Capitolul 4. Dependențe funcționale. A doua formă normalizată

Cea de-a doua formă normală (2NF) se bazează pe dependențele funcționale *totale* sau *elementare*. Drept care acest capitol începe cu prezentarea dependențelor funcționale (DF), apoi tratează câteva modalități de reprezentare a DF, iar în continuare sunt puse în evidență elementele care împart dependențele funcționale în elementare (totale) şi ne-elementare (parțiale). În final, pe baza acestei tipologii, sunt prezentate modalitățile de "rupere" a bazei de date în mai multe relații aflate în 2NF. În plus, după cum vom vedea, eliminarea dependențelor funcționale parțiale rezolvă şi problema bazelor de date a căror schemă în 1NF încalcă restricția de entitate.

4.1. Dependențe funcționale

Problema stabilirii dependenţelor dintre atribute este fundamentală în procesul elaborării bazelor de date relaţionale. Aducerea structurii bazei într-o variantă care minimizează redundanțele şi anomaliile manifestate la adăugarea/modificarea/ştergerea unor linii din tabele presupune parcurgerea unui proces de normalizare a relaţiilor, normalizare care este centrată tocmai pe conceptul de dependenţă dintre atributele bazei.

4.1.1. Definiție

Dependența funcțională reprezintă o generalizare a conceptului de cheie primară[1]. Desemnăm prin `Data2` un atribut sau combinație de atribute dintr-o relație oarecare şi prin `Data1` un alt atribut sau grup de atribute din aceeaşi relație. Se spune că `Data2` este în *dependență funcțională* (sau depinde funcțional) de `Data1` atunci când cunoaşterea unei valori pentru `Data1` permite determinarea (cunoaşterea) a maximum o valoare pentru `Data2`. `Data1` se numeşte *sursa* sau *determinantul* dependenţei funcţionale, iar `Data2` reprezintă *destinația* (*scopul* sau *determinatul*) dependenţei funcţionale.

Formal, dacă notăm relația R $\{A_1, A_2, ..., A_n\}$ - unde R este numele relației, A_i sunt atributele relației R - şi prin `X` şi `Y` două subansambluri din $\{A_1, A_2, ..., A_n\}$, se spune că există o dependenţă funcţională între `X` şi `Y` (simbolizată `X`$\longrightarrow$`Y`) dacă şi numai dacă :

- fiecare apariție (valoare) a lui `X` poate fi asociată unei singure apariții a lui `Y`,

[1] Vezi şi [Garcia-Molina s.a.02], [Date86]

- două apariţii identice ale lui X nu pot fi asociate decât aceleiaşi apariţii a lui Y.

Într-o prezentare şi mai explicită, Date[2] defineşte dependenţa funcţională după cum urmează: dată fiind o relaţie R, subansamblul de atribute Y din R depinde funcţional de subansamblul X (tot din R), dacă şi numai dacă, ori de câte ori două tupluri din R prezintă aceeaşi valoare pentru X, obligatoriu valoarea celor două tupluri este identică şi pentru Y.

Se spune despre o dependenţă că este *trivială* dacă şi numai dacă destinaţia este un subset al sursei: dacă $Y \subseteq X$, atunci $X \longrightarrow Y$.

Revenim asupra relaţiei universale VÎNZĂRI prezentată în capitolul 2 (tabelul 2.1) şi discutată în finalul capitolului 3 (paragraful 3.4). Cunoaşterea numărului facturii permite determinarea cu exactitate a codului clientului, deoarece o factură este (conform legii şi metodologiei în vigoare) întocmită pentru un singur client. Se poate scrie:

NrFact $\longrightarrow$ DataFact
NrFact $\longrightarrow$ CodCl
NrFact $\longrightarrow$ Obs
dar şi
NrFact $\longrightarrow$ (DataFact, CodCl)
NrFact $\longrightarrow$ (DataFact, CodCl, Obs)

Situaţia de mai sus se încadrează în cazurile în care sursa (determinantul) este cheia primară a relaţiei. Implicit, în orice relaţie R există dependenţele funcţionale:
cheia lui R $\longrightarrow$ *celelalte atribute ale lui R*.

Luăm un alt exemplu: un cod poştal nu poate desemna adrese din oraşe şi comune diferite. Se poate scrie, deci:

CodPoştal $\longrightarrow$ Localitate
CodPoştal $\longrightarrow$ Judeţ

Observaţie
Uneori, începătorii au probleme în ceea ce priveşte "sensul" dependenţelor funcţionale. Dependenţa funcţională CodPoştal $\longrightarrow$ Localitate nu înseamnă că o localitate are un singur cod poştal, ci, invers, că *unui cod poştal îi corespunde o singură localitate*.

O dependenţă funcţională în care scopul (destinaţia) este alcătuit dintr-un singur atribut se numeşte *dependenţă funcţională canonică* sau dependenţă funcţională în formă canonică[3].

---

[2] [Date86], p.365

[3] [Saleh4], p.76. Vezi şi [Garcia-Molina s.a 01], p.89

Două atribute **nu** sunt în dependenţă funcţională atunci când cunoaşterea unei valori oarecare a primului atribut:

- fie nu permite cunoaşterea nici uneia dintre valorile celui de-al doilea (nu există nici un "raport" între cele două atribute); în acest caz dependenţa nefuncţională echivalează cu inexistenţa vreunei dependenţe între cele două atribute;
- fie permite cunoaşterea mai multor valori ale celui de-al doilea atribut.

Relaţia VÎNZĂRI

Între atributele `DataFact` şi `CodCl` nu există dependenţă funcţională. O valoare a atributului `DataFact` nu furnizează nici o informaţie cu privire la valoarea de pe aceeaşi linie a atributului `CodCl`. Se scrie:

`DataFact` —/→ `CodCl`

De asemenea, deoarece pentru un client se pot întocmi mai multe facturi (aferente fiecărei vânzări):

`CodCl` —/→ `NrFact`

Într-o factură se poate consemna vânzarea mai multor produse/servicii; prin urmare, unei valori a `NrFact`, îi corespund mai multe valori ale atributului `CodPr`. Se notează:

`NrFact` —/→ `CodPr`.

Relaţia LINII_ARTICOLE_CONTABILE

Să examinăm tabela din figura 4.1.

LINII_ARTICOLE_CONTABILE

NotaContabilă	Data	Operaţiune	ContDebitor	ContCreditor	Suma
150	30.11.2002	1	300	401	10000000
150	30.11.2002	1	4426	401	1800000
150	30.11.2002	2	401	5311	7000000
150	30.11.2002	2	401	5121	4800000
151	01.12.2002	1	5121	700	5250000

Figura 4.1. Tabela LINII_ARTICOLE_CONTABILE

- notă contabilă este alcătuită din mai multe articole contabile:

(`NotaContabilă, Data`) —/→ `Operaţiune`

- un articol contabil (înregistrare) poate fi compus, adică alcătuit din mai multe corespondenţe `ContDebitor` = `ContCreditor`:

(`NotaContabilă`, `Data`, `Operaţiune`) —/→ `ContDebitor`

(`NotaContabilă`, `Data`, `Operaţiune`) —/→ `ContCreditor`

(`NotaContabilă`, `Data`, `Operaţiune`, `ContDebitor`) —/→ `ContCreditor`

(`NotaContabilă`, `Data`, `Operaţiune`, `ContCreditor`) —/→ `ContDebitor`

Atunci când firmele îşi centralizează operaţiunile, utilizând de la o lună la alta aceleaşi numere pentru notele contabile,

`NotaContabilă` —/→ `Data`

Pe lângă aceste două exemple, am mai putea aminti faptul că localități cu aceeași denumire există în mai multe județe. Spre exemplu, localitatea *Vînători* există în județele Vrancea, Galați, Iași, Neamț etc. Aceasta înseamnă că:

Localitate —/→ Judeţ

Dependențele funcționale reprezintă legături semantice între atribute ale bazei de date. Au fost propuşi algoritmi de identificare şi gestionare a dependențelor funcționale într-o relație. Cea mai mare parte a acestora sunt total inutili, pentru că, efectiv, nimeni nu poate stabili o dependență fără a cunoaşte regula sau restricția care se ascunde în spatele acesteia, altfel spus, fără a cunoaşte temeinic realitatea modelată. Cu atât mai mult calculatorul care are mari probleme în a lucra la nivel semantic, ca să nu mai vorbim de cel pragmatic.

