Modelul relațional al datelor

Christian Mancaş

Ovidius University Press, Constanța
2004

CUPRINS

1 Introducere

Modelul relațional al datelor (*MRD*), primul model matematic în baze de date (*bd*), atât în ordine cronologică, cât şi a importanței, constituie piatra unghiulară în domeniu din cel puțin următoarele patru puncte de vedere:

1. Majoritatea uneltelor de bd existente pe piață se bazează pe acest model de peste două decenii, fie că este vorba de cele elementare, pentru PC-uri (e.g. *MS SQL Server*, *Access*, *FoxPro*, *Delphi*, *Paradox*, *dBase* etc.), fie că este vorba de cele pentru mini şi mainframes (e.g. *IBM DB/2*, *Oracle*, *Sybase SQL Server*, *Tandem NonStop SQL*, *Postgress* etc.) şi, în viitorul previzibil, nici un alt model nu pare capabil a i se substitui în această privință.
2. Orice model al datelor de nivel mai înalt trebuie să aibă în componență un algoritm de traducere a bd în model relațional, deoarece cea mai naturală modalitate de memorare a datelor discrete o constituie tabelele.
3. Majoritatea limbajelor de interogare, definire şi manipulare a datelor existente se bazează pe cele relaționale (în special pe *SQL* şi *QBE*).
4. Teoria matematică a proiectării şi interogării bd relaționale (*bdr*) este încă deschisă, de actualitate şi fundamentează (sau măcar inspiră) majoritatea demersurilor similare în alte modele ale datelor şi/sau cunoştințelor.

Provocarea de a scrie o monografie asupra *MRD* este deci aproape obligatorie pentru orice specialist în domeniu, chiar dacă literatura este deja foarte bogată, cel puțin pentru a prezenta punctul de vedere propriu actual (şi în limba maternă): care sunt subiectele şi demonstrațiile cele mai importante, care sunt principalele limite ale *MRD*, în general deci ce şi cât trebuie aprofundat în domeniu? Pe lângă răspunsurile proprii la aceste întrebări, prezenta monografie încearcă să furnizeze şi o colecție bogată de exemple concrete şi corecte, care să sprijine intuiția cititorului şi să-i servească drept „bibliotecă” de soluții primare în domeniu.

Prezentarea începe cu aspectele "statice", structurale, interesând doar proiectarea *schemelor de bd* (în primele 6 capitole); apoi (în ultimul capitol) sunt tratate şi *interogările*, ce țin de "dinamica" bd; dată fiind însă strânsa interdependență între *algebra relațională* şi *descompunerea fără pierderi* sau *dependențele join*, secțiunea 1.5 conține deja o introducere în *limbajul algebric relațional de interogare*.

După un scurt istoric (în secțiunea 1.1), introducerea *relațiilor* în secțiunea 1.2 (inclusiv motivarea diferențelor în raport cu noțiunea matematică omonimă) şi a unei prime formalizări a *MRD* în secțiunea 1.3, urmează, în secțiunea 1.4, o scurtă trecere în revistă a *primei forme normale*.

Capitolul 2 stăruie asupra *constrângerilor de integritate*, analizând *cheile* (inclusiv cele *primare* şi *străine*, dar, în special, *dependențele cheie*), *valorile nule*, *constrângerile tuplu*, cele *implicate*, cele *triviale*, ca şi *dependențele de incluziune tipate* şi *funcțional dependențele*.

Sunt apoi prezentate în capitolul 3 *anomaliile de actualizare a datelor* care, împreună cu cheile, impun şi justifică necesitatea *formei normale Boyce-Codd*.

Capitolul 4 tratează *teoria dependențelor elementare*; sunt abordate *problema implicației*, *regulile de derivare*, *tablourile* şi *algoritmii de vânare*, ca şi *descompunerile fără pierderi ale funcțional dependențelor*, precum şi *problema implicației pentru dependențele de incluziune* şi *decidabilitatea implicației finite şi a celei nerestricționate*.

Capitolul 5 este dedicat *dependențelor join* şi *multivaluate*, respectiv *formelor normale* (*3,3*), *4*, *5* (*proiecție-join*) şi *domenii-chei*.

Lucrarea este folosită şi drept suport pentru mai multe cursuri universitare de specialitate. De exemplu, primele trei capitole sunt prezentate în toate cursurile; capitolul 7 şi secţiunile 5.4, 6.1, 6.2 şi 6.5 sunt adăugate cursurilor de un an ale studenţilor din anii IV, iar capitolul 4 şi secţiunile 5.1, 5.2, 5.3, 6.3 şi 6.4 sunt studiate la masterat.

1.1 Istoric şi referinţe bibliografice

Modelul relaţional al datelor a fost introdus în 1970 de matematicianul englez E.F.Codd, cercetător IBM [106], având drept scop declarat asigurarea independenţei datelor (de nivelul fizic, al oricăror implementări posibile) şi depăşirea restricţiilor impuse de modelele datelor existente până atunci. Primele modele ale datelor în ordine cronologică, cel ierarhic şi apoi cel reţea, includ referinţe explicite la caracteristici de implementare la nivel fizic, precum pointerii, care aproape exclud dintre potenţialii utilizatori ai bd de acest tip pe nespecialiştii în programarea calculatoarelor. În plus, aceste modele nu dispun de definiţii formale ce le-ar putea asigura consistenţa şi precizia. Desigur că succesul *MRD*, nu se explică însă numai prin înaltul grad de independenţă logică asigurat (faţă de orice posibilă implementare fizică) sau prin formalizarea sa matematică; el se datorează şi simplităţii unicei structuri folosite pentru organizarea datelor (o variantă a relaţiei matematice *n*-are) precum şi naturaleţei şi expresivităţii reprezentării acesteia sub formă de tabele.

Există foarte multe cărţi de specialitate ce prezintă şi discută aceste prime trei modele ale datelor, precum Atzeni şi De Antonellis [30], Date[118, 120], ElMasri şi Navathe [134], Korth şi Silberschatz [211], Maier [239], Tsichritzis şi Lochovski [334], Ullman [337, 338, 339, 342, 343].

Primele implementări ale modelului relaţional au apărut la începutul anilor 1980 (abia la un deceniu după definiţia sa abstractă!); şi asupra acestora există cărţi, precum Date [118,120] sau Stonebraker [323]. Începând cu 1985, o adevărată avalanşă de implementări pe calculatoare personale a înclinat definitiv balanţa în favoarea modelului relaţional, făcând aproape complet uitate şi desuete modelele ierarhic şi reţea. ElMasri şi Navathe [134], Korth şi Silberschatz [211], Valduriez şi Gardarin [344] indică şi discută bună parte din aceste implementări, furnizând multe alte referinţe suplimentare.

Cheile şi funcţional dependenţele au fost studiate prima oară de Codd [108,110]. Constrângerile tuplu nu au fost niciodată studiate formal; desigur însă că importanţa lor nu a putut fi subestimată de nici o implementare (a se vedea, de exemplu, Date [118]). Dependenţele de incluziune au fost luate în considerare mult mai târziu (începând cu Casanova ş.a. [86,87]), lucru oarecum surprinzător dacă avem în vedere atât înţelesul lor, mult mai concret decât al multor alte clase de constrângeri, cât şi marea lor importanţă practică în orice implementare.

Regulile de derivare au constituit prima tehnică folosită în studiul implicaţiei constrângerilor, odată cu introducerea de către Armstrong [21] a unei mulţimi de reguli solide şi complete (nu identică, dar echivalentă cu cea prezentată în problema 13 din secţiunea precedentă). Oricât de ciudat ar părea acest lucru, existenţa unei mulţimi solide şi complete de reguli de derivare pentru orice clasă de constrângeri (aşa numita *axiomatizare completă* a clasei respective) a fost multă vreme considerată mai importantă decât cea a decidabilităţii problemei implicaţiei. Pentru unele dintre aceste clase, s-a descoperit că nu există o *axiomatizare completă limitată* (adică o axiomatizare completă în care fiecare regulă trebuie să aibă un număr de antecedente cel mult egal cu o valoare fixată), chiar dacă problema implicaţiei pentru aceste clase este decidabilă (vezi Kanellakis ş.a. [192] pentru subclasa dependenţelor de incluziune unare sau Casanova ş.a. [87] pentru reuniunea claselor de *FD* şi de *DIN*). Pe de altă parte însă,

pentru unele clase de constrângeri a căror problemă a implicației este nedecidabilă (vezi Chandra ş.a. [99] pentru dependențele implicaționale încapsulate sau Vardi [346] pentru dependențele templet tipate), există o axiomatizare completă limitată (vezi Beeri şi Vardi [62], respectiv Sadri şi Ullman [304]). Ca atare, existența sau nonexistența unei axiomatizări complete limitate şi decidabilitatea (sau complexitatea) problemei implicației nu sunt legate între ele. Desigur însă că existența unei axiomatizări complete simple este un rezultat teoretic nu doar „frumos" în sine, ci îndeobşte şi foarte util în rezolvarea problemei implicației.

Dependențele multivaluate au fost independent introduse de Fagin [138] şi de Zaniolo [364], iar dependențele join de către Rissanen [293]. Alte clase de constrângeri, mai generale, au fost introduse în domeniu de Fagin [141], Yannakakis şi Papadimitriou [361], Beeri şi Vardi [62,63]. Menționăm aici doar faptul că mulțimea tuturor tipurilor de dependențe cunoscute poate fi împărțită în două mari clase [62,63]:

1. *dependențe generatoare de egalități* (care includ *FD*) ce egalizează întotdeauna valori ale unor coloane pe anumite linii; şi
2. *dependențe generatoare de tupli* (care includ *DJ* şi *DMV*) ce întotdeauna adaugă noi tupli la conținutul relațiilor.

Formele normale *FN1*, *FN2*, *FN3* şi *FNBC* au fost introduse de Codd [108] (primele trei, în 1972) şi [110] (*FNBC*, în 1974). Formele normale *FN4*, *FNPJ* (*FN5*) şi *FNDC* au fost introduse de Fagin în 1977 [138], 1979 [139] şi respectiv 1981 [140]. Formele normale *FN(3,3)* şi *FNKE* au fost propuse de Smith [316], respectiv de Zaniolo [364]. Breazu şi Mancaş [78] au introdus alte câteva forme normale: *FN2L* şi *FN3L* ("locale" în raport cu o cheie oarecare), *FN2S* şi *FN3S* ("speciale", în care se renunță la condiția ca atributele din partea dreaptă a *FD* să fie neprime), dovedind că *FN2L* şi *FN2S*, respectiv *FN3L*, *FN3S* şi *FNBC* sunt echivalente. Figura 0 prezintă sinoptic ierarhia formelor normale discutate în această monografie.

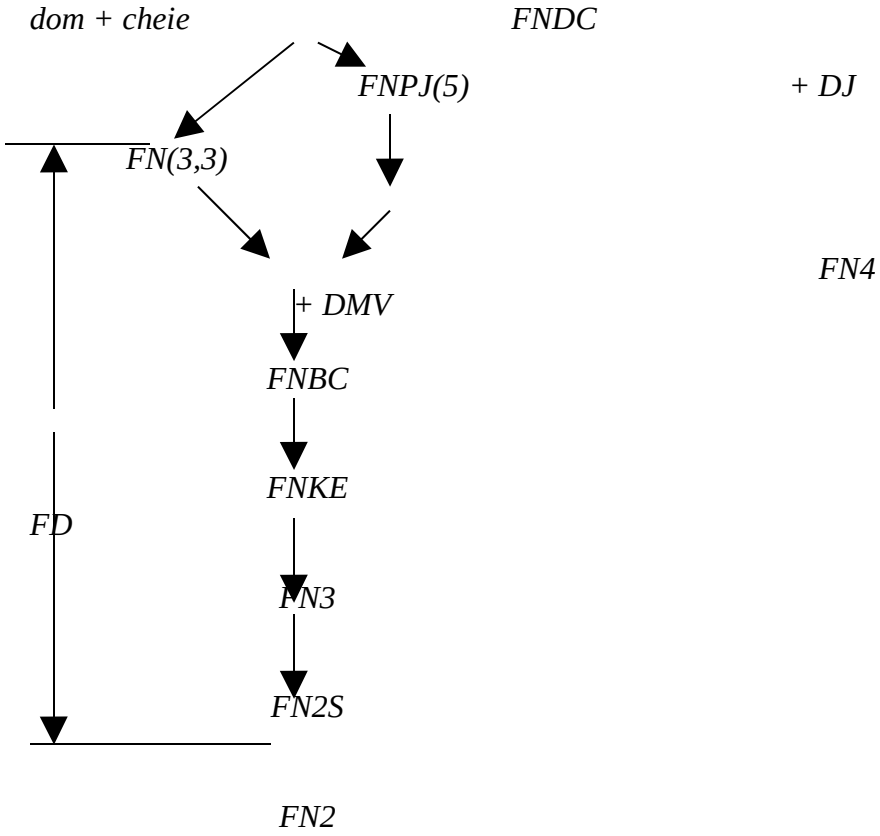


Figura 0: Ierarhia formelor normale relaționale şi constrângerile asociate

Studiul formal al anomaliilor se datorează lui Bernstein şi Goodman [67] (1980), LeDoux şi Parker [221] (1982) şi mai ales lui Chan [94] (1989).

Valorile nule necunoscute (împreună cu o extensie a algebrei relaționale ce include o logică cu trei valori şi un aşa numit principiu de substituție a nulurilor) au fost introduse de Codd [81]; critica abordării sale se datorează lui Grant [117]. Biskup [51,52] extinde şi aprofundează cercetările în domeniu.

Vassiliou [224,226] consideră atât nulurile necunoscute, cât şi pe cele inexistente în termenii semanticii denotaționale. Valorile inexistente au fost studiate şi de Lien [152] şi de Lerat şi Lipsky [151]; nulurile lipsite de informație au fost abordate de Zaniolo [233,235], Keller [134], Atzeni şi Morfuni [28,29] şi Atzeni şi De Bernardis [26].

Lipski [155,156] propune o primă generalizare a valorilor nule, prin folosirea valorilor *parțial specificate* (submulțimi nevide ale domeniului atributului corespunzător în care nulurile pot lua valori). O altă generalizare, bazată pe asocierea de probabilități fiecărei valori dintr-un domeniu, se datorează lui Wong [230] (o valoare specificată se poate reprezenta asignându-i probabilitatea 1, în timp ce toate celelalte valori ale domeniului au probabilitatea 0; un nul inexistent se reprezintă asociind probabilitatea 0 tuturor valorilor domeniului; un nul necunoscut, prin probabilitatea 1/*n* asociată tuturor valorilor, unde *n* este cardinalul domeniului respectiv[1]; iar nulurile parțial specificate de o submulțime de cardinalitate *k*, asociind probabilitatea 1/*k* tuturor acestor valori şi probabilitatea 0 restului domeniului).

Korth şi Ullman [145], Maier [160], Sagiv [191], Imielinski şi Lipski [125] au propus şi studiat folosirea de indici pentru a distinge între diversele valori nule (zise şi *nuluri marcate*, acestea permit, de exemplu, memorarea informației că anumite valori nule trebuie să coincidă).

Studiul comportării *FD* pentru valorile nule necunoscute se datorează, în esență, lui Vassiliou [225]; rezultatele sale în domeniu sunt însă strâns legate de cele obținute de Honeyman [120] privind satisfacerea *FD* privite drept constrângeri interrelaționale.

Lien [153] şi Atzeni şi Morfuni [28,29] s-au ocupat de *FD* şi valorile nule lipsite de informații (ultimii doi considerând şi interacțiunea *FD* cu constrângerile asupra valorilor nule).

Constrângerile de existență se datorează lui Maier [160]. Alte constrângeri asupra valorilor nule au fost propuse de Sciore [198] (*obiecte*) şi respectiv Maier [160] (*constrângeri de existență disjunctive*, care au fost apoi aprofundate de Goldstein [113] şi Atzeni şi Morfuni [29]).

O abordare interesantă inedită a dependențelor (de orice tip) este prezentată de Imielinski şi Lipski [126].

Deşi inițial normalizarea a dorit să ofere o proiectare automată a schemelor de bd, în ultimii ani, din ce în ce mai mulți cercetători în domeniu sunt de acord că o asemenea abordare este greşită. În locul ei, se recunoaşte fie şi numai implicit (dar, de exemplu, în [30] şi explicit!) că proiectantul de bd trebuie să facă întâi uz de un model "conceptual" ("semantic") al datelor, măcar de tip entități-asociații, urmând ca schema astfel rezultată să fie tradusă (manual sau automat) într-o schemă relațională normalizată. În acest context, teoria normalizării asigură doar repere şi caracterizează ținta unor asemenea proceduri. Capitolul 6 se bazează pe [206].

Majoritatea operatorilor algebrei relaționale au fost definiți de Codd [106], odată cu introducerea modelului relațional în 1970. Prezentările ulterioare ale altor autori [123,134,211,337,342] diferă foarte puțin de cele originale. Pentru capitolul de față, am preferat să urmărim prezentarea din [30].

[1] Dacă domeniul este infinit, trebuie recurs, desigur, la distribuții !

Studiul puterii expresive a algebrei relaționale prezentat în subsecțiunea 7.2.1 se datorează lui Paredaens [278] (1978); rezultatele similare pentru calculul relațional au fost obținute de Bancilhon [44] în acelaşi an.

Calculul relațional a fost introdus tot de Codd (în 1971-72) [107,109], care a propus limbajul *CRT* numit *ALPHA*, ce avea deja încorporată o formă de declarații ale domeniilor de valori. În 1978, Pirotte [280] a făcut distincția între *CRT* şi *CRD*, oferind o analiză detaliată a diverselor forme ale *CR*, inclusiv asupra problemelor ce apar la transformarea unui calcul aplicat generic într-un limbaj de interogare.

Deşi introdusă în 1969 de Di Paola [130], deci înaintea modelului relațional al datelor, chiar dacă într-un context mai larg, noțiunea de independență a domeniilor s-a impus foarte încet în teoria bazelor de date. Indezirabilitatea expresiilor al căror rezultat ar putea fi constituit de întregul domeniu al valorilor a fost totuşi detectată încă de la introducerea *CRT* şi *CRD* şi a condus imediat la impunerea noțiunilor de tip, respectiv domeniu de valori [109,280]; cu toate acestea, accentul era pus la început asupra dificultății (sau chiar imposibilității) calculului efectiv al rezultatelor [239,337]. În contextele formale ce nu făceau apel la declarații de domeniu al valorilor au fost propuse două tipuri de soluții:

- restrângerea limbajului la o subclasă a expresiilor care se comportă corect (numite diferit de diverşii autori: *cu domenii restrânse* [269], *sigure* [337], *evaluabile* [128] sau *permise* [333]);
- interpretarea interogărilor în raport cu domeniul activ şi nu cu întreg domeniul (*interpretare limitată* [239]).

Abia în 1982, Fagin [141] recunoaşte importanța independenței domeniilor ca noțiune semantică (în contextul formulelor ce descriu constrângeri de integritate); nedecidabilitatea ei a fost dedusă din rezultate anterioare [130, 345], iar restrângerea domeniilor, siguranța, evaluabilitatea, permisiunea şi limitarea interpretării au fost privite drept condiții suficiente pentru independența domeniilor. Aceasta este de altfel şi abordarea pe care am adoptat-o în subsecțiunea 7.1.1, cu excepția faptului că folosirea declarațiilor de domeniu al valorilor are avantajul de a oferi o justificare formală proprietăților pe care le au limbajele comerciale oferite de SGBD curente.

Echivalența între limbajele algebrice şi cele bazate pe calculul relațional a fost demonstrată tot de Codd [109]; demonstrații diferite au fost apoi propuse de Ullman [337] şi de Maier [239]. Această echivalență l-a determinat pe Codd să introducă şi noțiunea de completitudine [109]; deja pentru *ALPHA* însă, el a simțit nevoia de mai mult decât completitudine şi a introdus funcțiile tuplu şi cele de agregare. Completitudinea a fost apoi caracterizată de Paredaens [278] şi Bancilhon [44]. În 1982, Klug [209] a oferit demonstrația echivalenței între algebra şi calculul relațional extinse cu funcții de agregare.

Teorema 193, care dovedeşte că închiderea tranzitivă nu poate fi exprimată în algebra relațională, datează din 1979 şi se datorează lui Aho şi Ullman [19] (care au formulat-o şi demonstrat-o pentru *AR*; demonstrația pentru *CRD* din secțiunea 7.4 este mai uşoară şi conduce, evident, la acelaşi rezultat – în virtutea echivalenței între cele două formalisme). Noțiunea de *CH*-completitudine a fost introdusă de Chandra şi Harel [98] în 1980 sub denumirea de *interogări calculabile*. Noțiunile înrudite privind interogările generice au fost studiate de Hull [177].

Chandra [97] constituie o excelentă trecere în revistă a problematicii limbajelor de interogare (incluzând şi limbaje mai puternice decât cele prezentate în acest capitol), ce prezintă ierarhii de limbaje bazate pe putere expresivă, complexitate şi primitive de programare. În plus, lucrarea include şi o impresionantă bibliografie cu lucrări din logica matematică ce conțin şi unele din rezultatele pe care le-am prezentat în acest capitol şi care au fost de fapt demonstrate anterior, într-un context mai general.

Dintre operatorii algebrici propuşi suplimentar şi relevanţi doar în manipularea valorilor nule menţionăm: *augmentarea*, unar, producând o relaţie peste atributele operandului plus alte atribute şi ai cărui tupli sunt proveniţi din cei ai operandului prin extinderea lor cu valori nule; *joinul* (*natural*) *extern* (*„outer join”*, adică LEFT şi RIGHT) [148]; şi *proiecţia totală* [191], o proiecţie uzuală urmată de eliminarea tuturor tuplilor ce conţin valori nule. Este uşor de demonstrat că aceşti operatori sunt derivaţi, căci se pot obţine din cei fundamentali şi din relaţii constante conţinând valori nule.

Cel de-al doilea tip de abordare a nulurilor (i.e. cel bazat pe constrângeri asupra acestora) a fost introdus şi dezvoltat de Imielinski şi Lipsky [127], Abiteboul şi colectivul [7] şi de Grahne [116]. Cel de-al treilea, bazat pe predicatul *NULL*, nu provine din cercetarea în domeniu, ci din industria SGBD relaţionale!

SQL provine din *SEQUEL* (acronimul pentru „*Structured English QUEry Language*”), creaţie a cercetătorilor corporaţiei *IBM* (Chamberlin şi Boyce [371], 1974) care au lucrat la prototipul primului SGBD relaţional din lume, *System R* [27, 92].

QBE se datorează altui distins cercetător al *IBM*, Moshe Zloof [369] şi datează din 1977.

