_1, A_2, \ldots, A_n\}$. Per satisfer aquesta restricció, l'extensió de la relació ha de ser tal que *X* determini de manera única el valor de *Y*; és a dir, si dues tuples tenen els mateixos valors als atributs *X*, també tenen els mateixos valors als atributs de *Y*. En aquest cas diem que *Y* depèn funcionalment de *X* o, alternativament, que *X* determina funcionalment *Y* (i, per això, *X* és anomenat el determinant de la dependència). O sigui, 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.

Per acabar de presentar aquest concepte, l'il.lustrem amb un exemple. Analitzem la relació:

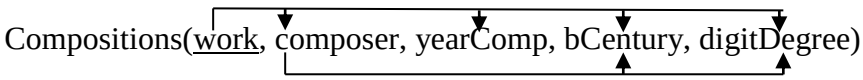
Compositions(<u>work</u>, composer, yearComp, bCentury, digitDegree)

per identificar-hi dependències funcionals. Hi trobem les següents:

- {work} → {composer, yearComp, bCentury, digitDegree}. Aquesta dependència correspon a la clau primària.
- {composer} → {bCentury, 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 totes dues vegades amb els mateixos valors per als atributs *bCentury* i *digitDegree*.

En canvi, {composer} → {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 denotar les dependències que hi ha a una relació, farem servir una notació a base de fletxes, tal i com mostra en següent exemple:



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 plena 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 plena perquè hi ha la segona i es verifica que {work} és un subconjunt propi de {work, composer}.

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

3.3 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 recolzen en gran mesura en el concepte de dependència funcional plena que acabem de presentar en l'apartat 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 eliminen 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 poder modificar una relació que no està en una determinada forma normal, amb l'objectiu de que hi estigui. Aquest procés de modificació s'anomena normalització. Com anirem veient, una mateixa relació no normalitzada pot ser modificada 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 darerra (5FN) en la dependència de projecció-combinació.

Aquests dos darrers tipus de dependències seran presentats en el moment de definir les formes normals corresponents.

Tractarem en un primer bloc les quatre primeres formes normals per passar seguidament a presentar un parell d'algoritmes capaços de normalitzar a aquest nivell. Finalment, veurem les dues darreres formes normals.

3.3.1 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 il.lustrar aquest concepte, fixem-nos en la figura 3.3.1 on hi tenim una relació que permet recollir informació dels compositors.

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

Fig. 3.3.1. Una relació amb atributs no atòmics

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

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

A partir de la relació de la figura 3.3.1 obtenim la relació de la figura 3.3.2.

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

Fig. 3.3.2 La relació després d'aplanar els atributs

Cal notar 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 com que {work} → {composer}, la clau primària està formada exclusivament per l'atribut *work*.

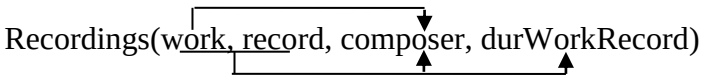
Cal tenir present que el model relacional, i els SGBD relacionals, ja garateixen aquesta primera forma normal en qualsevol relació que hi poguem definir.

3.3.2 Segona forma normal.

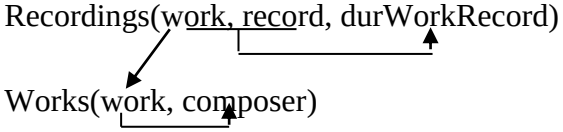
Una relació està en **segona forma normal (2FN)** si, i només si, està en primera forma normal i tot atribut que no forma part d'una clau candidata depèn plenament de totes les claus candidates de la relació.

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, la dependència segur que és plena.

Com a exemple d'aquesta problemàtica considerem la relació següent, que emmgatzema quines obres hi ha a cadascun dels enregistraments (discs, cintes, arxius mp3, ...) que tenim. Una mateixa obra (*work*) pot estar repetida a 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} → {composer}. L'exemple ens permet veure amb claredat quin és l'objectiu de la segona forma normal: impedir que barregem dos fets elementals que comparteixen part de la clau (els enregistraments de les obres que tenim, per una banda, i els compositors de les obres, per una altra) en una mateixa relació. D'aquesta manera evitarem redundàncies com la de tenir replicat el compositor d'una obra tantes vegades com enregistraments tenim que contenen l'obra. Per 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:

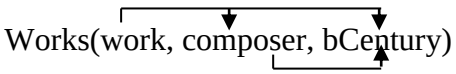


Com es pot observar, les dues relacions resultants les hem de lligar amb una clau forana a través de la part de la clau compartida pels dos fets. Cal no confondre les fletxes que representen dependències amb la que representa aquesta clau forana.

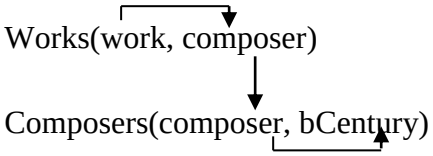
3.3.3 Tercera forma normal.