4.1.2. Dependențe funcționale cu sursa compusă

Există dependențe funcționale care prezintă în partea stângă (sursa dependenței funcționale sau determinantul) două sau mai multe atribute. Acestea sunt dependențele funcționale cu sursa compusă. Astfel, în relația VÎNZĂRI:

- o factură conține mai multe linii: NrFact —/→ Linie
- pe o factură pot apărea mai multe produse: NrFact —/→ CodPr
- un produs apare, de obicei, pe mai multe facturi: CodPr —/→ NrFact
- pentru fiecare factură, o linie este unică:

(NrFact, Linie) ⟶ CodPr
(NrFact, Linie) ⟶ Cantitate
(NrFact, Linie) ⟶ PreţUnit

- dacă stabilim că un produs nu poate apărea pe o factură decât o singură dată, atunci sunt valabile DF:

(NrFact, CodPr) ⟶ Linie
(NrFact, CodPr) ⟶ Cantitate
(NrFact, CodPr) ⟶ PreţUnit

Relația LINII_ARTICOLE_CONTABILE

În tabela din figura 4.1 stabilirea dependențelor funcționale depinde de modul în care firma îşi organizează evidența contabilă. Astfel, marea majoritate a firmelor îşi centralizează operațiunile, pe tipuri (intrări de materiale, consumuri de materiale, operațiuni legate de casierie, conturile din bancă etc.), utilizând, de la o lună la altă, aceleaşi numere pentru notele contabile (ex: nota 1 - încasări prin casierie, nota 10 - pentru înregistrarea amortizării etc.). În acest caz, există dependența funcțională:

(1) (NotaContabilă, Data, Operaţiune, ContDebitor, ContCreditor) ⟶ Suma

Dacă unele firme, cu un volum de activitate mai redus, ar conta fiecare document primit/întocmit, generând astfel o nouă notă contabilă, dependențele funcționale s-ar putea scrie sub forma:

(2) NotaContabilă ⟶ Data

(3) (NotaContabilă,Operaţiune, ContDebitor, ContCreditor) ⟶ Suma

Fireşte, este valabilă şi dependența:

(4) (NotaContabilă, Operaţiune, ContDebitor, ContCreditor) ⟶ Data

Relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

Tabela din figura 3.7. (paragraful 3.3.2), aflată în prima formă normală, prezintă ca şi cheie primară combinația (Cota, Autor, CuvântCheie). De aceea, se pot inventaria următoarele cinci dependențe funcționale, în care destinații sunt atributele necheie.

(1) (Cota, Autor, CuvântCheie) ⟶ ISBN

(2) (Cota, Autor, CuvântCheie) ⟶ Titlu

(3) (Cota, Autor, CuvântCheie) ⟶ Editura

(4) (Cota, Autor, CuvântCheie) ⟶ LocSediuEd

(5) (Cota, Autor, CuvântCheie) ⟶ AnApariţie

Alte elemente luate în discuție pentru stabilirea DF:

- o carte poate fi scrisă de mai mulți autori: ISBN —/→ Autor
- un autor scrie mai multe cărți: Autor —/→ ISBN şi
- despre mai multe subiecte: Autor —/→ CuvântCheie
- într-o carte sunt tratate mai multe subiecte: ISBN —/→ CuvântCheie
- pentru o carte sunt în depozit mai multe exemplare: ISBN —/→ Cota

Cota este codul care identifică în mod unic o carte aflată în depozit, aşadar:

(6) Cota ⟶ ISBN

(7) Cota ⟶ Titlu

(8) Cota ⟶ Editura

(9) Cota ⟶ LocSediuEd

(10) Cota ⟶ AnApariţie

Fiecare titlu de carte este identificat fără ambiguitate de ISBN:

(11) ISBN ⟶ Titlu

(12) ISBN ⟶ Editura

(13) ISBN ⟶ LocSediuEd

(14) ISBN ⟶ AnApariţie

O editură are un singur sediu central:

(15) Editura ⟶ LocSediuEd

Prin urmare, dintre cele 15 DF inventariate, primele cinci au sursa compusă, celelalte sursa simplă.

Relația DOTARE_JUCĂRII_1

Pentru identificarea dependențelelor funcționale care se manifestă în relația prezentată în figura 3.14 (finalul paragrafului 3.3.3), discuția poate fi purtată după cum urmează:

- codul identifică o familie:
  (1) `CodFamilie` $\longrightarrow$ `NumeFamilie`

- o familie are mai mulți copii: `CodFamilie` $\not\longrightarrow$ `Copil`

- un copil face parte dintr-o singură familie, însă atribut `Copil` nu identifică în mod unic un singur copil, deoarece care se referă numai la prenume (*Ion*, *Vasile* etc.): `Copil` $\not\longrightarrow$ `CodFamilie`

- cu toată recesiunea, un copil are mai multe jucării:
  (`CodFamilie`, `Copil`) $\not\longrightarrow$ `Jucărie`

- o jucărie aparține unui singur copil, însă două "instanțe" ale aceleaşi jucării nu pot fi diferențiate prin denumire, iar acelaşi tip de jucărie poate fi cumpărat mai multor copii, aşa că:
  `Jucărie` $\not\longrightarrow$ `CodFamilie`
  `Jucărie` $\not\longrightarrow$ `NumeFamilie`
  `Jucărie` $\not\longrightarrow$ `Copil`
  `Jucărie` $\not\longrightarrow$ `DataCumpărării`

- dacă un copil are o singură jucărie de acelaşi tip, se poate scrie:
  (2) (`CodFamilie`, `Copil`, `Jucărie`) $\longrightarrow$ `DataCumpărării`[4]

Relația LOCALITĂȚI_CODURI_VECHI

Continuăm cu relația din figura 3.15 (paragraful 3.4) care se referă la vechiul sistem de codificare poştală:

- pentru un oraş sau comună exista un singur cod poştal:
  (1) `CodPoştal` $\longrightarrow$ `OraşComună`
  (2) `CodPoştal` $\longrightarrow$ `Judeţ`

- într-o comună, mai multe sate partajau un singur cod poştal:

[4] Dealtminteri, această DF decurge din însăşi calitatea de cheie primară a sursei

CodPoştal —/→ Sat

Relația CODURI_NOI_V2

Pentru noul sistem de codificare poştală lucrurile stau diferit. Astfel condițiile care guvernează relațiile tabelei din figura 3.17 (paragraful 3.4) pot fi sistematizate după cum urmează:

- un cod poştal nu poate fi alocat la două localități, cu atât mai puțin la două comune sau județe diferite:
  (1) CodPoştal ⟶ Localitate
  (2) CodPoştal ⟶ Comuna
  (3) CodPoştal ⟶ Judeţ

- un acelaşi cod poate fi asociat imobilelor din două străzi diferite:
  CodPoştal —/→ Strada, CodPoştal —/→ TipNr,
  CodPoştal —/→ NrIniIniţial, CodPoştal —/→ LitIniţială

- porțiuni din aceaşi stradă pot avea coduri diferite:
  (Localitate, Strada) —/→ CodPoştal

- intervalele adreselor sunt disjuncte la acordarea codurilor, aşa că putem formula şi două DF cu sursa compusă:
  (4) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ NrFinal
  (5) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ LitFinală

Baza de date VÎNZĂRI

Dat fiind numărul de atribute şi specificul aplicației, în acest caz discuțiile sunt ceva mai laborioase:

- fiecare județ are un nume şi un indicativ unice:
  (1) Jud ⟶ Regiune
  (2) Judeţ ⟶ Regiune
  (3) Jud ⟶ Judeţ
  (4) Judeţ ⟶ Jud
- pe baza discuției de la exemplul anterior (CODURI_NOI_V2):
  (5) CodPost ⟶ Localitate
  (6) CodPost ⟶ Comuna
  (7) CodPost ⟶ Jud
  (8) CodPost ⟶ Judeţ

- fiecare firmă-client este identificată în mod unic atât prin cod (CodCl):
  (9) CodCl ⟶ DenCl
  (10) CodCl ⟶ CodFiscal

(11) CodCl $\longrightarrow$ StradaCl
(12) CodCl $\longrightarrow$ NrStradaCl
(13) CodCl $\longrightarrow$ BlocScApCl
(14) CodCl $\longrightarrow$ TelefonCl
(15) CodCl $\longrightarrow$ CodPost
(16) CodCl $\longrightarrow$ Localitate
(17) CodCl $\longrightarrow$ Comuna
(18) CodCl $\longrightarrow$ Jud
(19) CodCl $\longrightarrow$ Judet
(20) CodCl $\longrightarrow$ Regiune

- cât şi prin cod fiscal (CodFiscal):
(21) CodFiscal $\longrightarrow$ DenCl
(22) CodFiscal $\longrightarrow$ CodCl
(23) CodFiscal $\longrightarrow$ StradaCl
(24) CodFiscal $\longrightarrow$ NrStradaCl
(25) CodFiscal $\longrightarrow$ BlocScApCl
(26) CodFiscal $\longrightarrow$ TelefonCl
(27) CodFiscal $\longrightarrow$ CodPost
(28) CodFiscal $\longrightarrow$ Localitate
(29) CodFiscal $\longrightarrow$ Comuna
(30) CodFiscal $\longrightarrow$ Jud
(31) CodFiscal $\longrightarrow$ Judet
(32) CodFiscal $\longrightarrow$ Regiune

- elementul care identifică un produs este codul acestuia:
(33) CodPr $\longrightarrow$ DenPr
(34) CodPr $\longrightarrow$ UM
(35) CodPr $\longrightarrow$ Grupa
(36) CodPr $\longrightarrow$ ProcTVA