Timp de foarte mulţi ani, s-a dat foarte puţină atenţie limbajelor de actualizare a datelor: se credea că acestea pot fi specificate doar cu ajutorul unor operaţii simple (adăugarea, modificarea şi ştergerea de tupli ce satisfac anumite condiţii) care pot fi reduse la operaţii algebrice cu mulţimi. Mai nou însă, de prin 1988, Abiteboul şi Vianu [8,9,10,11] au dovedit că există şi în acest domeniu probleme de adâncime, care au determinat apariţia unui nou câmp de cercetări încă activ. O trecere interesantă în revistă a aspectelor fundamentale în această direcţie o constituie [1].

1.2 Relaţii

O *bd relaţională* este reprezentată ca o colecţie de tabele, fiecare dintre ele având ataşat un nume unic în cadrul bd. Liniile tabelelor reprezintă o legătură (în sensul de asociaţie, relaţie) între elementele unor mulţimi de valori. Capul de tabel conţine nume de *atribute* ce trebuie să fie distincte în cadrul fiecărei tabele. Pentru fiecare coloană în parte există o mulţime de valori posibile, numită *domeniu*[2].

Figura 1 prezintă o reprezentare tabulară a unor informaţii despre cărţi şi vânzarea lor. Analizând cele patru tabele, se poate afirma că fiecare linie este un *n-tuplu* ordonat de valori $<d_1, d_2, ..., d_n>$, în care fiecare valoare d_j aparţine domeniului coloanei j, $j = 1,2,...,n$, unde $n \geq 1$ este numărul de atribute al tabelei.

În teoria matematică a relaţiilor, dat fiind un şir de mulţimi $D_1, D_2, ..., D_n$ (nu neapărat distincte!), o *relaţie* este o submulţime a produsului cartezian $D_1 \times D_2 \times ... \times D_n$. $D_1, D_2, ..., D_n$ se zic *domeniile* relaţiei, iar n se zice *gradul* (sau *aritatea*) ei. O relaţie este deci o mulţime ordonată de n-tupli de forma $<d_1, d_2, ..., d_n>$ astfel încât fiecare valoare d_j aparţine domeniului D_j, pentru $j = 1,2,..., n$.

Una dintre reprezentările posibile ale relaţiilor este, desigur, cea tabulară (lipsită însă de numele coloanelor).

Numărul de n-tupli ai unei relaţii se zice *cardinalul*[3] ei. În general, dacă domeniile sunt infinite, atunci şi cardinalitatea relaţiei poate fi infinită. Evident însă că în aplicaţii concrete de bd cardinalul oricărei relaţii este finit. În cele ce urmează, implicit, vom presupune relaţii finite peste domenii infinite.

Conform proprietăţilor fundamentale ale mulţimilor şi relaţiilor, rezultă următoarele:

[2] Aici înţeles mai degrabă ca în programare, drept „domeniu de valori”, deşi este vorba de *codomeniul* atributului privit ca funcţie definită pe relaţia din care face parte.

[3] În general, funcţia *cardinal* (notată *card*(M) sau *card*M, unde *M* e o mulţime) este definită pe mulţimea tuturor mulţimilor şi cu valori în mulţimea naturalilor, asociind fiecărei mulţimi numărul ei de elemente. Trivial, de exemplu, $card\emptyset = 0$.

1. (i) ordinea *n*-tuplilor într-o relație este irelevantă
(ii) *n*-tuplii unei relații sunt distincți (nu există dubluri).

AUTORI

#Autor	*Prenume*	*Nume*
1	Emil	Cioran
2	Constantin	Noica
3	Mircea	Ciobanu
4	Mircea	Eliade

VÂNZĂTORI

#Vânzător	*Vânzător*
1	Librăria din fundul curții
2	Dalles
3	Eminescu

CĂRȚI

#Carte	*Autor*	*Titlu*	*Preț*
1	1	Silogismele amărăciunii	6.000
12	2	Devenirea întru ființă	8.000
29	3	Convorbiri cu Regele Mihai I	6.000
37	4	De la Zalmoxis la Gingis-Han	12.000

VÂNZĂRI

#Vânzare	*Vânzător*	*Carte*	*Cantitate*
1	1	1	670
2	1	12	210
3	1	29	2.890
4	1	37	915
5	2	29	4.770
6	3	1	820
7	3	29	150

Figura 1: Reprezentarea tabulară a unei bd

2. orice *n*-tuplu este o mulțime ordonată de valori (adică orice a *i*-a valoare aparține celui de-al *i*-lea domeniu); ca atare, ordinea domeniilor unei relații matematice este semnificativă.

După cum se poate observa în exemplul 1, pentru a interpreta corect înțelesul unei relații matematice, referirea la domenii trebuie făcută folosind poziția acestora în şirul definind relația.

Exemplul 1 Să reprezentăm printr-o relație matematică, numită *orar*, informații privind cursurile ținute într-o facultate: *orar* $\subseteq$ *şir* $\times$ *şir* $\times$ *dată* $\times$ *dată*,

orar = { <Ionescu, Logică, 15.07.94, 30.05.95>,

<Popescu, Algebră, 15.07.94, 30.06.95>,

<Nicolescu, Fizică, 15.07.95, 20.01.96>}

Domeniul *dată* are aici două roluri distincte, indicând data de începere a cursurilor, respectiv cea de sfârşit. Singurul mod de a distinge între ele este doar faptul că acestea ocupă poziția a treia, respectiv a patra în şirul domeniilor. Acelaşi lucru este valabil, desigur, şi pentru domeniul *şir*.

Pentru a evita acest neajuns, este de obicei adoptată în bdr o definiție alternativă a relațiilor: fiecărei apariții a unui domeniu i se asignează un *rol*. Rolurile sunt referite prin nume simbolice zise *atribute*, care corespund cât mai apropiat cu putință numelor coloanelor din reprezentarea tabulară. Formal, legătura între atribute şi domenii este stabilită de o funcție *dom* : $X \rightarrow D$ definită pe mulțimea atributelor $X = \{A_1, A_2, \ldots, A_n\}$ şi

cu valori în mulțimea domeniilor *D*. Similar, un tuplu peste mulțimea atributelor *X* este o funcție *t* care asociază fiecărui atribut $A_i \in X$ o valoare din domeniul $dom(A_i)$; această valoare se notează $t[A_i]$. Notația se poate extinde la mulțimi de atribute $Y \subseteq X$: $t[Y]$ indică restricția funcției *t* la atributele din *Y*.[4]

Exemplul 2 În exemplul 1 s-ar putea defini următoarele atribute:

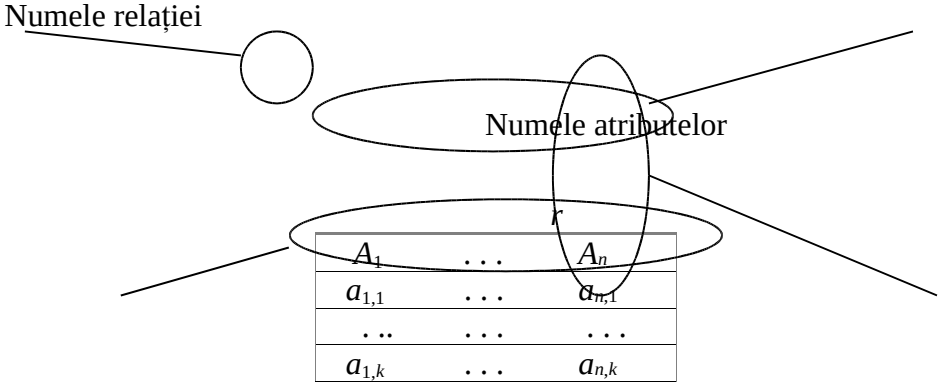
Profesor cu *dom*(*Profesor*) = *şir*
Curs cu *dom*(*Curs*) = *şir*
Început cu *dom*(*Început*) = *dată*
Sfârşit cu *dom*(*Sfârşit*) = *dată*

Tuplul <Ionescu, Logică, 15.07.94, 30.05.95> din exemplul 1 poate fi descris funcțional astfel: $t[Profesor]$ = "Ionescu", $t[Curs]$ = "Logică",

$t[Început]$ = 15.07.94, $t[Sfârşit]$ = 30.05.95.

Revenind la reprezentarea tabulară, putem acum afirma şi formal că o relație poate fi prezentată în mod natural ca o tabelă ale cărei linii corespund tuplilor şi în care coloanele conțin valori ale câte unui atribut al relației (vezi figura 2). Este însă important de subliniat faptul că o relație este o descriere formală a corespondenței între elementele unor mulțimi, în timp ce o tabelă este doar una dintre posibilele reprezentări ale relației. Analizând reprezentarea tabulară a relațiilor, se observă următoarele:

- Valorile oricărei coloane sunt omogene (i.e. valorile unui atribut aparțin unui acelaşi domeniu: întregi, şiruri de caractere etc.).
- Liniile sunt distincte (relațiile sunt mulțimi de tupli, deci nu conțin niciodată duplicate).
- Ordinea coloanelor este irelevantă (căci ele sunt întotdeauna identificate printr-un nume unic în cadrul relației şi nu prin poziție).
- Ordinea liniilor este irelevantă (căci ele sunt identificate prin conținut şi nu prin poziție).



[4] Această notație conține o incoerență: dacă *A* este un atribut, atunci $t[A]$ este o valoare; simultan, dacă *X* este un atribut, atunci $t[X]$ este un tuplu, adică o funcție. De asemenea, după cum vom vedea, mulțimile conținând un singur atribut sunt de obicei notate cu numele atributului; urmează că, dacă *X*=*A*, atunci $t[A]$ este atât o valoare cât şi un tuplu. Ambiguitatea aceasta este însă deseori irelevantă, dar trebuie clarificată semnificația notațiilor în toate cazurile relevante. Desigur că totul ar fi mai clar dacă tuplii n-ar fi considerați în *MRD* funcții, ci valori ale produselor de funcții (i.e. atributelor) alcătuind relațiile.

Valorile unui atribut

Figura 2: Atribute, tupli şi valori într-o relaţie

1.3 Definiţia formală a modelului relaţional

Un *model al datelor* este o unealtă formală pentru descrierea informaţiilor de interes în diverse aplicaţii. Există două niveluri de descriere posibile:

1. *Nivelul intensional*, implicând proprietăţile generale ale claselor de valori, precum şi ale claselor de corespondenţe între acestea.
2. *Nivelul extensional*, implicând valorile propriu-zise şi corespondenţele între ele.

În modelul relaţional, primul nivel corespunde descrierii independente de timp a relaţiilor (*scheme*), iar cel de-al doilea corespunde conţinutului de date al relaţiilor (*instanţe*) la un moment dat. Definiţiile exacte urmează:

Definiţia 1 Schema relaţiei constă dintr-un nume (*numele relaţiei*) şi o mulţime de nume de atribute. Notaţia pentru o relaţie numită R cu atribute A_1, A_2, …, A_n este $R(A_1, A_2, \ldots, A_n)$. Prima literă a numelui relaţiilor, prin convenţie, este literă mare.

Schema unei bazei de date constă dintr-o mulţime de scheme de relaţii, fiecare din ele având nume distincte. Prin convenţie, notaţia pentru schema unei bd utilizează litere îngroşate, de exemplu $\boldsymbol{R}$.

Definiţia 2 Conţinutul (*instanţa*) *relaţiei* (sau, simplu, *relaţia*) definită peste o schemă $R(X)$ constă dintr-o mulţime finită de tupli peste X. Prin convenţie, conţinutul unei relaţii se notează cu numele schemei sale scris însă cu literă mică (de exemplu r).

Conţinutul (*instanţa*) *bazei de date* (sau, simplu, *baza de date*) definită peste o schemă $\boldsymbol{R} = \{R_1(X_1), R_2(X_2), ..., R_n(X_n)\}$ constă dintr-o mulţime de relaţii $\{r_1, r_2, ..., r_n\}$, unde fiecare r_i este definită peste schema corespunzătoare R_i.[5]

Exemplul 3 Fie informaţiile despre cărţi şi vânzări din figura 1.
Schema bd conţine patru scheme de relaţie:

AUTORI (*#Autor*, *Prenume*, *Nume*)
VÂNZĂTORI (*#Vânzător*, *Vânzător*)
CĂRŢI (*#Carte*, *Autor*, *Titlu*, *Preţ*) şi
VÂNZĂRI (*#Vânzare*, *Vânzător*, *Carte*, *Cantitate*)

Conţinuturile relaţiilor corespunzătoare sunt:

$autori = \{t_1, t_2, t_3, t_4\}$
$vânzători = \{t_5, t_6, t_7\}$
$cărţi = \{t_8, t_9, t_{10}, t_{11}\}$ şi
$vânzări = \{t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}\}$,

unde: $t_1[\#Autor] = 1$, $t_1[Prenume]$ = "Emil", $t_1[Nume]$ = "Cioran",...,
$t_8[Titlu]$ = "Silogismele amărăciunii", $t_8[Preţ]$ = 6.000 ş.a.m.d.

Pentru a uşura citirea, reprezentăm relaţiile cu ajutorul tabelelor, adoptând notaţia formală doar pentru valori individuale ale datelor. Pentru a reprezenta conţinutul unei bd, încadrăm pe figura corespunzătoare toate tabelele componente într-un dreptunghi, ca în figura 1.

Alte convenţii notaţionale uzuale sunt următoarele: atributele se notează cu litere mari de la începutul alfabetului, eventual cu indici şi apostroafe ($A,B,...,A_1,A',...$); mulţimile de atribute la fel, dar folosind litere de la sfârşitul alfabetului ($X,Y,...,X_1,X'$, ...) sau prin juxtapunerea numelor atributelor implicate (de exemplu, $X = A_1A_2...A_k$); similar, re-

[5] Mai precis (dar mai puţin clar), o bază de date ar putea fi definită ca o aplicaţie asociind fiecare relaţie r peste $R(X)$ cu schema sa $R(X)$.

uniunea mulțimilor de atribute se notează prin concatenarea numelor acestora (de exemplu, *XA* înseamnă $X \cup \{A\}$); câteodată, mai ales în cazul operanzilor din expresiile algebrice relaționale, în loc de numele relației se foloseşte notația $r(X)$, cu semnificația "relația *r* definită peste mulțimea de atribute *X*".

1.4 Prima formă normală

În exemplele de relații de până acum, toate valorile oricărui atribut au fost atomice (adică indivizibile în bd). Asemenea atribute se zic *simple*. Există însă şi alte două tipuri de atribute la care ne vom opri în cele ce urmează[6].

Un atribut se zice *multivaluat* dacă valorile sale sunt elemente ale mulțimii părților unei mulțimi (de valori).

Exemplul 4 Atributul *CoduriDisciplineAbsolvite* din relația prezentată în figura 3 este multivaluat. Domeniul său este mulțimea părților mulțimii tuturor cursurilor oferite studenților de facultatea în cauză.

ABSOLVIRE

CodStudent	*AnAbsolvire*	*CoduriDisciplineAbsolvite*
0962	1995	{MA101, MA102, F100, PC101}
0962	1996	{MA201, MA202, PC201, BD201, SD201}
1087	1996	{}
1152	1996	{MA101, F100}

Figura 3: O relație cu un atribut multivaluat

EXAMINĂRI

CodStudent	*DataExaminării*	*NoteObținute*
0962	20.06.95	<MA101,9>
0962	15.02.96	<MA202,10>
1152	08.02.96	<F100,6>

Figura 4: O relație cu un atribut structurat

CATALOG

CodStudent	*Secția*	*Examene*		
		Data	*Disciplina*	*Nota*
0962	Calculatoare	20.06.95	MA101	9
		30.06.95	MA102	10
		07.07.95	PC101	10
		14.07.95	F100	6
		15.02.96	MA202	10
1087	Automatizări	14.07.95	F100	4
		15.02.96	MA202	4

Figura 5: O relație cu un atribut relație (adică şi multivaluat şi structurat)

CATALOG

CodStudent	*Secția*	*Data*	*Disciplina*	*Nota*
0962	Calculatoare	20.06.95	MA101	9
0962	Calculatoare	30.06.95	MA102	10
0962	Calculatoare	07.07.95	PC101	10
0962	Calculatoare	14.07.95	F100	6

[6] Tipuri provenind, de fapt, din modelele de date ierarhic şi rețea.

0962	Calculatoare	15.02.96	MA202	10
1087	Automatizări	14.07.95	F100	4
1087	Automatizări	15.02.96	MA202	4

Figura 6: Relația plată echivalentă celei din figura 5

Un atribut se zice *structurat* dacă valorile sale sunt tupli (de valori), deci dacă domeniul său este o relație.

Exemplul 5 Atributul *NoteObținute* din relația prezentată în figura 4 este structurat. Domeniul său este produsul cartezian între mulțimea disciplinelor şi mulțimea notelor posibile.

Domeniile pot avea structuri şi mai complexe, construite din compuneri de multivaluări şi structurări.

Exemplul 6 Atributul *Examene* din relația prezentată în figura 5 este atât multivaluat, cât şi structurat. Valorile sale sunt relații (adică mulțimi de tupli)! În asemenea cazuri, desigur, sunt necesare convenții de reprezentare tabulară care să evidențieze nivelurile de structurare.

Definiția 3 O schemă de relație se zice că este în *prima formă normală (FN*1*)* (sau "*plată"*) dacă toate atributele ei sunt simple. Altfel, relația se zice *"încuibată"* (*"nested"* în engleză).

În modelul relațional al datelor sunt considerate doar relații plate, astfel încât modul de reprezentare al datelor să fie simplu şi uniform. Se poate arăta în mod trivial că orice relație încuibată se poate transforma într-o relație plată echivalentă.

Exemplul 7 Figura 6 prezintă o relație plată echivalentă cu cea din figura 5. "Aplatizarea" a fost evident realizată prin înlocuirea atributului multivaluat structurat *Examene* cu componentele sale *Data*, *Disciplina*, *Nota*.

1.5 Introducere în algebra relațională

Bazele de date nu sunt utilizate doar pentru memorarea (chiar dacă structurată) a datelor din aplicații, ci, mai ales, pentru a le putea regăsi, actualiza şi extrage din ele informații agregate (exemple: totaluri, medii, maxime, minime etc.). Ca atare, modelele de care ne ocupăm includ şi limbaje pentru interogarea şi actualizarea datelor.

Actualizările pot fi văzute ca funcții definite pe şi cu valori în mulțimea stărilor bd; la rândul lor, interogările sunt funcții definite pe mulțimea stărilor şi cu valori în spațiul relațiilor peste toate schemele posibile. Prin urmare, cele două familii de limbaje au multe lucruri în comun. Literatura de specialitate acordă însă o mult mai mare atenție limbajelor de interogare (care sunt mai complexe) decât celor de actualizare.

De fapt, chiar şi schema unei bd poate fi interpretată ca o colecție de întrebări, iar conținutul ei ca pe mulțimea răspunsurilor corespunzătoare. De exemplu, schema bd prezentată în figura 1 (şi formalizată relațional în exemplul 3) este constituită din următoarele patru întrebări:

1. „Care sunt codul, prenumele şi numelor autorilor de interes?”
2. „Care sunt codul şi denumirea vânzătorilor de interes?”
3. „Care sunt codul autorului, codul, titlul şi prețul cărților de interes?"
4. „Care sunt codul vânzării, al vânzătorului şi al cărții (la care se referă întrebările 2 şi 3 de mai sus), precum şi cantitatea (i.e. numărul de exemplare) vândute?"

Desigur că răspunsul la fiecare dintre aceste întrebări în parte este oferit de relațiile *AUTORI*, *VÂNZĂTORI*, *CĂRȚI*, respectiv *VÂNZĂRI*.

În abordările logice ale modelării datelor, se consideră chiar că fiecare tuplu în parte constituie răspunsul la o întrebare: de exemplu, primul tuplu al relației *CĂRȚI* din figura 1 este interpretat drept "Este *adevărat* faptul că, printre cărțile de interes, se numără şi

cea cu titlul ‘Silogismele amărăciunii’, cu codul 1 şi preţul de vânzare 6.000, având autorul cu cod 1"; iar primul tuplu al relaţiei *VÂNZĂRI* e interpretat drept "Este *adevărat* faptul că din cartea având codul 1 s-au vândut de către vânzătorul cu codul 1 cantitatea de 670 exemplare."

Pe baza acestor întrebări şi răspunsuri (fundamentale) memorate de orice bd, se pot formula în orice moment o mulţime de alte întrebări ("derivate") la care răspunsul poate fi calculat. De exemplu, referindu-ne din nou la bd din figura 1: "Ce autor, ce cod şi ce preţ are cartea cu titlul ‘Devenirea întru fiinţă’?"; "Există vreo carte de autorul ‘Gabriel Liiceanu’ printre cele de interes?"; "Ce titluri de cărţi ale autorului ‘Mircea Eliade’ au fost editate şi la ce preţuri?"; "Câte exemplare s-au revândut în total din cartea cu titlul ‘Convorbiri cu Regele Mihai I’?"; "Care este mulţimea codurilor de cărţi ce nu au fost revândute în cel puţin 100 de exemplare?"; "Care este totalul încasărilor vânzătorului ‘Dalles’ pentru cărţile de interes?" etc.

Desigur că sunt posibile şi întrebări la care răspunsul nu poate fi calculat pornind doar de la informaţiile memorate de bd, fie din lipsa unor valori, fie pentru că întrebările nu sunt corect sau sunt ambiguu formulate.

De exemplu, revenind din nou la figura 1, la întrebarea "Ce preţ are cartea cu titlul ‘Pe culmile disperării’?" nu s-ar putea răspunde decât că "Momentan, nu există nici o carte cu titlul ‘Pe culmile disperării’ printre cele despre care sunt deţinute date!"; sau la întrebarea "În ce an a fost scrisă cartea cu titlul ‘De la Zalmoxis la Gingis-Han’?" nu se poate răspunde decât cel mult că "Pentru nici o carte nu este memorat anul scrierii!"; în timp ce la întrebarea "Care este codul cărţilor vândute într-o cantitate mai mare decât titlul lor?" nu se poate răspunde decât ceva de tipul "Întrebare incorect formulată, deoarece cantităţile nu sunt comparabile cu titlurile!".

Limbajele de interogare a datelor sunt tocmai uneltele ce permit formularea, verificarea corectitudinii şi calcularea răspunsurilor la toate întrebările corecte, posibil de derivat pe marginea oricărei bd. Deşi abia capitolul 7 se ocupă în detaliu de interogări, este nevoie deja să prezentăm în următoarea secţiune cei trei operatori fundamentali ai *algebrei relaţionale*, principalul limbaj de interogare al modelului relaţional, şi anume: *selecţia*, *proiecţia* şi *join*-ul[7] natural.

