

Capitolul 5. A treia formă normalizată și forma normală Boyce-Codd

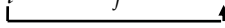
A treia formă normalizată (3NF) prezintă un interes aparte, deoarece este considerată de mulți specialiști drept un minimum acceptabil pentru structura unei baze de date relaționale. Bazată pe a doua formă normală și dependențele funcționale tranzitive (de fapt, pe eliminarea lor), 3NF a cunoscut două definiții majore, una "originală", formulată de Codd însuși (cu variantele sale "folclorice") și a doua, ce acoperă câteva lipsuri ale primeia, numită, după autorii săi, *Boyce-Codd Normal Form*, în traducere oarecum liberă *forma normală Boyce-Codd*, mai pe românește, *BCNF*.

5.1. Dependențe funcționale directe/tranzitive

O dependență funcțională $Da\breve{t}a1 \longrightarrow Da\breve{t}a2$ este directă atunci când nu există o $Da\breve{t}a3$ care angajează o dependență funcțională tranzitivă de genul:

$$Da\breve{t}a1 \longrightarrow Da\breve{t}a3 \longrightarrow Da\breve{t}a2.$$

Formal, dacă $A_i \longrightarrow A_j$ și $A_j \longrightarrow A_k$ atunci $A_i \longrightarrow A_k$ și se notează

$$A_i \longrightarrow A_j \longrightarrow A_k.$$


Pentru relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 (figura 3.7.), dependențele (7), (8), (9) și (10) din paragraful 4.1.2, sunt tranzitive deoarece:

$$Cota \longrightarrow ISBN \longrightarrow Titlu$$

$$Cota \longrightarrow ISBN \longrightarrow Editura$$

$$Cota \longrightarrow ISBN \longrightarrow LocSediuEd$$

$$Cota \longrightarrow ISBN \longrightarrow AnApariție$$

În plus și DF (13) este tranzitivă, întrucât:

$$ISBN \longrightarrow Editura \longrightarrow LocSediuEd$$

Pentru relația VÎNZĂRI dependențele tranzitive simple sunt mai numeroase:

$$CodPost \longrightarrow Jud \longrightarrow Județ$$

$$CodPost \longrightarrow Jud \longrightarrow Regiune$$

$$CodCl \longrightarrow CodPost \longrightarrow Localitate$$

$$CodCl \longrightarrow CodPost \longrightarrow Comuna$$

$$CodCl \longrightarrow CodPost \longrightarrow Jud \longrightarrow Județ$$

$$CodCl \longrightarrow CodPost \longrightarrow Jud \longrightarrow Regiune$$

$$NrFact \longrightarrow CodCl \longrightarrow DenCl$$

$$NrFact \longrightarrow CodCl \longrightarrow CodFiscal$$

$$NrFact \longrightarrow CodCl \longrightarrow StradaCl$$

$$\begin{array}{l}
 \text{NrFact} \longrightarrow \text{CodCl} \longrightarrow \text{NrStradaCl} \\
 \text{NrFact} \longrightarrow \text{CodCl} \longrightarrow \text{BlocScApCl} \\
 \text{NrFact} \longrightarrow \text{CodCl} \longrightarrow \text{TelefonCl} \\
 \text{NrFact} \longrightarrow \text{CodCl} \longrightarrow \text{CodPost} \\
 \text{NrFact} \longrightarrow \text{CodCl} \longrightarrow \text{CodPost} \longrightarrow \text{Localitate} \\
 \dots
 \end{array}$$

dar și

$$\begin{array}{l}
 (\text{NrFact}, \text{Linie}) \longrightarrow \text{CodPr} \longrightarrow \text{DenPr} \\
 (\text{NrFact}, \text{Linie}) \longrightarrow \text{CodPr} \longrightarrow \text{UM} \\
 (\text{NrFact}, \text{Linie}) \longrightarrow \text{CodPr} \longrightarrow \text{ProcTVA} \\
 (\text{NrFact}, \text{Linie}) \longrightarrow \text{CodPr} \longrightarrow \text{Grupa}
 \end{array}$$

Pe baza noțiunilor prezentate se poate defini *închiderea tranzitivă* a unui ansamblu de dependențe funcționale. *Închidere tranzitivă* reprezintă ansamblul *dependențelor totale directe + dependențele totale tranzitive*. *Acoperirea minimală*, definită în capitolul anterior, este subansamblul minimal de dependențe funcționale elementare care permit determinarea tuturor celorlalte dependențe funcționale. Cu alte cuvinte: (1) nici o dependență funcțională nu este redundantă și (2) orice dependență funcțională totală face parte din închiderea tranzitivă. Spre deosebire de ceea ce am discutat în capitolul anterior, acum suntem în măsură să eliminăm din ansamblul DF și pe cele tranzitive.

Astfel, pentru relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3, acoperirea minimală este alcătuită din patru DF:

- Cota \longrightarrow ISBN
- ISBN \longrightarrow Titlu
- ISBN \longrightarrow Editura
- ISBN \longrightarrow AnApariție
- Editura \longrightarrow LocSediuEd

iar închiderea tranzitivă este următorul ansamblu de dependențe:

- Cota \longrightarrow ISBN
- Cota \longrightarrow Titlu
- Cota \longrightarrow Editura
- Cota \longrightarrow LocSediuEd
- Cota \longrightarrow AnApariție
- ISBN \longrightarrow Titlu
- ISBN \longrightarrow Editura
- ISBN \longrightarrow LocSediuEd
- ISBN \longrightarrow AnApariție
- Editura \longrightarrow LocSediuEd

Acoperirea minimală poate fi ușor reprezentată prin graful DF. Dependențele tranzitive sunt foarte ușor de identificat și eliminat. Spre exemplu, dacă din figura 4.8 eliminăm săgețile ce reprezintă drumul direct dintre două atribute/grupe și un atribut destinație, atâta timp cât între respectivele atribute/conectori există un “traseu ocolitor”, se obține graful acoperirii minimale – vezi figura 5.1.

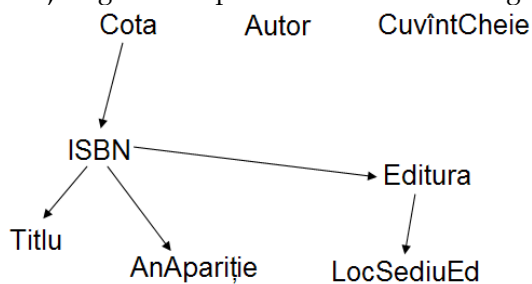


Figura 5.1. Graful DF din acoperirea minimală pentru relația
BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

Este drept, atributele Autor și CuvîntCheie atârnă inutil în figură, stricând imaginea grafului. Le-am păstrat însă pentru a nu omite nici un atribut din relația universală și de a indica cheia primară a relației.

Pentru baza de date VÎNZĂRI acoperirea minimală, adică setul inițial de DF din care au fost eliminate dependențele simetrice (redundante), parțiale și tranzitive este următorul set de dependențe funcționale reprezentat în figura 4.9.

- Jud → Județ
- Jud → Regiune
- CodPost → Localitate
- CodPost → Comună
- CodPost → Jud
- CodCl → DenCl
- CodCl → CodFiscal
- CodCl → StradaCl
- CodCl → NrStradaCl
- CodCl → BlocScApCl
- CodCl → TelefonCl
- CodCl → CodPost
- CodPr → DenPr
- CodPr → UM
- CodPr → Grupa
- CodPr → ProctVA
- NrFact → CodCl
- NrFact → DataFact
- NrFact → Obs
- (NrFact, Linie) → Cantitate

- (NrFact, Linie) \longrightarrow PrețUnit
- CodÎnc \longrightarrow DataÎnc
- CodÎnc \longrightarrow CodDoc
- CodÎnc \longrightarrow NrDoc
- CodÎnc \longrightarrow DataDoc
- (CodÎnc, NrFact) \longrightarrow Tranșă

5.2. Aducerea relațiilor în a treia formă normală

Prima definiție a celei de-a treia forme normale a unei relații poate fi formulată relativ simplu și incremental, prin raportare la 2NF. O relație se află în 3NF dacă:

1. *Se găsește în 2NF.*
2. *Toate atributele care nu aparțin cheii primare nu depind funcțional de un alt atribut (ansamblu de atribute) care nu face parte din cheie.*

A doua condiție poate fi exprimată și în maniera: *toate dependențele funcționale care leagă cheia primară de celelalte atribute sunt directe (netranzitive).*

O altă formulare a 3NF utilizează tot două condiții:

- toate atributele ne-cheie din R sunt independente unele de altele, în sensul că nici unul dintre atributele neparticipante în cheia primară nu apare în vreo DF în care sursa este constituită dintr-o combinație de celelalte atribute ne-cheie;
- toate atributele ne-cheie din R sunt dependente ireductibil de cheia primară.

Important ! Chris Date este poate singurul care, precaut, la începutul discuțiilor legate de primele trei formele normale, afirmă că situațiile luate în considerare pornesc de la premisa că nu există chei candidat. Premisa este foarte importantă, întrucât atunci când nu se elimină dependențele simetrice și redundante pomenite în capitolul anterior, iar o relație are două mai multe chei candidat, sunt șanse mare să apară complicații. Vom reveni la sfârșitul acestui paragraf cu analiza bazei de date VÎNZĂRI și vom vedea că și precauția lui Date este insuficientă, deoarece, în fapt, problema nu este numai a cheilor candidat ale unei relații, ci a dependențelor simetrice și redundante.

Trecerea din 2NF în 3NF presupune eliminarea DF tranzitive și se poate face, pentru o relație, în următoarea manieră:

a) Se identifică toate atributele ce nu fac parte din cheie și sunt surse ale unor dependențe funcționale.

b) Pentru toate atributele identificate la punctul a), se constituie câte o relație în care cheia primară va fi atributul respectiv, iar celelalte atribute destinațiile din dependențele considerate.

Operațiile a) și b) se repetă și pentru relațiile "proaspăt" obținute prin decompoziție. De fiecare dată, din relațiile supuse descompunerii se elimină atributele

destinație ale surselor non-cheie (sună bine, nu-i așa ?), păstrându-se atributele sursă, în vederea stabilirii legăturilor între tabele (cheile străine) și, implicit, declarării restricțiilor referențiale.

Relația LINII_ARTICOLE_CONTABILE

Relația LINII_ARTICOLE_CONTABILE din figura 4.1 a fost adusă în capitolul anterior în 2NF prin descompunerea în două relații, NOTE_CTB și ARTICOLE_CTB, prezentate în figura 4.14.

NOTE_CTB {NotaContabilă, Data}

ARTICOLE_CTB {NotaContabilă, Operațiune, ContDebitor, ContCreditor, Suma}

Întrucât nici una din cele două relații nu conține dependențe funcționale tranzitive, în 3NF baza de date are aceeași structură ca și în 2NF.

Relația BIBLIOTECĂ_TUPLURI_NOI SOLUȚIA 3

Relația prezentată în figura 3.7 (paragraful 3.3.2) a fost adusă în 2NF în paragraful 4.4, prin ruperea sa în CĂRȚI {Cota, ISBN, Titlu, Editura, LocSediuEd, AnApariție} și TITLURI_AUTORI_CUVINTECHEIE {Cota, Autor, CuvântCheie} (vezi figura 4.15).

Cea de-a doua este alcătuită exclusiv din attribute-cheie, deci nu se pune în discuție 3NF (adică relația este deja în 3NF). Altfel stau lucrurile în CĂRȚI. Relația este în 2NF și conține dependențe tranzitive. Există două attribute din afara cheii primare, ISBN și Editura, care sunt surse de DF, astfel încât dependențele (7), (8), (9), (10) și (13) sunt tranzitive. Trecerea în 3NF se realizează prin constituirea a două relații pe care o să le numim EDITURI și TITLURI:

EDITURI {Editura, LocSediuEd}

și

TITLURI {ISBN, Titlu, Editura, AnApariție}

iar din CĂRȚI rămâne:

EXEMPLARE {Cota, ISBN}

În concluzie, în 3NF baza de date BIBLIOTECĂ este alcătuită din patru tabele:

EDITURI {Editura, LocSediuEd}

TITLURI {ISBN, Titlu, Editura, AnApariție }

EXEMPLARE { Cota, ISBN}

TITLURI_AUTORI_CUVINTECHEIE {Cota, Autor, CuvântCheie}

Conținutul acestora este cel din figura 5.2.