- o factură se întocmeşte pentru un singur client:
(37) NrFact $\longrightarrow$ CodCl
(38) NrFact $\longrightarrow$ CodFiscal

- o factură se întocmeşte o singură dată:
(39) NrFact $\longrightarrow$ DataFact

- în plus:
(40) NrFact $\longrightarrow$ Obs

- o factură conţine mai multe linii: NrFact $\not\longrightarrow$ Linie

- fiecare linie dintr-o factură se referă la vânzarea unui produs (CodPr), indicând cantitatea şi preţul unitar la care s-a făcut vânzarea:
  (41) (NrFact, Linie) ⟶ Cantitate
  (42) (NrFact, Linie) ⟶ PreţUnit
  (43) (NrFact, Linie) ⟶ CodPr
  (44) (NrFact, Linie) ⟶ DenPr
  (45) (NrFact, Linie) ⟶ UM
  (46) (NrFact, Linie) ⟶ ProcTVA

- într-o factură, un produs apare pe o singură linie:
  (47) (NrFact, CodPr) ⟶ Cantitate
  (48) (NrFact, CodPr) ⟶ PreţUnit
  (49) (NrFact, CodPr) ⟶ Linie

- o încasare are la bază un document primar (*Ordin de plată*, *Chitanţă* etc.):
  (50) CodÎnc ⟶ DataÎnc
  (51) CodÎnc ⟶ CodDoc
  (52) CodÎnc ⟶ NrDoc
  (53) CodÎnc ⟶ DataDoc

- documentele de încasare pot prezenta numere similare de la un client la altul; spre exemplu, *Ordinul de plată cu nr. 101* poate fi emis de mai mulţi clienţi (bineînţeles, este vorba de trei documente diferite care au însă acelaşi număr, poate chiar şi dată):
  (CodDoc, NrDoc) —/→ DataDoc
  (CodDoc, NrDoc) —/→ NrFact
  (CodDoc, NrDoc) —/→ CodCl

- o factură este încasată în mai multe tranşe: NrFact —/→ CodÎnc

- o încasare poate achita mai multe facturi: CodÎnc —/→ NrFact

- într-o încasare, o tranşă se referă la o singură factură:
  (54) (CodÎnc, NrFact) ⟶ Tranşă,
- nu însă şi (CodÎnc, Tranşă) ⟶ NrFact, deoarece, într-o aceeaşi încasare pot exista două tranşe egale (ex. 5 000 000 lei) referitoare la două facturi diferite.

4.1.3. Proprietăţi ale dependenţelor funcţionale, închidere şi acoperire minimală

Dependenţele funcţionale prezintă câteva proprietăţi pe care le discutăm, în cea mai mare parte, pe calapodul relaţiei VÎNZĂRI.

1. *Reflexivitate*[5] sau *incluziune*[6]: Dacă $Y \subseteq X$, atunci $X \longrightarrow Y$, adică un ansamblu oarecare de atribute determină pe oricare dintre atributele sale. Exemplu: Y = NrFact, X = (CodÎnc, NrFact). Se poate scrie: (CodÎnc, NrFact) $\longrightarrow$ NrFact.

2. *Extensie* (*Creştere* sau *Multiplicare*): Dacă $X \longrightarrow Y$, atunci $XZ \longrightarrow YZ$. Cu alte cuvinte, dacă X determină pe Y, cele două ansambluri de atribute pot fi "îmbogățite" printr-un al treilea. Exemplu: NrFact $\longrightarrow$ CodCl, prin urmare (NrFact, CodÎnc) $\longrightarrow$ (CodCl, CodÎnc).

3. *Tranzitivitate*: Dacă $X \longrightarrow Y$ şi $Y \longrightarrow Z$, atunci $X \longrightarrow Z$. Exemplu: Deoarece NrFact $\longrightarrow$ CodCl şi CodCl $\longrightarrow$ DenCl, atunci NrFact $\longrightarrow$ DenCl. Practic, dacă aplicăm această proprietate, obținem următoarele DF:

(55) NrFact $\longrightarrow$ DenCl
(56) NrFact $\longrightarrow$ CodFiscal
(57) NrFact $\longrightarrow$ StradaCl
(58) NrFact $\longrightarrow$ NrStradaCl
(59) NrFact $\longrightarrow$ BlocScApCl
(60) NrFact $\longrightarrow$ TelefonCl
(61) NrFact $\longrightarrow$ CodPost
(62) NrFact $\longrightarrow$ Localitate
(63) NrFact $\longrightarrow$ Comuna
(64) NrFact $\longrightarrow$ Jud
(65) NrFact $\longrightarrow$ Judet
(66) NrFact $\longrightarrow$ Regiune

Aceste trei proprietăți ale dependenţelot funcţionale sunt denumite *axiomele lui Armstrong*. Alte reguli se referă la:

4. *Autodeterminare*: $X \longrightarrow X$.

5. *Uniune*: Dacă $X \longrightarrow Y$ şi $X \longrightarrow Z$, atunci $X \longrightarrow YZ$. Exemplu: Întrucât NrFact $\longrightarrow$ CodCl şi NrFact $\longrightarrow$ DataFact, rezultă NrFact $\longrightarrow$ (CodCl, DataFact).

6. *Pseudo-tranzitivitate*: Dacă $X \longrightarrow Y$ şi $WY \longrightarrow Z$, atunci $WX \longrightarrow Z$. Să discutăm cazul relației CĂRȚI_AUT {Cota, ISBN, Autor, NrPagini}, unde NrPagini indică numărul de pagini din cartea curentă scrise de autorul curent

[5] Denumirea de *reflexivitate* apare în [Date04], p.338, [Elmasri & Navathe 00], p.479, [Garcia-Molina s.a. 02], p.99 etc.

[6] Termenul *incluziune* este folosit în [O'Neil & O'Neil 01], p.362

(din fiecare tuplu al relației). Întrucât `Cota` ⟶ `ISBN` şi (`Autor, ISBN`) ⟶ `NrPagini`, este valabilă şi dependența (`Autor, Cota`) ⟶ `NrPagini`.

7. *Descompunere*: Dacă `X` ⟶ `YZ`, atunci `X` ⟶ `Y` şi `X` ⟶ `Z`. Este opusa uniunii.

Hugh Darwen a formulat *Teorema Unificării Generale* (General Unification Theorem) prin care o parte de regulile de mai sus devin cazuri particulare ale următorului enunț:

8. Dacă `X` ⟶ `Y` şi `Z` ⟶ `W`, atunci `X` ∪ (`Z` – `Y`) ⟶ `YW`[7].

Chiar pentru cazuri simple, ansamblul tuturor DF într-o bază de date poate să devină extrem de stufos. De aceea, în procesul normalizării trebuie avut în vedere un echilibru între:

- identificarea tuturor DF ce reflectă restricții, reguli ale fenomenului/procesului modelat, şi
- includerea în setul DF numai a celor relevante, adică a unui ansamblu minimal din care, prin reguli de genul celor de mai sus, se pot obține toate celelalte DF.

În teorie, dat fiind un set de DF notat S, ansamblul tuturor DF determinate pe baza acestuia se numeşte *închidere* a lui şi se notează S^+. Multe lucrări consacrate bazelor de date prezintă algoritmi pentru calculul închiderilor, atât pentru toată relația, cât şi pentru un grup de atribute, dat fiind un set de DF de plecare. Utilizatea închiderilor în practica proiectării bazelor de date rămâne încă de demonstrat.

Mult mai importantă este determinarea *acoperii minimale*, adică a setului minim de DF prin care pot fi determinate toate celelalte dependențe din relație. După Chris Date, un set *S* de dependețe funcționale este ireductibil dacă şi numai dacă satisface următoarele trei proprietăți[8]:

1. Partea dreaptă (destinația) a fiecărei dependențe din S implică un singur atribut.
2. Sursa fiecărei DF este ireductibilă, adică nici un atribut nu poate fi eliminat fără distrugerea dependenței; se spune că DF este ireductibilă la stânga.
3. Nici o DF din S nu poate fi eliminată fără afectarea închiderii S^+, altfel spus, fără pierderea a o serie de informații.

Din păcate, deşi elegant postulată, acoperirea minimală ridică probleme majore chiar şi la un număr mediu de atribute, ca să nu vorbim cât de alunecoasă este

---

[7] Hugh Darwen - *The Role of Functional Functional Dependence in Query Decomposition*, în Date, C.J., Darwen, H. - *Relational Database Writings 1989-1991*, Addison-Wesley, Reading, Mass., 1992

[8] [Date00], p.338. Vezi şi [Garcia-Molina00] p.482

proprietatea a treia formulată de Date. În plus, de multe ori, pentru o bază de date există mai multe acoperiri minimale[9]. Lucrurile pot fi însă simplificate parțial eliminând din discuție dependențele simetrice şi redundante.

