Three of Normalization Glitches

Prof.dr. Marin FOTACHE
Catedra de Informatică Economică, Universitatea „Alexandru Ioan Cuza", Iaşi

Among the database designers there is a strong feeling that normalization haven't fulfilled one of it's main objectives - being an appropriate methodology for guiding any database designer in achieving a good schema, according to the application requirements. One of the reasons could be too much formalism. But this paper is focused on a different area, trying to propose solutions for three types of problems which occurs quite often in practice: the apparent transitivity of some FDs, canonical FDs, and some semantic relationships between attributes which cannot be expressed as FDs.
Keywords: *normalization, functional dependencies (FD), apparently transitive FD, 3rd normal form, Boyce-Codd normal form, non-canonic FD.*

Introducere

Toți autorii importanți atrag atenția asupra faptului că, deşi riguros formalizată, normalizarea nu este un panaceu. Există numeroase reguli semantice neincorporabile în normalizare. Pe de altă parte, unele cazuri reclamă o examinare mai atentă a DF sau chiar o anumită doză de artificialitate. Prezenta lucrare îşi propune să identifice şi să propună soluții trei tipuri situaționale ce pot afecta calitatea unei scheme de baze de date obținută în urma normalizării: DF aparent tranzitive, DF necanonice şi relații semantice dintre atribute ce nu pot fi modelate sub forma DF.

Dependențe funcționale aparent tranzitive

Problema a fost studiată tangențial şi nu prea zgomotos. Dintre cele câteva lucrări în care este analizată, merită amintită ce a lui Paolo Atzeni şi Stott Parker (Atzeni & Parker 82). Să luăm un exemplu foarte simplu: R {Profesor, Birou, Catedră, Telefon} care conține date despre "domiciliul" în facultate al fiecărui profesor. Un profesor are un singur birou (Birou identifică fără ambiguitate sala respectivă): (1) Profesor $\longrightarrow$ Birou. Un profesor face parte dintr-o singură catedră:(2) Profesor $\longrightarrow$ Catedră. Într-un birou pot "locui" mai mulți profesori: Birou $\not\longrightarrow$ Profesor. Fiecare birou este arondat unei singure catedre (Economie politică, Contabilitate etc.): (3) Birou $\longrightarrow$ Catedră. Fiecare birou are un număr de telefon unic:(4) Birou $\longrightarrow$ Telefon.

Din (3) ar reieşi că (2) este DF tranzitivă, iar relația ar trebui descompusă numai în R1 {Birou, Catedră, Telefon} şi R2 {Profesor, Birou}. Apare însă o situație ce aruncă în aer algoritmul: 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 şi 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 şi catedrele la care sunt afiliați profesorii: R {Profesor, CatedrăProfesor, Birou, CatedrăBirou, Telefon}:

Acum dependențele ar fi:

(1) Profesor $\longrightarrow$ Birou
(2) Profesor $\longrightarrow$ CatedrăProfesor
(3) Birou $\longrightarrow$ CatedrăBirou
(4) Birou $\longrightarrow$ Telefon

Noua schemă ar fi: R1 {Birou, CatedrăBirou, Telefon} şi R2 {Profesor, CatedrăProfesor, Birou}.

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

Dependențe tranzitive necanonice

Nici a o doua problemă pe care o vom ridica în continuare nu a beneficiat de prea multă atenție din partea celebrităților normalizării, deşi implicațiile sunt serioase. Mai toate lucrările recomandă folosirea exclusivă a depedențelor funcționale în formă canonică – cele în care destinația este alcătuită dintr-un singur atribut (de ex. [Saleh94], [Garcia-Molina s.a 02]). Lucrurile nu sunt grozav de complicate, identificarea depedențelor 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 ⟶ B, A ⟶ C, A ⟶ D, A ⟶ E, dependențe ce derivă din rolul de cheie primară pe care îl îndeplineşte atributul A, să se verifice dacă nu cumva există fie o dependență funcțională simplă, de genul B ⟶ D, fie o dependență funcțională cu sursa compusă, de genul (B, C) ⟶ D. Dacă una din cele două afirmații de mai sus este adevărată, atunci dependența funcțională A ⟶ E nu este directă. De fapt, că problema poate fi pusă în legătură uneori cu forma normală Boyce-Codd.