EDITURI

Editura	LocSediuEd
Polirom	Iași

TITLURI

ISBN	Titlu	Editura	AnApariție
973-683-889-7	Visual FoxPro. Ghidul dezvoltării aplicațiilor profesionale	Polirom	2002
973-683-709-2	SQL. Dialecte DB2, Oracle și Visual FoxPro	Polirom	2001

EXEMPLARE

Cotă	ISBN
III-13421	973-683-889-7
III-13422	973-683-889-7
III-13423	973-683-889-7
III-10678	973-683-709-2
III-10679	973-683-709-2

TITLURI_AUTORI_CUVINTECHEIE (fragment)

Cota	Autor	CuvântCheie
III-13421	Marin Fotache	baze de date
III-13421	Marin Fotache	SQL
III-13421	Marin Fotache	proceduri stocate
III-13421	Marin Fotache	FoxPro
III-13421	Marin Fotache	formulare
III-13421	Marin Fotache	orientare pe obiecte
III-13421	Marin Fotache	client-server
III-13421	Marin Fotache	web
	...	

Figura 5.2. Relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 în 3NF

Din păcate, ne rămâne pe conștiință ultima tabelă, TITLURI_AUTORI_CUVINTECHEIE care conține un grad de ridicat de redundanță a datelor, problemă rezolvată pe baza unui alt tip de dependențe - multivaloare (vezi capitolul viitor).

Relația DOTARE_JUCĂRII_1

Relația DOTARE_JUCĂRII_1 (figura 3.14) a fost adusă în 2NF (paragraful 4.4) sub forma celor două relații, FAMILII și JUCĂRII, din figura 4.16. Ambele relații sunt și în 3NF, deoarece nici una nu prezintă vreo dependență tranzitivă.

Relațiile universale LOCALITĂȚI_CODURI_VECI, CODURI_ORAȘE și STUDENȚI_EXAMENE

Nici una dintre relațiile ORAȘE_COMUNE / SATE / CODURI_OR, CODURI_ADRESE / CODURI_LOCALITĂȚI, și STUDENȚI / DISCIPLINE / EXAMENE nu conțin dependențe tranzitive, deci sunt și în 3NF.

Relația CODURI_NOI_V3

Această relație obținută în paragraful 3.4 (vezi figura 3.20), care are drept cheie primară atributul Id și care nu a intrat în discuție la 2NF, prezintă următoarele dependențe:

- (1) Id \longrightarrow CodPoștal
- (2) Id \longrightarrow Localitate
- (3) Id \longrightarrow Strada
- (4) Id \longrightarrow TipNr
- (5) Id \longrightarrow NrInițial
- (6) Id \longrightarrow LitInițială
- (7) Id \longrightarrow NrFinal
- (8) Id \longrightarrow LitFinală
- (9) Id \longrightarrow Comuna
- (10) Id \longrightarrow Județ

dar și

- (11) CodPoștal \longrightarrow Localitate
- (12) CodPoștal \longrightarrow Comuna
- (13) CodPoștal \longrightarrow Județ

Pe baza DF (11), (12) și (13), dependențele (2), (9) și (10) sunt tranzitive, iar relația se rupe în:

CODURI_LOC {CodPoștal, Localitate, Comuna, Județ}

și

CODURI_ADRESE2 {Id, CodPoștal, Strada, TipNr, NrInițial, LitInițială, NrFinal, LitFinală }

Relația VÎNZĂRI

Ultimul exemplu din pagraful 4.4 a adus relația universală VÎNZĂRI în a doua formă normalizată, prin ruperea acesteia în patru tabele: ÎNCAS_FACT, ÎNCASĂRI, LINII_FACTURI și FACTURI. Dar, după cum am văzut în paragraful precedent, există o serie de dependențe tranzitive. Să examinăm pe rând cele cinci relații.

ÎNCAS_FACT {CodÎnc, NrFact, Tranșa} are un singur atribut ne-cheie, deci nici vorbă de DF tranzitive.

ÎNCASĂRI {CodÎnc, DataÎnc, CodDoc, NrDoc, DataDoc} nu prezintă nici un atribut în afara CodÎnc care să fie sursă de DF.

Cu LINII_FACTURI {NrFact, Linie, Cantitate, PrețUnit, CodPr, DenPr, UM, ProctVA, Grupa} situația este diferită. Atributul ne-cheie CodPr este determinant în o serie de dependențe, (33)-(36), pe baza acestora construindu-se relația PRODUSE {CodPr, DenPr, UM, ProctVA, Grupa}; din LINII_FACTURI se elimină toate destinațiile celor patru DF, obținându-se LINII_FACT {NrFact, Linie, CodPr, Cantitate, PrețUnit}.

În tabela FACTURI {NrFact, CodCl, DataFact, Obs, DenCl, CodFiscal, StradaCl, NrStradaCl, BlocScApCl, TelefonCl, CodPost, Localitate, Comuna, Jud, Județ, Regiune} lucrurile sunt și mai interesante:

- mai întâi, se cuvine de observat că Jud este sursă a DF (1) și (3), așa încât FACTURI se descompune în:
 - JUDEȚE{ Jud, Județ, Regiune}
 - FACTURI_1 {NrFact, CodCl, DataFact, Obs, DenCl, CodFiscal, StradaCl, NrStradaCl, BlocScApCl, TelefonCl, CodPost, Localitate, Comună, Jud}
- dar și FACTURI_1 prezintă DF; în dependențele (5), (6) și (7) CodPost este sursă, așa încât FACTURI_1 se descompune în:
 - CODURI_POȘTALE {CodPost, Localitate, Comună, Jud}
 - FACTURI_2 {NrFact, CodCl, DataFact, Obs, DenCl, CodFiscal, StradaCl, NrStradaCl, BlocScApCl, TelefonCl, CodPost}
- în FACTURI_2 a mai rămas un singur atribut ne-cheie ce este sursă în DF (8)-(16), CodCl, astfel încât și aceasta se descompune în:
 - CLIENȚI {CodCl, DenCl, CodFiscal, StradaCl, NrStradaCl, BlocScApCl, TelefonCl, CodPost}
 - FACT {NrFact, CodCl, DataFact, Obs}

În concluzie, în 3NF relația inițială VÎNZĂRI este “spartă” în următoarele opt relații:

ÎNCAS_FACT {CodÎnc, NrFact, Tranșa}

ÎNCASĂRI {CodÎnc, DataÎnc, CodDoc, NrDoc, DataDoc}

PRODUSE {CodPr, DenPr, UM, ProctVA, Grupa}

LINII_FACT {NrFact, Linie, CodPr, Cantitate, PrețUnit}

JUDEȚE{ Jud, Județ, Regiune}

CODURI_POȘTALE {CodPost, Localitate, Comună, Jud}

CLIENTI {CodCl, DenCl, CodFiscal, StradaCl, NrStradaCl, BlocScApCl, TelefonCl, CodPost}

FACT {NrFact, CodCl, DataFact, Obs}

Cheile candidat și influența lor asupra 3NF

Promiteam în deschiderea paragrafului câteva detalii despre implicațiile cheilor alternative asupra aducerii unei relații în 3NF, implicații prea puțin evocate în literatura de profil. Astfel, revenim la relația în 2NF LINII_FACTURI {NrFact, Linie, Cantitate, PrețUnit, CodPr, DenPr, UM, ProcTVA, Grupa} din baza de date VÎNZĂRI. Dacă luăm în considerație că și denumirea produsului este unică, altfel spus, nu există două sortimente cu nume absolut identic, rezultă o serie de paralelă de dependențe (după cum am discutat în paragraful 4.1.4):

CodPr \longrightarrow DenPr	DenPr \longrightarrow CodPr
CodPr \longrightarrow UM	DenPr \longrightarrow UM
CodPr \longrightarrow Grupa	DenPr \longrightarrow Grupa
CodPr \longrightarrow ProcTVA	DenPr \longrightarrow ProcTVA

Dacă urmărim algoritmul prescris de trecere a relației LINII_FACTURI din 2NF în 3NF, ar trebui să consituim câte o relație pentru fiecare sursă de DF care nu face parte din cheia primară a relației, iar în relația ce rămâne din LINII_FACTURI (LINII_FACT_2) să păstrăm pe post de chei străine atât CodPr, cât și DenPr:

PRODUSE_1 {CodPr, DenPr, UM, ProcTVA, Grupa}

PRODUSE_2 {DenPr, CodPr, UM, ProcTVA, Grupa}

LINII_FACT_2 {NrFact, Linie, CodPr, DenPr, Cantitate, PrețUnit}

Gogomănia este evidentă, însă, ca în atâtea alte cazuri, fundamentată științific ! Iar dacă demonstrația nu a fost suficient de impresionantă, ce ziceți de relația FACTURI în 2FN, în care orice client poate fi identificat fără ambiguitate atât de CodCl, cât și de DenCl și de CodFiscal.

CodCl \longrightarrow DenCl	DenCl \longrightarrow CodCl	CodFiscal \longrightarrow CodCl
CodCl \longrightarrow CodFiscal	DenCl \longrightarrow CodFiscal	CodFiscal \longrightarrow DenCl
CodCl \longrightarrow StradaCl	DenCl \longrightarrow StradaCl	CodFiscal \longrightarrow StradaCl
CodCl \longrightarrow NrStradaCl	DenCl \longrightarrow NrStradaCl	CodFiscal \longrightarrow NrStradaCl
CodCl \longrightarrow BlocScApCl	DenCl \longrightarrow BlocScApCl	CodFiscal \longrightarrow BlocScApCl
CodCl \longrightarrow TelefonCl	DenCl \longrightarrow TelefonCl	CodFiscal \longrightarrow TelefonCl
CodCl \longrightarrow CodPost	DenCl \longrightarrow CodPost	CodFiscal \longrightarrow CodPost
CodCl \longrightarrow Localitate	DenCl \longrightarrow Localitate	CodFiscal \longrightarrow Localitate
CodCl \longrightarrow Comuna	DenCl \longrightarrow Comuna	CodFiscal \longrightarrow Comuna
CodCl \longrightarrow Jud	DenCl \longrightarrow Jud	CodFiscal \longrightarrow Jud
CodCl \longrightarrow Judet	DenCl \longrightarrow Judet	CodFiscal \longrightarrow Judet
CodCl \longrightarrow Regiune	DenCl \longrightarrow Regiune	CodFiscal \longrightarrow Regiune

Așa că din LINIIFACTURI obținem în prima fază a trecerii la 3NF nu mai puțin de patru relații, apoi să continuăm cu descompunem celor trei constituite pe baza seriilor de dependențe de mai sus s.a.m.d.

Soluția este cea pe care am sugerat-o în capitolul 4. Pentru identificarea oricărei entități (produs, client, factură) păstrăm doar cheia primară (practic, renunțăm la celelate chei candidată), iar toate dependențele ce decurg din calitatea de cheie alternativă nu sunt luate în considerare.

5.3. Aducerea relațiilor în 3NF prin grafuri și matrice ale DF

Modul în care a fost prezentată transformarea succesivă a relației universale inițiale în 1NF, 2NF și 3NF prin *descompuneri succesive* are certe valențe pedagogice și aplicabilitate practică. Descompunerea nu este însă singura manieră de normalizare a unei baze de date relaționale.

Philip A. Bernstein este autorul unui model de normalizare prin *sinteza* unor relații binare (construite pe baza DF) în relații mai mari și, astfel, aducerea bazei în 3NF¹. În plus, autorul a demonstrat că schema obținută conține un număr minim de relații. Algoritmul este menționat în majoritatea cărților dedicate bazelor de date, deși nu toți autorii îl descriu complet. Iată pașii:

1. Se elimină atributele exterioare din sursele tuturor dependențelor funcționale (setul F de DF). Un atribut este extern dacă eliminarea sa nu alterează închiderea lui F . Prin eliminarea din F se obține un alt set de DF notat G .
2. Se determină acoperirea neredundată H din setul de DF notat G .
3. Se partiționează H în grupuri, astfel încât toate DF dintr-un grup au surse identice.
4. Se fuzionează cheile echivalente. Fie $J = \emptyset$. Pentru toate perechile de grupuri H_i și H_j , cu sursele X , respectiv Y , se fuzionează H_1 și H_2 în cazurile în care se manifestă bijectivitatea $X \leftrightarrow Y$ în H^+ . Pentru fiecare asemenea bijectivitate, se adaugă în J dependențele $X \rightarrow Y$ și $Y \rightarrow X$. Pentru orice $A \in Y$, dacă există dependența $X \rightarrow A$ în H , aceasta se șterge din H . Analog se procedează pentru fiecare dependență $Y \rightarrow B$ din H , unde $B \in X$.
5. Se elimină DF tranzitive. Se identifică seturi $H' \subseteq H$, care îndeplinesc condiția $(H' + J)^+ = (H + J)^+$, și nici un subset din H' nu îndeplinește această condiție. Se adaugă fiecare DF din J în grupul său corespondent din H' .

¹ [Bernstein76]

6. Pentru fiecare group, se construiește o relație ce conține toate atributele grupului respectiv. Atributele din sursa DF vor alcătui cheia primară a relației respective.