1.5.1 Selecţie

Operatorul unar de selecţie este definit cu ajutorul formulelor logice ale calculului propoziţional.

Definiţia 4 Fie r o relaţie peste mulţimea X de atribute; o *formulă propoziţională* (sau, simplu, *propoziţie*) F peste X se defineşte recursiv, pornind de la atomi şi folosind operatorii logicii propoziţionale, astfel:

Atomii peste X sunt de forma $A_1 \Theta A_2$ sau $A_1 \Theta a$, unde $A_1, A_2 \in X$, a este o constantă, iar Θ este unul dintre operatorii de comparaţie: $=, \neq, \leq, \geq, <, >$.[8] Orice atom peste X este o formulă propoziţională peste X; dacă F_1, F_2 sunt propoziţii peste X, atunci (F_1), $\neg(F_1)$, $F_1 \wedge F_2$, $F_1 \vee F_2$ sunt propoziţii peste X. Nimic altceva nu este formulă propoziţională.

[7] În engleză, "*join*" înseamnă, printre altele, "a îmbina/îmbinare", sens care redă cel mai exact în română semnificaţia avută în vedere de E.F.Codd pentru denumirea acestui operator. Am fi putut să-l traducem deci perfect prin "operatorul de îmbinare a relaţiilor", dar nu facem acest lucru şi preferăm în loc acest barbarism întrucât "join" a intrat deja de mult în jargonul de bd.

[8] Desigur că aceşti operatori de comparaţie cer ca domeniile implicate să fie mulţimi ordonate, ceea ce constituie probabil o presupunere rezonabilă pentru orice aplicaţie: în general, bd memorează numere, şiruri de caractere, constante logice şi date calendaristice. Evident însă că, de exemplu, cel mai adesea, compararea cu alt operator decât egalitatea a conţinutului sau adreselor unor fotografii, fişiere audio sau video nu are sens!

O propoziție peste X poate fi privită ca fiind o funcție booleană ce asociază fiecărui tuplu o valoare de adevăr. Un atom are valoarea *adevărat* pentru un tuplu t dacă $t[A_1]\Theta \Theta t[A_2]$ (respectiv $t[A_1]\Theta a$) este *adevărat*, iar în caz contrar, desigur, atomul are valoarea *fals*.

O propoziție de forma (F) este *adevărată* pentru t dacă şi numai dacă F este *adevărată* pentru t; o propoziție de forma $\neg(F)$ este *adevărată* pentru $t \Leftrightarrow F$ este *falsă* pentru t. $F_1 \wedge F_2$ este *adevărată* pentru $t \Leftrightarrow$ atât F_1 cât şi F_2 sunt *adevărate* pentru t; $F_1 \vee F_2$ este *adevărată* pentru $t \Leftrightarrow$ cel puțin una dintre propozițiile F_1, F_2 este *adevărată* pentru t.

Definiția 5 Fie o relație $r(X)$ şi o propoziție F peste X; se zice *selecția* lui r în raport cu F şi se notează cu $\sigma_F(r)$ relația având tot schema X, dar care conține doar acei tupli din r ce satisfac F, adică: $\sigma_F(r) = \{ t \in r \mid F(t) = adevărat\}$.

Figura 7 prezintă un exemplu de selecție asupra relației *VÂNZĂRI* din figura 1 şi anume formalizarea relațională a întrebării "Ce vânzări s-au făcut în cantități depăşind 1.000?", precum şi răspunsul aferent.

Se observă că, în general, selecția permite formalizarea întrebărilor de tipul "Ce tupli ai relației ... satisfac predicatul logic următor: ...?".[9]

$\sigma_{Cantitate > 1.000}(VÂNZĂRI)$

#Vânzare	*Vânzător*	*Carte*	*Cantitate*
3	1	29	2.890
5	2	29	4.770

Figura 7: Un exemplu de selecție

1.5.2 Proiecție

Definiția 6 Fie o relație $r(X)$ şi o submulțime proprie Y a lui X; se zice *proiecția* lui r pe Y şi se notează cu $\pi_Y(r)$ relația peste atributele Y ce conține restricția tuplilor lui r la atributele din Y, adică: $\pi_Y(r) = \{ t[Y] \mid t \in r \}$.[10]

Proiecția este, într-un fel, ortogonală selecției: în timp ce proiecția ia în considerare toți tuplii operandului peste o submulțime a atributelor acestuia, selecția consideră o submulțime a tuplilor operandului peste toate atributele sale. De aceea, se mai spune că proiecția calculează *descompuneri verticale*, în timp ce selecția calculează *descompuneri orizontale* ale relațiilor.

Figura 8 prezintă un exemplu de proiecție asupra relației *VÂNZĂRI* din figura 1 şi anume formalizarea relațională a întrebării "Care este submulțimea vânzărilor restrânsă la codul cărții şi cantitatea vândută?", precum şi răspunsul aferent.

Se observă că, în general, proiecția permite formalizarea întrebărilor de tipul "Ce tupli are relația obținută restrângând tuplii relației ... la submulțimea următoare a atributelor ei: ...?".

De notat că, datorită cerinței de unicitate a tuplilor impusă de definiția matematică a mulțimilor, prin proiecție se pot pierde informații în raport cu cele memorate de relația proiectată: de exemplu, chiar dacă în *VÂNZĂRI* ar mai fi existat şi tuplul <8,2,1, 670> rezultatul proiecției din figura 8 nu s-ar fi schimbat cu nimic!

Evident că singurul mod în care acest tip de pierdere a informațiilor poate fi evitat este includerea unei chei (iar în cazul existenței valorilor nule, aceasta trebuie neapărat să fie cheia primară!) în lista atributelor după care se face proiecția (vezi problema 86).

[9] Evident că dacă toți tupli unui conținut r satisfac F, atunci $\sigma_F(r)$ este aplicația unitate, în timp ce dacă nici unul nu o satisface, atunci conținutul calculat de acest operator este vid.

[10] Evident că, pentru orice relație r, dacă $Y = X$, atunci π_Y este aplicația unitate, în timp ce $\pi_\varnothing = \varnothing$.

Pentru a feri programatorii neavizați sau neatenți de asemenea pierderi de informații, majoritatea implementărilor comerciale (bazate pe limbajul *SQL*) nu elimină automat duplicatele din răspuns (a se vedea predicatele *SQL* DISTINCT şi DISTINCTROW).

$\pi_{\#Carte,\ Cantitate}(VÂNZĂRI)$

#Carte	*Cantitate*
1	670
12	210
29	2.890
37	915
29	4.770
1	820
29	150

Figura 8: Un exemplu de proiecție

1.5.3 Join natural

Definiția 7 Fie $r_1(YX)$ şi $r_2(XZ)$ două relații astfel încât $YX \cap XZ = X$; se zice *join natural*[11] (sau simplu, *join*) între r_1 şi r_2 şi se notează cu $r_1 \bowtie r_2$ relația definită peste YXZ şi care conține toți tupli (peste YXZ) ce rezultă din concatenarea acelor tupli din r_1 şi din r_2 ale căror valori pentru atributele X coincid, adică:

$$r_1 \bowtie r_2 = \{\ t \text{ peste } YXZ \mid \exists(t_1 \in r_1, t_2 \in r_2),\ t[XY] = t_1[XY] \wedge t[XZ] = t_2[XZ]\ \}.$$

Definiția 8 Se spune că doi tupli $t_1 \in r_1$, $t_2 \in r_2$ sunt *joinabili* dacă ei contribuie la formarea joinului, adică dacă $t_1[X] = t_2[X]$. Dacă un tuplu nu contribuie la formarea joinului, el se zice *"încurcă-lume"* (*"dangling"* în engleză). Dacă nici una din cele două relații nu conține asemenea tupli încurcă-lume, atunci se zice că rezultatul joinului lor natural este un *join complet*.

Se observă că, în general, joinul natural permite formalizarea întrebărilor de tipul "Ce tupli are relația obținută prin concatenarea tuplilor relației ... cu toți tuplii relației ... care au aceleaşi valori pentru atributele comune celor două relații?".

> *Exemplul 8* Figura 9 prezintă joinul (complet) dintre relațiile *CĂRȚI* şi *VÂNZĂRI* din figura 1. Se observă, de exemplu, că primul tuplu din *CĂRȚI* este joinabil numai cu primul şi cu penultimul dintre tuplii *VÂNZĂRI*. Acest join reprezintă formalizarea relațională a întrebării "În ce cantități au fost vândute cărțile de interes de către fiecare vânzător în parte?", precum şi răspunsul aferent.

CĂRȚI $\bowtie$ *VÂNZĂRI*

#Carte	*Autor*	*Titlu*	*Preț*	*#Vânzare*	*Vânzător*	*Cantitate*
1	1	Silogismele amărăciunii	6.000	1	1	670
1	1	Silogismele amărăciunii	6.000	2	3	820
12	2	Devenirea întru ființă	8.000	3	1	210
29	3	Convorbiri cu Regele Mihai I	6.000	4	1	2.890
29	3	Convorbiri cu Regele Mihai I	6.000	5	2	4.770
29	3	Convorbiri cu Regele Mihai I	6.000	6	3	150
37	4	De la Zalmoxis la Gingis-Han	12.000	7	1	915

Figura 9: Un exemplu de join natural

Este interesant de remarcat faptul că definiția 7 are sens chiar dacă X este mulțimea vidă (adică cele două relații nu au nici un atribut comun). În acest caz, orice tupli $t_1 \in r_1$,

[11] Vom vedea că operatorul join este mai general decât cel "natural" definit aici: se poate defini Θ-*joinul* pentru orice operator de comparație Θ aplicabil domeniilor respective: de exemplu, oricare dintre =, ≠, ≤, ≥, <, >. Dacă Θ este operatorul "=", atunci joinul se zice *echijoin*. Trivial, joinul natural este un caz particular de echijoin. Mai general însă, join-ul este de fapt compunerea selecției cu produsul cartezian!

$t_2 \in r_2$ sunt joinabili, iar joinul natural astfel obținut este chiar produsul cartezian al celor două relații.[12]

Din definiția sa, rezultă că joinul este, într-un anumit sens, inversul proiecției: așa cum se poate spune despre proiecție că generează descompuneri verticale, putem afirma că joinul calculează *compuneri verticale*; vom clarifica însă natura exactă a dualității între aceşti doi operatori în subsecțiunea următoare.

După cum am afirmat încă de la începutul acestui capitol, în modelul relațional nu există pointeri şi nici vreun alt fel de referințe explicite între date, toate informațiile fiind memorate folosind exclusiv valori ale datelor. Drept urmare, legăturile între relații sunt reprezentate prin existența unor atribute identice în schema relațiilor implicate, ceea ce face ca joinul natural să fie operatorul fundamental în corelarea datelor.

1.5.4 Expresii algebrice relaționale

Compunând operatorii relaționali introduşi în cele trei subsecțiuni anterioare, se pot obține *expresii de algebră relațională* oricât de complexe. Făcând din nou apel la relațiile din figura 1, exemplificăm acest lucru cu relația din figura 10, descrisă de următoarea *expresie relațională SPJ* (*Selecție-Proiecție-Join*) :

$$\pi_{Titlu,Cantitate}(CĂRȚI \bowtie \sigma_{Cantitate \geq 1.000}(VÂNZĂRI))$$

Titlu	*Cantitate*
Convorbiri cu Regele Mihai I	2.890
Convorbiri cu Regele Mihai I	4.770

Figura 10: Rezultatul evaluării unei expresii algebrice relaționale SPJ

Evident că aceasta reprezintă formalizarea relațională a întrebării "Ce titluri de cărți au fost vândute şi în ce cantități, pentru cantități de cel puțin 1.000 exemplare?", precum şi răspunsul aferent.

1.5.4.1 Expresii implicând operatorii join şi proiecție

Expresiile relaționale care implică operatorii join şi proiecție se bucură de foarte multe proprietăți interesante. Evident că, de exemplu, operatorul join este asociativ şi comutativ. Ca atare, se poate scrie $\bowtie^{m}_{i=1} r_i = r_1 \bowtie r_2 \bowtie ... \bowtie r_m$, $\forall m \in \mathbf{N}^*$, iar joinul poate fi deci considerat şi ca un operator n-ar, $n \geq 1$, după cum urmează: date fiind $r_1(X_1)$, ..., $r_n(X_n)$, se poate defini: $\bowtie^{n}_{i=1} r_i = \{t[X_1 ... X_n] \mid \exists(t_1 \in r_1,..., t_n \in r_n), t[X_i] = t_i[X_i], \forall i=1,2,...,n\}$.

Mult mai interesante sunt însă proprietățile ce clarifică natura exactă a dualității existente între join şi proiecție, despre care am amintit deja în subsecțiunea anterioară.

> *Exemplul 9* Dacă proiectăm relația din figura 9 pe atributele *#Vânzare, Vânzător, Carte, Cantitate* obținem la loc relația *VÂNZĂRI*. Similar, dacă o proiectăm pe atributele *Carte, Autor, Titlu, Preț* obținem relația *CĂRȚI*. De notat că aceste egalități sunt adevărate doar în cazul joinurilor complete; în general, dacă joinul nu este complet, se obțin prin proiecție submulțimi ale relațiilor originale ce conțin toți tupli cu excepția celor încurcă-lume.

Lema următoare (a cărei demonstrație este lăsată în seama cititorului – vezi problema 8) arată că acest lucru nu este întâmplător:

LEMA 9 Fie relațiile $r_1(X_1)$, ..., $r_m(X_m)$, $m \geq 1$;

1. $\pi_{Xj}(\bowtie^{m}_{i=1} r_i) \subseteq r_j$, $\forall j$, $1 \leq j \leq m$.
2. $\pi_{Xj}(\bowtie^{m}_{i=1} r_i) = r_j$, $\forall j$, $1 \leq j \leq m \Leftrightarrow r_1$, ..., r_m au un join complet.

[12] În plus, evident, dacă $Y = Z = \varnothing$ şi $r_1 = r_2 = r$, atunci $r_1 \bowtie r_2 = r \bowtie r = r$, deci în aceste condiții joinul este idempotent.

3. $\pi_{X_j}(\bowtie_{k=1}^{m}(\pi_{X_k}(\bowtie_{i=1}^{m} r_i))) = \pi_{X_j}(\bowtie_{i=1}^{m} r_i)$, $\forall j, 1 \le j \le m$.

Primul punct al lemei stabileşte că proiectând rezultatul joinului a *m* relaţii peste schema oricărui operand se obţine o relaţie inclusă în acel operand. Al doilea punct precizează că incluziunea de mai sus devine egalitate numai în cazul joinurilor complete. Ultimul punct al lemei stabileşte că operatorul compus de la primul punct este *idempotent* (adică aplicarea sa repetată de oricâte ori produce exact acelaşi rezultat ca şi o singură aplicare).

Rezultate oarecum duale celor de mai sus pot fi demonstrate pentru expresiile ce compun în ordine inversă proiecţia şi joinul natural (vezi problema 9):

LEMA 10 Fie $r(X)$ o relaţie şi X_1, ..., X_m mulţimi de atribute a.î. $\cup_{j=1}^{m} X_j = X$.

1. $\bowtie_{i=1}^{m}(\pi_{X_i}(r)) \supseteq r$
2. $\bowtie_{k=1}^{m}(\pi_{X_k}(\bowtie_{i=1}^{m}(\pi_{X_i}(r)))) = \bowtie_{i=1}^{m}(\pi_{X_i}(r))$.

1.6 Descompuneri fără pierderi

Există şi o noţiune duală celei de join complet:

Definiţia 11 Se zice că *r* are o *descompunere fără pierderi* în raport cu X_1, ..., X_m dacă $\bowtie_{i=1}^{m}(\pi_{X_i}(r)) = r$, adică dacă *r* poate fi reconstruit exact din proiecţiile sale.

Figurile 11 şi 12 prezintă o asemenea decompoziţie, respectiv o *descompunere cu pierderi*[13]. Vom studia în detaliu descompunerile în capitolul 4.

r

Localitate	*Judeţ*	*Primar*	*Prefect*
Sibiu	Sibiu	Ionescu	Popescu
Oradea	Bihor	Georgescu	Popescu

$\pi_{Localitate,Primar,Prefect}(r)$

Localitate	*Primar*	*Prefect*
Sibiu	Ionescu	Popescu
Oradea	Georgescu	Popescu

$\pi_{Localitate,Judeţ}(r)$

Localitate	*Judeţ*
Sibiu	Sibiu
Oradea	Bihor

$\pi_{Localitate,Primar,Prefect}(r) \bowtie \pi_{Localitate,Judeţ}(r)$

Localitate	*Judeţ*	*Primar*	*Prefect*
Sibiu	Sibiu	Ionescu	Popescu
Oradea	Bihor	Georgescu	Popescu

Figura 11: Un exemplu de descompunere fără pierderi

[13] De remarcat că nu conţinutul, ci schema relaţiilor este răspunzătoare de faptul că o descompunere se face sau nu cu pierderi: în acest exemplu, datorită constrângerilor suplimentare induse de semantica funcţiilor f : *Localitate* → *Judeţ*, g : *Localitate* → *Primar*, h : *Primar* → *Localitate*, i : *Primar* → *Prefect*, j : *Judeţ* → *Prefect* şi k : *Prefect* → *Judeţ* (unde, desigur, $h = g^{-1}$, $k = j^{-1}$), schema *r* este supraîncărcată de fapt cu înţelesul a două relaţii (una strict de localităţi, iar cealaltă strict de judeţe). În plus, pentru o modelare corectă, acestei scheme ar trebui să-i fie evident impuse şi următoarele patru constrângeri de tip comutativitate de diagrame de funcţii (imposibil de exprimat în MRD):

$$j \circ f = i \circ g,\ f \circ h = k \circ i,\ i = j \circ f \circ h,\ f = k \circ i \circ g.$$

r

Localitate	*Judeţ*	*Primar*	*Prefect*
Sibiu	Sibiu	Ionescu	Popescu
Oradea	Bihor	Georgescu	Popescu

$\pi_{Localitate,Primar,Prefect}(r)$

Localitate	*Primar*	*Prefect*
Sibiu	Ionescu	Popescu
Oradea	Georgescu	Popescu

$\pi_{Judeţ,Prefect}(r)$

Judeţ	*Prefect*
Sibiu	Popescu
Bihor	Popescu

$\pi_{Localitate,Primar,Prefect}(r) \bowtie \pi_{Judeţ,Prefect}(r)$

Localitate	*Judeţ*	*Primar*	*Prefect*
Sibiu	Sibiu	Ionescu	Popescu
Sibiu	Bihor	Ionescu	Popescu
Oradea	Sibiu	Georgescu	Popescu
Oradea	Bihor	Georgescu	Popescu

Figura 12: Un exemplu de descompunere cu pierderi[14]

2. Constrângeri de integritate

În general, nu orice conținut al unei relații este acceptabil (valid) în interpretarea dorită pentru o bd, chiar dacă tupli ei au aritatea corectă iar valorile componentelor aparțin domeniilor corespunzătoare.

Exemplul 10 Fie relația din figura 13, memorând date despre studenții unei facultăți. Evident că este imposibil ca un student să aibă vârsta peste 140 de ani, după cum e interzis ca doi studenți să aibă acelaşi număr matricol!

ÎNSCRIERE

#Student	*NrMatricol*	*Student*	*AnNaştere*
1	32897	Ionescu Gabriel	1973
2	44135	Georgescu Radu	1856
3	44135	Vasilescu Mihai	1975

Figura 13: O relație cu valori imposibile

Ca atare, modelarea datelor permite impunerea unor condiții ce ar trebui respectate de toți tupli bd, astfel încât interpretarea lor să poată fi întotdeauna plauzibilă. Pentru aceasta, modelele datelor includ *constrângeri de integritate* ce trebuie satisfăcute de orice conținut al schemei bd corespunzătoare. În exemplul de mai sus, desigur, o asemenea constrângere ar fi, de exemplu: $\forall x$, $1966 \le AnNaştere(x) \le 1977$.

Constrângerile de integritate sunt ustensile pentru îmbunătățirea modelării datelor. Vom vedea în secțiunile următoare că unele dintre tipurile de constrângeri existente (chei, funcțional dependențe etc.) pot fi folosite în evaluarea calității schemelor relaționale şi în sugerarea diverselor descompuneri posibile ale acestora. Deoarece constrân-

[14] De remarcat că denumirea consacrată de descompunere cu/fără pierderi nu trebuie să inducă în eroare: de fapt, nu este niciodată vorba de pierderi de informație ci, dimpotrivă, de apariția nejustificată semantic a unor informații false!

gerile de integritate joacă un rol crucial în modelarea datelor suntem obligați să le studiem aprofundat.

Definiția 12 Se zice *constrângere de integritate* asupra schemei unei bd ***R*** orice funcție γ ce asociază fiecărui conținut de date ***r*** al ***R*** o valoare booleană γ (***r***). O bd ***r*** peste ***R*** *satisface* γ dacă γ (***r***) = *adevărat* și *violează* γ dacă γ (***r***) = *fals*. În primul caz se mai zice și că γ *ține* în ***r***.

Definiția 13 Fie o schemă ***R*** și Γ mulțimea de constrângeri asociată ei; se zice că sunt *consistente* (sau *valide*, sau *legale*) doar acele bd ale ***R*** ce satisfac Γ (adică satisfac toate constrângerile din Γ); altfel, ele se zic *inconsistente* (*invalide*, *ilegale*).

Fiecare constrângere formalizează deci o proprietate ce trebuie satisfăcută de orice conținut acceptabil al bd corespunzătoare. Există mai multe clase de constrângeri. Desigur că există atât notații (*sintaxa*) pentru a exprima constrângerile din fiecare clasă, precum și reguli ce definesc funcția booleană asociată fiecărui tip de constrângeri în parte (*semantica*). În majoritatea cazurilor, aceste funcții logice pot fi exprimate cu ajutorul propozițiilor închise ale unui calcul predicativ de ordin întâi.

În modelul relațional al datelor, clasele constrângerilor de integritate pot fi împărțite în două categorii:

1. *Constrângeri intrarelaționale* (ce implică doar o singură relație a schemei bd)
2. *Constrângeri interrelaționale* (ce implică schemele mai multor relații).

Când ne ocupăm de constrângeri intrarelaționale este uneori comod să ne referim doar la relația implicată, omițând orice referire nu doar la alte relații, ci chiar și la cea curentă (care este astfel presupusă implicit).

2.1 Chei

Definiția 14 O submulțime K a atributelor unei relații r se zice *cheie* a lui r dacă satisface următoarele două proprietăți:

- *Identificare unică*: r nu conține nici o pereche de tupli care să aibă aceleași valori pentru toate atributele din K (i.e. $(\forall t_1, t_2)$, $t_1[K] \neq t_2[K]$)
- *Minimalitate*: nici o submulțime proprie a lui K nu satisface proprietatea de identificare unică.