Luăm în discuție tabela COMENZI, prezentată în figura 1, tabelă care gestionează comenzile pe care o firmă le-a trimis furnizorilor (în care se solicită materialele/produsele/serviciile de care are nevoie). Pe baza unei comenzi, un furnizor trimite produsele solicitate, întocmind o factură din care 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

Fig.1. 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 furnizar poate onora una, două sau mai multe comenzi.

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

(1) NrComanda ⟶ NrFactură
(2) NrComanda ⟶ CodFurnizor
(3) NrComanda ⟶ DataFactură
(4) NrComanda ⟶ Valoare
(5) NrComanda ⟶ TVADeduct

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

(6) (NrFactură, CodFurnizor) ⟶ DataFactură
(7) (NrFactură, CodFurnizor) ⟶ Valoare
(8) (NrFactură, CodFurnizor) ⟶ TVADeduct

Întrucât o factură primită poate onora mai multe comenzi:

(NrFactură, CodFurnizor) —/→ NrComanda

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

NrComanda ⟶ (NrFactură, CodFurnizor) ⟶ DataFactură
NrComanda ⟶ (NrFactură, CodFurnizor) ⟶ Valoare
NrComanda ⟶ (NrFactură, CodFurnizor) ⟶ TVADeduct

după cum se observă în graful DF din figura 2.

Economia de spațiu şi diminuarea redundanței sunt mai degrabă simbolice, ba chiar lucrurile se complică prin introducerea unei restricții referențiale şi, normal, prin apariția de două ori a atributelor de legătură NrFactură şi CodFurnizor. Există, totuşi, un merit indiscu-

tabil al noilor relaţii. În relaţia COMENZI nu se poate prelua nici o factură fără a-i cunoaşte comanda (eterna problemă a restricţiei de entitate – nici un atribut component al cheii nu poate avea nule). Practica economico-financiară de la noi, dar şi din alte părţi, cunoaşte suficiente cazuri în care firmele primesc facturi direct, fără a înainta o comandă prealabilă. Utilizând proaspetele tabele, FACTURI_PRIMITE şi COMENZI_FACTURI, o factură fără comandă va apărea numai în prima din cele două relaţii. Deşi, la prima vedere, am adăugat, şi nu eliminat (aşa cum, generos, sună unul dintre obiectivele normalizării) redundanţă, noua structură este mai "sănătoasă" din punct de vedere relaţional.

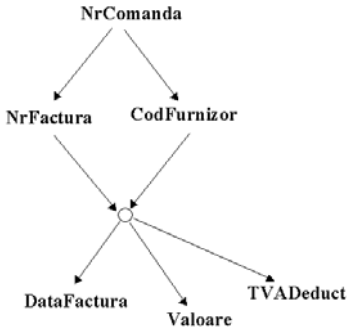

Fig.2. DF tranzitive necanonice

Pe baza tranzitivităţii, tabela COMENZI se poate sparge în două relaţii FACTURI_PRIMITE şi COMEZI_FACTURI, ca în figura 3.

FACTURI_PRIMITE

NrFactura	CodFurnizor	DataFactura	Valoare	TVADeduct
12344	12	20.06.2004	119000000	19000000
11987	5	15.05.2004	67830000	10830000
45963	2	14.05.2004	71400000	11400000
12456	12	02.07.2004	67592000	10792000
12344	5	27.06.2004	106691830	17034830
47890	2	29.05.2004	93886835	14990335

COMENZI_FACTURI

NrComanda	NrFactura	CodFurnizor
1501	12344	12
1502	11987	5
1503	45963	2
1504	12456	12
1505	12344	5
1505	12344	12
1507	47890	2

Fig.3. Eliminarea DF tranzitive din tabela COMENZI

Practic, avantajul celor două relaţii din figura 3 este cu atât mai evident cu cât sunt numărul facturilor ce onorează două sau mai multe comenzi este mai mare.