Deși s-a demonstrat că algoritmul este corect, aplicarea sa efectivă ridică destule dificultăți, deoarece, după cum am mai discutat, închiderea și acoperirea minimală ale seturilor de DF chiar și pentru relații nu foarte voluminoase sunt greu de manevrat. Chris Date reproșează modelului (și Bernstein acceptă observația) că operațiunile executate în cadrul algoritmului de sinteză sunt exclusiv de natură sintactică și nu iau în considerare latura semantică². Însă exemplul luat de Date pentru a pune în dificultate algoritmul de sinteză constituie o capcană și pentru varianta descompunerii.

O variantă mult mai directă și relevantă pentru practicieni aparține lui Henry Smith care a publicat în *Communications of the ACM* un foarte interesant articol în care arată că o structură deplin normalizată poate fi obținută pe baza unei riguroase diagrame de DF³. De fapt, diagramele sale seamănă mai degrabă cu grafuri, decât forma lor consacrată⁴. Smith afirmă că descompunerea clasică, pornind de la relația universală este greoaie, iar construirea relației universale reclamă, de multe ori, un efort important. Lucrarea lui Smith este cu atât mai tentantă cu cât nu operează cu algoritmi pentru determinarea închiderilor și acoperirilor de seturi de DF. Ținând seama că, în 3NF, relațiile nu trebuie să conțină dependențe parțiale și tranzitive, se va reprezenta grafic acoperirea minimală.

De fapt, un avantaj decisiv al grafurilor DF este posibilitatea identificării imediate și eliminării atât a dependențelor parțiale, cât și a celor tranzitive. În plus, după cum vom vedea în paragraful 5.5.2, există situații în care normalizarea „clasică” generează probleme, în timp ce, folosind graful, schema obținută este net superioară.

Baza de date BIBLIOTECĂ

În capitolul 3 am demarat discuțiile legate de normalizarea bazei de date BIBLIOTECĂ pentru care am convenit, nu fără strângere de inimă, că cea mai acceptabilă prima formă normală ar fi BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 (vezi figura 3.7). Pe baza dependențelor identificate în capitolul 4, graful acoperirii minimale pentru universală BIBLIOTECĂ1 este reprezentat în figura 5.1. Totuși, pentru ilustrarea modului de identificare și eliminare dependențelor parțiale și tranzitive pornim de la graful din figura 4.8 care va fi reprezentat în stângă figurii 5.3. Spre deosebire de figura „sursă”, săgețile ce reprezintă dependențe parțiale sau tranzitive sunt punctate. Eliminându-le, obținem graful din dreapta figurii pe care delimităm câte o relație pentru fiecare sursă, simplă sau compusă. Relația cores-

² [Date00], pp. 377-378

³ [Smith85]

⁴ vezi, spre exemplu, [Date86] sau [Pratt & Adamski91]

pondentă va conține sursa, drept cheie primară, plus toate destinațiile sale directe (și totale).

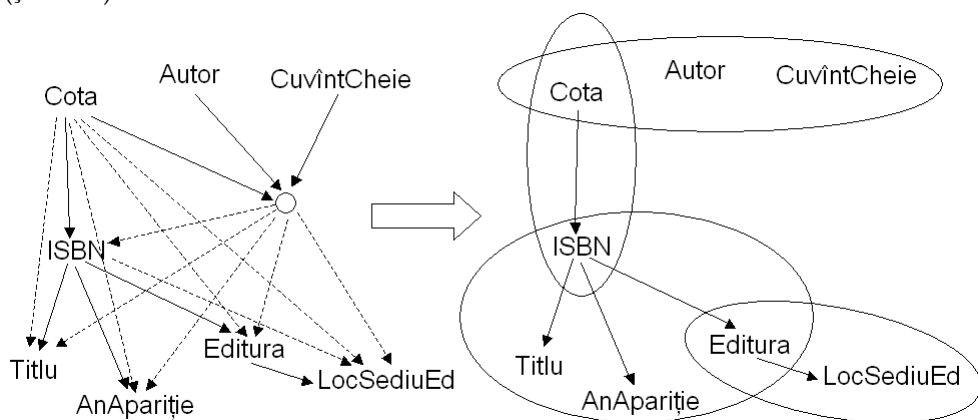


Figura 5.3. Cele patru tabele în 3NF pentru relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

În final, nu ne mai rămâne decât să dăm câte un nume fiecăreia dintre cele patru relații: EDITURI, TITLURI, EXEMPLARE, TITLURI_AUTORI_CUVINTECHEIE.

Relația STUDENȚI_EXAMENE

Cele 11 dependențe identificate în paragraful 4.4 pot fi lesne reprezentate ca un graf - vezi figura 5.4. Dependențele punctate sunt parțiale, deoarece destinațiile lor sunt legate de surse simple, așa că în final sunt eliminate din calcul.

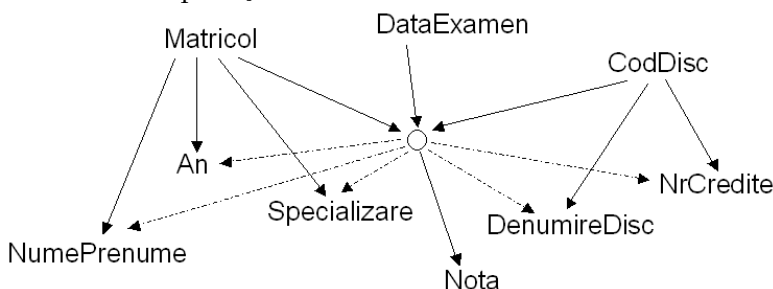


Figura 5.4. Aducerea relației STUDENȚI_EXAMENE în 3NF cu ajutorul grafului DF

Pe graf sunt trei surse, așa încât vom obține trei relații, cu următoarele atribute: {Matricol, NumePrenume, An, Specializare}, {CodDisc, DenumireDisc, NrCredite} și {Matricol, CodDisc, DataExamen, Nota}. Schema seamănă leit cu cea din paragraful 4.4.

Baza de date VÎNZĂRI

Prin reprezentarea sub formă de graf, economia de timp pentru aducerea în 3NF este și mai spectaculoasă, după cum reiese și din figura 5.5. Singura problemă ține de atenție în delimitarea tuturor surselor și dependențelor lor, ceea ce poate crea oarecare confuzie.

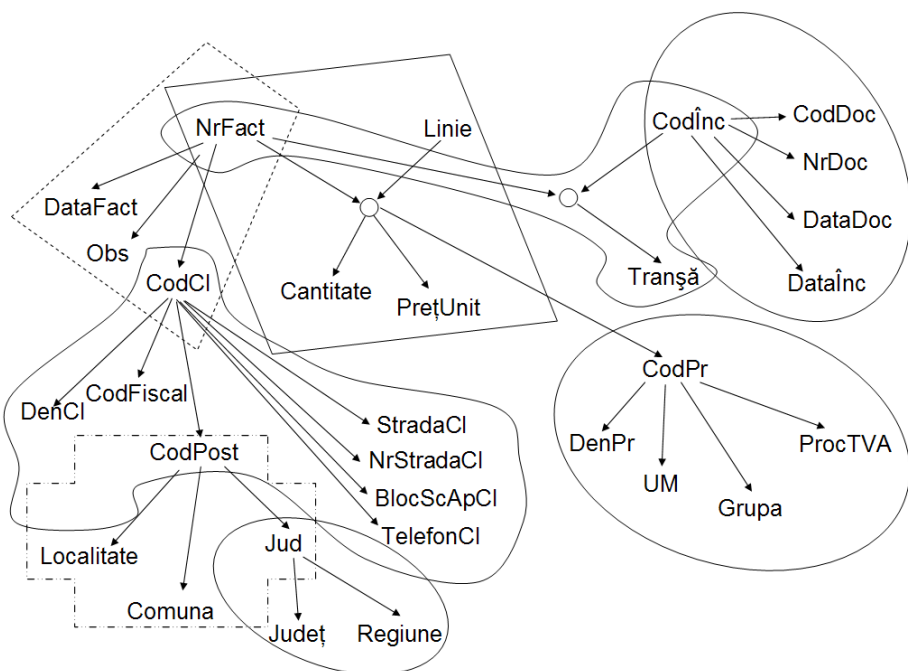


Figura 5.5. Decuparea relațiilor în 3NF pentru baza de date VÎNZĂRI

Pentru majoritatea cazurilor întâlnite în practică, construirea grafului dependențelor funcționale ce alcătuiesc acoperirea minimală, este una din cele mai simple și la îndemână modalități de aducere a bazei de date în 3NF. De o manieră asemănătoare se pot determina relațiile celei de-a treia forme normale utilizând matricea dependențelor incluse în acoperirea minimală.

5.4. Forma normală Boyce-Codd

Definirea celei de-a treia forme normale, așa cum a fost formulată inițial de Codd, ridică o serie de probleme în anumite circumstanțe, probleme pe care formularea din acest paragraf le rezolvă. Chris Date este de părere că situațiile în care definiția originală a 3NF este insuficientă se traduce prin trei condiții simultane⁵:

- relația are mai multe chei candidate;

⁵ [Date86], p.374

- o parte din cheile candidate sunt compuse;
- cheile compuse au atribute în comun.

Întrucât îndeplinirea simultană a celor trei condiții se întâlnește destul de rar în practică, a treia formă normală, așa cum a fost formulată în paragraful precedent, este suficientă. Se spune despre o relație că este în forma normală Boyce-Codd (BCNF) dacă:

1. se află deja în 3NF;
2. nu există nici o DF a cărei sursă să fie un atribut ne-component al cheii, iar destinația un atribut din cheie.

Într-o altă formulare, o relație R este în BCNF dacă și numai dacă orice determinant (sursă de dependență funcțională sau supercheie) este o cheie-candidat. Această exprimare are și avantajul de a nu lega BCNF de 3NF, unii autori preferând să discute problema normalizării direct în termenii BCNF, fără a trece prin formele intermediare⁶. Numele formei normalizate este discutabil, deoarece primul care elaborează o definiție echivalentă a ceea ce reprezintă astăzi BCNF este Ian Heath. În penultima ediție a cărții sale ce a împlinit recent 25 de ani, Chris Date afirmă că ar fi fost mai indicată titulatura *forma normală a lui Heath* (Heath Normal Form)⁷.

Cele trei condiții formulate de Chris Date au fost preluate și în alte lucrări, precum cele scrise de Rebecca M. Riordan⁸ sau, la noi, de Dumitru Oprea⁹ și de Robert Dollinger¹⁰. Este drept că Date, precaut, afirmă că suprapunerea (cel puțin un atribut în comun) cheilor candidate nu atrage întotdeauna necesitatea transformării relației în BCNF¹¹.

Primul contraexemplu

Curios lucru însă, din exemplele discutate în ultimele capitole putem servi un contraexemplu - tabela LINII_FACT {NrFact, Linie, CodPr, Cantitate, PrețUnit} pe care am obținut-o prin normalizare în paragraful anterior:

- tabela are două chei candidate, (NrFact, Linie) și (NrFact, CodPr);
- cele două surse de DF (superchei) sunt compuse;
- atributul NrFact este comun ambelor superchei.

Toate cele trei condiții sunt îndeplinite; cu toate acestea, relația LINIIFACT {NrFact, Linie, CodPr, Cantitate, PrețUnit} este și în 3NF și în BCNF, deoarece ambii determinanți sunt chei candidate ale relației.

⁶ Vezi, spre exemplu, [Garcia-Molina s.a.]

⁷ [Date00], p.366

⁸ [Riordan99], p.38

⁹ [Oprea99], p.351

¹⁰ [Dollinger98], p.160. Dollinger preia numai parțial ideea lui Date, dar greșește afirmând că definițiile 3NF (formularea "clasică") și BCNF sunt echivalente numai când atunci când relația R supusă normalizării are o singură cheie.

¹¹ [Date86], p.378

Exemplificarea relațiilor aflate în 3NF dar nu și în BCNF este destul de zgârcită și confuză în literatura de specialitate¹². Rebecca Riordan ia în discuție, după modelul lui Date¹³, următoarea relație¹⁴: R {CodFurnizor, NumeFurnizor, CodProdus, Cantitate, PrețUnitar}. Cele două chei candidat ale lui R sunt (CodFurnizor, CodProdus) și (NumeFurnizor, CodProdus). Fiind compuse și având un atribut comun, cazul pare ideal pentru aducerea în BCNF.

Fără a dramatiza prea mult lucrurile, mai întâi cred că exemplul nu prea are relevanță practică, deoarece datele despre produsele achiziționate sunt preluate din facturi, nefiind legate de furnizori decât la raportare (agregări, analize ale datelor). Mult mai coerentă este structura: FP {CodFurnizor, NumeFurnizor, NrFactura, DataFact, ValoareTotală}.