O mulțime K ce satisface doar proprietatea de identificare unică se zice *supercheie*.

Evident, orice cheie este supercheie, iar orice supercheie formată dintr-un singur atribut este cheie. Deoarece relațiile sunt mulțimi și nu pot deci conține duplicate, mulțimea tuturor atributelor oricărei relații este supercheie. Ca atare, orice relație are cel puțin o cheie.

Desigur că, în general, este posibil ca o relație să aibă mai multe chei. De exemplu, în figura 14 este evident (din regulile și practica domeniului aplicației modelate) că atributele *NrMatricol* și *CodNPersonal* trebuie să fie chei. Cheile sunt cele mai importante constrângeri de integritate din schema fiecărei relații. Identificarea și includerea tuturor cheilor în schema bd este crucială pentru modelarea oricărei aplicații.

Pe lângă cheile „semantice”, practica proiectării bd a evidențiat necesitatea includerii în schema oricărei relații a unei chei „sintactice”, cu rol de identificare unică minimală, numită „*cheie surogat*”; domeniul ei este mulțimea numerelor întregi.

STUDENȚI

#Student	*CodNPersonal*	*Prenume*	*Nume*	*NrMatricol*	*DataÎnscrierii*
1	2730812400217	Mihaela	Ionescu	5497	15.07.91
2	1751229310018	Gabriel	Ionescu	8970	15.07.93
3	1770107200412	Ioan	Gavrilă	9999	15.02.96
4	2790530100782	Irina	Dumitrescu	1020	15.07.95

5	1780714500356	Vlad	Georgescu	4850	15.07.94

Figura 14: O relație cu mai multe chei

Exemplul 11 Fie relația *STUDENȚI* din figura 14. Evident că nu pot exista două persoane având același cod numeric personal și deci, cu atât mai mult, orice doi studenți trebuie să aibă coduri personale distincte.

Similar, numărul matricol asignat fiecărui student este unic în cadrul universității. Rezultă că atât *CodNPersonal*, cât și *NrMatricol* trebuie să fie chei ale relației, iar aceste două constrângeri de integritate trebuie obligatoriu adăugate schemei bd.

Desigur că, de exemplu, este posibil ca, în general, să nu existe studenți înscriși simultan și având exact același prenume și același nume; aceasta ar putea duce la concluzia că și mulțimea de atribute K = {*Prenume*, *Nume*, *DataÎnscrierii*} constituie o supercheie (și, deoarece nici una dintre submulțimile ei proprii nu ar putea identifica unic tupli relației, K ar fi cheie). Impunerea acestei constrângeri însă ar fi nu doar nenaturală, ci și inutil de restrictivă, nepermițând niciodată memorarea în bd a unor asemenea cazuri care, totuși, ar putea apărea la un moment dat.

De observat și că, întâmplător, și valorile *Prenume* sunt distincte; evident însă că, în general, acest lucru nu poate fi adevărat.

În sfârșit, cheia surogat *#Student* este prin definiție cheie.

În concluzie, niciodată nu trebuie asertată vreo cheie care nu trebuie să fie cheie, pentru că astfel s-ar interzice conținuturi plauzibile, dar nici nu trebuie vreodată ignorată vreo cheie, căci aceasta ar avea ca efect posibilitatea memorării de conținuturi implauzibile.

2.2 Valori nule

Exemplele de până acum sugerează faptul că orice relație poate fi văzută ca o reprezentare a unei părți a informației disponibile la un moment dat într-o anume aplicație de interes. Fiind însă posibil ca nu toate datele corespunzătoare unei bd să fie cunoscute în orice moment, este rezonabil să admitem ca relațiile să poată conține și valori nespecificate. Pentru aceasta, domeniile relațiilor sunt extinse prin adăugarea unei așa-numite *valori nule*, care se notează cu simbolul $\perp$ (sau ϕ sau NULL sau nimic).[15]

De remarcat că dacă ar fi permise valori nule și pentru chei, atunci desigur că ar trebui relaxată definiția identificării unice (în sensul restrângerii ei la tuplii ce nu au valori nule în atributele implicate în chei). Lucrul acesta ar conduce însă la unele situații nedorite. Exemplul 12 sugerează că e preferabil să fie restrânsă permisiunea apariției unor asemenea cazuri și că adăugarea cheilor surogat este benefică.

STUDENȚI

CodPersonal	*Prenume*	*Nume*	*NrMatricol*	*DataÎnscrierii*
$\perp$	Victor	Popescu	$\perp$	15.07.91
1751229310018	Gabriel	Ionescu	$\perp$	15.07.93
$\perp$	Irina	Dumitrescu	9999	15.02.96
2790530100782	Irina	Dumitrescu	$\perp$	$\perp$

Figura 15: O relație cu valori nule pentru chei

Exemplul 12 Fie bd din exemplul 11, dar cu conținutul din figura 15, unde primul tuplu al relației are valori nule pentru ambele chei și, în consecință, nu poate fi identificat în nici un mod. Dacă, de exemplu, am vrea să adăugăm

[15] Pentru a nu exista pericolul de confuzie cu denumirea cifrei 0, vom folosi "*valori non-nule*" pentru negația valorilor nule (și nu "*valori nenule*", care înseamnă non-zero).

relației încă un tuplu cu *Prenume* = "Victor" şi *Nume* = "Popescu", n-am putea şti dacă acesta s-ar referi la acelaşi student ca şi primul tuplu (ceea ce ar fi o eroare) sau nu.

O problemă similară apare şi pentru ultimii doi tupli ai acestei relații: deşi fiecare dintre ei are o valoare non-nulă pentru câte una dintre chei, nu putem şti dacă ei se referă la studenți diferiți, aşa cum ar trebui să se întâmple întotdeauna.

Raportul *ANSI* [17] distinge între 14 tipuri de valori nule: 11 dintre acestea se datorează însă considerentelor de implementare sau sunt particularizări ale celorlalte 3 pentru diverse subuniversuri de interes în modelare; aceste trei tipuri generale sunt următoarele: *valoare inexistentă* (*atribut inaplicabil*), *valoare* (*temporar*) *necunoscută* şi disjuncția logică a acestora, zisă *valoare lipsită de informație*[16] (i.e. nu se ştie dacă această valoare există sau nu, iar în caz că ea ar exista nu se ştie nimic despre natura ei).

Exemplul 13 Fie bd din figura 16. Mărturiile istorice nu lasă loc îndoielilor cu privire la erudiția domnitorului Dimitrie Cantemir, care vorbea fluent, printre altele, atât franceza, engleza, cât şi rusa. Cum însă nici una dintre aceste trei limbi nu exista în zorii primului mileniu creştin, este evident că cele trei valori nule din dreptul lui Decebal au semnificația de valoare inexistentă.

Cu toții l-am putut auzi pe M.S. Regele Mihai I vorbind engleză şi franceză, dar niciodată rusă; cum n-am întâlnit nici o referire în literatură în legătură cu acest subiect, valoarea nulă corespunzătoare este de tip necunoscută (temporar, deoarece chestiunea s-ar putea încă lămuri).

În schimb, deşi la cumpăna dintre secolele XVII şi XVIII toate aceste trei limbi moderne erau de mult în uz, iar Constantin Brâncoveanu a fost unul dintre cei mai luminați domni români (şi nu numai), istoria nici nu ne pomeneşte nimic despre acest subiect şi nici nu se întrevăd şanse pentru viitor în acest sens; ca atare, valorile nule din dreptul său sunt lipsite de informație.

POLIGLOȚI

#Personalitate	*Personalitate*	*Engleză*	*Franceză*	*Rusă*
1	Decebal	ϕ	ϕ	ϕ
2	Constantin Brâncoveanu	ϕ	ϕ	ϕ
3	Dimitrie Cantemir	Da	Da	Da
4	Mihai I	Da	Da	ϕ

Figura 16: O relație cu toate cele trei tipuri de valori nule

Pentru a evita notații excesiv de complicate, nu vom prezenta aici definiții formale pentru aceste trei tipuri de valori nule. Schițăm totuşi în continuare formalizarea lor în logica de ordin întâi (pentru cazul simplu în care un tuplu conține o singură valoare nulă): date fiind o schemă de relație oarecare $R(A_1, A_2, \ldots, A_n)$ şi predicatul R asociat ei, pentru fiecare tuplu t al unui conținut r al acesteia se obține propoziția:

$$R(t[A_1], t[A_2], \ldots, t[A_n]).$$

Valorile temporar necunoscute pot fi tratate cu ajutorul cuantificatorului existențial: dacă t are o asemenea valoare nulă pentru atributul A_k, formula corespunzătoare devine $\exists x(R(t[A_1], t[A_2], \ldots, t[A_{k-1}], x, t[A_{k+1}], \ldots, t[A_n]))$. În cazul exemplului de mai sus: $\exists x$ (*Poliglоți*(Mihai I, *Da, Da, x*)).

Valorile inexistente pot fi, evident, tratate simetric folosind formule negate de tipul: $\neg\exists x(R(t[A_1], t[A_2], \ldots, t[A_{k-1}], x, t[A_{k+1}], \ldots, t[A_n]))$; acestea însă ascund într-un anumit sens informația pozitivă din cele n-1 valori precizate; de aceea, o asemenea formulă trebuie cuplată cu una pozitivă peste un predicat (n-1)-ar, provenită tot din R dar fără A_k:

[16] De notat că acest tip de nul este folositor în special în SGBD-urile ce permit modificarea schemelor relațiilor fără oprirea operațiilor de actualizare curente.

$$R_{-k}(t[A_1], t[A_2], \ldots, t[A_{k-1}], t[A_{k+1}], \ldots, t[A_n]).$$

De exemplu, pentru relația din figura 16, ar fi necesară următoarea propoziție:

$$\neg\exists x(\neg\exists y(\neg\exists z(\textit{Poligloți}(\text{Decebal}, x, y, z)))) \wedge \textit{Poligloți}_{-2\text{-}3\text{-}4}(\text{Decebal})$$

În sfârșit, cel mai indicat mod de a exprima lipsa completă de informație despre valoarea unui atribut este folosirea unui predicat care să nu-l menționeze: $R_{-k}(t[A_1], ..., t[A_{k-1}], t[A_{k+1}], ..., t[A_n])$. Corespunzător, pentru exemplul de mai sus:

$$\textit{Poligloți}_{-2\text{-}3\text{-}4}(\text{Constantin Brâncoveanu})$$

Necesitatea celei de-a doua propoziții (pozitive) pentru cazul valorilor nule inexistente sugerează faptul că tupli relaționali încapsulează mai multă informație decât cea oferită de propozițiile bazate pe predicatul R asociat relației respective: de fapt, pe lângă informațiile asociate întregii mulțimi de atribute X a schemei $R(X)$, orice asemenea tuplu conține și informații despre submulțimile lui X. Formalizarea acestei constatări impune, evident, introducerea mai multor simboli predicativi per relație, i.e. câte unul pentru fiecare asemenea submulțime de interes a X. Se pot astfel asocia relației formule cuantificate generale ce statuează implicația între fiecare predicat m-ar (corespunzând unei submulțimi oarecare $Y \subseteq X$), $1 \le m \le n$, și toate predicatele (m-1)-are (corespunzând submulțimilor lui Y având cardinalitate (m-1)):

$$\forall a_1(...\forall a_n(R(a_1, ..., a_n) \rightarrow R_{-k}(a_1, ..., a_{k-1}, a_{k+1}, ..., a_n)) ...).$$

Lăsăm cititorului ca exercițiu (vezi problema 2) demonstrarea faptului că propoziția corespunzând valorilor nule lipsite de informație ($R_{-k}(t[A_1], ..., t[A_{k-1}], t[A_{k+1}], ..., t[A_n])$) este echivalentă cu disjuncția formulelor asociate celorlalte două tipuri de valori nule:

$$\exists x(R(t[A_1], ..., t[A_{k-1}], x, t[A_{k+1}], ..., t[A_n])) \vee$$
$$\vee\ (\neg\exists x(R(t[A_1], ..., t[A_{k-1}], x, t[A_{k+1}], ..., t[A_n])) \wedge R_{-k}(t[A_1], ..., t[A_{k-1}], t[A_{k+1}], ..., t[A_n])).$$

Înainte de a încheia această subsecțiune, merită semnalată și problema *„nulurilor murdare”*, adică a șirurilor de caractere care deși nu sunt nule, par a fi astfel, deoarece sunt alcătuite exclusiv din caractere netipăribile (de exemplu: spațiu, tab, CR/LF etc.). Unele SGBD (de exemplu *Access*, dar nu și *MS SQL Server*!) le consideră drept *„șiruri de lungime nulă”* și le pot interzice; evident, dacă o asemenea facilitate există, ea trebuie folosită, iar în caz contrar, ea merită adăugată explicit aplicațiilor (în trăgaciuri).

2.3 Chei primare

Cea mai simplă soluție ce permite identificarea unică a fiecărui tuplu al unei relații este aceea de a alege una dintre cheile relației (care se zice atunci *cheie primară*) și a nu permite pentru ea, niciodată, nici o valoare nulă. Pentru toate celelalte chei ale relației însă, valorile nule pot fi permise fără restricții. Cheia primară este scoasă în evidență prin sublinierea atributelor ce o compun, ca în exemplul din figura 17, unde *#Student* a fost ales drept cheie primară. Cea mai bună practică este aceea de a alege drept primară cheia surogat. Majoritatea sistemelor de gestiune a bd (SGBD) oferă pentru ele o mulțime „cu autonumărare” („autonumber” sau „counter”) ale cărei valori sunt automat asignate de sistem la crearea fiecărei noi linii (fie incremental, fie aleatoriu).

STUDENȚI

#Student	*CodPersonal*	*Prenume*	*Nume*	*NrMatricol*	*DataÎnscrierii*
1	⊥	Victor	Popescu	7432	15.07.91
2	⊥	Irina	⊥	9999	15.02.96
3	2790530100782	Irina	Dumitrescu	9305	⊥

Figura 17: O relație cu valori nule, dar nu și în cheia primară

În concluzie, relații precum cea din figura 15 nu sunt permise, în timp ce relația din figura 17 sau cele din figurile până la 14 inclusiv sunt valide. În general însă, dacă nu este explicit menționat altfel, presupunem în cele ce urmează că relațiile au doar valori non-nule.

2.4 Constrângeri primitive

Definiția 15 O constrângere se zice *primitivă* dacă este de tip dependență cheie sau constrângere de domeniu.

2.4.1 Dependențe cheie

Pe baza subsecțiunilor anterioare, putem acum introduce formal cea mai uzuală constrângere intrarelațională în bd şi anume dependența cheie:

Definiția 16 Dată fiind o schemă relațională $R(U)$ şi o submulțime $K \subseteq U$, o *dependență cheie* este exprimată prin *cheie*(K) şi este satisfăcută de o relație r dacă r nu conține nici o pereche de tupli distincți t_1, t_2 care să aibă aceleaşi valori peste K (adică dacă $t_1[K] \neq t_2[K]$, $\forall t_1, t_2 \in r$).

Rezultă că dependența de cheie *cheie*(K) este satisfăcută de un conținut de relație r $\Leftrightarrow$ K este o supercheie a lui r (nu neapărat minimală).

2.4.2 Constrângeri de domeniu

Vom vedea (în secțiunea 7.4) cu ocazia definirii formei normale domenii-chei că, în anumite condiții, şi mulțimile de valori ale atributelor (deci "domeniile" în jargon relațional) joacă un rol similar cu cel al constrângerilor.

Definiția 17 Definițiile domeniilor relaționale se zic *constrângeri de domeniu*, iar mulțimea de constrângeri asociată oricărei scheme de relație conține, pentru fiecare domeniu al relației, câte o asemenea constrângere (care are forma $dom(A) = D_A$, unde D_A este mulțimea tuturor valorilor admisibile pentru atributul A).

De exemplu: *dom*(*Sex*) = {'M','F'}; *dom*(*Validat*) = {Da,Nu}; *dom*(*AnNaştere*) = = $\{n \in \text{NAT}(4) \mid 1920 \leq n \leq 1982\}$; *dom*(*Nume*) = CHAR(32) etc.

2.5 Constrângeri tuplu

O altă clasă interesantă de constrângeri, care precizează restricții asupra unor tupli individuali, este formată din *constrângerile tuplu*. Ele sunt exprimate cu ajutorul formulelor propoziționale ce au fost introduse la 1.5.1 mai sus, cu restricția că există o unică variabilă, implicit cuantificată existențial (care, de obicei, se şi omite, deoarece nu există nici un pericol de ambiguitate). Funcția booleană asociată unei asemenea constrângeri poate fi deci reprezentată de un predicat cu o singură variabilă cuantificată universal per relație (variabilă ce semnifică tuplii ce trebuie să satisfacă constrângerea în cadrul relației respective).

Exemplul 14 Fie relația (din figura 18) *SALARII* (*#Angajat, Angajat, SalariuBrut, Reținere, SalariuNet*). O constrângere evidentă ar fi aceea că, pentru orice angajat, salariul net trebuie să fie egal cu diferența dintre salariul brut şi rețineri. În plus, s-ar putea adăuga condiția ca salariul net să fie întotdeauna strict mai mare ca zero. Putem exprima formal aceste cerințe cu ajutorul constrângerii tuplu: $SalariuNet = SalariuBrut - Rețineri \wedge SalariuNet > 0$.

Predicatul asociat ar fi, evident, următorul:

$(\forall t \in salarii)((t[SalariuNet]=t[SalariuBrut] - t[Retineri]) \wedge (t[SalariuNet] >0))$.

Trivial, relația din figura 18 satisface această constrângere, în timp ce relația din figura 19 nu o satisface, din cauza celui de-al doilea tuplu al ei.

SALARII

#Angajat	Angajat	SalariuBrut	Reținere	SalariuNet
1	Ion	500.000	100.000	400.000
2	Robert	650.000	200.000	450.000
3	Mihai	480.000	95.000	385.000

Figura 18: O relație satisfăcând constrângerea tuplu din exemplul 14

SALARII

#Angajat	Angajat	SalariuBrut	Reținere	SalariuNet
1	Ion	500.000	100.000	400.000
2	Robert	650.000	200.000	550.000
3	Mihai	480.000	95.000	385.000

Figura 19: O relație violând constrângerea tuplu din exemplul 14

2.6 Dependențe de incluziune. Chei străine

Poate cel mai interesant tip de constrângeri (în general interrelaționale, dar putând fi şi intrarelaționale) este constituit de *dependențele de incluziune* (*DIN*). Acestea sunt folosite în modelul relațional pentru a exprima incluziunea între proiecții ale relațiilor. Pentru simplitate, prezentăm deocamdată doar un caz special de asemenea dependențe.

Definiția 18 Dată fiind o schemă $\boldsymbol{R} = \{R_1(X_1), R_2(X_2), ..., R_n(X_n)\}$, o *dependență tipată de incluziune* (*DTIN*) se notează $R_i[Y] \supseteq R_j[Y]$, unde $Y \subseteq X_i \cap X_j$, $1 \le i, j \le n$. Ea este satisfăcută de o bd $\boldsymbol{r} = \{r_1, r_2, ..., r_n\}$ a lui $\boldsymbol{R}$ dacă $\pi_Y(r_i) \supseteq \pi_Y(r_j)$.

Exemplul 15 Fie o bd având schema $\boldsymbol{R} = \{R_1(\underline{\#Autor}, Prenume, Nume), R_2(\underline{\#Carte}, Carte, Autor)\}$, în care prima relație memorează mulțimea autori-lor, iar cea de-a doua memorează mulțimea cărților (de interes) scrise de aceş-tia. Este naturală impunerea cerinței ca, în a doua relație, pentru a se putea menționa, tipări, clasifica etc. autorul fiecărei cărți, acesta să figureze printre cei din prima relație.

Evident că *DIN* $R_1(\#Autor) \supseteq R_2(Autor)$ exprimă exact această constrângere. Se poate uşor verifica că bd din figura 20 satisface această *DTIN*, în timp ce bd din figura 21 nu o satisface.

r_1

#Autor	Prenume	Nume
1	Lucian	Blaga
2	Emil	Cioran
3	Mircea	Ciobanu
4	Mircea	Eliade

r_2

#Carte	Carte	Autor
1	Eonul dogmatic	1
2	Cunoaşterea luciferică	1
3	Cenzura transcendentă	1
4	Noi convorbiri cu Regele Mihai I al României	3
5	Regele Mihai şi exilul românesc	3

6	Erotica mistică în Bengal	4

Figura 20: O bază de date ce satisface DTIN R_1(#Autor) $\supseteq$ R_2(Autor)

r_1

#Autor	Prenume	Nume
1	Lucian	Blaga
2	Emil	Cioran
3	Mircea	Ciobanu

r_2

#Carte	Carte	Autor
1	Eonul dogmatic	1
2	Cunoaşterea luciferică	1
3	Cenzura transcendentă	1
4	Noi convorbiri cu Regele Mihai I al României	3
5	Regele Mihai şi exilul românesc	3
6	Erotica mistică în Bengal	4

Figura 21: O bază de date ce violează DTIN R_1(#Autor) $\supseteq$ R_2(Autor)

Partea dreaptă a dependențelor de incluziune se zice „cheie străină”:

Definiția 19 O submulțime X a atributelor unei relații r_1 se zice *cheie străină* (a lui r_1, din r_2), dacă mulțimea valorilor tuplilor din r_1 pentru X este inclusă în mulțimea valorilor pentru cheia (primară sau nu a) unei relații r_2.

Cheile străine furnizează principalul mijloc de a lega între ei tupli din relații diferite. Evident că, deşi se pot oricând alege orice alte chei ale relației din partea stângă a unei *DIN*, cea mai bună soluție este întotdeauna alegerea cheii surogat corespunzătoare (atât ca eficiență, cât şi ca eleganță).

> *Exemplul 16* Fie bd din figura 1, unde cheile primare sunt *#Autor*, *#Vânzător*, *#Carte* şi *#Vânzare*; evident mai există următoarele chei: {*Prenume, Nume*}, *Vânzător*, {*Autor*, *Titlu*} şi {*Vânzător*, *Carte*}. În relația *VÂNZĂRI*, atributul *Carte* este cheie străină (căci valorile sale trebuie să fie mereu incluse printre cele ale cheii primare *#Carte* din relația *CĂRȚI*). Acest lucru nu este, desigur, întâmplător, căci tocmai (şi numai!) valorile acestui atribut identifică unic în *VÂNZĂRI* titlurile de cărți vândute de fiecare vânzător în parte. Similar, evident, acelaşi lucru este valabil şi pentru *Vânzător*, ca şi pentru *Autor* din relația *CĂRȚI*.