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

- stabilirea cheii primare a relaţiei iniţiale (universale), apoi
- eliminarea, dintre depedenţele funcţionale ce decurg din calitatea de cheie primară a relaţiei iniţiale, a celor parţiale, apoi
- eliminarea, din toate relaţiile obţinute în pasul anterior, a tuturor dependenţelor tranzitive (în care surse sunt cheile primare ale respectivelor relaţii),

trebuie verificat dacă în relaţia iniţială mai existau depedenţe cu sursa compusă ce nu apar drept cheie primară sau candidat în nici una dintre relaţiile finale obţinute.

Relaţii semantice imposibil de preluat în dependenţe fucţionale

Un alt aspect delicat al normalizării ţine de dificultatea preluării în schemă, sub formă de depedenţe funcţionale, a unor relaţii semantice. Pentru o primă ilustrare să luăm relaţia PROFI_DISCIPLINE din figura 4 ce conţine date legate de profesori şi cursurile predate într-o facultate.

PROFI_DISCIPLINE

IdProf	NumeProf	Catedră	Cod Disc	DenDisc	LbStrProf	LbStrDisc
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 Marin	Informatică Economică	AI1101	Introducere în informatica economică	FR	FR

Fig.4. Cursuri predate în limbi străine

Un profesor este identificat prin IdProf şi face parte dintr-o singură catedră:
(1) IdProf ⟶ NumeProf
(2) IdProf ⟶ Catedră.
Un curs este identificat prin cod şi arondat unei singure catedre:
(3) CodDisc ⟶ DenDisc
(4) CodDisc ⟶ Catedră.
În schimb:
- un profesor predă mai multe discipline: IdProf —/→ CodDisc
- o disciplină poate fi predată de mai mulţi profesori: CodDisc —/→ IdProf
- un profesor poate preda în mai multe limbi străine: IdProf —/→ LbStrProf
- un curs poate fi predat în mai multe limbi străine: CodDisc —/→ LbStrDisc
- un profesor poate preda acelaşi curs în mai multe limbi străine: (IdProf, CodDisc) —/→ LbStrDisc
- un profesor poate preda în aceaşi limbă străină două sau mai multe cursuri:
(IdProf, LbStrDisc) —/→ CodDisc şi
(IdProf, LbStrProf) —/→ CodDisc
Practic, graful DF se prezintă ca în figura 5.

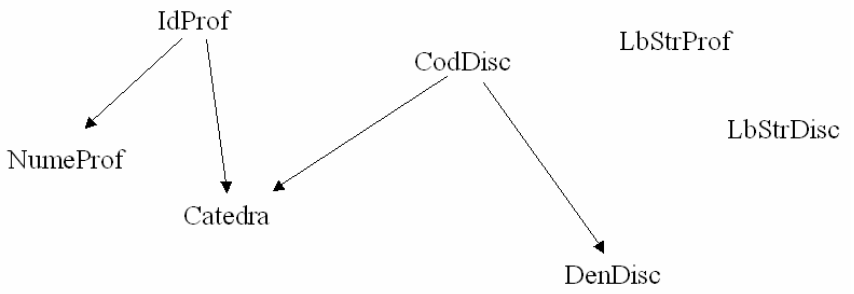

Fig.5. Graful DF pentru relaţia PROFI_DISCIPLINE

Nemulţumirea legată de informaţia reprezentată prin graf ţine de faptul că un profesor ştie 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 informaţiile: care sunt limbile străine cunoscute de fiecare profesor şi care sunt limbile străine în care se predă fiecare disciplină.
Atributele LbStrProf şi LbStrDisc rămân suspendate în graf. Ce-i de făcut ? O idee, nu întotdeauna rezonabilă, este a le plasa într-o relaţie împreună cu celelate "rădăcini" ale grafului, obţinându-se, astfel, schema:
PROFI {IdProf, NumeProf, Catedra}, DISCIPLINE {CodDisc, DenDisc, Catedra} şi
PROFI_DISC_LBSTR { IdProf, CodDisc, LbStrProf, LbStrDisc}
Predicatul care ar exprima oricare tuplu al ultimei relaţii ar fi: un profesor (IdProf) care cunoaşte o limbă străină (LbStrProf) predă o disciplină (CodDisc) într-una din limbile străine pe care le vorbeşte (LbStrDisc). Firește, apare întrebarea: Ce relaţie trebuie să existe între valorile LbStrProf şi LbStrDisc într-un tuplu, ştiind că disciplina este predată şi în alte limbi decât LbStrProf, iar profesorul ştie şi alte limbi străine decât LbStrDisc ?
O soluţie acceptabilă în multe cazuri constă în introducerea unor atribute suplimentare, tocmai pentru a "fixa" sub formă de depedenţe restricţiile respective. Spre exemplu, pentru a "formaliza" restricţia un profesor ştie anumite anumite limbi străine, se poate introduce un atribut, Nivel, pentru a ilustra gradul de comunicare al profesorului în limba străină respectivă: (IdProf, LbStrProf) ⟶ 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 şi case în responsabilitatea unui singur profesor:
(CodDisc, SerieCurs) ⟶ LbStrDisc,
(CodDisc, SerieCurs) ⟶ IdProf.
Graful obţinut nu mai conţine atribute "suspendate" - vezi figura 6 - iar schema desprinsă din acesta este mult mai bună:

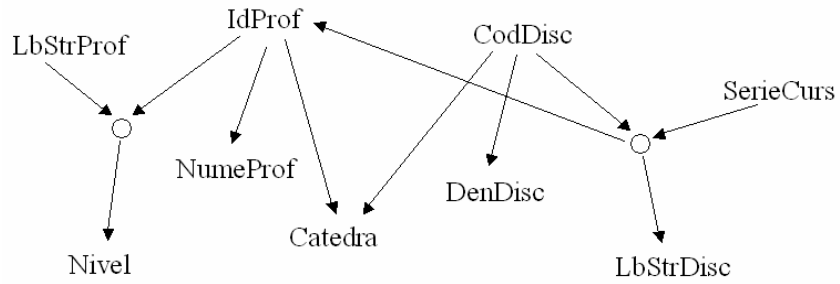

Fig.6. Noul graful DF pentru relația PROFI_DISCIPLINE ameliorată

PROFI {IdProf, NumeProf, Catedra},
DISCIPLINE {CodDisc, DenDisc, Catedra},
PROFI_LBSTR {IdProf, LbStrProf, Nivel}
şi
SERII_CURS_LBSTR { CodDisc, SerieCurs, LbStrDisc, IdProf}.

Deşi oarecum artificială, tehnica introducerii de atribute suplimentare, care nu sunt evidente în momentul construirii relației universale, este una salutară de multe ori, 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 paşi până se obține o schemă considerată mulțumitoare.

Concluzii

Prezenta lucrare încearcă să concilieze, pe calapodul a trei categorii situaționale, teoria şi practica normalizării bazelor de date relaționale. Deşi fără o încărcătură metodologică copleşitoare, cele trei probleme expuse sunt mai mult decât episodice pentru proiectanții de baze de date. Departe de a cădea în păcatul multor lucrări din domeniu, ce rezolvă problemele printr-o formulă matematică sau o teoremă şi două leme, soluțiile prezentate sunt mai degrabă comune, simple, dar şi aplicabile.

Bibliografie

[Atzeni & Parker 82] Atzeni, P., Parker, S. - *Assumptions in relational Database Theory*, Proc. of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems, Los Angeles, 1982

[Date04] Date, C.J. – *An Introduction to Database Systems*, 8th edition, Pearson Addison-Wesley, Boston, 2004

[Elmasri & Navathe 00] Elmasri, R., Navathe, S.R. - *Fundamentals of Database Systems*, Addison-Wesley, Reading, Massachussetts, 2000

[Fotache01-1] Fotache, M. – *Câteva probleme ale formelor normalizate 2 şi 3*, Net Report nr.104, mai, 2001

[Fotache01-2] Fotache, M. – *Cui îi mai este frică de formele normalizate Boyce-Codd, 4 şi 5 ?*, Net Report nr.105, iunie, 2001

[Fotache04] Fotache, M. – *Do practitioners need database normalization ?*, Proc. of the Central and East European Conference in Business Information Systems, Ed. Risoprint, Cluj-Napoca, 2004

[Garcia-Molina s.a. 02] Garcia-Molina, H., Ullman, J., Widom, J. - *Database Systems. The complete Book*, Prentice Hall, Upper Saddle Riner, New Jersey, 2002

[Saleh94] Saleh, I. – *Les bases de données relationneles. Conception et réalisation*, Hermes, Paris, 1994