4.1.4. Dependențe funcționale simetrice şi redundante

Deseori, o bază de date sau relație prezintă mai multe chei candidat. Alegerea dintre acestea a cheii primare are în vedere, pe lângă unicitate, compoziție minimală şi valori nenule ale oricărui atribut component, şi elemente ce țin de facilitatea utilizării: uşurința reținerii (de către utilizator), lungime cât mai mică, constanță în timp etc.

În procesul normalizării, problemele ridicate de aceste dependențe reciproce dintre cheile candidat trebuie tratate cu prudență. De cele mai multe ori, este recomandat ca dintre dependențele ale unei relații să fie eliminate toate cele în care sursa este o cheie candidat şi păstrarea numai celor în care determinantul este atributul cheie-primară.

Revenim la cele 54 de dependențe din relația universală VÎNZĂRI tratate în finalul paragrafului 4.1.2. Este evident că dependențele (3), `Jud` $\longrightarrow$ `Judeţ`, şi (4), `Judeţ` $\longrightarrow$ `Jud`, sunt simetrice. Deşi corecte, şi dependențele (1) - `Jud` $\longrightarrow$ `Regiune` - şi (2) - `Judeţ` $\longrightarrow$ `Regiune` - "se calcă pe picior". Pentru comoditate, putem alege atributul `Jud` ca identificator "de bază" al unui județ, eliminând (pe nedrept), pe lângă (4), toate DF în care sursa simplă este atributul `Judeţ` iar destinațiile identice cu cele ale dependențelor în care sursă este `Jud`. Ce-i drept, efortul nu este chiar supraomenesc, deoarece este o singură dependență de acest tip, (2).

La fel se pune problema şi pentru perechile de atribute `CodCl` şi `DenCl` şi `CodCl` şi `CodFiscal`. Toate identifică un acelaşi tip de entitate - o firmă client. Investim `CodCl` drept "reprezentant unic" al unui client. `DenCl` nefiind luat prea mult în seamă în setul de DF, avem de eliminat doar DF în care `CodFiscal` este sursă simplă: (21) - (32). În privința perechii `CodPr` şi `DenPr` nu avem a ne epuiza, deoarece, aproape intenționat, au fost omise din setul DF cele în care `DenPr` apare ca sursă.

Atenție ! Nu facem acelaşi lucru în relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 cu atributele `Cota` şi `ISBN`, deoarece primul se referă la un exemplar "fizic" al unei cărți din bibliotecă, în timp ce al doilea se referă la un titlu (carte publicată la o editură).

Tipul acesta de redundanță e uşor de identificat şi eliminat, chiar dacă nu se leagă vizibil de nici o teoremă sau lemă ce constituie fundamentul normalizării. Setul DF conține însă şi un alt tip de redundanță/circularitate. Astfel, deoarece fiecare linie dintr-o factură se referă la un produs vândut la un preț unitar şi într-o

[9] [Garcia-Molina00] p.482

anumită cantitate, sursele DF în care destinaţiile sunt `Cantitate`, `PreţUnit` (plus `DenPr`, `UM` şi `ProcTVA`) pot fi atât (`NrFact`, `Linie`), cât şi (`NrFact`, `CodPr`). Procedând analog situaţiei de mai sus (dar cu mai multă strângere de inimă), decidem ca fiecare linie/produs dintr-o factură să fie identificat de sursa (`NrFact`, `Linie`). Aşa că eliminăm dependenţele în care sursă este "concurentul", adică (`NrFact`, `CodPr`) şi destinaţiile sunt identice: (47), (48) şi (49).

4.2. Reprezentări ale dependenţelor funcţionale

Elaborarea structurii bazelor de date prin tehnica normalizării necesită o gestiune riguroasă a tuturor dependenţelor funcţionale. În practică, parcurgerea tuturor relaţiilor dintre atributele unei baze de date se poate solda cu identificarea a zeci sau chiar sute de dependenţe. Conectarea lor, determinarea redundanţelor, incluziunilor, tranzitivităţilor ridică astfel uneori probleme apăsătoare. Drept pentru care au fost elaborate diverse instrumente menite a simplifica vizualizarea şi lucrul cu DF. În acest paragraf vom prezenta trei asemenea instrumente: diagrame, grafuri şi matrice ale dependenţelor funcţionale.

4.2.1. Diagrame ale DF

Diagramele DF au o utilizare limitată[10], deoarece la un număr mare de atribute reprezentarea dependenţelor este foarte anevoiasă. O dependenţă dintre două atribute este indicată printr-o linie (sau un număr de segmente) ce porneşte din sursă şi se termină printr-o săgeată "înfiptă" în destinaţie (determinant). Când sursele sau destinaţiile sunt compuse, atunci, mai întâi, se unesc atributele din grupul determinant/destinaţie, iar terminaţiile dependenţelor vizează segmentele de grup.

O porţiune din relaţia VÎNZĂRI

Dacă luăm în considerare DF de la (33) la (36) şi de la (41) la (49) discutate în paragraful 4.1.2, diagrama acestora ar arăta ca în figura 4.2.

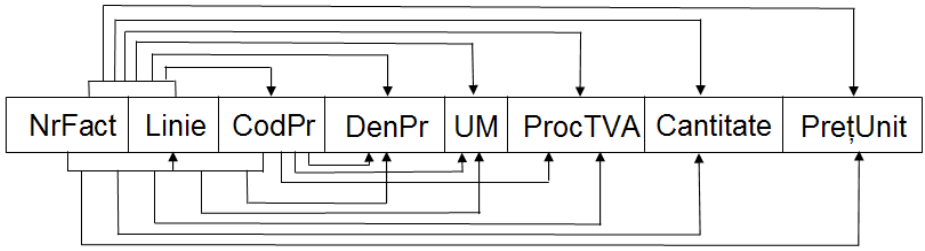

Figura 4.2. Diagrama DF - o parte din atributele relaţiei VÎNZĂRI

[10] Lucrarea din care am preluat diagramele DF este [Pratt&Adamski 91], p.261. La noi, lucrarea cea mai cunoscută în care sunt folosite este [Oprea 99]

După cum se observă, există un grad apăsător de redundanță, aşa încât ne vom conforma recomandărilor din paragraful 4.1.4 şi vom elimina DF simetrice/redundante, obținând diagrama din figura 4.3.

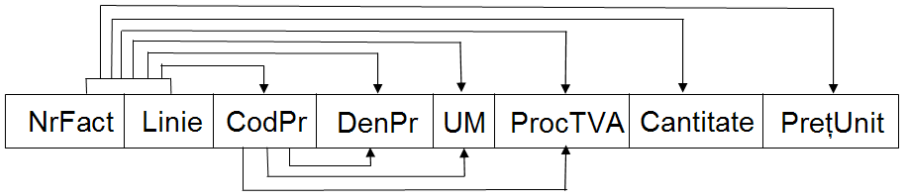

Figura 4.3. Diagrama DF din figura 4.2, însă fără dependenţele redundante

Relația LINII_ARTICOLE_CONTABILE

Pentru relația din figura 4.1, dacă se foloseşte primul sistem de contare, constant de la lună la lună, în care dependența funcțională este (1), diagrama are forma din figura 4.4, iar pentru al doilea tip, cel în care se manifestă dependențele (2) (3) şi (4), diagrama este cea din figura 4.5.

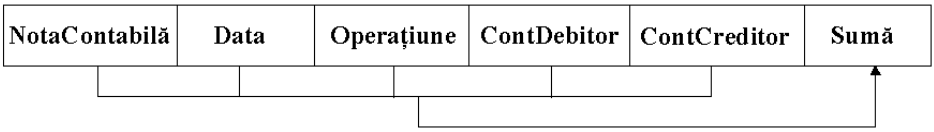

Figura 4.4. Diagrama DF - exemplul 2 - primul tip de contare

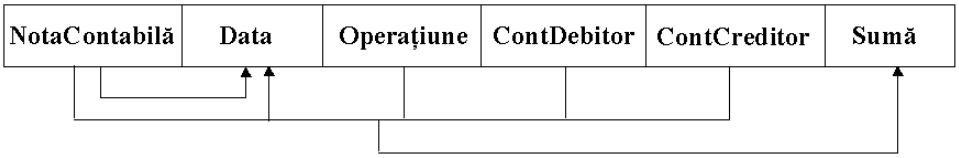

Figura 4.5. Diagrama DF - exemplul 2 - al doilea tip de contare

Relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

Diagrama dependențelor acestei relații (figura 3.7), dependențe amintite în paragraful 4.1.2 este prezentată în figura 4.6.