2.7 Funcțional dependențe

Definiția 20 Fie două atribute A şi B ale unei relații r; B se zice *funcțional dependent* de A (sau că A *determină funcțional* pe B) $\Leftrightarrow$ oricărei valori a lui A îi corespunde o unică valoare a lui B. Simbolic, acest lucru se notează $A \rightarrow B$ (sau $B \leftarrow A$). Orice asemenea aserțiune se numeşte *funcțional dependență* (sau simplu, *dependență*, iar prescurtat: *FD*).

Această noțiune se poate extinde şi la mulțimi de atribute: fie X şi Y două asemenea submulțimi nevide X şi Y ale U[17]*;* spunem că Y este funcțional dependent de X în r dacă oricărui (sub)tuplu având valoarea x pentru atributele X din r îi corespunde în r un

[17] Evident că $\varnothing \rightarrow \varnothing$ şi $X \rightarrow \varnothing$ sunt *FD triviale* (i.e. țin în orice conținut), complet neinteresante, în timp ce *FD* de tipul $\varnothing \rightarrow Y$, care implică faptul că Y este o funcție constantă, ar trebui factorizată afară din orice relație, eliminând Y, devenit complet neinteresant, din schema bd.

singur tuplu având valoarea *y* pentru atributele *Y*. *X* se zice *partea stângă*, iar *Y partea dreaptă* a *FD*.

Definiția 21 Formal, date fiind $XY \subseteq U$, $r(U)$ *satisface FD* $X \to Y$ (sau, echivalent, *FD* $X \to Y$ *ține* în *r*), dacă: $(\forall t_1,t_2 \in r)(t_1[X] = t_2[X]) \Rightarrow (t_1[Y] = t_2[Y])$.

Deşi definițiile de mai sus[18] sunt formulate în termenii conținutului relațiilor, orice *FD* corespunde de fapt unei constrângeri de integritate asupra schemei relației; aceasta înseamnă că ea constituie aserțiunea formală a unei proprietăți ce trebuie să fie validă pentru orice conținut posibil al schemei relaționale.

Împreună cu dependențele cheie, clasa de constrângeri cea mai studiată este cea a funcțional dependențelor: pe baza *FD* se definesc mai multe *forme normale*, dintre care introducem pe cea mai importantă (*FNBC*) în cele ce urmează (vezi secțiunea 3.1). Evident că dacă $XY = U$, $X \to Y$ ține în $r \Leftrightarrow$ *cheie*(X) ține în *r* (vezi problema 83). Rezultă că dependențele de cheie sunt un caz particular al *FD*.

Exemplul 17 Relația din figura 22 satisface dependența de cheie *cheie*(*Tip, NrCurent*) şi *FD Localitate* $\to$ *Primar*, dar nu satisface nici dependența *cheie* (*Tip*), nici *FD Localitate* $\to$ *NrCurent*.

Se poate demonstra uşor (vezi problema 10) următorul rezultat, fundamental în abordarea relațională:

Propoziția 22 Fie relația $r(U)$ şi X_1, X_2, $X \subset U$ astfel încât $X_1X_2 = U \wedge X = X_1 \cap X_2$; descompunerea $r(U)$ peste X_1, X_2 este fără pierderi dacă $r(U)$ satisface cel puțin una dintre *FD* $X \to X_1$ sau $X \to X_2$.

2.8 Constrângeri implicate şi triviale. Închiderea mulțimilor de constrângeri

Fie o schemă de bd generică ***R***; dată fiind o mulțime de constrângeri Γ, se întâmplă cel mai adesea ca, dacă o bd peste ***R*** satisface Γ, atunci ea să mai satisfacă şi alte constrângeri (ce nu aparțin Γ).

Definiția 23 Se zice că Γ *implică* o constrângere γ, dacă γ ține în toate bd ce satisfac Γ.

Un caz special de constrângeri îl reprezintă cele implicate de mulțimea vidă (adică acele constrângeri ce țin în orice conținut al bd):

Definiția 24 Dacă $\Gamma = \varnothing$ şi Γ implică γ, atunci γ se zice *trivială*.

Exemplul 18 Fie schema bd din exemplul 14; evident că orice relație satisface constrângerea: (*SalariuNet* = *SalariuBrut*) $\vee$ (*SalariuNet* $\neq$ *SalariuBrut*) care este, deci, trivială. Similar, sunt triviale şi constrângerile tuplu:

SalariuBrut = *SalariuNet* + *Rețineri*, *SalariuBrut* > *Rețineri*.

Următorul fapt, a cărui demonstrație e banală, este folosit în multe dintre rezultatele teoretice ce urmează (vezi problema 3):

Propoziția 25 O mulțime de constrângeri Γ nu implică o constrângere $\gamma \Leftrightarrow$ există un conținut al bd ce satisface Γ, dar nu satisface γ.

Definiția 26 Date fiind o clasă oarecare de constrângeri C şi o submulțime Γ a ei, submulțimea $(\Gamma)_C^+ \subset C$ a tuturor constrângerilor din C implicate de Γ se zice *închiderea* lui Γ în raport cu C.

Trebuie subliniat faptul că închiderea unei mulțimi de constrângeri Γ în raport cu o altă clasă C' este în general diferită de închiderea sa în raport cu C. Dacă însă contextul este neambiguu, atunci C poate fi omis, iar închiderea Γ se notează simplificat prin Γ^+ (de exemplu, cel mai adesea, C este clasa tuturor funcțional dependențelor).

[18] O definiție mult mai elegantă pentru *FD* face apel la relația de echivalență „nucleu" (vezi [211]).

Propoziția următoare precizează cele mai importante proprietăți evidente ale închiderilor (demonstrația ei este propusă cititorului de problema 4):

Propoziția 27 Fie Γ şi Δ două mulțimi oarecari de constrângeri;

(*i*) $\Gamma \subseteq \Gamma^+$

(orice mulțime de constrângeri e inclusă în închiderea ei)

(*ii*) $(\Gamma^+)^+ = \Gamma^+$

(operatorul de închidere este idempotent)

(*iii*) $\Gamma \subseteq \Delta \Rightarrow \Gamma^+ \subseteq \Delta^+$

(operatorul de închidere este monoton)

(*iv*) $\Gamma^+ \cup \Delta^+ \neq (\Gamma \cup \Delta)^+$ (în general, operatorul de închidere nu este aditiv)

Definiția 28 O mulțime de constrângeri se zice *închisă în raport cu implicația* dacă ea este egală cu propria-i închidere: $\Gamma = \Gamma^+$.

Definiția 29 Două mulțimi de constrângeri Γ_1 şi Γ_2 se zic *echivalente* (sau că se *acoperă* una pe cealaltă) dacă au închiderile egale: $\Gamma_1^+ = \Gamma_2^+$.

Definiția 30 Constrângerea γ se zice *redundantă* dacă Γ şi $\Gamma - \{\gamma\}$ sunt echivalente.

Definiția 31 O *acoperire* se zice *minimală* dacă nu conține nici o constrângere redundantă.

2.9 Problema implicației

Testarea echivalenței şi redundanței constrângerilor se poate reduce la aşa-zisa *problemă a implicației*: *date fiind Γ şi γ oarecari, implică Γ pe γ*? Această problemă poate fi rezolvată eficient pentru unele clase de constrângeri (liniar, de exemplu, pentru *FD*), foarte ineficient pentru altele (e.g. *FD* şi *DMV* sau *DJ*, vezi capitolul 5) şi este nedecidabilă în multe cazuri (exemplu: *FD* şi *DIN*, vezi 4.7).

Studiul problemei implicației se face folosind una dintre următoarele două tehnici:

- *reguli de derivare* sau
- *tablouri* şi *algoritmul de vânare*.

De exemplu, problema implicației pentru *FD* şi *DIN* este studiată pe îndelete în capitolul 4.

2.9.1 Reguli de derivare a constrângerilor

Informal, dată fiind o mulțime Γ de constrângeri ale unei clase C oarecare fixată, *regulile de derivare* (*inferență*) generează o mulțime ce conține Γ, precum şi, în general, alte constrângeri (zise *derivabile*). Deoarece interesează doar generarea constrângerilor implicate de orice asemenea mulțime Γ, este nevoie numai de reguli pentru derivarea constrângerilor din $(\Gamma)_C^+$. Orice regulă având această proprietate se zice *solidă*.

În plus, pentru a putea calcula închiderea, este obligatoriu ca mulțimea regulilor de inferență să permită derivarea tuturor constrângerilor aparținând închiderii respective.

Definiția 32 O mulțime de reguli se zice *completă* în raport cu o clasă dată C, dacă ea permite derivarea $(\Gamma)_C^+$ pentru orice submulțime $\Gamma \subseteq C$.

Regulile de derivare sunt interesante însă şi în sine: existența unei mulțimi solide şi complete de reguli simple este nu doar un rezultat elegant (eventual permițând rezolvarea eficientă a problemei implicației pentru acea clasă), ci şi un instrument ce permite o înțelegere mai profundă a problemei implicației (deoarece, pentru majoritatea algoritmilor de testare a implicației constrângerilor, se foloseşte completitudinea regulilor pentru a demonstra completitudinea algoritmilor).

Capitolul 4 prezintă reguli de derivare atât pentru *FD*, cât şi pentru *DIN*.

2.9.2 Tablouri şi algoritmi de vânare

Intuitiv, un *tablou* este o relație ale cărei valori sunt nu doar constante, ci şi variabile. Un *algoritm de vânare* (“*chase*”) porneşte de la un tablou T şi o mulțime de constrângeri Γ, cu scopul de a găsi un tablou care satisface Γ şi este cât mai asemănător cu putință cu T.

În esență, un asemenea algoritm încearcă să construiască o relație contraexemplu (a faptului că orice mulțime de constrângeri este inclusă în închiderea sa) cu variabile, forțând-o să satisfacă toate constrângerile din Γ; o constrângere dată γ este implicată de Γ dacă şi numai dacă şi ea ține în tabloul astfel rezultat.

Această metodă este interesantă şi fiindcă este parametrizabilă (într-o anumită măsură) în raport cu clasa C a constrângerilor considerate.

Capitolul 4 prezintă tablouri şi algoritmi de vânare atât pentru *FD*, cât şi pentru *DIN*.

3. Forma normală Boyce-Codd

Exemplul 19

Fie bd electorală referitoare la alegerile locale (deja folosită în figurile 11 şi 12) şi fie relaţia *ALEGERI* (*Tip, NrCurent, Judeţ, Localitate, Circumscripţie, Primar, Prefect*) din figura 22. Pe lângă împărţirea teritorial-administrativă uzuală în localităţi şi judeţe, pentru alegeri ţara este împărţită în circumscripţii electorale; niciodată o circumscripţie electorală nu reuneşte localităţi din judeţe diferite, invers însă fiind posibile cazuri în care există mai multe circumscripţii într-un acelaşi judeţ (de exemplu, Bucureşti); fiecare localitate aparţine unui judeţ; orice circumscripţie electorală este constituită dintr-o mulţime de centre de votare; pentru fiecare centru de votare interesează şi tipul său (de exemplu: municipal, urban, rural, UM MApN, UM MI, vapoare civile, vapoare militare, ambasade etc.); ca atare, centrele sunt unic identificate de perechea de atribute {*Tip, NrCurent*}, care a fost aleasă drept cheie primară a relaţiei *ALEGERI*; valorile atributului *NrCurent* sunt unice în cadrul fiecărui tip în parte, dar nu sunt unice pe ţară; fiecare centru de votare este organizat într-o localitate; fiecare localitate are un primar; fiecare judeţ are un prefect.

Inspectând această relaţie este evident că ea prezintă foarte multe redundanţe, precum, de exemplu:

- judeţul şi prefectul se repetă pentru toate localităţile unui acelaşi judeţ
- localitatea şi primarul se repetă pentru toate centrele de votare dintr-o aceeaşi localitate.

ALEGERI ({*Tip,NrCrt*})

Tip	*NrCrt*	*Judeţ*	*Localitate*	*Circumscripţie*	*Primar*	*Prefect*
Urban	01	Alba	Sebeş	1	Pop	Văcărescu
Rural	02	Alba	Lancrăm	1	Octavian	Văcărescu
suburban	01	Călăraşi	Oltenița	3	Ilici	Serghei
municipal	01	Vaslui	Vaslui	6	Vasilescu	Văcărescu
municipal	02	Vaslui	Vaslui	9	Vasilescu	Văcărescu
UM MApN	01	Alba	Alba-Iulia	8	Ionescu	Văcărescu

Figura 22: O bd cu o singură relaţie pentru aplicaţia "alegeri locale"

3.1 Anomalii de actualizare a datelor

Foarte multe probleme apar atunci când trebuie actualizate datele acestei relaţii. Prima din ele, zisă *anomalie de modificare*, este datorată tocmai redundanţelor semnalate mai sus: de exemplu, dacă se schimbă primarul unei localităţi, trebuie actualizaţi toţi tuplii referind acea localitate (deci datele tuturor centrelor de votare din acea localitate), în loc de un singur tuplu, cum ar fi firesc. Aceeaşi anomalie se observă şi pentru cazul în care s-ar schimba prefectul unui judeţ sau apartenenţa unei localităţi la un judeţ etc.

Există încă alte două tipuri de anomalii, zise de inserţie, respectiv de ştergere, după cum rezultă, de exemplu, din următoarele două consideraţii:

- faptul că, dacă nu se cunoaşte tipul sau numărul curent al unui nou centru de votare, nu se pot insera în relaţie informaţii despre o nouă circumscripţie sau localitate sau despre un nou judeţ se zice *anomalie de inserţie*, iar
- faptul că, dacă se şterg din bd toate centrele de votare dintr-o localitate, atunci se pierd şi informaţiile despre acea localitate, se zice *anomalie de ştergere*.

În general, prezența anomaliilor de tipul celor semnalate mai sus (şi care sunt desemnate generic prin "*anomalii de actualizare a datelor*") se datorează supraîncărcării unei singure relații cu prea multe informații despre structura aplicației modelate.

În particular, revenind la exemplul de mai sus, conținutul relației *ALEGERI* încearcă să țină cont de toate informațiile furnizate de primul paragraf al acestei secțiuni, în timp ce schema sa memorează doar faptul că perechea (*Tip, NrCurent*) este cheie.

Pentru a defini formal anomaliile de actualizare, este nevoie de introducerea în prealabil a noțiunii de *compatibilitate* (*semantică*) între conținutul (instanța) unei relații şi tupli. Sintactic, orice tuplu al unei relații este trivial compatibil cu relația respectivă (din moment ce îi aparține, tuplul are exact aceleaşi coloane cu relația). Pentru a insera un nou tuplu într-o relație, este iar trivială cerința compatibilității sale sintactice cu relația. Evident însă că această compatibilitate nu este suficientă în nici unul din cele două cazuri de mai sus, deoarece atât ştergerea unui tuplu dintr-o relație, cât şi inserția unuia nou pot viola constrângerile de integritate asociate relației.

Definiția 32 Fie r o relație peste schema $R(U)$ şi t un tuplu peste U, $t \notin r$; t se zice *compatibil* (*semantic*) cu r dacă el nu violează constrângerile primitive:

1. $\forall A \in U$, $t[A] \in dom(A)$ (t nu violează constrângerile de domeniu);
2. $\forall K$ cheie a R şi $\forall t' \in r$, $t[K] \neq t'[K]$ (t nu violează dependențele cheie).

Deci t este compatibil cu r, dacă $r \cup \{t\}$ nu violează constrângerile primitive.

Definiția 33 O schemă de relație R se zice a avea o *anomalie de inserție* dacă există:

1. un conținut valid r al R şi
2. un tuplu t compatibil cu R

astfel încât $r \cup \{t\}$ nu este un conținut valid al R.

Definiția 34 Similar, o schemă de relație R se zice a avea o *anomalie de ştergere* dacă există:

1. un conținut valid r al R şi
2. un tuplu $t \in r$

astfel încât $r - \{t\}$ nu este un conținut valid al R.

Definiția 35 O schemă de relație R se zice a avea o *anomalie* (*de actualizare a datelor*) dacă ea are vreo anomalie de inserție sau de ştergere.

Studiul problemelor legate de eliminarea acestor anomalii a condus la definirea *formelor normale* (*FN*) din ce în ce mai înalte pentru scheme de relații; aceste *FN* iau în considerare tocmai proprietățile de tip constrângeri de integritate ale aplicațiilor modelate (de exemplu, "orice județ are un unic prefect" etc.). De departe, cea mai importantă dintre constrângerile avute în vedere de modelul relațional este *FD*.

3.2 Relații în forma normală Boyce-Codd

Definiția 36 O schemă de relație $R(U)$ se zice a fi în *forma normală Boyce-Codd* (*FNBC*) dacă $\forall X \rightarrow Y$, cu $Y \not\subset X$, X este supercheie în $R(U)$[19].

Altfel spus, nici un atribut nu poate depinde funcțional de vreo mulțime de atribute ce nu conține o cheie. În esență, aceasta înseamnă că orice *FD* ar trebui să poată fi descrisă de chei.

Revenind la bd *ALEGERI* din figura 22, se pot identifica următoarele şase *FD*:

Tip, NrCurent $\rightarrow$ *Localitate*

Circumscripție $\rightarrow$ *Județ*

Localitate $\rightarrow$ *Județ, Primar, Circumscripție*

Județ $\rightarrow$ *Prefect*

[19] Dacă $Y \subseteq X$, $X \rightarrow Y$ este trivială (vezi lema 37); evident că *FD* triviale nu contrazic *FNBC* chiar dacă X nu este supercheie!

Primar → *Localitate*

Prefect → *Județ*

Deoarece, de exemplu, *Circumscripție* nu este cheie (după cum nici *Localitate* şi nici *Județ* nu sunt, deşi toate aceste trei atribute alcătuiesc singure partea stângă a câte unei *FD*) rezultă că relația *ALEGERI* nu este în *FNBC*. În schimb, schema de bd următoare este alcătuită numai din relații în *FNBC*:

CENTRE(*Tip, NrCurent,Localitate,Circumscripție*)

JUDEȚE(*Județ, Prefect*)

CIRCUMSCRIPȚII(*Circumscripție, Județ*)

cheie(*Prefect*)

LOCALITĂȚI(*Localitate,Județ,Primar,Circumscripție*)

cheie(*Primar*)

Bd obținută, prin proiecții, conform acestei scheme din relația *ALEGERI* este prezentată în figura 23. Se zice că schema acestei bd a fost obținută prin *descompunerea* unicei relații *ALEGERI* a bd "echivalente".

Este uşor de remarcat faptul că din această bd care este în *FNBC* au fost eliminate toate anomaliile de actualizare a datelor semnalate anterior, minus cea de inserție (care nu poate fi rezolvată decât prin adăugarea unei chei primare surogat *#Centru*). De exemplu, schimbarea primarului unei localități, a prefectului unui județ, sau a apartenenței unei localități la un județ impun acum doar modificarea tuplului corespunzător din *LOCALITĂȚI* sau *JUDEȚE*; se pot insera noi circumscripții, localități sau județe, fără ca ele să aibă vreun centru de votare deja memorat în bd; ştergerea tuturor centrelor de votare dintr-o localitate nu conduce şi la pierderea informațiilor despre acea localitate (care sunt independent păstrate în relația *LOCALITĂȚI*) etc.

Exemplul de *normalizare* prezentat mai sus (adică înlocuirea unei scheme nenormalizate cu una normalizată echivalentă) sugerează că, în procesul de proiectare a schemei unei bd, ar putea fi nevoie să descompunem acele relații a căror schemă este nesatisfăcătoare, în mai multe relații mai mici, dar a căror schemă respectă o anumită formă normală. Pentru a răspunde însă la unele interogări asupra bd, ar putea fi necesară şi schema relațiilor originare. De exemplu, dacă am vrea să aflăm prefecții în a căror jurisdicție intră centrele de votare, relația *ALEGERI* trebuie reconstruită pentru a fi apoi din nou proiectată pe atributele *Tip, NrCurent, Prefect*:

$$\pi_{Tip,NrCurent,Prefect}(CENTRE \bowtie LOCALITĂȚI \bowtie JUDEȚE)$$

De remarcat că joinul relațiilor *CENTRE*, *LOCALITĂȚI* şi *JUDEȚE* reconstruieşte întotdeauna exact relația *ALEGERI*, ceea ce, am văzut, înseamnă că descompunerea *ALEGERI* conform acestor trei scheme de relații se face fără pierderi. Se mai poate observa că, similar, acelaşi lucru este adevărat şi pentru descompunerea ce include şi relația *CIRCUMSCRIPȚII*[20].

[20] Faptul că: $CENTRE \bowtie LOCALITĂȚI \bowtie JUDEȚE \bowtie CIRCUMSCRIPȚII = ALEGERI = CENTRE \bowtie LOCALITĂȚI \bowtie JUDEȚE = ALEGERI \bowtie CIRCUMSCRIPȚII$, precum şi acela că ambele descompuneri ale *ALEGERI*, atât cea exclusiv, cât şi cea inclusiv *CIRCUMSCRIPȚII* se fac fără pierderi se datorează constrângerii suplimentare de tip comutativitate de diagrame de funcții (ce nu se poate evident aserta în *MRD*) care afirmă în acest caz că:
(*LOCALITĂȚI* → *JUDEȚE*) = (*CIRCUMSCRIPȚII* → *JUDEȚE*) ° (*LOCALITĂȚI* → *CIRCUMSCRIPȚII*)

Vom vedea în secțiunile următoare că proprietatea de descompunere fără pierderi este fundamentală. După cum știm deja, ea nu este însă garantabilă oricăror descompuneri. Exemplul următor ne arată că există și descompuneri în *FNBC* ce nu au această proprietate.

Exemplul 20

Fie schema *R*(*Localitate, Județ, Primar, Prefect*) cu aceleași date ca în *ALEGERI* din figura 22 și cu aceleași *FD* între atributele ei ca mai sus și anume: *Localitate* → *Județ*, *Localitate* → *Primar*, *Județ* → *Prefect*, *Primar* → *Localitate*, *Prefect* → *Județ*.

Descompunerea lui *R* în relațiile *LOCALITĂȚI*(*Localitate, Primar, Prefect*) și *JUDEȚE*(*Județ, Prefect*) este în *FNBC* dar se face cu pierderi și, ca atare, reconstrucția *R* din aceste două descompuneri ale sale nu este posibilă.