Trecând însă peste "legitimitatea" relației, problema are rezolvare foarte simplă. Codul identifică fără ambiguitate un furnizor, deci CodFurnizor \rightarrow NumeFurnizor. Este drept că și reciproca este valabilă, dar dacă eliminăm dependențele simetrice, avem de-a face în R cu o dependență funcțională (CodFurnizor, CodProdus \rightarrow NumeFurnizor) parțială. Prin urmare, relația R poate fi descompusă încă din 2NF în R1 {CodFurnizor, NumeFurnizor} și R2 {CodFurnizor, CodProdus, Cantitate, PrețUnitar}.

Relația PROIECTE

Candace Fleming și Barbara von Halle exemplifică utilitatea BCNF prin relația PROIECTE din figura 5.6¹⁵.

PROIECTE		
ProiectNr	Membriu	Supervizor
1	John	Oprea
1	Cătălin	Marin
1	Gabi	Oprea
2	John	Airinei
2	Cătălin	Airinei
3	Mircea	Luminița
3	Doina	Fătu
3	John	Fătu

Figura 5.6. Relație aflată în 3NF, nu însă și în BCNF

Restricții:

- la fiecare proiect există mai mulți supervizori și mai mulți membri;

¹² În [Lungu s.a.95], p.170 și 184-185, precum și [Băscă97], pp.110-111 exemplele sunt strict teoretice, iar în [Florescu s.a.99], p.85 exemplul relației în BCNF este neexplicat și inexplicabil.

¹³ [Date86], p.376

¹⁴ [Riordan99], pp.38-39

¹⁵ Exemplul este preluat și în alte lucrări, precum [Teorey99]

- un membru poate participa la oricâte proiecte;
- o persoană poate superviza un singur proiect;
- un supervisor poate coordona (în cadrul aceluiași proiect) mai mulți membri;
- într-un proiect, un membru este coordonat de un singur supervisor.

În aceste condiții, se verifică următoarele DF:

$$(1) (\text{ProiectNr}, \text{Membru}) \longrightarrow \text{Supervizor}$$

$$(2) (\text{Supervizor}, \text{Membru}) \longrightarrow \text{ProiectNr}$$

Prin urmare, relația are două chei candidat. Totodată, însă, există și DF $\text{Supervizor} \longrightarrow \text{ProiectNr}$, ceea ce face din dependența (2) una parțială. Atributul **Supervizor** este determinant fără a fi cheie candidat. În aceste condiții, relația **nu** este în BCNF. Pentru a o aduce în această formă normalizată este necesară transformarea grafului DF, așa cum arată figura 5.7 (pentru discuții privind grafurile DF în condițiile BCNF - vezi și paragraful 5.5.5).

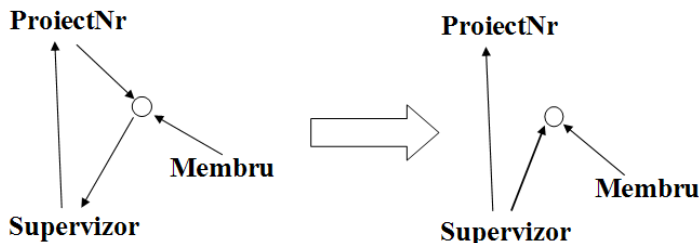


Figura 5.7. Transformarea grafului DF pentru relația PROIECTE

Cele două relații obținute sunt:

SUPERVIZORI {Supervizor, ProiectNr}
și
MEMBRI {Supervizor, Membru}

reprezentate în figura 5.8.

SUPERVIZORI	
Supervizor	ProiectNr
Oprea	1
Marin	1
Airinei	2
Luminița	3
Fătu	3

MEMBRI	
Supervizor	Membru
Oprea	John
Marin	Cătălin
Oprea	Gabi
Airinei	John
Airinei	Cătălin
Luminița	Mircea
Fătu	Doina
Fătu	John

Figura 5.8. Aducerea relației PROIECTE în BCNF

O prima problemă semnalată de cele două autoare se referă la pierderea unei restricții de integritate: *la un proiect, un membru are un singur supervisor*. Cu alte cuvinte, noua structură a BD permite ca unui membru să-i fie asociați doi supervizori în cadrul aceluiași proiect, ceea ce este incorect în condițiile date. În plus, exemplului prezentat i se poate reproșa și rigiditatea, deoarece în decursul anilor un supervisor poate răspunde de mai multe proiecte, dar acest din urmă neajuns este mai degrabă unul mărunț.

Niște profesori, cursuri și specializări

Cred că, de la C.J. Date încoace, cel mai frecvent întâlnit exemplu pentru ilustrarea formei normale Boyce-Codd este cel privitor la cursuri-profesori-studenți. Nu vom face rabat de la “regulă” și definim tabela SCP, reprezentată în figura 5.9, în condițiile în care *un profesor poate predă la oricâte specializări, dar numai un singur curs* (asta da profilare) !

SCP

Specializare	Curs	Profesor
Contabilitate	Bazele informaticii economice	Fătu
Finanțe	Bazele informaticii economice	Fătu
Contabilitate	Proiectarea sistemelor informatice	Oprea
Contabilitate	Baze de date	Airinei
Finanțe	Monedă	Turliuc
Statistică	Bazele informaticii economice	Fotache
Finanțe	Baze de date	Filip

Figura 5.9. Relația SCP

Să analizăm dependențele funcționale.

- un profesor predă un singur curs:
 $\text{Profesor} \longrightarrow \text{Curs}$
- un profesor predă la mai multe specializări:
 $\text{Profesor} \longrightarrow \text{Specializare}$
- un același curs poate fi ținut de mai mulți profesori:
 $\text{Curs} \longrightarrow \text{Profesor}$
- un același curs poate fi în planul de învățământ al mai multor specializări:
 $\text{Curs} \longrightarrow \text{Specializare}$
- la o specializare, în planul de învățământ sunt incluse mai multe cursuri (din păcate):
 $\text{Specializare} \longrightarrow \text{Curs}$
- la o specializare predau cursuri mai mulți profesori:
 $\text{Specializare} \longrightarrow \text{Profesor}$
- la o specializare, un curs este predat de un singur profesor:
 $(\text{Specializare}, \text{Curs}) \longrightarrow \text{Profesor}$
- un profesor poate ține cursul la mai multe specializări:
 $(\text{Profesor}, \text{Curs}) \longrightarrow \text{Specializare}$

Cuplul de atribute (*Specializare*, *Curs*) este cheie primară. Pe baza dependenței funcționale *Profesor* \longrightarrow *Curs* în BCNF tabela SCP va fi descompusă în două tabele, SP și PC, prezentate în figura 5.10.

SP		PC	
Specializare	Profesor	Profesor	Curs
Contabilitate	Fătu	Fătu	Bazele informaticii economice
Finanțe	Fătu	Oprea	Proiectarea sistemelor informatice
Contabilitate	Oprea	Airinei	Baze de date
Contabilitate	Airinei	Turliuc	Monedă
Finanțe	Turliuc	Fotache	Bazele informaticii economice
Statistică	Fotache	Filip	Baze de date
Finanțe	Filip		

Figura 5.10. Tabelele SP și PC aflate în BCNF

Relația TELEFOANE

Un exemplu sugestiv de relație aflată în 3NF, dar nu și în BCNF l-am întâlnit în lucrarea lui Robert Dollinger¹⁶ care ia în discuție relația TELEFOANE {*Oraș*, *Județ*, *Prefix*} ce conține prefixele telefonice ale tuturor orașelor (și, eventual, comunelor ce au centrală automată) din țară.

Tabela are două chei-candidat, (*Oraș*, *Prefix*) și (*Oraș*, *Județ*):

- (1) (*Oraș*, *Prefix*) \longrightarrow *Județ*
- (2) (*Oraș*, *Județ*) \longrightarrow *Prefix*

Iată care sunt premisele autorului:

- în țară există mai multe comune cu același nume, plasate în județe diferite:
Oraș \longrightarrow *Prefix*
Oraș \longrightarrow *Județ*
Prefix \longrightarrow *Oraș*
- un prefix nu poate fi alocat pentru localități din mai multe județe:
 (3) *Prefix* \longrightarrow *Județ*

În condițiile în care cheia primară este combinația (*Oraș*, *Județ*), în relație există un atribut din afara cheii (*Prefix*, cine altul ?) în postura de sursă a unei DF în care destinația este un atribut din cheie, lucru pus în evidență de figura 5.11.

¹⁶ [Dollinger98], pp.160-161

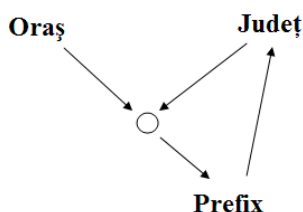


Figura 5.11. Graful DF pentru relația TELEFOANE

Graful din figura 5.11 nu este unul complet. Pe bună dreptate, întrucât forma din figura 5.12 nu este una dezirabilă, deoarece dependența $\text{Prefix} \longrightarrow \text{Județ}$ face din $(\text{Oraș}, \text{Prefix}) \longrightarrow \text{Județ}$ una parțială, care trebuie, deci, eliminată.

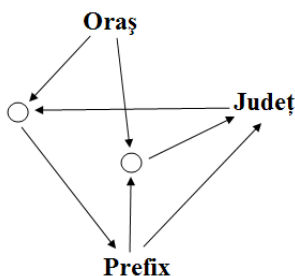


Figura 5.12. Graful complet al dependențelor relației TELEFOANE

Conformitatea cu regulile BCNF este realizată prin spargerea relației TELEFOANE în:

PREFIXE_JUD {Prefix, Județ}
 și
 ORCOM_PREFIX {Prefix, Oraș}

S-ar cuveni câteva discuții vis-a-vis de modul de tratare a acestui exemplu de către autor. Mai întâi, Robert trece sub tăcere faptul că există o dependență simetrică pentru (3), și anume:

(4) $\text{Județ} \longrightarrow \text{Prefix}$

deoarece pe tot cuprinsul unui județ operează un singur prefix. Pierderea, prin descompunere, dependenței (2) - $(\text{Oraș}, \text{Județ}) \longrightarrow \text{Prefix}$, este, totuși, nesemnificativă, atâta timp cât aceasta, datorită (4), este una parțială.

Consecvenți discuțiilor din capitolul precedent vis-a-vis de dependențele simetrice, eliminăm una dintre depdențele (3) și (4). Rezultatul nu suferă modificări, întrucât dependența (3) o face pe (1) parțială, în timp ce (4) o face pe (2) parțială.

Un exemplu sugestiv de relație aflată în 3NF ce întrunește condițiile de a fi convertită în BCNF este STUD_DOMENIU_CONDICĂTOR propusă de Dumitru Oprea¹⁷. Relația gestionează informațiile legate de lucrările de diplomă ale studenților care și-au propus să absolve o facultate: {Matricola_Student, Domeniu, Conducător}. Deși nu chiar reală, situația aleasă este cea în care un student își poate alege la diplomă unul, două, sau chiar mai multe domenii, de fiecare domeniu fiind responsabili o serie de profesori¹⁸. Cheia primară este combinația (Matricola_Student, Domeniu).

În condițiile în care:

- un student își poate alege o temă din mai multe domenii;
 - la fiecare domeniu, un student „beneficiază” de un singur profesor conducător;
 - fiecare domeniu este acoperit de mai mulți conducători și
 - fiecare conducător coordonează un singur domeniu,
- relația are două chei candidat, (Matricola_Student, Domeniu) și (Matricola_Student, Conducător), iar Conducător \rightarrow Domeniu. Aducerea relației în BCNF presupune descompunerea sa în

CONDUCĂTORI_DOMENII {Conducător, Domeniu} și
STUDENȚI_CONDUCĂTORI {Matricola_Student, Conducător}

Al doilea contraexemplu BCNF

La începutul acestui paragraf, prezentăm cele trei condiții care, îndeplinite, atrag necesitatea aducerii relației în BCNF. Reluăm tabela SCP, dar schimbăm ipoteza de lucru: *un profesor poate ține mai multe cursuri, dar fiecare curs îl predă la o singură specializare, iar la o specializare orice curs este ținut de un singur profesor*, noua relație fiind denumită SCP2 - vezi figura 5.13.