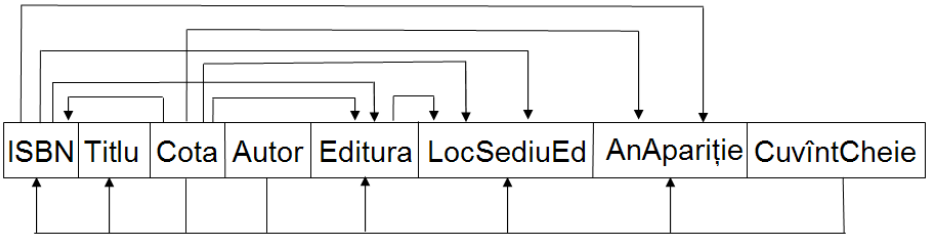

Figura 4.6. Diagrama DF - exemplul 3

Dacă pentru exemplele de mai sus diagramele pot fi trasate cu oarecare uşurinţă, pentru exemplul 7 din paragraful 4.1.2, cel dedicat bazei de date VÎNZĂRI, în care apar mult mai multe atribute şi dependenţe, diagrama este practic imposibil de reprezentat pe o coală A4. În asemenea situaţii, care formează "grosul" cazurilor practice, mult mai la îndemână sunt graful şi matricea DF.

4.2.2. Graful dependenţelor funcţionale

Într-un graf al dependenţelor funcţionale, acestea sunt reprezentate prin săgeţi ce leagă sursa şi destinaţia. La dependenţele funcţionale simple săgeata uneşte cele două atribute. Când sursa este compusă, se foloseşte un conector care leagă, în prima instanţă, atributele-determinant, iar săgeata uneşte acest conector cu atributul destinaţie. Astfel, diagrama din figura 4.3 îmbracă forma grafului din figura 4.7.

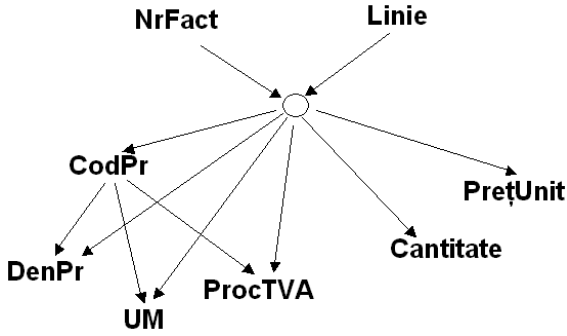

Figura 4.7. Reprezentarea sub formă de graf a diagramei din figura 4.3.

Cât priveşte dependenţele exemplului 3 (paragraful 4.1.2), corespunzător diagramei din figura 4.6 este graful din figura 4.8. În ciuda densităţii de săgeţi, se remarcă un plus de claritate şi folosirea mai judicioasă a spaţiului.

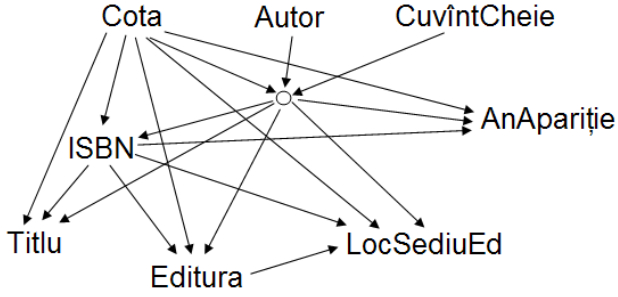

Figura 4.8. Reprezentarea sub formă de graf a diagramei din figura 4.6.

Spre deosebire de diagramă, cu graful DF se poate reprezenta voluminosul ansamblu al DF pentru baza de date VÎNZĂRI – exemplul 7 (paragraful 4.1.2). Pentru simplificare, vom reprezenta numai DF (1), (3), (5)-(7), (9)-(15), (33)-(46), (50)-(54), adică, după cum vom vedrea în pagraful 5.1, acoperirea minimală- vezi figura 4.9[11].

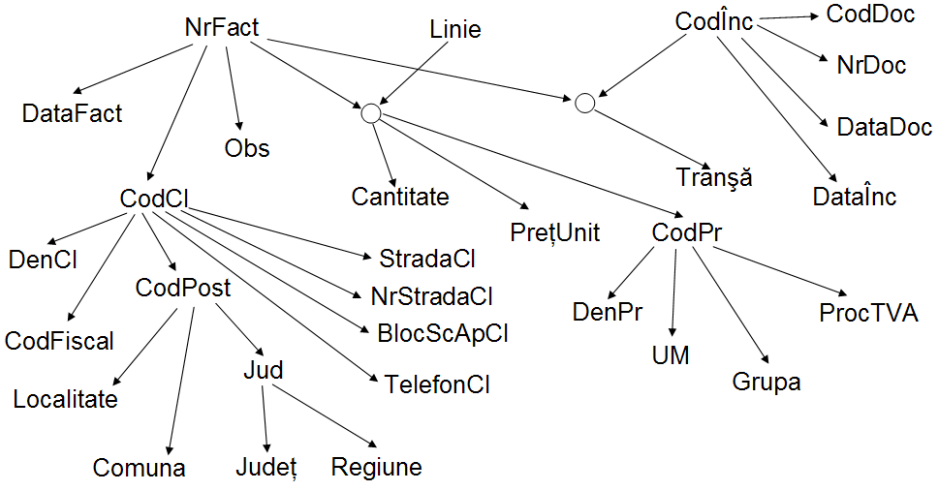

Figura 4.9. Graful DF corespunzător acoperirii minimala a BD VÎNZĂRI

4.2.3. Matricea dependenţelor funcţionale

Construirea matricei dependenţelor funcţionale demarează prin crearea unui tabel în care fiecărui atribut i se va asocia o linie şi o coloană. După includerea tuturor atributelor, se adaugă toate grupurile de atribute care constituie surse de dependenţe funcţionale. Apoi, se parcurg, pe rând, coloanele, în fiecare celulă scriindu-se 1 atunci *atributul de pe coloană este sursă a unei DF în care destinaţia este atributul/grupul de pe linie.*

Astfel, corespunzător diagramei din figura 4.3 şi grafului din figura 4.7, matricea DF s-ar putea prezenta ca în figura 4.10.

		Surse de dependenţe funcţionale								
		1	2	3	4	5	6	7	8	9
1	NrFact									
2	Linie									
3	CodPr									1
4	DenPr			1						1
5	UM			1						1
6	ProcTVA			1						1
7	Cantitate									1
8	PreţUnit									1

[11] De fapt, eliminăm atât depedenţele simetrice şi redundante, dar şi pe cele tranzitive, luând-o astfel înainte cu „materia", deoarece acesta este subiectul capitolului următor.

9	(NrFact, Linie)									

Figura 4.10. Matricea corespunzătoare diagramei din fig. 4.2 şi grafului din fig. 4.10

Pentru simplificarea figurii, de obicei se recurge la eliminarea coloanelor care nu sunt surse de dependențe funcționale, ca în figura 4.11.

		Surse de DF	
		3	**9**
1	NrFact		
2	Linie		
3	CodPr		1
4	DenPr	1	1
5	UM	1	1
6	ProcTVA	1	1
7	Cantitate		1
8	PreţUnit		1
9	(NrFact, Linie)		

Figura 4.11. Matricea anterioară simplificată

Păstrând paralelismul, figura 4.12 prezintă matricea DF corespunzătoare diagramei din figura 4.6 şi grafului din figura 4.8.

		Surse de DF			
		1	**2**	**6**	**9**
1	Cota				
2	ISBN	1			1
3	Autor				
4	CuvântCheie				
5	Titlu	1	1		1
6	Editura	1	1		1
7	LocSediuEd	1	1	1	1
8	AnApariţie	1	1		1
9	(Cota, Autor, CuvântCheie)				

Figura 4.12. Matricea corespunzătoare diagramei din fig. 4.6 şi grafului din fig. 4.8

Fireşte, cea mai complicată matrice este cea a bazei de date VÎNZĂRI (exemplul 7 (paragraful 4.1.2), figura 4.13 fiind transpunerea grafului din figura 4.9.

		Surse DF							
		1	**4**	**7**	**14**	**19**	**25**	**31**	**32**
1	Jud								
2	Judeţ	1							
3	Regiune	1							
4	CodPost								
5	Localitate		1						
6	Comună		1						
7	CodCl								
8	DenCl			1					
9	CodFiscal			1					
10	StradaCl			1					
11	NrStradaCl			1					
12	BlocScApCl			1					

13	TelefonCl			1					
14	CodPr							1	
15	DenPr				1				
16	UM				1				
17	Grupa				1				
18	ProcTVA				1				
19	NrFact								
20	DataFact					1			
21	Obs					1			
22	Linie								
23	Cantitate							1	
24	PreţUnit							1	
25	CodÎnc								
26	DataÎnc						1		
27	CodDoc						1		
28	NrDoc						1		
29	DataDoc						1		
30	Tranşă								1
31	(NrFact, Linie)								
32	(CodÎnc, NrFact)								

Figura 4.13. Matricea simplificată a DF a bazei de date VÎNZĂRI

4.3. Dependenţe funcţionale totale

Noţiunea de *dependenţă funcţională totală* a fost introdusă de Melkanoff. O dependenţă funcţională `Data1` $\longrightarrow$ `Data2` este *totală* dacă nu există nici un atribut/combinaţie `Dată3` ca subansamblu al `Data1`, care să verifice dependenţa funcţională `Dată3` $\longrightarrow$ `Dată2`.