CENTRE

JUDEȚE

Tip	*NrCurent*	*Localitate*	*Circumscripție*
Urban	01	Sebeș	1
Rural	02	Lancrăm	1
Suburban	01	Oltenița	3
Municipal	01	Vaslui	6
Municipal	02	Vaslui	9
UM MApN	01	Alba-Iulia	8

Județ	*Prefect*
Alba	Văcărescu
Călăraşi	Serghei
Vaslui	Văcărescu

CIRCUMSCRIPȚII

LOCALITĂȚI

Circumscripție	*Județ*
1	Alba
3	Călăraşi
6	Vaslui
8	Alba
9	Vaslui

Localitate	*Județ*	*Circumscripție*	*Primar*
Alba-Iulia	Alba	1	Ionescu
Sebeș	Alba	1	Pop
Lancrăm	Alba	3	Octavian
Oltenița	Călăraşi	6	Ilici
Vaslui	Vaslui	9	Vasilescu

Figura 23: O bd descompusă în FNBC pentru aplicația "alegeri locale"

Desigur că numai adăugând chei surogat tuturor relațiilor (şi suntem convinşi că ple-doaria de până acum în favoarea lor este convingătoare) şi continuând şi semantic nor-malizarea (prin abstractizarea conceptelor de *TIPURI_CENTRE* şi *PERSOANE*) ca în fi-gura 24 se pot elimina complet toate anomaliile de actualizare (pe care nu *FNBC*, ci doar *FNDC*, *forma normală domenii-chei*, vezi secțiunea 5.4, le elimină complet).

CENTRE

PERSOANE

#C	*Tip*	*NrCurent*	*Localitate*	*Circumscripție*
1	1	01	2	1
2	2	02	3	1
3	3	01	4	3
4	4	01	5	6
5	4	02	5	9

6	5	01	1	8

CIRCUMSCRIPȚII

Circumscripție	*Județ*
1	1
3	2
6	3
8	1
9	3

LOCALITĂȚI

#Persoana	*Persoana*
1	Văcărescu
2	Serghei
3	Văcărescu
4	Ionescu
5	Pop
6	Octavian
7	Ilici
8	Vasilescu

TIPURI_CENTRE

#TipCentru	*TipCentru*
1	Urban
2	Rural
3	Suburban
4	Municipal
5	UM MApN

#Loc	*Localitate*	*Județ*	*Circumscripție*	*Primar*
1	Alba-Iulia	1	1	4
2	Sebeş	1	1	5
3	Lancrăm	1	3	6
4	Oltenița	2	6	7
5	Vaslui	3	9	8

JUDEȚE

#Județ	*Județ*	*Prefect*
1	Alba	1
2	Călăraşi	2
3	Vaslui	3

Figura 24: O bd în FNDC pentru aplicația "alegeri locale"

4. Teoria dependențelor elementare

În esență, studiem în această secțiune problema implicației pentru *FD* şi *DIN*; în plus, vom analiza şi problematica interacțiunii între valorile nule şi *FD*. Atragem atenția de la început asupra faptului că rostul rezolvării problemei implicației (nu doar pentru *FD* şi *DIN*, ci pentru orice alte clase de constrângeri) nu este în nici un caz acela de a o aplica în modelarea datelor! Sarcina principală a proiectantului de bd este de a depista şi formaliza constrângerile necesare subuniversului modelat pentru a le include în schema proiectată; în cursul acestui proces însă, în special în bd de anvergură, ce presupun zeci sau chiar sute ori mii de constrângeri, el poate greşi, cerând SGBD impunerea unor constrângeri derivabile din celelalte. Ca atare, un SGBD performant trebuie să dispună de un optimizator al mulțimilor de constrângeri care, atunci când e cazul, să înlocuiască

mulțimea constrângerilor impuse de proiectant cu o acoperire minimă, optimă a ei (fie automat şi transparent pentru proiectant, fie doar cu acordul acestuia).

4.1 Reguli de derivare pentru funcțional dependențe

Prezentăm în această subsecțiune o mulțime de reguli de inferență pentru *FD* şi le demonstrăm corectitudinea. Pentru simplitate, presupunem că toate *FD* sunt definite peste schema unei unice relații $R(U)$; ca atare, toate mulțimile de atribute considerate sunt submulțimi nevide al U.

Următoarele trei leme furnizează condiții suficiente pentru implicația *FD*; ele vor fi folosite ca bază pentru construcția unei mulțimi de reguli de inferență solide.

Lema 37 O *FD* $X \to Y$ este trivială dacă şi numai dacă $X \supseteq Y$.

Demonstrație:

($\Rightarrow$)(*dacă*) Fie $X \supseteq Y$; arătăm că $X \to Y$ este trivială, adică, conform definiției, că $X \to Y$ ține în orice conținut al R. Considerăm o pereche de tupli $t_1, t_2 \in r$; dacă $t_1[X] = t_2[X]$, atunci $t_1[A] = t_2[A]$, $\forall A \in X$. Cum $X \supseteq Y$, rezultă că $t_1[A] = t_2[A]$, $\forall A \in Y$, ceea ce este echivalent cu a spune că $t_1[Y] = t_2[Y]$.

($\Leftarrow$)(*şi numai dacă*) Arătăm că, dacă $Y \not\subset X$, atunci $X \to Y$ este netrivială. Cum $Y \not\subset X$, $\exists A \in Y - X$; fie r o relație conținând doi tupli[21] t_1, t_2 cu proprietatea că $t_1[X] = t_2[X]$ şi $t_1[B] \neq t_2[B]$, $\forall B \notin X$. Deoarece $A \notin X$, relația r nu satisface $X \to Y$ şi deci $t_1[Y] \neq t_2[Y]$ q.e.d.

Lema 38 Dacă r satisface $X \to Y$, iar $Z \supseteq X$ şi $W \subseteq Y$, atunci r satisface şi $Z \to W$.

Demonstrație: Fie r satisfăcând $X \to Y$; arătăm că r satisface şi $Z \to W$. Fie $t_1, t_2 \in r$ cu proprietatea că $t_1[Z] = t_2[Z]$; arătăm că $t_1[W] = t_2[W]$. Cum $Z \supseteq X$, din $t_1[Z] = t_2[Z]$ rezultă că $t_1[X] = t_2[X]$; deoarece, în plus, r satisface $X \to Y$, rezultă că $t_1[Y] = t_2[Y]$. Ca atare, fiindcă $W \subseteq Y$, rezultă evident că $t_1[W] = t_2[W]$

q.e.d.

Lema 39 Dacă r satisface $X \to Y$ şi $Z \to W$, $Z \subseteq Y$, atunci r satisface şi $X \to YW$.

Demonstrație: Fie r satisfăcând $X \to Y$ şi $Z \to W$, iar $t_1, t_2 \in r$ cu proprietatea că $t_1[X] = t_2[X]$. Cum r satisface $X \to Y$, urmează că $t_1[Y] = t_2[Y]$ şi deci, deoarece $Z \subseteq Y$, că $t_1[Z] = t_2[Z]$; fiindcă r satisface şi $Z \to W$, rezultă că $t_1[W] = t_2[W]$, deci şi că $t_1[YW] = t_2[YW]$ q.e.d.

Definiția 40 Se zice *regulă de inferență* (*derivare*) orice propoziție de forma:

dacă $f_1, f_2, \ldots, f_k$ *satisfac* P, *atunci* f_{k+1}

unde f_i, $1 \leq i \leq k+1$, sunt *FD*[22] de forma $X_i \to Y_i$,[23] iar P este o condiție decidabilă asupra $(k+1)$-tuplului $(f_1, f_2, \ldots, f_k, f_{k+1})$. În plus, $f_1, f_2, \ldots, f_k$ se zic *antecedentele* regulii, în timp ce f_{k+1} se zice *consecința* ei.

Exemplul 21

dacă $X_1 \to Y_1$, $X_2 \to Y_2$ *satisfac* $Y_1 = X_2 \wedge X_1 = X_3 \wedge Y_2 = Y_3$, *atunci* $X_3 \to Y_3$

este, evident, formalizarea ca regulă de inferență a tranzitivității *FD* (i.e. a compunerii funcțiilor).

În majoritatea cazurilor se folosesc prescurtări pentru reguli; acestea încapsulează părți ale condiției P în mulțimile de atribute implicate. Astfel, regula din exemplul 21 se scrie prescurtat *dacă* $X_1 \to Y_1$, $Y_1 \to Y_2$,, *atunci* $X_1 \to Y_2$ sau, şi mai uşor de citit:

dacă $X \to Y$, $Y \to Z$,, *atunci* $X \to Z$.

---

[21] Evident, aceasta presupune că domeniile atributelor au, fiecare, cel puțin două valori (ceea ce este întotdeauna rezonabil în practică); de fapt, în general, presupunem că domeniile sunt infinit numărabile.

[22] Conceptul de regulă de inferență poate fi evident extins cu efort minim la orice tip de constrângeri.

[23] Mai precis, $X_i \to Y_i$ este o „*schemă* (*familie*) de *FD*", deoarece ea reprezintă orice *FD* care, împreună cu celelalte *FD* din definiție, satisface condiția P.

Definiția 41 Se zice (*schemă de*) *axiomă*[24] orice regulă de inferență fără antecedente. Axiomele sunt scrise sub forma: *dacă* $\perp$ *satisface P, atunci f*.

Regulile de inferență reduc problema implicației la manipulări sintactice: constrângerile sunt derivate cu ajutorul acestor reguli, ceea ce este mult mai simplu decât demonstrarea implicației pentru fiecare *FD* în parte doar pe baza tuplilor şi valorilor.[25] Noțiunile de soliditate şi completitudine leagă sintaxa de semantică:

Definiția 42 O *regulă de inferență* se zice *solidă* dacă, $\forall R(U)$ şi $\forall(f_1, f_2, \ldots, f_k, f_{k+1})$, ($k$+1)-tuplul de *FD* peste $R(U)$ satisfăcând *P*, mulțimea de *FD* $\{f_1, f_2, \ldots, f_k\}$ implică f_{k+1}.

O consecință imediată a lemelor 37, 38 şi 39 este soliditatea următoarelor reguli de inferență:[26]

FD1. *Reflexivitate*: *dacă* $\perp$, *atunci* $X \to X$.

FD2. *Descompunere*: *dacă* $X \to YZ$, *atunci* $X \to Y$.

FD3. *Tranzitivitate* (*extinsă*): *dacă* $X \to YW$, $W \to Z$, *atunci* $X \to YWZ$.[27]

Dată fiind o regulă de inferență oarecare, se mai zice că f_{k+1} *rezultă din* $f_1, f_2, \ldots, f_k$ *prin aplicarea regulii* R,[28] dacă (k+1)-tuplul $(f_1, f_2, \ldots, f_k, f_{k+1})$ satisface condiția asociată R. De exemplu, *FD* $A \to BCD$ rezultă imediat din $A \to BC$ şi $BC \to CD$ prin aplicarea regulii FD3 (cu $X=A$, $Y=\varnothing$, $W=BC$, $Z=CD$).

Definiția 43 Date fiind o mulțime *F* de *FD* şi o *FD* f oarecari, se zice *derivare a lui* f *din F cu ajutorul unei mulțimi* S *de reguli* o secvență finită de *FD* astfel încât:

(1) orice *FD* din secvență ori aparține *F*, ori rezultă din *FD* anterioare ei în secvență prin aplicarea unei reguli de inferență; iar

(2) f apare în secvență (de obicei drept ultim element).

Definiția 44 Dacă există cel puțin o derivare a lui f din *F* cu ajutorul lui S, se zice că f *este derivabilă din F cu ajutorul regulilor* S. Ca de obicei, dacă S este subînțeles din context, el este cel mai adesea omis.[29]

Exemplul 22

Următoarea secvență constituie o derivare a $AC \to DE$ din $F = \{BC \to DE, A \to B, AEG \to H\}$, cu ajutorul regulilor FD1, FD2 şi FD3 de mai sus:

1. $AC \to AC$ (din FD1)
2. $A \to B$ (din *F*)
3. $AC \to ABC$ (din 1. şi 2., cu ajutorul FD3)
4. $AC \to BC$ (din 3., cu ajutorul FD2)
5. $BC \to DE$ (din *F*)
6. $AC \to BCDE$ (din 4. şi 5., cu ajutorul FD3)
7. $AC \to DE$ (din 6., cu ajutorul FD2)

---

[24] În literatura bd relaționale, termenii de axiomă şi de regulă de inferență sunt adesea considerați sinonimi, ceea ce nu este cazul în domeniul sistemelor formale (de unde au fost de fapt împrumutate aceste concepte)!

[25] De notat că regulile de inferență sunt folosite în multe alte domenii, nu doar în bd, ca instrumente de manipulare sintactică a frazelor încărcate cu o semantică oarecare.

[26] De fapt, aceste trei leme implică soliditatea unor reguli şi mai generale!

[27] În literatură, în locul tranzitivității extinse, se folosește în general *tranzitivitatea*: *dacă* $X \to Y$, $Y \to Z$, *atunci* $X \to Z$. Demonstrațiile următoarelor rezultate sunt însă mai puțin laborioase dacă dispunem de această variantă extinsă.

[28] Aici, k este egal cu numărul de antecedente din regula de inferență; evident că, pentru FD1 k=0, pentru FD2 k=1, iar pentru FD3 k=2.

[29] Metamatematic, derivabilitatea este definiția unei demonstrații matematice: *F* este mulțimea ipotezelor unei teoreme (leme, propoziții etc.), f este rezultatul (ceea ce trebuie demonstrat), S este mulțimea axiomelor şi a teoremelor anterior demonstrate, iar derivarea lui f este chiar demonstrația teoremei!

Definiția 45 O *derivare* se zice *neredundantă* dacă:

(1) orice *FD* derivată este ulterior utilizată (nu se derivează reguli inutile)

(2) nici o *FD* nu este derivată de două sau mai multe ori (derivarea e aciclică)

(3) nu este derivată nici o *FD* din *F* (derivarea nu e tautologică)

Soliditatea regulilor de inferență garantează că, prin aplicarea lor repetată, pot fi derivate doar *FD* implicate de antecedentele regulilor. Demonstrația acestei afirmații, formalizată de următoarea lemă, este lăsată cititorului ca exercițiu (vezi problema 13):

Lema 46 Dacă f este derivabilă din F cu ajutorul regulilor FD1, FD2, FD3, atunci F implică f (i.e. $f \in F^+$).

Desigur că o mulțime de reguli solide este cu adevărat interesantă numai dacă se bucură și de proprietatea duală solidității, și anume *completitudinea* (adică dacă toate *FD* implicate sunt și derivabile).

Definiția 47 O *mulțime de reguli de inferență* se zice *completă* (*în raport cu implica-ția FD*) dacă $\forall F$, F mulțime de *FD*, f este derivabilă din F dacă $f \in F^+$.

Ca atare, o mulțime de reguli de inferență este solidă și completă dacă $\forall F$, F mulțime de *FD*, f este derivabilă din F dacă și numai dacă $f \in F^+$.

Pe de altă parte, este important de subliniat independența acestor două noțiuni: există mulțimi solide dar incomplete (de exemplu, orice mulțime formată doar cu două dintre regulile FD1, FD2 și FD3), precum și mulțimi solide care nu sunt însă complete.

Exemplul 23

Regula *dacă* $\perp$, *atunci* $X \to Y$ este completă, deoarece permite derivarea oricărei *FD*; evident însă că ea nu este solidă, căci, în general, nu toate *FD* posibile aparțin F^+. De exemplu, dacă $F = \{Localitate \to Județ\}$, această regulă ar deriva și $Județ \to Localitate$, deși, evident, această *FD* nu aparține F^+.

Tot ceea ce mai rămâne din această subsecțiune este dedicat demonstrării solidității și completitudinii mulțimii de reguli {FD1,FD2,FD3} pentru *FD*.

Începem prin a statua câteva proprietăți suplimentare interesante ale *FD*:

Lema 48 Fie F o mulțime de *FD*:

i. Orice relație satisface $X \to Y \Leftrightarrow$ ea satisface $X \to A$, $\forall A \in Y$.

ii. $X \to Y$ este derivabilă din F cu ajutorul FD1, FD2 și FD3 $\Leftrightarrow X \to A$ este derivabilă din F cu ajutorul FD1, FD2 și FD3, $\forall A \in Y$.

iii. $X \to Y \in F^+ \Leftrightarrow X \to A \in F^+$, $\forall A \in Y$.

Demonstrația acestei leme este lăsată în seama cititorului (vezi problema 14).

De remarcat că, pentru moment, (ii) și (iii) din lema de mai sus au înțelesuri distincte: prima se referă la derivare, iar cea de-a doua la implicație. Când vom dovedi completitudinea regulilor însă, ele vor deveni echivalente.

O consecință imediată a lemei 48(iii) este aceea că, pentru orice mulțime de *FD*, există o mulțime echivalentă conținând în partea dreaptă a fiecărei *FD* doar câte un atribut (și nu o mulțime de atribute). De aceea, oricând, fără a pierde cu nimic din generalitate, putem considera că toate *FD* au doar un singur atribut în partea lor dreaptă.

Înainte de a putea demonstra soliditatea și completitudinea mulțimii de reguli {FD1,FD2,FD3} pentru *FD*, mai trebuie să precizăm și să caracterizăm noțiunea de închidere a unei mulțimi de atribute în raport cu o mulțime de *FD*. Date fiind o mulțime F

oarecare de *FD* şi o mulțime *X* oarecare de atribute, fie $\wp$ următoarea mulțime de mulțimi de atribute: $\wp = \{Y \mid X \to Y$ este derivabilă din *F* cu ajutorul FD1 şi FD3$\}$.

Definiția 49 Se zice *închiderea lui X în raport cu F* şi se notează X^+ mulțimea:

$$X^+ = \cup_{Y \in \wp} Y.$$

Lema următoare stabileşte proprietățile fundamentale ale închiderilor:

Lema 50

i. $X \to X^+$ este derivabilă din *F* cu ajutorul FD1 şi FD
ii. Dacă $Y \subseteq X^+$, atunci $X \to Y$ este derivabilă din *F* cu ajutorul FD1, FD2 şi FD
iii. Dacă $A \in X^+$, atunci $X \to A$ este derivabilă din *F* cu ajutorul FD1, FD2 şi FD

Demonstrație:

i. Fie $X \to Y_1$, $X \to Y_2$, ..., $X \to Y_k$ toate *FD* derivabile din *F* cu ajutorul FD1 şi FD3; de remarcat că unul dintre aceşti Y_i trebuie să fie egal cu *X*, deoarece $X \to X$ este derivabilă cu ajutorul FD1. Se poate obține deci $X \to Y_1Y_2...Y_k$ cu ajutorul unei derivări ce include întâi toate derivările pentru cele *k FD* de mai sus şi care, apoi, aplică pentru $i = 1,2, ..., k$ FD3 asupra $X \to XY_1Y_2...Y_{i-1}$ şi $X \to Y_i$ pentru a genera $X \to XY_1Y_2...Y_i$. Cum $XY_1Y_2...Y_k = Y_1Y_2...Y_k$, ultimul pas generează $X \to \to Y_1Y_2...Y_k$.
ii. Conform i., $X \to X^+$ este derivabilă din *F*; deoarece $Y \subseteq X^+$, aplicând FD2, rezultă trivial că $X \to Y$.
iii. Similar cu ii

q.e.d.

Putem acum demonstra următoarele proprietăți, ce justifică pentru X^+ denumirea de „închidere” (vezi problema 18):

Corolar 51

(1) $X \subseteq X^+$ (orice mulțime de atribute este inclusă în închiderea ei)
(2) $(X^+)^+ = X^+$
(operatorul de închidere este idempotent)
(3) X^+ este cea mai mică mulțime ce satisface (1) şi (2) de mai sus.

Exemplul 24

Dată fiind mulțimea *F* de *FD* din exemplul 22, aplicând exhaustiv regulile, se poate demonstra că mulțimile *Y* cu proprietatea că $AC \to Y$ este derivabilă din *F* cu ajutorul FD1 şi FD3 sunt *AC, B, ABC, ABCDE* şi deci că $(AC)^+ = ABCDE$.

Cea mai importantă proprietate a închiderilor, de care avem nevoie în demonstrarea completitudinii FD1, FD2 şi FD3, este stabilită de lema următoare:

Lema 52 Dacă $X \to A \in F^+$, atunci $A \in X^+$.

Demonstrație:

Arătăm că dacă $A \notin X^+$, atunci $X \to A \notin F^+$. Datorită propoziției 25, este suficient să găsim drept contraexemplu o relație ce satisface *F* dar nu satisface şi $X \to A$. Afirmăm că orice relație *r* cu doi tupli t_1 şi t_2 având proprietatea că $t_1[B] = t_2[B] \Leftrightarrow B \in X^+$ constituie un asemenea contraexemplu.

(*r nu satisface* $X \to A$) este trivială, deoarece $A \notin X^+$.

(*r satisface F*) Arătăm că dacă $f \in F$, atunci *r* satisface *f*. Fie $f = W \to Z \in F$. Dacă $t_1[W] \neq t_2[W]$, atunci *f* este trivial satisfăcută; altminteri, din construcția *r*, $W \subseteq X^+$. Ca atare, conform lemei 50, cum $X \to X^+$ este derivabilă cu ajutorul FD1 şi FD3, rezultă că şi $X \to ZX^+$ este derivabilă cu ajutorul FD1 şi FD3 (şi anume prin aplicarea FD3 asupra $X \to X^+$ şi $W \to Z$). Dar, prin definiție, X^+ conține toate submulțimile *Y* cu proprietatea

că $X \rightarrow Y$ este derivabilă şi deci trebuie ca $Z \subseteq X^+$. De aceea, $t_1[Z] = t_2[Z]$, ceea ce înseamnă că $f = W \rightarrow Z$ este satisfăcută

q.e.d.

Din lemele 50 şi 52 rezultă evident că:

- X^+ conține un atribut $A \Leftrightarrow X \rightarrow A \in F^+$
- X^+ conține un atribut $A \Leftrightarrow X \rightarrow A$ este derivabilă din F cu ajutorul FD1,FD2,FD3.

Suntem acum în măsură să demonstrăm principalul rezultat al acestei subsecțiuni:

Teorema 53 Mulțimea de reguli de inferență {FD1,FD2,FD3} este solidă şi completă în raport cu implicația *FD.*

Demonstrație:

Lema 46 garantează soliditatea; pentru a demonstra şi completitudinea, trebuie arătat că $\forall F$, F mulțime de *FD*, dacă $f \in F^+$, atunci f este derivabilă din F cu ajutorul regulilor. Dacă $X \rightarrow Y \in F^+$, atunci, conform lemei 48(iii), $X \rightarrow A \in F^+$, $\forall A \in Y$. Însă, conform lemei 52, $\forall A \in Y$, $A \in X^+$ şi deci $Y \subseteq X^+$. Ca atare, conform lemei 50(ii), $X \rightarrow Y$ este derivabilă

q.e.d.