SCP2

Specializare	Curs	Profesor
Contabilitate	Bazele informaticii economice	Fătu
Finanțe	Bazele informaticii economice	Filip
Contabilitate	Proiectarea sistemelor informatice	Oprea
Contabilitate	Baze de date	Airinei
Finanțe	Monedă	Turliuc
Statistică	Bazele informaticii economice	Fotache
Statistică	Bazele informaticii economice	Airinei
Finanțe	Baze de date	Filip

Figura 5.13. Tabela SPC2

În aceste noi condiții, singurele DF sunt:

¹⁷ [Oprea99], pp.351-352

¹⁸ Situația nu este chiar reală, deoarece ar însemna ca o lucrare de diplomă să fie coordonată de doi, sau chiar mai mulți profesori din domenii diferite. Or, după cum știți, profesorii nu se prea pot înghiți unii pe alții.

$$(\text{Profesor}, \text{Curs}) \longrightarrow \text{Specializare}$$

$$(\text{Specializare}, \text{Curs}) \longrightarrow \text{Profesor}$$

Deoarece un profesor poate predă două cursuri la o aceeași specializare,

$$(\text{Profesor}, \text{Specializare}) \not\rightarrow \text{Curs}$$

Tabela SCP2 îndeplinește cele trei condiții:

- Relația are două chei candidate: (Profesor, Curs) și (Specializare, Curs)
- Cele două chei candidate sunt compuse.
- Cheile compuse au atributul Curs în comun.

Aceasta ar însemna că SCP este în 3NF, și trebuie adusă în BCNF. Totuși, relația SCP2 este deja în BCNF, deoarece, conform definiției, *oricare determinant* (în SCP2 sunt doi determinanți) *este cheie candidată a relației*.

5.5. Câteva probleme ale normalizării

Aproape toți autorii importanți atrag atenția asupra faptului că, deși riguros formalizată, normalizarea nu este un panaceu. Există numeroase reguli semantice pe care normalizarea nu le poate încorpora. În plus, o bază de date excesiv normalizată poate crea probleme la implementare, cum ar fi o viteză de acces inacceptabilă. Multe situații trebuie tratate cu precauție, iar uneori intuiția este preferabilă rigidității unui formalism.

Dacă în capitolul anterior discutam despre dependențe simetrice, prin eliminarea cărora scăpăm de multe complicații, în cele ce urmează ne vom ocupa de alte aspecte semantice ale normalizării care sunt "alunecoase", mai ușor de identificat sau chiar insolubile.

5.5.1. DF aparent tranzitive dar care conțin informații neredundante

Să luăm un exemplu foarte simplu: $R \{\text{Profesor}, \text{Birou}, \text{Catedră}, \text{Telefon}\}$ care conține date despre "domiciliul" în facultate al fiecărui profesor.

- Un profesor are un singur birou (Birou identifică fără ambiguitate sala respectivă):

$$(1) \text{ Profesor} \longrightarrow \text{Birou}$$

- Un profesor face parte dintr-o singură catedră:

$$(2) \text{ Profesor} \longrightarrow \text{Catedră}$$

- Într-un birou pot "locui" mai mulți profesori (deși, uneori, lucrul acesta îi nemulțumește profund):

$$\text{Birou} \not\rightarrow \text{Profesor}$$

- Fiecare birou este arondat unei singure catedre (*Economie politică, Contabilitate etc.*):

$$(3) \text{ Birou} \longrightarrow \text{Catedră}$$

- Fiecare birou are un număr de telefon unic:

$$(4) \text{ Birou} \longrightarrow \text{Telefon}$$

Din (3) ar reieși că (2) este DF tranzitivă, iar relația ar trebui descompusă numai în $R1 \{\underline{\text{Birou}}, \text{Catedră}, \text{Telefon}\}$ și $R2 \{\underline{\text{Profesor}}, \text{Birou}\}$.

Apare însă o situație ce aruncă în aer algoritmul nostru: *un profesor are un birou alocat altei catedre din care face parte !* Situația este reală. Catedrele pot "împrumuta", temporar sau definitiv, birouri sau câte un loc-două din anumite birouri. Pentru profesorii care ocupă un birou alocat altei catedre, schema bazei alcătuită din $R1$ și $R2$ furnizează eronat răspunsul la întrebarea: *Din ce catedră face parte fiecare profesor ?*

O soluție ar fi să se delimiteze, semantic, catedrele care dețin birouri și catedrele la care sunt afiliați profesorii:

$$R \{ \underline{\text{Profesor}}, \text{CatedrăProfesor}, \text{Birou}, \text{CatedrăBirou}, \text{Telefon} \};$$

Acum dependențele sunt:

$$\begin{aligned} (1) & \text{ Profesor} \longrightarrow \text{Birou} \\ (2) & \text{ Profesor} \longrightarrow \text{CatedrăProfesor} \\ (3) & \text{ Birou} \longrightarrow \text{CatedrăBirou} \\ (4) & \text{ Birou} \longrightarrow \text{Telefon} \end{aligned}$$

Noua schemă ar fi:

$$R1 \{ \underline{\text{Birou}}, \text{CatedrăBirou}, \text{Telefon} \}$$

și

$$R2 \{ \underline{\text{Profesor}}, \text{CatedrăProfesor}, \text{Birou} \}.$$

Se poate obiecta că ambele atribute, *CatedrăBirou* și *CatedrăProfesor*, se referă la aceeași entitate, deci seamănă a redundanță. Cu toate acestea, soluția elimină problema pierderii de informații, deci poate fi însoțită la elaborarea schemei bazei de date.

5.5.2. Dependențe tranzitive necanonice

În exemplele de până acum am avut de lucrat cu dependențe funcționale în formă canonică – cele în care destinația este alcătuită dintr-un singur atribut. Lucrurile nu sunt grozav de complicate, identificarea dependențelor fiind o operațiune lesnicioasă. Din păcate, în unele situații este posibil ca tranzitivitatea să se stabilească prin dependențe ale căror destinații sunt compuse. La normalizarea relațiilor este recomandabil ca, în situația unor dependențe funcționale de genul: A

$\longrightarrow B, A \longrightarrow C, A \longrightarrow D, A \longrightarrow E$, dependențe ce derivă din rolul de cheie primară pe care îl îndeplinește atributul A, să se verifice dacă nu cumva există:

- fie o dependență funcțională simplă, de genul $B \longrightarrow D$,
- fie o dependență funcțională cu sursa compusă, de genul $(B, C) \longrightarrow D$.

Dacă una din cele două afirmații de mai sus este adevărată, atunci dependența funcțională $A \longrightarrow E$ nu este directă.

Caz nr. 1 - baza de date COMENZI

Luăm în discuție tabela COMENZI, prezentată în figura 5.14, tabelă care gestionează comenzile pe care firma noastră le-a trimis furnizorilor (în care solicităm materialele/produsele/serviciile de care avem nevoie). Pe baza unei comenzi, un furnizor trimite produsele solicitate, întocmind o factură din care ne va fi remis un exemplar.

COMENZI						
NrComanda	NrFactura	CodFurnizor	DataFactura	Valoare	TVADeduct	
1501	12344	12	20.06.2004	119000000	19000000	
1502	11987	5	15.05.2004	67830000	10830000	
1503	45963	2	14.05.2004	71400000	11400000	
1504	12456	12	02.07.2004	67592000	10792000	
1505	12344	5	27.06.2004	106691830	17034830	
1505	12344	12	20.06.2004	101692640	16236640	
1507	47890	2	29.05.2004	93886835	14990335	

Figura 5.14. Relația COMENZI

Stabilim următoarele restricții:

- o comandă are un număr unic, stabilit de întreprinderea noastră, fiind întocmită pentru un singur furnizor;
- pentru o comandă, furnizorul va întocmi o singură factură;
- în schimb, o factură primită de la furnizor poate onora una, două sau mai multe comenzi.

Astfel, atributul NrComanda este cheia primară. Ca urmare, se poate scrie:

- (1) $NrComanda \longrightarrow NrFactura$
- (2) $NrComanda \longrightarrow CodFurnizor$
- (3) $NrComanda \longrightarrow DataFactura$
- (4) $NrComanda \longrightarrow Valoare$
- (5) $NrComanda \longrightarrow TVADeduct$

Facturile sunt întocmite de furnizori (care le numerotează independent între ei):

- (6) $(NrFactura, CodFurnizor) \longrightarrow DataFactura$
- (7) $(NrFactura, CodFurnizor) \longrightarrow Valoare$
- (8) $(NrFactura, CodFurnizor) \longrightarrow TVADeduct$

Întrucât o factură primită poate onora mai multe comenzi:

$(NrFactura, CodFurnizor) \longrightarrow NrComanda$

În virtutea ultimelor trei dependențe funcționale, rezultă că dependențele (3), (4) și (5) sunt tranzitive,

$$\begin{aligned} \text{NrComanda} &\longrightarrow (\text{NrFactură}, \text{CodFurnizor}) \longrightarrow \text{DataFactură} \\ \text{NrComanda} &\longrightarrow (\text{NrFactură}, \text{CodFurnizor}) \longrightarrow \text{Valoare} \\ \text{NrComanda} &\longrightarrow (\text{NrFactură}, \text{CodFurnizor}) \longrightarrow \text{TVADeduct} \end{aligned}$$

după cum se observă în graful DF din figura 5.15.

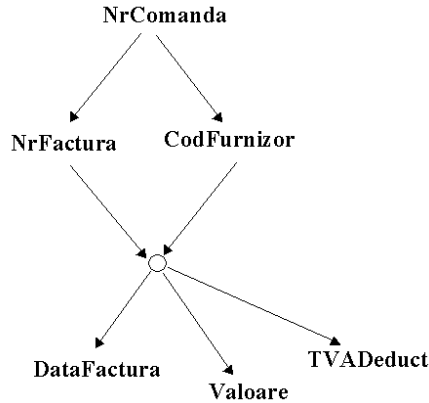


Figura 5.15. DF tranzitive necanonice

Pe baza tranzitivității, tabela COMENZI se poate sparge în două relații FACTURI_PRIMITE și COMENZI_FACTURI, ca în figura 5.16.

FACTURI PRIMITE					COMENZI FACTURI		
NrFactura	CodFurnizor	DataFactura	Valoare	TVADeduct	NrComanda	NrFactura	CodFurnizor
12344	12	20.06.2004	119000000	19000000	1501	12344	12
11987	5	15.05.2004	67830000	10830000	1502	11987	5
45963	2	14.05.2004	71400000	11400000	1503	45963	2
12456	12	02.07.2004	67592000	10792000	1504	12456	12
12344	5	27.06.2004	106691830	17034830	1505	12344	5
47890	2	29.05.2004	93886835	14990335	1505	12344	12
					1507	47890	2

Figura 5.16. Eliminarea DF tranzitive din tabela COMENZI

La drept vorbind, nu trebuie un talent deosebit de cârcotaș pentru a recunoaște că economia de spațiu și diminuarea redundanței sunt mai degrabă simbolice, ba chiar lucrurile se complică prin introducerea unei restricții referențiale și, normal, prin apariția de două ori a atributelor de legătură NrFactură și CodFurnizor.

Există, totuși, un merit indiscutabil al noilor relații. În relația COMENZI nu se poate prelua nici o factură fără a-i cunoaște comanda (eterna problemă a restricției de entitate – nici un atribut component al cheii nu poate avea nule). Practica economico-financiară de la noi, dar și din alte părți, cunoaște suficiente cazuri în care firmele primesc facturi direct, fără a înainta o comandă prealabilă. Utilizând proaspetele tabele, FACTURI_PRIMITE și COMENZI_FACTURI, o factură fără comandă va apărea numai în prima din cele două relații. Deși, la prima vedere, am

adăugat, și nu eliminat (așa cum, generos, sună unul dintre obiectivele normalizării) redundanță, noua structură este mai “sănătoasă” din punct de vedere relațional.

Practic, avantajul celor două relații din figura 5.15 este cu atât mai evident cu cât sunt numărul facturilor ce onorează două sau mai multe comenzi este mai mare.

Caz nr. 2 - baza de date EXAMENE

Pentru un plus de credibilitate, să reluăm cazul bazei de date destinată evidențierii rezultatelor obținute de studenți la examene programate în sesiunile de pe parcursul unui universitar (STUDENTI_EXAMENE) căreia îi extindem structura:

R {Matricol, NumePren, An, Modul, Spec, Grupa, CodDisc, DenDisc, NrCredDisc, CodProf, NumeProf, GradProf, DataEx, SalaEx, Nota}, al cărei dicționar simplicat este prezentat în tabelul 5.1.