Formal, dacă `X` = { A_i, A_j,...., A_k } reprezintă un grup de atribute ale relaţiei R, iar A_m un atribut al aceleiaşi relaţii, unde $A_m \not\subset$ `X`.

Se spune că există o dependenţă funcţională totală între X şi A_m dacă:

1. `X` $\longrightarrow$ A_m ,

2. nu există nici un subansamblu S de atribute, S $\subset$ X, pentru care S $\longrightarrow$ A_m .

Se notează: `X` $\xrightarrow{total}$ A_m

Dependenţa funcţională totală mai este denumită şi *elementară*, *plină, deplină* sau *completă*. Implicit, toate dependenţele în care sursa este simplă (alcătuită dintr-un singur atribut) sunt dependenţe funcţionale complete. Problema prezintă importanţă în cazul dependenţei funcţionale cu sursa compusă. Astfel, în BD VÎNZĂRI, dependenţele :

(`NrFact, Linie`) $\longrightarrow$ `CodPr`
(`NrFact, Linie`) $\longrightarrow$ `Cantitate`
(`NrFact, Linie`) $\longrightarrow$ `PretUnit`

sunt, toate, totale, deoarece: NrFact —/→ CodPr, NrFact —/→ Cantitate, NrFact —/→ PretUnit, Linie —/→ CodPr, Linie —/→ Cantitate, Linie —/→ PretUnit.

Dacă se utilizează cel de-al doilea sistem de contare, caz în care sunt valabile DF (2)-(4), atunci pentru LINII_ARTICOLE_CONTABILE din figura 4.1 dependența (4) este una parțială datorită (2).

În relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3 din figura 3.7 primele cinci dependențe funcționale:

(1) (Cota, Autor, CuvântCheie) ⟶ ISBN
(2) (Cota, Autor, CuvântCheie) ⟶ Titlu
(3) (Cota, Autor, CuvântCheie) ⟶ Editura
(4) (Cota, Autor, CuvântCheie) ⟶ LocSediuEd
(5) (Cota, Autor, CuvântCheie) ⟶ AnApariţie

sunt parțiale, deoarece:

(6) Cota ⟶ ISBN
(7) Cota ⟶ Titlu
(8) Cota ⟶ Editura
(9) Cota ⟶ LocSediuEd
(10) Cota ⟶ AnApariţie

Cu alte cuvinte, un atribut component al determinatului dependențelor (1)-(5) este, el singur, sursă într- serie de DF în care destinațiile sunt identice.

Ansamblul DF ce desemnează acoperirea minimală a relației universale VÎNZĂRI, ansamblu reprezentat în paragraful anterior sub formă de graf (figura 4.9) şi matrice (figura 4.13), nu sunt dependențele parțiale (ca din întâmplare).

4.4. Trecerea relațiilor în a doua formă normală (2NF)

Începând cu a doua formă normală, relațiile pot fi decupate în sub-relații, în scopul diminuării problemelor legate de stocare şi actualizare. În general, atunci când într-o relație R există o DF de tip X ⟶ Y şi X nu este cheie candidată, atunci R conține o anumită doză de redundanță[12].

Heath a demonstrat[13] că orice relație alcătuită din trei atribute, notată R(X, Y, Z), în care există dependența funcțională X ⟶ Y poate fi descompusă în două relații R1(X, Y) şi R2(X, Z); R1 şi R2 reprezentă proiecțiile relațiile R pe atributele X

[12] [Date00], p.333

[13] Heath, I.J. - *Unacceptable File Operations in a Relational Database*, Proc.1971 ACM SIGFIDET Workshop on Data Description, Access and Control, nov. 1971

şi Y, respectiv X şi Z. Esenţial este faptul că descompunerea să se facă fără pierdere de informaţii, adică R să poată fi "recompusă" prin joncţiunea tabelelor R1 şi R2. În alţi termeni, la spargerea oricărei relaţii trebuie ca toate dependenţele din acoperirea minimală să se regăsească în structura noilor relaţii[14]. Ca o consecinţă indirectă, se poate astfel rezolva şi problema cheii primare a relaţiei universale, în cazul încălcării restricţiei de entitate (un atribut din cheia primară prezintă valori nule).

O relaţie se află în 2NF dacă:

1. Se află în 1NF.

2. Toate DF ce leagă cheia primară la celelalte atribute sunt DF elementare (totale).

O bază de date este în 2NF când toate relaţiile care o alcătuiesc sunt în 2NF. Problema trecerii unei relaţii din prima în a doua formă normală se pune numai când cheia primară a relaţiei este compusă din mai multe atribute. Într-o formulare mai lejeră, se poate spune că *o relaţie în 2NF nu conţine DF parţiale între atributele-cheie şi celelalte atribute*.

În general, trecerea de la 1NF la 2NF se poate realiza după următoarea succesiune de paşi:

a) Se identifică posibila cheie primară a relaţiei universale, care conferă unicitate oricărui tuplu din relaţie, chiar în condiţiile violării restricţiei de entitate;

b) Se inventariază toate dependenţele dintre atributele relaţiei (închiderea DF), cu excepţia celor în care destinaţia este un atribut component al cheii primare.

c) Se trec în revistă toate dependenţele care au ca sursă (determinant) un atribut sau sub-ansamblu de atribute din cheia primară.

d) Pentru fiecare atribut (sau sub-ansamblu) al cheii de la pasul c), se creează o relaţie care va avea drept identificator primar atributul (subansamblul) respectiv, iar celelalte atribute vor fi cele care apar ca destinaţii în dependenţele de la pasul c).

e) Din relaţia iniţială sunt eliminate toate atributele destinaţie (noncheie) din relaţiile "proaspăt" obţinute, păstrându-se numai atributele cheie ale noilor relaţii (se asigură, astfel, cheile străine care vor face legăturile către noile relaţii).

f) În „rămăşiţele" relaţiei universale (iniţiale) se verifică dacă cheia primară iniţială respectă restricţia de entitate, iar dacă nu, se mai elimină dintre atribute, iar, la limită, chiar toată relaţia-„rămăşiţe".

Important ! Etapele sugerate sunt aplicabile mai ales în situaţiile în care cheia primară a relaţiei universale încalcă restricţia de entitate, adică atunci când combinaţia ar conferi unicitate, însă unul sau mai multe dintre atributele cheie ar risca valori nule (vezi discuţia din paragraful 3.4). Este vorba de violare ruşinoasă a "poruncilor" relaţionale, deoarece, după cum am promis, începând cu 2NF orice relaţie obţinută va respecta scupulos (şi) restricţia de entitate.

---

[14] Vezi şi [O'Neil & O'Neil 01], p.387

La limită, prin descompuneri, din relaţia universală (iniţială) este posibil să nu rămână decât atributele cheie. Acum, însă, orice pericol privind nulitatea vreunui atribut cheie va atrage după sine eliminarea atributului respectiv, sau, la limită a relaţiei respective, după cum vom vedea chiar în acest paragraf, pentru relaţia VÎNZĂRI.

Relaţia LINII_ARTICOLE_CONTABILE

Pentru relaţia din figura 4.1, dacă se utilizează primul sistem de contare, în care singura dependenţă dintre cele discutate în paragraful 4.1.2 este (1), atunci nu se mai pune problema trecerii în 2NF. În al doilea caz, în care DF sunt valabile (2), (3) şi (4), ultima este parţială. Urmăm cei şase paşi:

a) Cheia primară este combinaţia: (NotaContabilă, Operaţiune, ContDebitor, ContCreditor)

b) Inventarierea tuturor DF: (2), (3), (4).

c) Singura dependenţă în care determinantul un atribut din cheia primară este (2): NotaContabilă $\longrightarrow$ Data

d) Pentru atributul NotaContabilă se constituie relaţie care îl va avea identificator primar, iar cel de-al doilea atribut Data. Din tabela iniţială se elimină atributul necheie preluat în noua relaţie.

e) Din relaţia iniţială rămâne: R' {NotaContabilă, Operaţiune, ContDebitor, ContCreditor, Suma}'

f) În R' cheia primară nu periclitează restricţia entităţii, deci R' îşi poate păstra structura.

Prin urmare, în 2NF relaţia LINII_ARTICOLE_CONTABILE se descompune în două relaţii, NOTE_CTB şi ARTICOLE_CTB prezentate în figura 4.14.

NOTE_CTB {NotaContabilă, Data}

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

NOTE_CTB

Nota_Contabilă	Data
150	30.11.2000
151	01.12.2000

ARTICOLE_CTB

Nota_Contabilă	Operaţiune	Cont_Debitor	Cont_Creditor	Suma
150	1	300	401	1 000 000
150	1	4426	401	180 000
150	2	401	5311	700 000
150	2	401	5121	480 000
151	1	5121	700	525 000

Figura 4.14. Relaţia LINII_ARTICOLE_CONTABILE adusă în 2NF

La drept vorbind, nu putem spune că economia de spațiu şi diminuarea anomaliilor la actualizări sunt spectaculoase din cale-afară prin aducerea relației în 2NF. Din rațiuni de simplitate, exemplul este însă util.