Vom vedea acum, în sfârşit, cum se pot folosi regulile de inferență pentru a rezolva problema implicației *FD*.

4.2 Problema implicației pentru funcțional dependențe

Existența unei mulțimi de reguli de inferență solide şi complete pentru *FD* garantează decidabilitatea problemei implicației: date fiind $R(U)$ şi F, putem aplica exhaustiv regulile de inferență pentru a genera F^+, care poate apoi fi folosit pentru a testa care *FD* sunt implicate de F şi care nu.

Această metodă nu este însă practică, deoarece F^+ are adesea cardinalitate exponențială în raport cu cea a lui F. De exemplu, pentru schema $R(AB)$, $F=\{A \rightarrow B\}$, închiderea F^+ conține şapte *FD* (dintre care, e drept, cinci sunt triviale). În general, numărul de *FD* triviale peste o schemă $R(U)$ este întotdeauna exponențial în raport cu cardinalitatea lui U (vezi problema 11)[30].

În practică însă, din fericire, suntem arareori interesați de toate *FD* din F^+: pentru a decide dacă $X \rightarrow Y \in F^+$, este suficient, conform lemelor 48 şi 52, să calculăm doar X^+ şi să verificăm apoi dacă $Y \subseteq X^+$. După cum vom vedea în curând, acest lucru poate fi făcut foarte eficient; pentru aceasta, desigur, avem însă nevoie de un algoritm care să calculeze închiderea lui X în raport cu F. Lema 55 fundamentează un prim asemenea algoritm; pentru a o demonstra mai uşor, este utilă propoziția 54:

Propoziția 54 Fie X o mulțime oarecare de atribute, F o mulțime oarecare de *FD* ce nu conține $X \rightarrow X$, iar δ o derivare neredundantă de lungime $l = m+n+1$, conținând:

(1) la început, m *FD* din F

(2) apoi, $X \rightarrow X$

(3) şi, în sfârşit, n *FD* generate cu ajutorul regulilor FD1 şi FD;

$\forall j$, $1 \leq j \leq n$, $\exists \delta$' derivare a $X \rightarrow X^+$, identică cu δ în primii $l-j$ paşi, astfel încât, $\forall k > l-j$, *FD* derivate în pasul k au partea stângă egală cu X.

[30] În general, cardinalitatea lui F^+ nu este neapărat exponențială în raport cu cea a lui F; chiar dacă ea ar fi liniară însă, cardinalitatea lui F este deja exponențială în raport cu cea a lui U!

Demonstrație:

Folosim inducția după j. Baza (j=1) este trivială, căci $X \rightarrow X^+$ este derivată în ultimul pas. Să presupunem că *FD* f_k derivată în pasul k, $m+1 < k < l$, este ultima având partea stângă $Y \neq X$. Transformăm δ într-o altă derivare cu proprietățile următoare:

(1) produce $X \rightarrow X^+$ în ultimul pas

(2) coincide cu δ în primii k-1 pași și

(3) produce *FD* având doar X în partea stângă în toți pașii de la k încolo inclusiv.

Dacă regula folosită în pasul k este FD1, atunci $f_k = X \rightarrow Y$. Cum derivarea este neredundantă, f_k va fi folosită într-un pas ulterior h, în care se va aplica FD3. Conform ipotezei de inducție, deoarece $Y \neq X$, f_k nu poate fi folosită drept prim antecedent; dacă ea ar fi folosită drept al doilea antecedent, atunci, din $f_j = X \rightarrow Z$ (cu $j < k$) și $Y \rightarrow Y$, unde $Y \subseteq Z$, ar rezulta $f_h = X \rightarrow YZ$; aceasta ar însemna că $YZ = Z$ și deci $f_h = f_k$, cu $h \neq k$, ceea ce contrazice însă ipoteza neredundanței. Ca atare, regula folosită în pasul k nu poate fi FD1, ci trebuie să fie FD3.

Fie $Y \rightarrow VWZ$ *FD* generată în pasul k cu ajutorul FD3 din $Y \rightarrow VW$ și $W \rightarrow Z$; ea va fi la rândul ei folosită într-un pas ulterior drept al doilea antecedent într-o nouă aplicare a FD3, care va genera o *FD* de forma $X \rightarrow TYVWZ$ din $X \rightarrow TY$ și $Y \rightarrow VWZ$. Evident că se obține același rezultat eliminând pasul ce produce $Y \rightarrow VWZ$ și derivând $X \rightarrow TYVWZ$ în doi pași, primul generând $X \rightarrow TYVW$ din $X \rightarrow TY$ și $Y \rightarrow VW$, iar al doilea pe $X \rightarrow TYVWZ$ din $X \rightarrow TYVW$ și $W \rightarrow Z$; aceasta, evident, completează inducția, deoarece noua derivare astfel obținută:

(1) produce $X \rightarrow X^+$ (fiindcă nici unul dintre pașii generând *FD* având X drept parte stângă nu a fost modificat)

(2) este identică cu cea inițială pentru primii k-1 pași și

(3) toate *FD* generate în pașii ulteriori lui k-1 au X drept parte stângă

q.e.d.

Exemplul 25

Fie $F = \{BC \rightarrow DE, A \rightarrow B, AEG \rightarrow H\}$ (ca și în exemplul 22, unde am văzut că $AC^+ = ABCDE$). O posibilă derivare a $AC \rightarrow ABCDE$ din F este:

1. $ABC \rightarrow ABC$ (din FD1)
2. $BC \rightarrow DE$ (în F)
3. $ABC \rightarrow ABCDE$ (din 1., 2. și FD3)
4. $A \rightarrow B$ (în F)
5. $AC \rightarrow AC$ (din FD1)
6. $AC \rightarrow ABC$ (din 4., 5. și FD3)
7. $AC \rightarrow ABCDE$ (din 3., 6. și FD3)

Aplicând propoziția 54, putem transforma această derivare astfel:

1. $BC \rightarrow DE$ (în F)
2. $A \rightarrow B$ (în F)
3. $AC \rightarrow AC$ (din FD1)
4. $AC \rightarrow ABC$ (din 2., 3. și FD3)
5. $AC \rightarrow ABCDE$ (din 1., 4. și FD3).

Lema 55 Există o derivare a $X \rightarrow X^+$ care, pe lângă *FD* din F, implică numai *FD* a căror parte stângă este constituită doar din X, dintre care prima este $X \rightarrow X$, iar celelalte se obțin din aceasta prin aplicarea regulii FD3.

Demonstrație:

Conform lemei 52, știm că există o derivare δ din F a $X \rightarrow X^+$ care implică doar folosirea FD1 și FD3 și fie n numărul de *FD* astfel generate; fără a pierde nimic din generalitate, putem evident presupune că δ satisface condițiile cerute de propoziția 54

de mai sus. Aplicând rezultatul acestei propoziții pentru cazul particular $j = n$, demonstrația lemei este încheiată

q.e.d.

Putem acum prezenta un prim algoritm pentru calculul închiderii unei mulțimi de atribute (vezi figura 25), căruia îi vom şi demonstra corectitudinea în teorema 57.

Exemplul 26

Să executăm algoritmul 56 pentru a calcula închiderea mulțimii de atribute *AC* în raport cu mulțimea de *FD F* din exemplul 22: *XPLUS* este inițializat cu *AC*; la fiecare iterație a buclei interne, *FD* din *F* sunt examinate în ordine, de la stânga la dreapta; ca atare, în prima iterație se intră pe ramura *atunci* doar pentru $A \rightarrow B$, când *B* este deci adăugat la *XPLUS*, care devine astfel *ABC*;

Algoritm 56 (*calculul închiderii unei mulțimi de atribute pentru o mulțime de FD*)
Intrare: o mulțime de atribute X şi
o mulțime de k *FD*, $F = \{X_1 \rightarrow Y_1, \ldots, X_k \rightarrow Y_k\}$
Ieşire: o mulțime de atribute *XPLUS* (închiderea lui X)
Variabile locale:
bool gata;
int i;
Început
$XPLUS = X$;
repetă
gata = adevărat;
repetă pentru i de la 1 *la k*

dacă $X_i \subseteq XPLUS$ *şi* $Y_i \not\subset XPLUS$

atunci gata = fals;

$XPLUS = XPLUS \cup Y_i$;

sfârşit dacă;

sfârşit repetă;

până când gata;

Sfârşit algoritm 56

Figura 25: Un prim algoritm de calcul al închiderii unei mulțimi de atribute

similar, în a doua iterație se intră pe ramura *atunci* doar pentru $BC \rightarrow DE$, când *DE* este deci adăugat la *XPLUS*, care devine astfel *ABCDE*; la a treia iterație, nici o *FD* nu mai satisface condiția, deci nici un atribut nu se mai adaugă la *XPLUS*, ceea ce are ca efect terminarea execuției.

Teorema 57 Pentru orice X şi F, algoritmul 56 calculează corect închiderea lui X în raport cu F în timp $O(|F| \times |U|)$ (unde $|F|$ este numărul de *FD* din F, iar $|U|$ este numărul total al atributelor implicate).

Demonstrație:

Arătăm întâi că acest algoritm nu buclează la infinit: bucla sa externă se execută doar dacă în iterația precedentă s-a adăugat la rezultat cel puțin un nou atribut; cum numărul atributelor menționate în F este finit, execuția algoritmului se termină după un număr finit de iterații.

Privind corectitudinea propriu-zisă, arătăm că, pentru orice X şi F, la terminarea execuției sale, $XPLUS = X^+$.

($XPLUS \subseteq X^+$)

Demonstrăm această primă incluziune prin inducție după numărul de modificări al mulțimii rezultat *XPLUS*: vom dovedi că X_j, valoarea ei după modificarea j, este într-adevăr o submulțime a X^+.

Evident că pentru j=0 acest lucru este trivial ($X_0 = X \subseteq X^+$). Presupunem că $X_{j-1} \subseteq X^+$ şi fie $X_l \rightarrow Y_l$ *FD* implicată în pasul j; evident că $X_l \subseteq X_j$ şi că $X_j = X_{j-1}Y_l$. Din ipoteza inducției ($X_{j-1} \subseteq X^+$) rezultă că $X \rightarrow X_{j-1} \in F^+$; urmează că $X \rightarrow X_{j-1}Y_l \in F^+$, deoarece ea este derivabilă din $X \rightarrow X_{j-1}$ şi $X_l \rightarrow Y_l$ aplicând FD3. Ca atare, conform lemelor 48 şi 50, $X_j = X_{j-1}Y_l \subseteq X^+$.

($XPLUS \supseteq X^+$)

Conform lemei 55, există o derivare a $X \to X^+$ din F ce generează doar *FD* având partea stângă egală cu X: $X \to X_0$, $X \to X_1$,…, $X \to X_n$; dar $\forall j$, $1 \le j \le n$, $X \to X_j$ este derivat din $X \to X_{j-1}$ şi o *FD* de forma $X_l \to Y_l$, unde $X_l \subseteq X_{j-1}$, iar $X_j = X_{j-1}Y_l$. Ca atare, $X = X_0 \subseteq X_1 \subseteq \ldots \subseteq X_n = X^+$. Demonstrăm şi această a doua incluziune tot prin inducție după j, dovedind astfel că $XPLUS \supseteq X_j$.

Din nou, baza inducției este trivială, căci *XPLUS* este inițializat cu X. Presupunem că $XPLUS \supseteq X_{j-1}$; după atingerea acestui moment în cursul execuției algoritmului, la prima evaluare a condiției instrucțiunii *dacă* pentru *FD* $X_l \to Y_l$, se va intra pe ramura *atunci*, ceea ce va avea ca efect adăugarea Y_l la *XPLUS*. În consecință, $XPLUS \supseteq X_{j-1}Y_l$ şi deci $XPLUS \supseteq X_j$, ceea ce completează demonstrația corectitudinii calculului.

În sfârşit, este uşor de arătat că acest algoritm are complexitatea $O(|F|\times|U|)$: bucla interioară se execută de $|F|$ ori pentru fiecare iterație a celei exterioare; bucla exterioară se execută atât timp cât mai sunt adăugate noi atribute rezultatului: în cel mai rău caz, ea se va executa deci de $|U|$ - 1 ori

q.e.d.

Algoritmul 56 este simplu dar ineficient, deoarece verifică repetat fiecare *FD*, chiar dacă ea a mai fost deja verificată. Evident că acest algoritm poate fi îmbunătățit; ca şi în cazul celei de-a doua incluziuni din demonstrația teoremei 57, ideea fundamentală pentru aceasta ne este oferită tot de lema 55: simulăm derivarea $X \to X^+$ folosind fiecare *FD* din F o singură dată şi anume numai atunci când partea sa stângă este deja inclusă în rezultatul calculat până în acel punct; aceasta se poate realiza, de exemplu, asociind fiecărei *FD* un contor ce memorează numărul de atribute din partea sa stângă ce nu sunt încă incluse în rezultat: atunci când acest contor devine 0, *FD* respectivă poate fi luată în considerare astfel încât partea sa dreaptă să fie adăugată rezultatului. Desigur că, ori de câte ori se adaugă la rezultat un nou atribut, algoritmul trebuie să decrementeze valoarea contorului fiecărei *FD* conținând acel atribut în partea stângă. Pentru a executa eficient acest pas obligatoriu, trebuie însă menținută, pentru fiecare atribut, o listă a *FD* care îl conțin în partea lor stângă. Algoritmul 58 (vezi figura 26) implementează exact această strategie.

Definiția 59 Se zice *dimensiunea unei mulțimi de FD* F şi se notează cu $\|F\|$ suma cardinalităților *FD* din F, i.e. $\|F\| = \sum_{i=1}^{k}(|Xi|+|Yi|)$, unde $k = card(F)$.

Teorema 60 Pentru orice X şi F, algoritmul 58 calculează închiderea lui X în raport cu F în timp $O(\|F\|)$.

Demonstrație:

Terminarea şi corectitudinea se demonstrează uşor, ca şi pentru teorema 57.

Cât priveşte complexitatea, trebuie evident analizate atât inițializarea listelor, cât şi bucla exterioară ce urmează. Inițializarea listelor trebuie precedată de citirea F, care se face, evident, în timp $O(\|F\|)$. În cadrul inițializării propriu-zise, operația crucială este adăugarea, ce se execută câte o dată pentru fiecare atribut al fiecărei părți stânga: în total, ea se execută deci de $\sum_{i=1}^{k}(|Xi|)$ ori. În bucla principală, timpul este evident consumat în majoritate de cele două instrucțiuni din bucla interioară: decrementarea se execută de cel mult $|X_i|$ ori pentru fiecare *FD* din F, în timp ce adăugarea unui nou atribut rezultatului se execută tot câte o dată pentru fiecare *FD* din F, dar necesită timp de $|Y_i|$. Ca atare, şi bucla principală se execută tot în timp $O(\|F\|)$ şi deci aceasta este şi complexitatea întregului algoritm

q.e.d.

Exemplul 27

Reluăm exemplul 22, ca şi în precedentul exemplu, cu diferența însă că acum folosim algoritmul 58 în locul algoritmului 56. Vectorii *CST* şi *LST* au 3, respectiv 7 elemente (corespunzătoare $F = \{f_1=BC\rightarrow DE,\ f_2=A\rightarrow B,\ f_3=AEG\rightarrow \rightarrow H\}$) şi anume: $CTR[f_1]=2$, $CTR[f_2]=1$, $CTR[f_3]=3$; $LST[A]=<2,3>$, $LST[B]=<1>$, $LST[C]=<1>$, $LST[D]=<>$, $LST[E]=<3>$, $LST[G]=<3>$, $LST[H]= =<>$. *XPLUS* este iniţializat cu *AC* iar *XTMP* cu mulțimea vidă; presupunem că atributele sunt selectate din *XPLUS – XTMP* în ordine lexicografică; în pri-ma iterație deci este selectat *A*; ca atare, $CTR[f_2]$ şi $CTR[f_3]$ sunt decrementa-

Algoritm 58 (*calculul eficient al închiderii unei mulțimi de atribute*)
Intrare: o mulțime de atribute X şi
o mulțime de k *FD*, $F = \{f_1=X_1 \rightarrow Y_1, \ldots, f_k=X_k \rightarrow Y_k\}$
(atributele implicate în F şi X sunt în număr de m: $A_1A_2\ldots A_m$)
Ieşire: o mulțime de atribute *XPLUS* (închiderea lui X)
Variabile locale:

```
int CTR[k], LST[m];
multatrib XTMP;
int i,j;
Început
pentru i = 1 la m repetă LST[Ai] = listavidă;
repetă pentru i = 1 la k
                                        CTR[fi] = |Yi|;
                                pentru fiecare Aj ∈ Yi repetă

                                adaugă i la LST[Aj];
                                        sfârşit repetă;
                        sfârşit repetă;
                        XPLUS = X;
                        XTMP = ∅;
                        repetă
                                selectează Aj din XPLUS – XTMP;
                                XTMP = XTMP ∪ { Aj };
                                repetă pentru fiecare i ∈ LST[Aj]

                        CTR[fi] = CTR[fi] – 1;

                        dacă CTR[fi] == 0

                                atunci XPLUS = XPLUS ∪ Yi;

                        sfârşit dacă;
                                sfârşit repetă;
                până când XPLUS == XTMP;
```

Sfârşit algoritm 58
Figura 26: Un algoritm mai eficient de calcul al închiderii unei mulțimi de atribute

te cu 1, ceea ce face ca $CTR[f_2]$ să devină 0 şi, în consecință, B (partea dreaptă a f_2) este adăugat la *XPLUS*, care devine astfel *ABC*; similar, în a doua iterație este selectat B, se decrementează $CTR[f_1]$, dar nu se adaugă nimic la rezultat; în a treia iterație este selectat C, se decrementează $CTR[f_1]$ care devine astfel 0 şi, în consecință, D şi E (partea dreaptă a f_1) sunt adăugate la *XPLUS*, care devine astfel *ABCDE*; în următoarele două iterații, D şi E sunt adăugate la *XTMP*, dar nici un contor nu mai devine 0, deci nici un atribut nu se mai adaugă la *XPLUS*; cum însă *XTMP* a devenit egal cu *XPLUS*, execuția se termină.

4.3 Acoperiri ale funcțional dependențelor

Găsirea acoperirilor minimale pentru o mulțime oarecare de constrângeri este utilă din cel puțin două puncte de vedere:

(i) pentru a reduce timpul necesar impunerii lor asupra conținuturilor de bd corespunzătoare;

(ii) pentru a reduce timpul necesar executării algoritmului 58 (deoarece conform teoremei 60, calculul închiderii unei mulțimi de atribute este proporțional cu dimensiunea mulțimii de constrângeri).

Conform definiției 28, minimalitatea este definită în raport cu cardinalitatea mulțimilor de constrângeri; desigur că ea se poate însă defini şi în raport cu dimensiunea mulțimilor de constrângeri.

Definiția 61 O *mulțime de constrângeri* Γ se zice *redundantă* dacă există o submulțime proprie $\Delta \subset \Gamma$ echivalentă (i.e. $\Delta^+ = \Gamma^+$).

Definiția 62 O mulțime de constrângeri Γ se zice *minimă* dacă nu există nici o acoperire a sa Δ având mai puține constrângeri (i.e. $\forall \Delta, \Delta^+ = \Gamma^+ \Rightarrow |\Gamma| \le |\Delta|$).

Definiția 63 O mulțime de constrângeri Γ se zice *optimă* dacă nu există nici o acoperire a sa Δ având dimensiune mai mică (i.e. $\forall \Delta, \Delta^+ = \Gamma^+ \Rightarrow \|\Gamma\| \le \|\Delta\|$).

Definiția 64 O mulțime de constrângeri Δ se zice *acoperire neredundantă minimă* (*optimă)* a Γ dacă ea este echivalentă cu Γ, neredundantă şi minimă (respectiv optimă).

Este uşor de arătat că, în cazul particular al *FD*, o acoperire minimă este şi neredundantă. Vom demonstra (din greu!) în această subsecțiune şi că orice acoperire optimă este minimă (vezi teorema 73). Contraexemplul următor dovedeşte însă că reciprocele acestor propoziții nu sunt adevărate nici măcar în acest caz simplu.

Exemplul 28

Mulțimea de *FD* $\Gamma = \{AB \to C, C \to A, C \to B, ABD \to E\}$ este neredundantă însă nu minimă: $\Delta = \{AB \to C, C \to AB, ABD \to E\}$ este echivalentă cu Γ şi con-ține o *FD* mai puțin; se poate dovedi însă că Δ este minimă, dar ea nu e şi op-timă căci $\Psi = \{AB \to C, C \to AB, CD \to E\}$ este echivalentă cu Δ şi are dimen-siunea mai mică.

Având în vedere că putem rezolva problema implicației în timp $O(\|\Gamma\|)$, este evident faptul că putem obține o acoperire neredundantă în timp $O(|\Gamma| \times \|\Gamma\|)$ prin eliminarea una câte una a tuturor *FD* redundante. Problema găsirii unei acoperiri minimale nu este însă atât de simplă; în cele ce urmează, trebuie să introducem în consecință câteva noi concepte ajutătoare.

Definiția 65 Două *mulțimi de atribute* X şi Y se zic *echivalente* în raport cu o mulțime oarecare de *FD* dacă $X^+ = Y^+$; X se zice (*strict*) *mai slabă* decât Y dacă $X^+ \subseteq Y^+$ (respectiv $X^+ \subset Y^+$, $X^+ \ne Y^+$).

Conform lemei 48, putem reuni toate *FD* având o aceeaşi parte stângă într-o singură *FD*; ca atare, orice acoperire minimală conține cel mult atâtea *FD* câte părți stânga distincte există. De asemenea, evident, dacă o mulțime Γ de *FD* este minimă, atunci şi mulțimea $\{X \to X^+ \mid X \to Y \in \Gamma\}$ este minimă; de aceea, în cele ce urmează, presupunem câteodată că toate *FD* din Γ sunt de forma $X \to X^+$.

Definiția 66 O mulțime Γ de *FD* se zice a avea *atributele închise* dacă toate elementele sale sunt de forma $X \to X^+$.

Demonstrația lemei ce urmează este propusă cititorului drept exercițiu (vezi problema 21).

Lema 67 Dacă $Y \to W$ este folosită într-o derivare neredundantă a $X \to V$ din Γ, atunci $Y^+ \subseteq X^+$.

Lema 68 Dacă Γ şi Δ sunt echivalente şi neredundante,

atunci $\forall X \to V \in \Gamma, \exists Y \to W \in \Delta$ astfel încât X şi Y sunt echivalente.