Tabel 5.1. Dicționarul datelor pentru baza de date EXAMENE

Atribut	Descriere	Tip
Matricol	O combinație de litere și cifre ce identifică în mod unic un student	CHAR
NumePren	Numele și prenumele studentului	CHAR
An	Anul de studiu	NUMERIC
Modul	Modulul de studiu: este o literă ce desemnează un grup de specializări	CHAR
Spec	Specializarea ('C' de la contabilitate, 'I' de la Informatică economică etc.)	CHAR
Grupa	Grupa de studiu	NUMERIC
CodDisc	Codul disciplinei din planul de învățământ al specializării	CHAR
DenDisc	Denumirea disciplinei	CHAR
NrCredDisc	Numărul de credite alocate disciplinei	NUMERIC
CodProf	Codul profesorului	CHAR
NumeProf	Numele profesorului	CHAR
GradProf	P - profesor, C - conferențiar, L - lector, A - asistent, R - preparator, A - din afară (colaborator din afara universității)	CHAR
DataEx	Data examenului	DATE
SalaEx	Sala de desfășurare a examenului	CHAR
Nota	Nota obținută	NUMERIC

Pentru simplificare, am luat în calcul numai studenții de la profilul *Economic*, lungă durată (4-5 ani), cursuri de zi. O linie din R reprezintă rezultatul obținut de un student la o disciplină într-o sesiune de examinare. Spre exemplu, un student (matricol *EL001103*) poate obține la disciplina *Microeconomie* (cod *AE1001*) nota 4 (scuze) în prima sesiune de examinare – ianuarie 2005 (data examenului: *19 ianuarie 2005*) și nota 9 în a doua sesiune de examinare – februarie 2005 (data ex: *14/02/2005*). Cei mai puțin dispuși la efort se pot califica în faze superioare ale competiției (sesiunea mai 2005 s.a.m.d.).

Acestea fiind spuse, cheia primară a relației R este, ca și în cazul STUDENȚI_EXAMENE, (Matricol, CodDisc, DataEx), pe baza căreia pot fi scrise următoarele DF:

- (1) (Matricol, CodDisc, DataEx) \longrightarrow NumePren
- (2) (Matricol, CodDisc, DataEx) \longrightarrow An
- (3) (Matricol, CodDisc, DataEx) \longrightarrow Modul
- (4) (Matricol, CodDisc, DataEx) \longrightarrow Spec
- (5) (Matricol, CodDisc, DataEx) \longrightarrow Grupa
- (6) (Matricol, CodDisc, DataEx) \longrightarrow DenDisc
- (7) (Matricol, CodDisc, DataEx) \longrightarrow NrCredDisc
- (8) (Matricol, CodDisc, DataEx) \longrightarrow CodProf
- (9) (Matricol, CodDisc, DataEx) \longrightarrow NumeProf
- (10) (Matricol, CodDisc, DataEx) \longrightarrow GradProf
- (11) (Matricol, CodDisc, DataEx) \longrightarrow SalaEx
- (12) (Matricol, CodDisc, DataEx) \longrightarrow Nota

Relația R este în prima formă normalizată. După cum am văzut în capitolul precedent, pentru a demonstra că nu este în 2NF trebuie să găsim în ansamblul DF (1)-(12) măcar o DF parțială. Lucru destul de simplu, dacă ținem seamă de următoarele DF în care sursa o constituie unul dintre attributele componente ale cheii.

- Matricolul identifică în mod unic un student înscris la o specializare într-un an de studii:
 - (13) Matricol \longrightarrow NumePren
 - (14) Matricol \longrightarrow An
 - (15) Matricol \longrightarrow Modul
 - (16) Matricol \longrightarrow Spec
 - (17) Matricol \longrightarrow Grupa
- CodDisc identifică în mod unic fiecare disciplină din planul de învățământ al specializării:
 - (18) CodDisc \longrightarrow DenDisc
 - (19) CodDisc \longrightarrow NrCredDisc

Pe baza ultimelor șapte DF, relația universală R se poate descompune astfel:

R1 {Matricol, NumePren, An, Modul, Spec, Grupa}

R2 {CodDisc, DenDisc, NrCredDisc}

R3 {Matricol, CodDisc, CodProf, NumeProf, GradProf, DataEx, SalaEx, Nota}

Cele trei relații sunt în 2NF, iar până aici similitudinea cu discuția pricinuită de relația STUDENȚI_EXAMENE este tulburătoare. Aducerea BD în 3NF echivalează cu eliminarea dependențelor tranzitive din R1, R2 și R3. În R1 singurele DF existente sunt (13)-(17), deci această relație este în 3NF. Nici R2 nu conține dependențe tranzitive; în schimb R3 da:

- CodProf identifică în mod unic un profesor:

(20) CodProf \longrightarrow NumeProf

(21) CodProf \longrightarrow GradProf

Pe baza DF (20) și (21), putem spune că (9) și (10) sunt tranzitive, așa încât rupem R3 în R31 {CodProf, NumeProf, GradProf} și R32 {Matricol, CodDisc, CodProf, DataEx, SalaEx, Nota}. Astfel, în 3NF, relația universală R are o schemă alcătuită din patru relații, după cum urmează:

STUDENTI {Matricol, NumePren, An, Modul, Spec, Grupa}

DISCIPLINE { CodDisc, DenDisc, NrCredDisc }

PROFESORI {CodProf, NumeProf, GradProf }

NOTE1 {Matricol, CodDisc, CodProf, DataEx, SalaEx, Nota}

Nici una din cele patru relații nu conține DF tranzitive. Cu toate acestea, schema este departe de a fi optimă. Față de DF discutate, mai există două restricții neluate în calcul în normalizare:

- La o serie de curs (seriile de curs se constituie, pentru fiecare disciplină, la nivel de an, modul, specializare) titular este un singur profesor:

(22) (CodDisc, An, Modul, Spec) \longrightarrow CodProf

- Studenții unei specializări susțin examenul la o disciplină în aceeași sală (și aceeași zi):

(23) (An, Modul, Spec, DataEx) \longrightarrow SalaEx

Să recapitulăm: în relația NOTE există DF:

(8) (Matricol, CodDisc, DataEx) \longrightarrow CodProf

dar și

(22) (CodDisc, An, Modul, Spec) \longrightarrow CodProf

Deoarece există DF (24): Matricol \longrightarrow An, Modul, Spec

rezultă ca (8) este o DF necanonică, așa încât relația NOTE1 s-ar descompune în:

DISC_SPEC_PROFI { CodDisc, An, Modul, Spec, CodProf }

și

NOTE2 { Matricol, CodDisc, DataEx, SalaEx, Nota }

Câștigul este considerabil. Analog se poate proceda cu DF (23), discuția transferându-se în relația NOTE2, în care, din calitatea de cheie primară a celor trei atribute există dependența:

$$(11) (\text{Matricol}, \text{CodDisc}, \text{DataEx}) \longrightarrow \text{SalaEx}$$

dar și DF:

$$(23) (\text{An}, \text{Modul}, \text{Spec}, \text{CodDisc}, \text{DataEx}) \longrightarrow \text{SalaEx}.$$

Pe baza dependenței (24), relația NOTE2 se sparge în SALI_EX { An, Modul, Spec, CodDisc, DataEx, SalaEx } și NOTE { Matricol, CodDisc, DataEx, Nota }

Ajungem, astfel, la o cu totul altă structură a bazei de date, mult mai bună decât precedenta:

STUDENTI { Matricol, NumePren, An, Modul, Spec, Grupa }

DISCIPLINE { CodDisc, DenDisc, NrCredDisc }

PROFESORI { CodProf, NumeProf, GradProf }

DISC_SPEC_PROFI { CodDisc, An, Modul, Spec, CodProf }

SALI_EX { An, Modul, Spec, CodDisc, DataEx, SalaEx }

NOTE { Matricol, CodDisc, DataEx, Nota }

Ca de obicei, dependențele tranzitive, canonice sau necanonice se observă mult mai lejer pe graful DF - vezi figura 5.17. Este suficient să decupăm câte o relație prentu fiecare sursă simplă și compusă din graf, sursă ce devine automat cheie a relației respective și vom obține structura de mai sus.

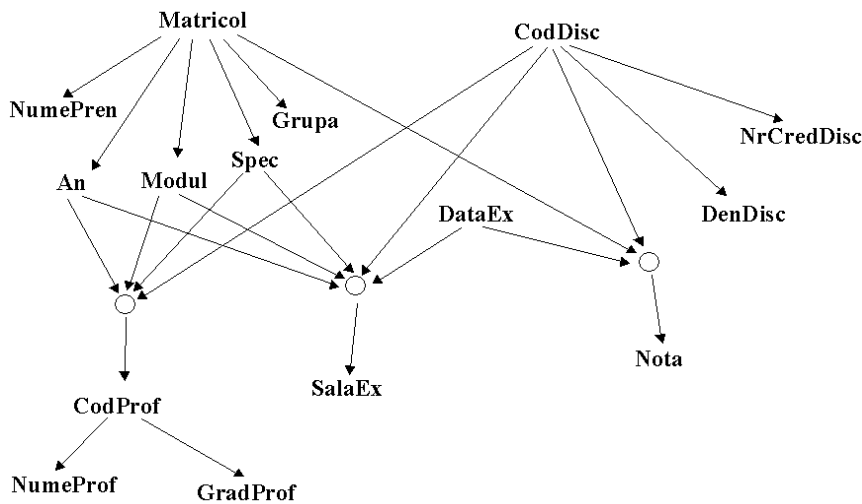


Figura 5.17. Graful DF pentru baza de date EXAMENE

Structura finală îndeplinește și cerințele formei normale Boyce-Codd ca fiecare sursă de DF să fie cheie candidată. Putem institui și o regulă: atunci când descompunerea se face urmând filiera clasică:

- stabilirea cheii primare a relației inițiale (universale), apoi
- eliminarea, dintre dependențele funcționale ce decurg din calitatea de cheie primară a relației inițiale, a celor parțiale, apoi
- eliminarea, din toate relațiile obținute în pasul anterior, a tuturor dependențelor tranzitive (în care surse sunt cheile primare ale respectivelor relații),

trebuie verificat dacă în relația inițială mai existau dependențe cu sursa compusă ce nu apar drept cheie primară sau candidat în nici una dintre relațiile finale obținute.

5.5.3. Relații semantice imposibil de preluat în dependențe funcționale

Un alt aspect delicat al normalizării ține de dificultatea preluării în schemă, sub formă de dependențe funcționale, a unor relații semantice. Pentru o primă ilustrare să luăm relația PROFI_DISCIPLINE din figura 5.18 ce conține date legate de profesori și cursurile predate într-o facultate.

PROFI_DISCIPLINE

Id Prof	Nume Prof	Catedră	Cod Disc	DenDisc	LbStr Prof	LbStr Disc
111	Airinei Dinu	Informatică Economică	AI1101	Introducere în informatica economică	FR	FR
112	Fotache Marin	Informatică Economică	AI1101	Introducere în informatica economică	EN	EN
112	Fotache	Informatică	AI1101	Introducere în informatica	FR	FR

	Marin	Economică		economică		
--	-------	-----------	--	-----------	--	--

Figura 5.18. Cursuri predate în limbi străine

Un profesor este identificat prin `IdProf` și face parte dintr-o singură catedră:

- (1) `IdProf` \longrightarrow `NumeProf`
- (2) `IdProf` \longrightarrow `Catedră`.

Un curs este identificat prin cod și arondat unei singure catedre:

- (3) `CodDisc` \longrightarrow `DenDisc`
- (4) `CodDisc` \longrightarrow `Catedră`.

În schimb:

- un profesor predă mai multe discipline:

`IdProf` $\text{---}/\rightarrow$ `CodDisc`

- o disciplină poate fi predată de mai mulți profesori:

`CodDisc` $\text{---}/\rightarrow$ `IdProf`

- un profesor poate predă în mai multe limbi străine:

`IdProf` $\text{---}/\rightarrow$ `LbStrProf`

- un curs poate fi predat în mai multe limbi străine:

`CodDisc` $\text{---}/\rightarrow$ `LbStrDisc`

- un profesor poate predă același curs în mai multe limbi străine:

$(\text{IdProf}, \text{CodDisc}) \text{---}/\rightarrow$ `LbStrDisc`

- un profesor poate predă în aceeași limbă străină două sau mai multe cursuri:

$(\text{IdProf}, \text{LbStrDisc}) \text{---}/\rightarrow$ `CodDisc` și

$(\text{IdProf}, \text{LbStrProf}) \text{---}/\rightarrow$ `CodDisc`

Practic, graful DF se prezintă ca în figura 5.19.