Relația BIBLIOTECĂ_TUPLURI_NOI_SOLUȚIA_3

Relația ilustrată în figura 3.7 prezintă, dintre cele discutate în paragraful 3.3.2, dependențe parțiale şi, astfel, constituie un bun "obiect" de normalizare în 2NF. Dealtminteri, relația abundă în redundanțe şi anomalii de genul celor discutate în paragraful 2.1:

- înregistrarea recepției unui titlu, introducerea datelor despre o carte (titlu) sunt imposibile până în momentul în care măcar o carte primeşte cotă;
- dacă un titlu a fost introdus inițial eronat, corecția trebuie operată pe destule linii ale tabelei;
- la fiecare achiziție ulterioară a unei aceleaşi cărți (`ISBN`) trebuie introduse din nou ISBN-ul, titlul, editura etc.

Parcurgem cei şase paşi, după cum urmează:

a) Cheia primară a relație inițiale este: (`Cota`, `Autor`, `CuvântCheie`);

b) Ansamblul DF: (1)-(15) - vezi paragraful 4.1.2;

c) În dependențele (6)-(10), atributul `Cota`, component al cheii relației, este sursă simplă.

d) Se constituie o relație în care `Cota` este cheie primară, iar celelalte atribute sunt destinațiile dependențelor (6), (7), (8), (9) şi (10).

e) Din relația inițială se elimină toate atributele-destinație preluate în proaspătă tabelă, rămânând doar atributele cheie: {`Cota`, `Autor`, `CuvântCheie`}.

f) Nici unul dintre cele trei atribute de la punctul e) nu periclitează restricția de entitate, aşa încât relația-„rămăşiță" se păstrează în această formă.

Prin urmare, în a doua formă normalizată, relația inițială (universala) s-ar putea descompune în:

CĂRȚI {`Cota`, `ISBN`, `Titlu`, `Editura`, `LocSediuEd`, `AnApariţie`}
şi
TITLURI_AUTORI_CUVINTECHEIE {`Cota`, `Autor`, `CuvântCheie` }

Conținutul acestor două noi relații este prezentat în figura 4.15.

CĂRȚI

Cotă	ISBN	Titlu	Editura	LocSe-diuEd	AnApa-riţie
III-13421	973-683-889-7	Visual FoxPro. Ghidul dezvoltării aplicațiilor profesionale	Polirom	Iaşi	2002
III-13422	973-683-889-7	Visual FoxPro. Ghidul dezvoltării aplicațiilor profesionale	Polirom	Iaşi	2002
III-13423	973-683-889-7	Visual FoxPro. Ghidul dezvoltării aplicațiilor profesionale	Polirom	Iaşi	2002
III-10678	973-683-709-2	SQL. Dialecte DB2, Oracle şi	Polirom	Iaşi	2001

		Visual FoxPro			
III-10679	973-683-709-2	SQL. Dialecte DB2, Oracle şi Visual FoxPro	Polirom	Iaşi	2001

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 4.15. Relaţia BIBLIOTECĂ_TUPLURI_NOI_SOLUŢIA_3 în 2NF

Chiar dacă o parte din redundanţe au fost eliminate, mai rămân destule, astfel încât structura de mai sus va constitui subiect atât pentru a trei formă normalizată (3NF), cât şi pentru cea de a patra (4NF).

Relaţia DOTARE_JUCĂRII_1

Nici tabela din figura 3.14 nu se află în 2NF, deoarece cheia primară este compusă - (`CodFamilie`, `Copil`, `Jucărie`), aşa că:

(1) (`CodFamilie`, `Copil`, `Jucărie`) $\longrightarrow$ `NumeFamilie`

(2) (`CodFamilie`, `Copil`, `Jucărie`) $\longrightarrow$ `DataCumpărării`

iar, după cum am discutat în paragraful 4.1.2,

(3) `CodFamilie` $\longrightarrow$ `NumeFamilie`

Prin urmare, DF (1) este parţială, iar relaţia se rupe în două tabele, după cum sugerează figura 4.16.

FAMILII

CodFamilie	NumeFamilie
1111	Popescu Ioan

JUCĂRII

CodFamilie	Copil	Jucărie	DataCumpărării
1111	Marc	Trenuleţ electric	14-05-1998
1111	Loredana	Păpuşă Barbie	15-09-1998
1111	Elvis	Puzzle 200 piese	23-12-1998
1111	Loredana	Puzzle 250 piese	14-05-1999
1111	Marc	Puzzle 200 piese	24-12-1999
1111	Loredana	Căsuţa Lego	08-03-2000

Figura 4.16. Relaţia DOTARE_JUCĂRII_1 adusă în 2NF

Relaţia LOCALITĂŢI_CODURI_VECHI

Pentru vechiul sistem de codificare poştală, relaţia LOCALITĂŢI_CODURI_VECHI din figura 3.15, cu cele câteva dependenţe discutate în acelaşi paragraf 4.1.2, nu are cheie primară, deoarece singura combinaţie plauzibilă –

(CodPoştal, Sat) – încalcă restricţia de entitate, întrucât pentru liniile care se referă la oraşe valoarea atributului Sat este NULL. Cu toate acestea:

(1) (CodPoştal, Sat) ⟶ OraşComună

(2) (CodPoştal, Sat) ⟶ Judeţ

Şi întrucât:

(3) CodPoştal ⟶ OraşComună

(4) CodPoştal ⟶ Judeţ

putem spune că (1) şi (2) sunt DF parţiale.

Ţinând seama de acest lucru, relaţia este ruptă astfel:

ORAŞE_COMUNE {CodPoştal, OraşComună, Judeţ}

şi

SATE {CodPoştal, Sat}

al căror conţinut este prezentat în figura 4.17.

ORAŞE_COMUNE

CodPoştal	OraşComună	Judeţ
5319	Vînători	Vrancea
5613	Roznov	Neamţ
5300	Focşani	Vrancea

SATE

CodPoşta	Sat
5319	Vînători
5319	Jorăşti
5319	Mirceştii-Vechi
5613	Roznov
5613	Slobozia

Figura 4.17. Relaţia LOCALITĂŢI_CODURI_VECHI adusă în 2NF cu rezolvarea problemei cheii primare

Pentru a verifica justeţea ruperii relaţii iniţiale în cele două noi tabele, ar trebui ca prin joncţiunea ORASE_COMUNE cu SATE să obţinem LOCALITĂŢI_CODURI_VECHI. Şi, într-adevăr prin fraza:

```
SELECT oc.codpostal, orascomuna, sat, judet
FROM orase_comune oc LEFT OUTER JOIN sate s
    ON oc.codpostal=s.codpostal
```

rezultatul are acelaşi conţinut informaţional ca şi relaţia iniţială. Atenţie, însă, folosim *joncţiunea externă*, şi nu cea naturală (sau echijoncţiunea) !

Avantajele descompunerii în cele două relaţii sunt vizibile. Mai întâi, ambele sunt în 1NF şi au cheie primară fără nici un risc în privinţa valorilor nule. În al doilea rând, se elimină redundanţele. În LOCALITĂŢI_CODURI_VECHI, dacă o comună are şase sate, de şase ori apare şi codul poştal al comunei, şi denumirea comunei şi a judeţului. Acum însă, denumirea comunei şi judeţului apare o singură dată în relaţia ORAŞE_COMUNE. De asemenea, pentru fiecare oraş apare o singură înregistrare în ORAŞE_COMUNE şi, firesc, nici una în SATE. La aşa normalizare reuşită, păcat că vechea codificare poştală nu mai este în vigoare !

Relaţia CODURI_ORAŞE

Pentru noul sistem de codificare poştală - relaţia CODURI_ORAŞE (figura 3.19 - paragraful 3.4) nu este în 2NF, deoarece un cod poştal nu poate fi alocat la două oraşe (localităţi):

(1) CodPoştal ⟶ Localitate
(2) CodPoştal ⟶ Judeţ

şi, pe de altă parte,

(3) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ Localitate
(4) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ Judeţ

aşa că putem rupe relaţia în:

CODURI_OR {CodPoştal, Localitate, Judeţ}
şi
CODURI_ADRESE {CodPoştal, Strada, NrIniţial, LitIniţială, TipNr, NrFinal, LitFinală}

Relaţia CODURI_NOI_V3

Lucrurile se prezintă foarte aproape de relaţia CODURI_ORAŞE discutată mai sus, aşa că putem scrie:

(1) CodPoştal ⟶ Localitate
(2) CodPoştal ⟶ Comuna
(3) CodPoştal ⟶ Judeţ

şi, pe de altă parte,

(6) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ Localitate
(7) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ Comuna
(8) (CodPoştal, Strada, NrIniţial, LitIniţială) ⟶ Judeţ.