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 {composer} → {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 a la relació. Per 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:

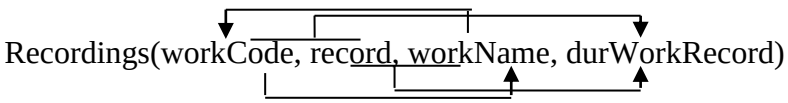


3.3.4 Forma normal de Boyce-Codd.

Una relació està en la **forma normal de Boyce-Codd (FNBC)** si, i només si, està en 1FN i els determinants de totes les dependències que presenta la relació en són claus candidates.

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

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



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 de l'obra hi hagi. 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:

I. {workCode} → {workName}
II. {workName} → {workCode}
III. {workCode, record} → {durWorkRecord}
IV. {workName, record} → {durWorkRecord}

Els determinants de les dues darreres són claus candidates de la relació però els determinant de les dues primeres no. Per tant, la relació no està en FNBC. La figura 3.3.3 mostra una possible extensió de la relació

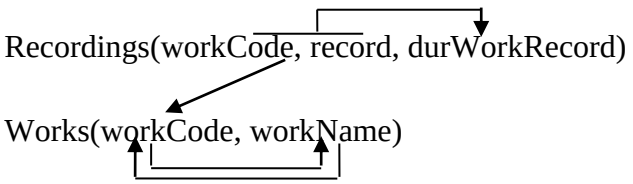
Recordings			
workCode	record	workName	durWorkRecord
MahS5	DeutscheGrammophon001	Symphony 5	52
MahS9	Decca036	Symphony 9	43
MahS5	Naxos201	Symphony 5	55
MahS5	Sony187	Symphony 5	51

Fig. 3.3.3. Exemple de relació que compleix la 3FN però no la FNBC

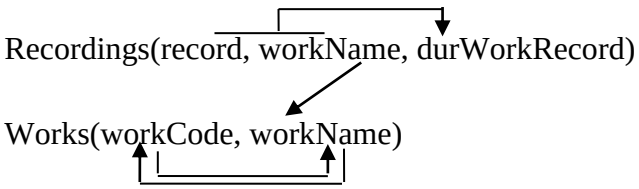
Fixeu-vos que, efectivament, es poden produir anomalies. Per exemple, si volem canviar el nom de la Simfonia 5, haurem de modificar tres tuples.

Per normalitzar a 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:

i. Eliminant {workCode} → {workName}



ii. Eliminant {workName} → {workCode}



Ens decidirem per una o altra opció en funció de si preferim accedir més aviat per codi o per nom de l'obra. Independenment de la 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).

3.3.5 Regles d'Armstrong.

Regles d'Armstrong. Tal i com ja hem comentat, un cop vistes les quatre primeres formes normals, presentarem dos algoritmes capaços de normalitzar un esquema relacional fins a FNBC. Aquests algoritmes 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 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 que 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*, deduint, 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 $X \cup Z \rightarrow W$

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 a partir dels axiomes les altres regles.

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:

- 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 conegudes de D_1 i D_2, es pot dir que D_1 i D_2 són equivalents si es verifica que $D_1^+ = D_2^+$.
- Prenent com a base algunes o totes d'aquestes regles, s'han definit algoritmes de normalització.

3.3.6 Algoritme d'anàlisi.

Algoritme d'anàlisi. La idea de l'algoritme 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'algoritme 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 FNBC. Si no en trobem cap, ja tenim un esquema normalitzat. Altrament, es tria una de les relacions que no estan en FNBC i es divideix en dues fent servir la dependència (o una qualsevol d'elles, si n'hi ha diverses) que viola FNBC en aquella relació. I anem repetint aquest procés fins que arribem a la normalització. Aquest procés és, de fet, el que intuïtivament seguim quan normalitzem un esquema, amb la diferència que habitualment partim no d'una relació universal sinó d'un conjunt de relacions resultants d'un disseny previ. Evidentment, podem iniciar l'algoritme partint d'un conjunt de relacions si ens convé més que iniciar-lo a partir d'una sola relació.

3.4. 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 algoritmes que normalitzen fins a FNBC. Un mateix esquema de partida es pot nomalitzar, en general, de més d'una manera (tant si ho fem de forma manual com si fem servir algun algoritme) 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 recomenable comprovar si el resultat final està normalitzat. Per exemple, si hem introduït a l'esquema un tipus de relació ternari 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 d'informació de la base de dades. Com que descomposa les relacions, separa atributs que en un esquema no normalitzat estarien a la mateixa relació. Això implica la necessitat d'executar més operacions de combinació per 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 el balanç 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 garantir que la base de dades es manté consistent. Es requereix, per tant, una anàlisi acurat 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 i/o costoses.