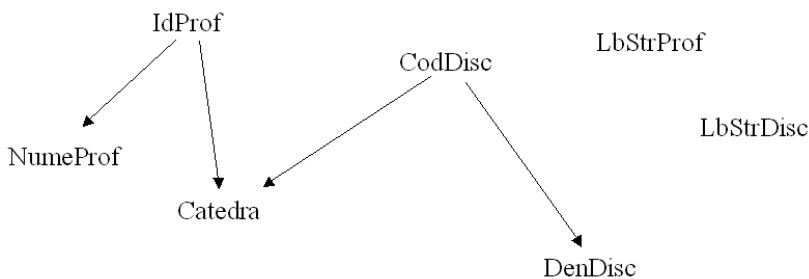


Figura 5.19. Graful DF pentru relația PROFIL_DISCIPLINE

Nemulțumirea legată de informația reprezentată prin graf ține de faptul că un profesor știe anumite limbi străine, iar o disciplină este predată în câteva limbi, nicidecum în toate. Cu alte cuvinte, schema de mai sus ridică probleme la stocarea informațiilor: *care sunt limbile străine cunoscute de fiecare profesor și care sunt limbile străine în care se predă fiecare disciplină.*

Atributele `LbStrProf` și `LbStrDisc` rămân suspendate în graf. Ce-i de făcut ? O idee, nu întotdeauna rezonabilă, este a le plasa într-o relație împreună cu celelate "rădăcini" ale grafului, obținându-se astfel schema:

`PROFI {IdProf, NumeProf, Catedra},`

`DISCIPLINE {CodDisc, DenDisc, Catedra} și`

`PROFI_DISC_LBSTR {IdProf, CodDisc, LbStrProf, LbStrDisc}`

Predicatul care ar exprima oricare tuplu al ultimei relații ar fi: un profesor (`IdProf`) care cunoaște o limbă străină (`LbStrProf`) predă o disciplină (`CodDisc`) într-una din limbile străine pe care le vorbește (`LbStrDisc`). Firește, apare întrebarea: *Ce relație trebuie să existe între valorile `LbStrProf` și `LbStrDisc` într-un tuplu, știind că disciplina este predată și în alte limbi decât `LbStrProf`, iar profesorul știe și alte limbi străine decât `LbStrDisc` ?*

O soluție acceptabilă în multe cazuri constă în introducerea unor atribute suplimentare, tocmai pentru a "fixa" sub formă de dependențe restricțiile respective. Spre exemplu, pentru a "formaliza" restricția *un profesor știe anumite anumite limbi străine*, se poate introduce un atribut, `Nivel`, pentru a ilustra gradul de comunicare al profesorului în limba străină respectivă:

$(IdProf, LbStrProf) \longrightarrow Nivel$.

De asemenea, pentru a indica în ce limbi străine este disponibil un curs, se introduce atributul `SerieCurs`. La un obiect, o serie de curs se constituie pentru o singură limbă străină și cade în responsabilitatea unui singur profesor:

$(CodDisc, SerieCurs) \longrightarrow LbStrDisc$

$(CodDisc, SerieCurs) \longrightarrow IdProf$.

Graful obținut nu mai conține atribute "suspendate" - vezi figura 5.20 - iar schema desprinsă din acesta este mult mai bună:

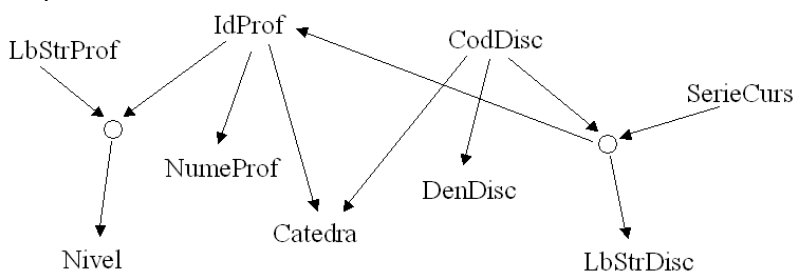


Figura 5.20. Noul graf DF pentru relația PROFIDISCIPLINE ameliorată

`PROFI {IdProf, NumeProf, Catedra},`

`DISCIPLINE {CodDisc, DenDisc, Catedra},`

`PROFI_LBSTR {IdProf, LbStrProf, Nivel} și`

```
SERII_CURS_LBSTR { CodDisc, SerieCurs, LbStrDisc, IdProf}.
```

Deși oarecum artificială, tehnica introducerii de atribute suplimentare, care nu sunt evidente în momentul construirii relației universale, este deseori salutară, fiind folosită pe scară largă de practicieni. În plus, putem vorbi de un argument suplimentar în favoarea ideii de ciclicitate a normalizării, de reluare a o serie de pași până se obține o schemă considerată mulțumitoare.

Relația BIBLIOTECĂ_TUPLURI_NOI SOLUȚIA_3 se întoarce !

Revenim la un caz “clasat” pentru 3NF, cel al relației BIBLIOTECĂ_TUPLURI_NOI SOLUȚIA_3. Combinația care diferențiază orice linie de toate celelalte este (Cota, Autor, CuvîntCheie). Intuim însă un aspect pe care dependențele și, implicit, normalizarea îl scapă, și anume că autorii și cuvintele cheie se referă la o carte (adică ISBN) și nu la un exemplar fizic din depozit (Cota). Deci graful DF nu ar trebui să arate ca în figura 5.3, ci precum în figura 5.21. Din păcate, acesta este un aspect pe care îl intuim, dar e mai greu de demonstrat “științific” prin normalizare. Un argument în plus pentru a nu considera normalizarea drept un proces infailibil, ci doar un ghid în construirea unei scheme a BDR cât mai bune.

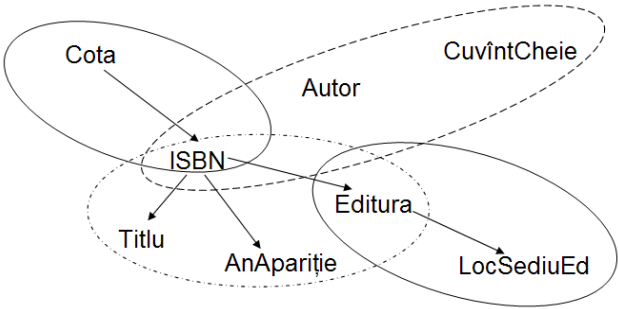


Figura 5.21. Autorii și cuvintele cheie corespund titlurilor, nu exemplarelor din cărți

Pe baza grafului de mai sus, din schema bazei de date în 3NF se schimbă ultima tabelă, TITLURI_AUTORI_CUVINTECHEIE2 {ISBN, Autor, CuvântCheie}.

Conținutul celor patru relații este acum cel din figura 5.22.

EDITURI

Editura	LocSediuEd
Polirom	Iași

TITLURI

<u>ISBN</u>	Titlu	Editura	AnApariție
973-683-889-7	Visual FoxPro. Ghidul dezvoltării aplicațiilor profesionale	Polirom	2002
973-683-709-2	SQL. Dialecte DB2, Oracle și Visual FoxPro	Polirom	2001

EXEMPLARE

Cotă	ISBN
III-13421	973-683-889-7
III-13422	973-683-889-7
III-13423	973-683-889-7
III-10678	973-683-709-2
III-10679	973-683-709-2

TITLURI_AUTORI_CUVINTECHEIE2

ISBN	Autor	CuvântCheie
973-683-889-7	Marin Fotache	baze de date
973-683-889-7	Marin Fotache	SQL
973-683-889-7	Marin Fotache	proceduri stocate
973-683-889-7	Marin Fotache	FoxPro
973-683-889-7	Marin Fotache	formulare
973-683-889-7	Marin Fotache	orientare pe obiecte
973-683-889-7	Marin Fotache	client-server
973-683-889-7	Marin Fotache	web
973-683-889-7	Ioan Brava	baze de date
973-683-889-7	Ioan Brava	SQL
973-683-889-7	Ioan Brava	proceduri stocate
973-683-889-7	Ioan Brava	FoxPro
973-683-889-7	Ioan Brava	formulare
973-683-889-7	Ioan Brava	orientare pe obiecte
973-683-889-7	Ioan Brava	client-server
973-683-889-7	Ioan Brava	web
973-683-889-7	Cătălin Strâmbei	baze de date
973-683-889-7	Cătălin Strâmbei	SQL
973-683-889-7	Cătălin Strâmbei	proceduri stocate
973-683-889-7	Cătălin Strâmbei	FoxPro
973-683-889-7	Cătălin Strâmbei	formulare
973-683-889-7	Cătălin Strâmbei	orientare pe obiecte
973-683-889-7	Cătălin Strâmbei	client-server
973-683-889-7	Cătălin Strâmbei	web
973-683-889-7	Liviu Crețu	baze de date
973-683-889-7	Liviu Crețu	SQL
973-683-889-7	Liviu Crețu	proceduri stocate
973-683-889-7	Liviu Crețu	FoxPro
973-683-889-7	Liviu Crețu	formulare
973-683-889-7	Liviu Crețu	orientare pe obiecte
973-683-889-7	Liviu Crețu	client-server
973-683-889-7	Liviu Crețu	web
973-683-709-2	Marin Fotache	baze de date
973-683-709-2	Marin Fotache	algebră relațională
973-683-709-2	Marin Fotache	SQL

Figura 5.22. O 3NF mai bună pentru relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

Se observă cu ochiul liber o diminuare a volumului ultimei tabele, astfel încât ne-am încumetat să reprezentăm toate liniile. Rămâne, în schimb, un grad de redundanță chiar și în forma "comprimată" a tabelei.

Similar relației PROFIL_DISCIPLINE din figura 5.18. putem încerca să adăugăm în BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 câteva atribute care să atragă în dependențe atributele Autor și CuvântCheie. Astfel, pornim de la faptul că, pe coperta unei cărți, autorii sunt trecuți într-o anumită ordine, deloc întâmplătoare,

iar la extragerea informațiilor despre cărți această ordine trebuie respectată. Prin urmare, folosim atributul *OrdineCopertă*, iar DF este:

$(ISBN, OrdineCopertă) \longrightarrow Autor.$

De asemenea, pentru a putea extrage cărțile legate de unul sau mai multe cuvinte cheie în ordinea relevanței, introducem atributul *Relevanță* ce indică nivelul la care este tratată într-o carte o anumită noțiune/sintagmă (cuvînt cheie):

$(ISBN, CuvîntCheie) \longrightarrow Relevanță.$

Pe baza acestor două noi dependențe, graful din figura 5.21 se modifică după cum este sugerat în figura 5.23.

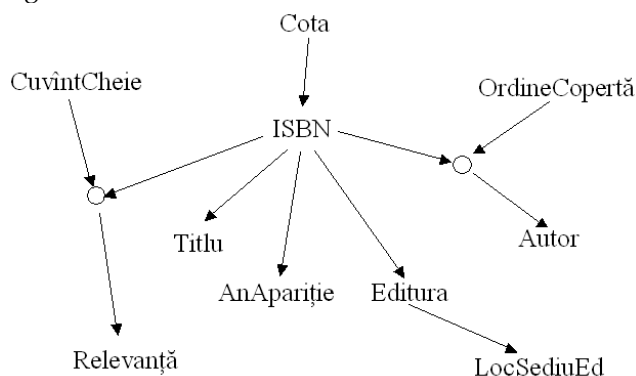


Figura 5.23. Noul graf pentru relația BIBLIOTECĂ (cu două atribute noi)

Din noul graf, relațiile construite sunt aproape impecabile:

EDITURI {Editura, LocSediuEd}

CĂRȚI {ISBN, Titlu, AnApariție, Editură}

AUTORI_CĂRȚI {ISBN, OrdineCopertă, Autor}

CĂRȚI_CUVINTE_CHEIE {ISBN, CuvîntCheie, Relevanță}

EXEMPLARE_CĂRȚI {Cota, ISBN}

5.5.4. Simetrie și temporalitate în dependențele funcționale

Revenim la relația COMENZI descrisă în paragraful 5.5.2 (figura 5.14), pentru care schimbăm condiția de derulare a operațiunilor. Astfel, dacă în paragraful 5.5.2 am inventariat dependențele în situația în care o comandă este onorată printr-o singură factură, dar o factură poate să corespundă mai multor comenzi, discutăm în cele ce urmează relația COMENZI2 pentru care, după recepționarea unei comenzi primite de la firma noastră, furnizorul o "rezolvă" expedind produsele comandate într-un singură livrare și, prin urmare, întocmește o singură factură. În

plus, o factură nu poate onora decât maxim o comandă. Graful DF ia forma din figura 5.24.

- COMENZI2**
- *O comandă este onorată de o singură factură*
 - *O factură onorează o singură comandă*

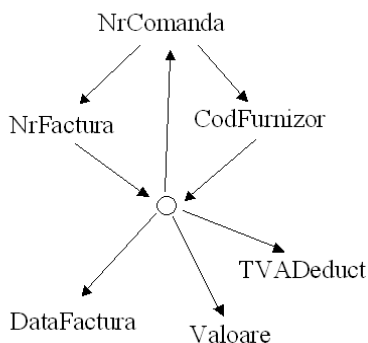


Figura 5.24. O (altă) dependență simetrică

De data aceasta, graful nu mai este așa de prompt în furnizarea schemei bazei de date în BCNF. Dacă am urma regula *o sursă = o relație*, atunci COMENZI2 s-ar descompune în:

COMENZI21 {NrComandă, NrFactură, CodFurnizor}

și

FACTURI21 {NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct, NrComandă}.