Rezultă că putem rupe relaţia în:

CODURI_LOCALITĂŢI {CodPoştal, Localitate, Comună, Judeţ}
şi
CODURI_ADRESE {CodPoştal, Strada, NrIniţial, LitIniţială, TipNr, NrFinal, LitFinală}

Descompunerea este salutară, deoarece rezolvă şi problema restricţiei de entitate; cea de-a doua relaţie va conţine numai codurile acordate la nivel de străzi, deci nici un atribut cheie nu va avea valori nule.

Relaţia STUDENŢI_EXAMENE

Senzaţia de deja-vu care vă încearcă este îndreptăţită. Acesta este relaţia din paragraful 2.2 (figura 2.3) cu care am început să închinăm osanale normalizării. Este o relaţie simplă care se pretează la normalizare, manifestând redundanţele şi anomaliile la actualizare expuse pe scurt în paragrafele 2.2.1, 2.2.2, 2.2.3 şi 2.2.4.

Cheia primară fiind combinaţia (Matricol, CodDisc, DataExamen), automat sunt valabile următoatele DF:

(1) (Matricol, CodDisc, DataExamen) ⟶ NumePrenume

(2) (Matricol, CodDisc, DataExamen) ⟶ An
(3) (Matricol, CodDisc, DataExamen) ⟶ Specializare
(4) (Matricol, CodDisc, DataExamen) ⟶ DenumireDisc
(5) (Matricol, CodDisc, DataExamen) ⟶ NrCredite
(6) (Matricol, CodDisc, DataExamen) ⟶ Nota

STUDENȚI_EXAMENE ar fi în 2NF dacă nici una din aceste şase dependențe nu ar fi parțială. Fără a ne întrebuința prea serios, e suficient să ne reamintim că matricolul este desemnat să identifice în mod unic fiecare student. Dintr-un foc avem trei dependențe:

(7) Matricol ⟶ NumePrenume
(8) Matricol ⟶ An
(9) Matricol ⟶ Specializare

Şi codul disciplinei are pretenții de identificare fără ambiguitare a fiecărei "materii" parcurse de studenți:

(10) CodDisc ⟶ DenumireDisc
(11) CodDisc ⟶ NrCredite

Din dependențele (7)-(9) construim relația:
STUDENȚI { Matricol, NumePrenume, An, Specializare}

din (10) şi (11):
DISCIPLINE { CodDisc, DenumireDisc, NrCredite}

iar din relația inițială rămâne:
EXAMENE { Matricol, CodDisc, DataExamen, Nota}
care nu are probleme cu valorile nule ale vreunui atribut-cheie.

Avem toate motivele să fim mulțumiți. Dacă ne raportăm la paragraful 2.2.1, observăm că redundanțele au fost eliminate. Pe de parte, numele, specializarea şi anul de studiu al fiecărui student apar acum o singură dată (în tuplul corespunzător studentului din relația STUDENȚI). Pe de altă parte, tot o singură dată apar şi denumirea unei discipline şi numărul de credite alocat acesteia (relația DISCIPLINE).

Lucrurile stau la fel de bine şi în ceea ce priveşte diminuarea anomaliilor manifestate la editarea datelor. Astfel, privitor la anomaliile de inserare (paragraful 2.2.2), din momentul înscrierii în anul I, până la cel al primului examen, studentul poate fi inserat în tabela STUDENȚI, urmând ca, odată cu prima sa sesiune de examene, să-i fie introduse înregistrări în EXAMENE. Modificarea denumirii unei discipline sau numărul de credite, sau corectarea unei eventuale erori strecurate în numele studentului sau al specializării/anului, toate se fac într-un singur loc, pe tuplul corespunzător din STUDENȚI sau DISCIPLINE, ceea ce înseamnă că am scăpat şi de anomaliile din paragraful 2.2.3. În fine, paragraful 2.2.4 atrăgea atenția asupra riscului pierderii unor informații valoroase la ştergerea unei linii din STU-

DENȚI_EXAMENE (un soi de aruncatul copilului odată cu albia). Nici acest pericol nu mai este de actualitate în noua structură.

Relația VÎNZĂRI

Şi relația universală VÎNZĂRI încalcă preceptele relaționalului (în fapt, am putea să-i spunem *tabelă*, pentru a respecta adevărul, tabela fiind asociată cu SQL-ul care este mult prea tolerant cu elemente ce fac dintr-o tabelă să nu fie o relație - unicitate, valori nule, lipsa cheii primare etc.), întrucât unul din atributele care asigură unicitate fiecărui tuplu din relația inițială, `CodÎnc` (codul încasării) are valori nule pentru facturile "proaspăt" emise, de fapt, pentru toate facturile din care nu s-a încasat nici o leţcaie. Dacă nu s-ar fi pus problema restricției de entitate, combinația de atribute care diferențiază cu siguranță orice tuplu de celelalte este (`NrFact, Linie, CodÎnc`). Lucrurile nu sunt atât de grave, deoarece în setul de DF (vezi paragraful 4.1.2) apar destule surse care sunt subansambluri ale pseudo-cheii primare. Aşadar, să parcurgem cele şase etape ale deja celebrului „îndrumar" de aducere a unei baze de date în 2FN.

a) Cheia primară a relației este, după cum am afirmat nu mai sus de câteva rânduri: (`NrFact, Linie, CodÎnc`).

b) Dependențele manifestate în relație sunt cele 54 din paragraful 4.1.2.

c) Dependențele (50), (51), (52) şi (53) au drept sursă `CodÎnc`. Cu totul altfel stau lucrurile pentru `NrFact`. Dacă în paragraful 4.1.2 identificasem dependențele (37), (38), (39) şi (40), am văzut în 4.1.3, când am discutat depre tranzitivitate, că acestora li se mai pot adăuga încă 12, adică (55)-(66). Am încheiat inventarierea DF în care sursele sunt simple şi atribute-cheie şi continuăm cu surse compuse, alcătuite din două dintre cele trei atribute, la acest „capitol" având o singură dependențele (54) şi cele de la (41) la (46).

d) Creăm noile relații.

Din (50), (51), (52) şi (53) rezultă:

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

Din (37), (38), (39) şi (40), plus (55)-(66) se obține:

FACTURI {`NrFact`, `CodCl`, `DataFact`, `Obs`, `DenCl`, `CodFiscal`, `StradaCl`, `NrStradaCl`, `BlocScApCl`, `TelefonCl`, `CodPost`, `Localitate`, `Comuna`, `Jud`, `Judeţ`, `Regiune`}

Din dependența (54) se poate "izola" relația:

ÎNCAS_FACT {`CodÎnc`, `NrFact`, `Tranşa`}

Pe baza DF (41)-(46) se construieşte:

LINII_FACTURI {`NrFact`, `Linie`, `Cantitate`, `PreţUnit`, `CodPr`, `DenPr`, `UM`, `ProcTVA`, `Grupa`}

e) Cu totul remarcabil este ceea ce mai rămâne din relaţia iniţială: FACTURI_INCASĂRI {NrFact, Linie, CodÎnc}.

f) În afara celor trei atribute cheie din relaţia iniţială - nimic. Dacă în 1NF eludam problema restricţiei de entitate, pe motiv că în 2NF lucrurile se rezolvă, acum ne propunem să tranşăm discuţia. Pentru ca să respecte restricţia de entitate, relaţia FACTURI_ÎNCASĂRI nu poate "primi" tupluri, decât în momentul unei prime încasări a unei facturi. Dar, de fapt, ceea ce se încasează într-o tranşă nu este o linie (un produs) dintr-o factură, ci *o parte* sau *întreaga* factură, lucrul pe care-l reflectă pe deplin dependenţa (54) şi relaţia ÎNCAS_FACT. Prin urmare, această ultimă relaţie nu ne este de nici un folos, şi, pur şi simplu, renunţăm la ea. Senzaţia este oarecum ciudată, mai ales că un asemenea demers nu este fundamentat prea riguros în teoria atât de matematizată a normalizării.

În concluzie, în 2NF, relaţia universală VÎNZĂRI va avea structura relaţiilor: ÎNCASĂRI, FACTURI, ÎNCAS_FACT şi LINII_FACTURI.

Forme normalizate optimale

Dacă avem în vedere teorema lui Heath, ruperea unei relaţii în mai multe tabele poate fi făcută cu oarecare frenezie. Astfel, în LOCALITĂŢI_CODURI_VECHI, pe baza DF: CodPoştal ⟶ OraşComună şi CodPoştal ⟶ Judeţ, baza de date poate fi adusă în 2NF şi astfel:

CODPOST_ORCOM {CodPoştal, OraşComună }

CODPOST_JUD {CodPoştal, Judeţ}
şi
SATE {CodPoştal, Sat}

Este clar că avem de-a face cu o exagerare, deşi toate cele trei relaţii respectă cu sfinţenie condiţiile formei a doua normale. De cele mai multe ori, în normalizare se aplică principiul obţinerii celui mai mic număr de relaţii care să respecte condiţiile uneia sau alteia dintre formele normalizate.