Redundanța este însă evidentă, deoarece graful conține două dependențe simetrice, $NrComandă \longrightarrow (NrFactură, CodFurnizor)$ și $(NrFactură, CodFurnizor) \longrightarrow NrComandă$. Pe care o eliminăm? Pare mai logic pe prima, deoarece sursa $(NrFactură, CodFurnizor)$ are și alte destinații. Practic, relația COMENZI2 ar păstra păstrează aceeași structură în 2NF, 3NF și BCNF:

COMENZI2{NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct, NrComandă}.

În paragraful 4.1.4 recomandam ca, atunci când pentru identificarea unui obiect/operațiune/tranzacție există mai multe atribute/combinatii, prin urmare, mai multe chei candidat, la reprezentarea dependențelor funcționale să se păstreze numai cele în care sursa este un identificator (cheie primară). Cele în care sursele sunt chei alternative, iar destinațiile identice cu ale identificatorului primar trebuie eliminate. Altfel spus, dacă:

$X \longrightarrow A, X \longrightarrow B, X \longrightarrow C, Y \longrightarrow A, Y \longrightarrow B, Y \longrightarrow C$

și $X \longrightarrow Y$, dar și $Y \longrightarrow X$,

atunci se alege X (sau Y) drept identificator primar și se păstrează numai dependențele:

$$X \longrightarrow A, X \longrightarrow B, X \longrightarrow C, X \longrightarrow Y.$$

Prelungim discuția de la relația COMENZI2 introducând cu bună știință atributul DataComandă pentru a ști în ce zi a fost întocmită fiecare comandă trimisă furnizorului. Altfel scris, relația conține attributele COMENZI3 {NrComanda, DataComanda, NrFactura, CodFurnizor, DataFactura, Valoare, TVADeduct}, iar raportul comenzi-facturi este de tip unu-la-unu. În postura "candidaților" X și Y avem atributul NrComandă, respectiv combinația (NrFactura, CodFurnizor). Seria de DF care se înregistrează în COMENZI3 este:

NrComanda \longrightarrow DataComanda	(NrFactura,CodFurnizor) \longrightarrow NrComanda
NrComanda \longrightarrow NrFactura	(NrFactura,CodFurnizor) \longrightarrow DataComanda
NrComanda \longrightarrow CodFurnizor	(NrFactura,CodFurnizor) \longrightarrow DataFactura
NrComanda \longrightarrow DataFactura	(NrFactura,CodFurnizor) \longrightarrow Valoare
NrComanda \longrightarrow Valoare	(NrFactura,CodFurnizor) \longrightarrow TVADeduct
NrComanda \longrightarrow TVADeduct	

Dacă, dintre cei doi candidați, s-ar alege NrComandă, atunci graful ar arăta precum varianta 1 din figura 5.25, în timp la alegerea combinației (NrFactura, CodFurnizor) forma finală a grafului ar fi cea din varianta 2.

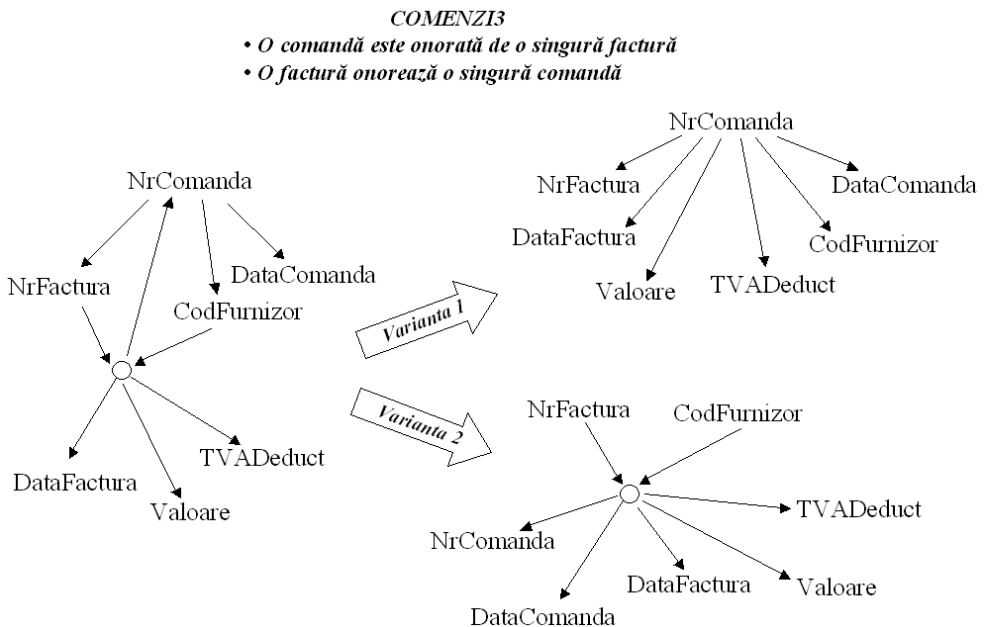


Figura 5.25. O (altă) dependență simetrică

În ambele variante relațiile obținute sunt identice, doar că alcătuirea cheii primare este diferită:

COMENZI3_VARIANTA1 {NrComandă, DataComanda, NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct}

COMENZI3_VARIANTA2 {NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct, NrComandă, DataComanda}.

Dacă, din punctul de vedere al dependențelor funcționale, oricare variantă este corectă, în practică relația COMENZI3_VARIANTA2 nu este dezirabilă, întrucât ordinea operațiunilor este rigidă: mai întâi se redactează comanda și se transmite furnizorului, iar apoi furnizorul o onorează. Comanda este, deci, anterioară facturii. Pentru relația incriminată, inserarea unei comenzi nu este posibilă decât în momentul onorării ei (altminteri atributele cheie ar avea valori NULLe), iar cunoașterea comenzilor (încă) neonorate este imposibilă.

Urmarea ar fi că, în situația dependențelor simetrice, eliminarea simetriei trebuie să țină cont și de eventuala succesiune a operațiunilor, privilegiindu-se sursele în ordinea precedenței lor. Din acest punct de vedere, COMENZI3_VARIANTA1 este relația ce pare a rezolva problema. Obiecția celor ce se opun folosirii valorilor NULL este una îndreptățită, deoarece din momentul întocmirii comenzi până la cel al livrării (facturii) corespunzătoare, atributele NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct sunt NULL.

Toate aceste discuții apar datorită faptului că dependențele funcționale și, implicit, graful acestora nu oferă întotdeauna informații legate de succesiunea operațiunilor, așa încât, cu riscul de a complica graful, putem apela la un artificiu care să sugereze ordinea momentelor la care este valabilă fiecare DF - vezi figura 5.26.

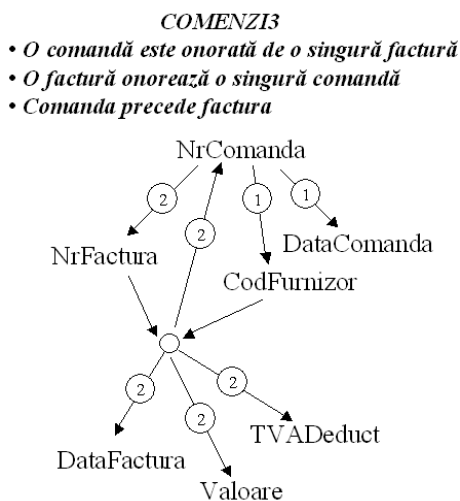


Figura 5.26. Graful din figura 5.24 completat cu elemente care să indice ordinea "producerii" dependențelor

Astfel, s-ar putea construi o relație pentru fiecare sursă de DF și "moment" consemnat în graf. Prima este asociată momentului întocmirii comenzii:

NUMAI_COMENZI3 {NrComandă, DataComanda, CodFurnizor}

Dintre DF simetrice (NrFactura, CodFurnizor) \longrightarrow NrComanda și NrComanda \longrightarrow NrFactura, respectiv NrComanda \longrightarrow CodFurnizor, se elimină prima, așa încât rezultă alte două relații:

COMENZI3_FACTURI {NrComandă, NrFactură}

și

FACTURI3 {NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct}.

Deși pare ciudat, iar fragmentarea bazei greu suportabilă, această structură este extrem de flexibilă, iar valorile NULL sunt eliminate în totalitate. Un alt efect colateral benefic este că, acum, se rezolvă și problema sosirii facturilor înaintea comenzii, sau chiar a facturilor fără comenzi. Practicienii știu că aceste situații sunt greu de evitat, mai ales în situația țării noastre în care rigoarea nu reprezintă o vocație națională.

5.5.5. Grafuri și BCNF

Revenim la relația COMENZI descrisă în paragraful 5.5.2 (figura 5.14), pentru care schimbăm, încă o dată, condițiile de derulare a operațiunilor (COMENZI4). Astfel, chiar dacă o factură nu poate corespunde decât unei singure comenzi, o comandă poate fi onorată în tranșe, prin mai multe livrări/transporturi, și, în consecință, mai multe facturi - vezi figura 5.27.

COMENZI4

- O comandă este onorată de mai multe facturi (de la același furnizor)
- O factură onorează o singură comandă

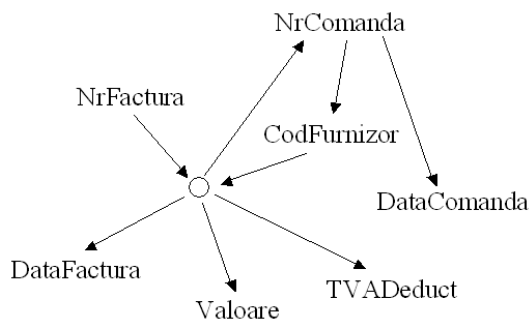


Figura 5.27. Situație ce necesită aducerea relației în BCNF

Urmând "preceptele" construirii relațiilor pe baza grafului, ar trebui să obținem schema:

COMENZI41 {NrComandă, CodFurnizor, DataComanda}

și

FACTURI41 {NrFactură, CodFurnizor, DataFactură, Valoare, TVADeduct, NrComandă}.

Ceea ce ne interesează în cea mai mare măsură în graf este, ză-i zicem, semi-circularitatea dependențelor, specifică discuțiilor legate de BCNF. Practic, descompunerea în COMENZI41 Și FACTURI41 este oarecum incorectă, deoarece schema nu este în BCNF. În COMENZI4 se manifestă atât dependența (NrFactură, CodFurnizor) \longrightarrow NrComandă, cât și NrComandă \longrightarrow CodFurnizor. Descompunerea corectă este:

COMENZI_FURNIZORI {NrComandă, CodFurnizor, DataComanda}

și

FACTURI {NrFactură, NrComandă, DataFactură, Valoare, TVADeduct}.

În concluzie, atunci când relațiile sunt construite direct din graful DF trebuie verificat în prealabil dacă, în graf, vreun atribut sau grup de attribute X dintr-o sursă compusă de genul:

(X, Y) \longrightarrow A,

(X, Y) \longrightarrow B,

(X, Y) \longrightarrow C

nu este destinație a unei dependențe de genul A \longrightarrow X. Dacă da, atunci se păstrează A \longrightarrow X, iar în celelalte DF sursa (X,Y) se înlocuiește cu (A,Y), adică:

(A, Y) \longrightarrow B,

(A, Y) \longrightarrow C

Ilustrarea acestui algoritm în transformării grafului din figura 5.27 este prezentată în figura 5.28.

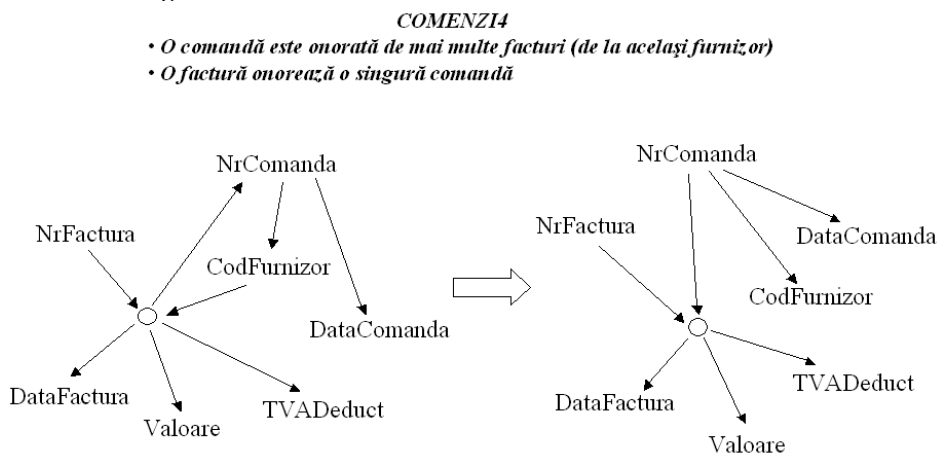


Figura 5.28. Transformarea grafului din figura 5.27 pentru aducerea relației în BCNF