Demonstraţie:

Fie oricare ar fi Γ şi Δ distincte şi echivalente; dacă $f = X \to V \in \Gamma$, rezultă că $f \in \Delta^+$ şi deci că există o derivare a lui f din Δ; fie Δ' o submulţime minimală a Δ folosită în derivare; cum Γ este neredundantă, rezultă că $\Delta' \not\subset \Gamma$; de asemenea, există cel puţin o *FD* $g = Y \to W \in \Delta'$ cu proprietatea că orice derivare a lui g din Γ implică f (în caz contrar, am putea găsi o derivare a lui $f \in FD$ din $\Gamma - \{f\}$, ceea ce ar contrazice ipoteza neredundanţei). Ca atare, conform lemei 67, rezultă că $X^+ = Y^+$

q.e.d.

Lema 68 implică faptul că toate acoperirile neredundante conţin acelaşi număr de părţi stânga neechivalente. Ca atare, acoperirile minimale au un număr minim de părţi stânga în fiecare clasă de echivalenţă.

Definiţia 69 Date fiind o mulţime de *FD* Γ şi două mulţimi de atribute X şi Y, se zice că X *determină direct* Y în raport cu Γ şi se notează $X \to\bullet_\Gamma Y$, dacă $X \to Y$ este derivabilă dintr-o submulţime a Γ ce conţine doar *FD* având partea stângă mai slabă decât X.

Următoarea lemă arată că definiţia determinării directe nu depinde de acoperirea aleasă pentru Γ.

Lema 70 Dacă $X \to\bullet_\Gamma Y$, atunci $X \to\bullet_\Delta Y$, $\forall \Delta$ echivalent cu Γ.

Demonstraţie:

Fie Δ echivalent cu Γ şi $X \to\bullet_\Gamma Y$. Conform definiţiei, $X \to Y$ poate fi derivat dintr-o submulţime Γ' a Γ ale cărei elemente au partea stângă mai slabă decât X. Cum Δ este echivalent cu Γ, rezultă că orice *FD* $V \to W \in \Gamma'$ este derivabilă dintr-o submulţime a lui Δ, $\Delta_{V \to W}$ care, conform lemei 67, implică doar *FD* $Z \to T$ cu proprietatea că $Z^+ \subseteq V^+ \subset X^+$. Reuniunea $\Delta' = \cup_{f \in \Gamma'} \Delta_f$ acestor mulţimi se bucură de următoarele trei proprietăţi:

(i) este o submulţime a Δ

(ii) include doar *FD* având partea stângă mai slabă decât X

(iii) închiderea sa conţine $X \to Y$.

Rezultă, evident, că $X \to\bullet_\Delta Y$

q.e.d.

În consecinţă, conform acestei leme, se poate omite din notaţia determinării directe referinţa la mulţimea de *FD* şi scrie simplu $X \to\bullet Y$. Următoarele două leme pregătesc demonstraţia teoremei de caracterizare a acoperirilor minimale.

Lema 71 Dacă Γ şi Δ sunt echivalente, neredundante şi având atributele închise, atunci $\forall X \to X^+ \in \Gamma, \exists Y \to Y^+ \in \Delta$ a.î. $X^+ = Y^+$ şi $X \to\bullet Y$.

Demonstraţie:

Fie $X \to X^+ \in \Gamma$; dacă $X \to X^+ \in \Delta$, atunci demonstraţia s-a încheiat deoarece $X \to\bullet X$ are loc în mod trivial. În caz contrar, fie Y o parte stângă din Δ echivalentă cu X şi cu proprietatea că pentru nici o altă parte stângă echivalentă Z din Δ nu există vreo derivare a $X \to Z$ din Δ mai scurtă decât cea mai scurtă derivare a $X \to Y$ din Δ. Susţinem că această derivare nu conţine nici o *FD* $V \to V^+$ cu proprietatea că $V^+ = X^+$ şi că, deci, $X \to\bullet Y$. Într-adevăr, dacă ar exista o asemenea *FD* în derivare, atunci derivarea $X \to V$ din Δ ar fi mai scurtă decât derivarea $X \to Y$, ceea ce ar conduce la o contradicţie

q.e.d.

Lema 72 O mulţime Γ de *FD* având atributele închise este minimă dacă şi numai dacă nu există nici o pereche X_1, X_2 de părţi stânga distincte şi echivalente cu proprietatea că $X_1 \to\bullet X_2$.

Demonstraţie:

($\Rightarrow$)(*dacă*)

Să presupunem că Γ nu este minimă şi fie Δ o acoperire minimală a sa; cum numărul de clase de echivalenţă ale părţilor stânga din Γ şi Δ este acelaşi, trebuie să existe în Γ cel puţin o asemenea clasă care să aibă mai multe elemente decât cea echivalentă din Δ. Fie Y_1, ..., Y_p elementele clasei din Γ, respectiv Z_1, ..., Z_q elementele clasei echivalente din Δ, cu $p > q$. Conform lemei 71, $\forall Y_i$, $\exists Z_j$ astfel încât $Y_i \rightarrow\bullet Z_j$. Cum sunt mai mulţi asemenea Y_i decât Z_j, trebuie să existe cel puţin o mulţime Z_j cu proprietatea că există cel puţin două mulţimi distincte Y_i, Y_k astfel încât $Y_i \rightarrow\bullet Z_j$ şi $Y_k \rightarrow\bullet Z_j$. Dar deoarece, tot conform lemei 71, $\forall Z_j$, $\exists Y_h$ cu proprietatea că $Z_j \rightarrow\bullet Y_h$ şi deoarece h nu poate fi în acelaşi timp egal şi cu i şi cu k (căci $i \neq k$), rezultă că $h \neq i$. Ca atare, urmează că $Y_i \rightarrow\bullet Y_h$ sau că $h \neq k$ şi deci că $Y_k \rightarrow\bullet Y_h$ (tranzitivitatea determinării directe putându-se demonstra prin juxtapunerea derivărilor corespunzătoare).

($\Leftarrow$)(*şi numai dacă*)

Dacă Γ conţine două funcţional dependenţe $X_1 \rightarrow Y$, $X_2 \rightarrow Y$ având proprietatea că $X_1^+ = X_2^+ = Y$ şi $X_1 \rightarrow\bullet X_2$, atunci $X_1 \rightarrow X_2$ poate fi derivată dintr-o submulţime Γ' a lui Γ ce conţine doar *FD* care au partea stângă mai slabă decât X_1. Ca atare, $X_1 \rightarrow Y$ poate fi derivată din $\Gamma' \cup \{X_2 \rightarrow Y\}$. Rezultă că Γ este echivalent cu $\Gamma - \{X_1 \rightarrow Y\}$, care conţine mai puţine *FD* decât Γ.

q.e.d.

Teorema 73 (*caracterizarea acoperirilor minimale*)

O mulţime Γ de *FD* având atributele închise este minimă dacă şi numai dacă nu este redundantă.

Demonstraţie:

Partea *şi numai dacă* a demonstraţiei rezultă, aşa cum am văzut mai sus, din definiţie. Partea *dacă* rezultă din lema 72: dacă Γ nu este minimă, atunci ea conţine o pereche de părţi stânga X_1, X_2 distincte şi echivalente cu proprietatea că $X_1 \rightarrow\bullet X_2$. Cum toate *FD* au atributele închise, rezultă că $X_1 \rightarrow X_1^+$ este redundantă în Γ

q.e.d.

Teorema 73 garantează corectitudinea următorului algoritm pentru calculul acoperirii minimale a unei mulţimi de *FD*[31]:

Algoritmul 74 (*calculul acoperirii minimale a unei mulţimi de FD*)

Intrare: o mulţime de *FD* $\Gamma = \{ X_i \rightarrow Y_i \mid 1 \leq i \leq n \}$

Ieşire:

o mulţime de *FD* Δ minimă, echivalentă cu Γ

Început:

$\Delta = \{ X_i \rightarrow X_i^+ \mid 1 \leq i \leq n \}$;

repetă pentru (i=1; $i \leq n$; i=i+1)

dacă ($X_i \rightarrow X_i^+$ este redundantă în Δ)

atunci $\Delta = \Delta - \{ X_i \rightarrow X_i^+ \}$;

sfârşit dacă;

sfârşit repetă;

[31] Evident că (vezi problema 79) acest algoritm este executabil în $O(|\Gamma| \times \|\Gamma\|)$.

Sfârşit algoritm 74;

Figura 27: Un algoritm de calcul al acoperirilor minimale

Exemplul 29

Fie subuniversul de interes compus din următoarele atribute: *#Angajat, Nume, Prenume, Categorie, Salariu, Departament, Timp, Şef, Lucrare* şi următoarea mulțime de *FD* (cu atributele prescurtate la prima lor literă): $\Gamma = \{A \rightarrow \rightarrow NS,\ NP \rightarrow AŞD,\ AN \rightarrow PCD,\ C \rightarrow S,\ D \rightarrow Ş,\ Ş \rightarrow D,\ ALD \rightarrow TA,\ NPCL \rightarrow T\}$. Aplicând algoritmul 74 asupra Γ se obține după prima instrucțiune: $\Delta = \{A \rightarrow \rightarrow ANPCSDŞ,\ NP \rightarrow ANPCSDŞ,\ AN \rightarrow ANPCSDŞ,\ C \rightarrow CS,\ D \rightarrow DŞ,\ Ş \rightarrow DŞ,\ ALD \rightarrow TLANPCSDŞ,\ NPCL \rightarrow TLANPCSDŞ\}$; apoi, în bucla *repetă* sunt eliminate pe rând *FD* redundante: întâi cea având partea stângă *AN*, iar apoi penultima, având partea stângă *ALD*[32]; rezultă acoperirea $\Delta = \{A \rightarrow ANPCSDŞ,\ NP \rightarrow ANPCSDŞ,\ C \rightarrow CS,\ D \rightarrow DŞ,\ Ş \rightarrow DŞ,\ NPCL \rightarrow TLANPCSDŞ\}$.

Ne îndreptăm acum atenția asupra acoperirilor optime. Deşi din acest punct de vedere contează dimensiunea *FD* şi nu numărul lor, propoziția următoare ne arată că din orice acoperire optimă se poate obține una echivalentă care să fie şi minimală.

Propoziția 75 Dacă o mulțime Γ de *FD* este optimă, atunci ea admite o acoperire minimală echivalentă.

Demonstrație:

Dacă Γ nu este minimală, atunci o putem transforma într-o mulțime echivalentă conform sugestiilor oferite de partea *dacă* a lemei 72, obținând astfel o acoperire cu dimensiune mai mică (căci $X_1 \rightarrow Y_1$, $X_2 \rightarrow Y_2$ sunt înlocuite cu $X_2 \rightarrow Y_1Y_2$), ceea ce contrazice ipoteza optimalității

q.e.d.

Din nefericire, din punctul de vedere al complexității, calculul acoperirilor optime este exponențial, aşa cum vom demonstra în curând[33]. Pentru aceasta însă, este mai uşor să demonstrăm întâi următorul rezultat, interesant şi doar în sine.

Teorema 76 Date fiind o schemă de relație, o mulțime de *FD* şi un întreg *k*, problema găsirii unei superchei de cardinalitate cel mult *k* este *NP*-completă.[34]

Demonstrație:

Apartenența acestei probleme la clasa *NP* se poate demonstra uşor cu ajutorul următoarei tehnici: arătăm că una dintre problemele *NP*-complete fundamentale (şi anume: *problema acoperirii nodurilor* unui graf) poate fi redusă în timp polinomial la problema noastră curentă.

Problema acoperirii nodurilor este următoarea: dat fiind un graf $G=(N,A)$ şi un întreg *k*, se cere găsirea unei acoperiri pentru noduri de cardinalitate cel mult *k*, adică a

[32] Aceasta are loc fiindcă ultimele două *FD* sunt echivalente şi $\psi = \{A \rightarrow ANPCSDŞ,\ NP \rightarrow ANPCSDŞ,\ C \rightarrow CS,\ D \rightarrow DŞ,\ Ş \rightarrow DŞ,\ ALD \rightarrow TLANPCSDŞ\}$ este o acoperire minimală, chiar optimă în raport cu Δ (căci dimensiunea ei este cu 1 mai mică); algoritmul 74 însă elimină redundanțele în ordinea găsirii lor.

[33] Pentru înțelegerea demonstrațiilor următoarelor două teoreme sunt necesare noțiuni de complexitatea calculului legate de *NP-completitudine* şi de tehnicile fundamentale de demonstrare a faptului că o problemă este *NP-grea* (i.e. nu admite soluții de complexitate polinomială; vezi pentru aceasta, de exemplu, [87]). De notat însă că cititorul poate ignora demonstrația lor, ba chiar şi aceste teoreme în sine, fără a pierde esența firului logic al expunerii.

[34] De remarcat că modelul relațional pune la dispoziție un algoritm pentru găsirea *unei* chei pentru orice relație (vezi problema 51); în practica modelării conceptuale a datelor însă, având în vedere, pe de-o parte, faptul că trebuie specificate în model *toate* cheile existente (altfel modelarea nu este corectă), iar pe de altă parte faptul că minimalitatea injectivă ține de semantica funcțiilor, *proiectantul de bd poate fi cel mult asistat, dar niciodată înlocuit de un algoritm de specificare a cheilor*!

unei submulțimi $M \subseteq N$ cu proprietatea că, pentru fiecare arc $\{i,j\} \in A$, cel puțin unul dintre nodurile *i* sau *j* aparțin lui *M*.

Vom arăta acum, folosind problema acoperirii nodurilor, că se poate decide în timp polinomial dacă o mulțime dată de atribute având cardinalitate $j \leq k$ este sau nu o supercheie; dat fiind un graf $G=(N,A)$ cu $card(N)=n$ considerăm următoarea schemă de bd și următoarea mulțime de *FD*:

(i) schema $R(U)$, unde U conține câte un atribut A_i pentru fiecare nod din N și câte un atribut $B_{i,j}$ pentru fiecare arc $\{i,j\} \in A$

(ii) mulțimea de *FD* Γ conținând $Z \rightarrow A_1 \ldots A_n$, unde $Z = \{B_{i,j} \mid \{i,j\} \in A\}$ și câte două *FD* $A_i \rightarrow B_{i,j}$, respectiv $A_j \rightarrow B_{i,j}$ pentru fiecare arc $\{i,j\} \in A$.

Susținem că o mulțime $M = \{h_1, \ldots, h_k\} \subseteq N$ constituie o acoperire a nodurilor grafului *G* dacă și numai dacă mulțimea de atribute $X = A_{h1} \ldots A_{hk}$ este o supercheie pentru $R(U)$, ceea ce, evident, completează și demonstrația teoremei 76.

Demonstrație:

($\Leftarrow$)(*și numai dacă*) Dacă $M = \{h_1, \ldots, h_k\}$ este o acoperire a nodurilor, atunci pentru orice arc $\{i,j\} \in A$, cel puțin unul dintre nodurile *i* sau *j* aparțin lui *M*; ca atare, pentru fiecare $B_{i,j}$, cel puțin unul dintre atributele A_i sau A_j aparține lui *X*, deci $Z \subseteq X^+$; cum $Z \rightarrow U \in \Gamma$, rezultă că $X^+ = U$.

($\Rightarrow$)(*dacă*) Fie $X = A_{h1} \ldots A_{hk}$ o supercheie a lui $R(U)$; ca atare, executând algoritmul 58 de calcul a închiderii lui *X* în raport cu Γ, toate atributele din *U* sunt incluse în X^+; dacă $X = A_1 \ldots A_n$, atunci $M = N$ și *M* constituie în mod trivial o acoperire. În caz contrar, fie $A_p \in U - XZ$; datorită modului de construcție a lui Γ, algoritmul 58 poate include A_p în X^+ doar dacă $Z \subseteq X^+$. Ca atare, pentru orice arc $\{i,j\} \in A$, $B_{i,j} \in X^+$ și deci cel puțin unul dintre A_i sau A_j trebuie să aparțină *X* (altfel $B_{i,j}$ nu ar fi inclus în X^+, căci $A_i \rightarrow B_{i,j}$, respectiv $A_j \rightarrow B_{i,j}$ sunt singurele *FD* având $B_{i,j}$ drept parte dreaptă). Rezultă, având în vedere și modul de construcție al *X*, că cel puțin unul dintre nodurile *i* sau *j* aparține lui *M*, $\forall \{i,j\} \in A$; aceasta demonstrează însă tocmai faptul că *M* este o acoperire a nodurilor grafului *G*

q.e.d.

Teorema 77 Problema găsirii pentru o mulțime de *FD* a unei acoperiri de dimensiune cel mult *k* este *NP*-completă.

Demonstrație:

Problema este în *NP* deoarece, dată fiind o mulțime Γ de *FD*, putem decide în timp polinomial dacă o mulțime Δ constituie o acoperire pentru Γ și are dimensiunea cel mult *k*; vom dovedi însă că, dacă ar exista un algoritm polinomial pentru această problemă, atunci ar exista și un algoritm polinomial pentru problema găsirii unei superchei minimale; dar, conform teoremei 76, aceasta din urmă este *NP*-completă, de unde rezultă că și problema curentă este *NP*-grea.

Fie Γ o mulțime de *FD* peste o schemă $R(U)$, *A* și *B* două atribute ce nu aparțin *U*, iar $\Delta = \Gamma \cup \{AU \rightarrow B\}$ o mulțime de *FD* peste schema $R(UAB)$. Susținem că orice acoperire optimă Ψ a Δ conține exact o *FD* în care apare *B* și anume $KA \rightarrow KB$, unde *K* este o cheie minimă pentru $R(U)$.

Demonstrație:

Fie Ψ o acoperire optimă a Δ; în mod evident, ea conține cel puțin o *FD* a cărei parte dreaptă include *B* (în caz contrar, cum $B \notin U$, Ψ nu ar implica $AU \rightarrow B$). Dacă Ψ conține o *FD* de forma $X \rightarrow BY$, atunci *X* trebuie să conțină *A* (căci toate *FD* netriviale din Δ^+ au *B* în partea dreaptă doar dacă îl au pe *A* în partea stângă) și, de asemenea, o supercheie *K'* a lui $R(U)$ (altfel ar fi implicate și alte *FD* decât cele din Δ^+). Cum $B \notin X$,

rezultă că $X = AK'$, cu K' supercheie a lui $R(U)$ (altfel, din nou, ar fi implicate şi alte *FD* decât cele din Δ^+). Rezultă că, dacă ar exista mai mult de o *FD* incluzând *B*, acestea ar putea fi cel puțin reduse (deoarece toate au în partea stângă pe *A* şi o cheie a lui $R(U)$), ceea ce ar contrazice ipoteza optimalității Ψ. Fie acum $AK' \rightarrow B$ unica *FD* având *B* în partea dreaptă: dacă există o cheie *K* mai mică decât *K'*, atunci am putea obține o acoperire cu dimensiune mai mică, înlocuind $AK' \rightarrow B$ cu $AK \rightarrow B$, ceea ce, evident, încheie demonstrația afirmației făcute

q.e.d.

O consecință imediată a afirmației tocmai demonstrate este aceea că, dacă ar exista un algoritm polinomial pentru problema acoperirii optime, atunci ar exista un algoritm polinomial şi pentru problema cheilor minimale, ceea ce contrazice însă *NP*-completitudinea acesteia (demonstrată de teorema 76)

q.e.d.

Cum problema găsirii de acoperiri optime s-a dovedit netratabilă, are sens să cercetăm dacă acoperirile minimale sunt într-adevăr cele mai bune acoperiri ce pot fi obținute. Vom vedea că se pot calcula în timp polinomial acoperiri minimale de o formă specială, care sau au dimensiunea mai mică decât cele calculate de algoritmul 74 sau se bucură de alte proprietăți suplimentare. Pentru aceasta însă trebuie introduse câteva alte noi concepte.

Definiția 78 O *mulțime* minimală Γ de *FD* se zice *L-minimă* dacă nu există nici o *FD* $X \rightarrow Y \in \Gamma$ cu proprietatea că Γ este echivalent cu $\Gamma - \{X \rightarrow Y\} \cup \{Z \rightarrow Y\}$, unde $Z \subset X$; atributele din $X - Z$ se zic *neesențiale*.

Definiția 79 O *mulțime* minimală Γ de *FD* se zice *LR-minimă* dacă nu există nici o *FD* $X \rightarrow Y \in \Gamma$ cu proprietatea că Γ este echivalent cu $\Gamma - \{X \rightarrow Y\} \cup \{X \rightarrow W\}$, unde $W \subset Y$.

Definiția 80 O *mulțime LR-minimă* Γ de *FD* se zice *circulară* dacă pentru orice clasă de echivalență γ a părților stânga din Γ există o ordonare $Y_0, Y_1, ..., Y_k$ a elementelor din γ astfel încât Γ conține *FD* $Y_0 \rightarrow ZW_1$, $Y_1 \rightarrow W_2$, ..., $Y_{k-1} \rightarrow W_k$, $Y_k \rightarrow W_0$, unde $W_h \subseteq Y_h$, $\forall h, 0 \leq h \leq k$.

Evident, introducerea acoperirilor L-minimale şi LR-minimale este motivată de reducerea dimensiunii, în timp ce circularitatea impune o structurare a *FD* în vederea obținerii unei sporite claritățI: într-o mulțime de *FD* LR-minimală circulară, pentru fiecare clasă de echivalență a părților stânga $\{Y_0, Y_1, ..., Y_k\}$, există cel mult o *FD* a cărei parte dreapta conține atribute ce nu fac parte din clasa de echivalență respectivă.

Pentru orice mulțime de *FD* se poate obține în mod trivial o acoperire satisfăcând oricare dintre cele trei definiții de mai sus. Mai mult, fiecare dintre aceste trei noțiuni suplimentare de minimalitate este mai puternică decât precedenta. Următorul exemplu arată că această ordonare este adevărată.

Exemplul 30

Mulțimea $\{A \rightarrow B, B \rightarrow A, ABC \rightarrow D\}$ este minimală, dar nu şi L-minimă; ea este însă echivalentă cu $\{A \rightarrow B, B \rightarrow A, AC \rightarrow D\}$ care este L-minimă.

Mulțimea $\{A \rightarrow BC, B \rightarrow C\}$ este L-minimă, dar nu este şi LR-minimă; ea este însă echivalentă cu $\{A \rightarrow B, B \rightarrow C\}$ care este LR-minimă.

Mulțimea $\{A \rightarrow BC, B \rightarrow AD\}$ este LR-minimă, dar nu este şi circulară, căci *A* şi *B* sunt părți stânga echivalente, în timp ce părțile dreapta conțin *C*, respec-tiv *D*; ea este însă echivalentă cu mulțimea circulară $\{A \rightarrow BCD, B \rightarrow A\}$.

Mulțimea $\{A \to B, B \to A, AC \to BD, BD \to AC\}$ este L-minimă, dar nu este şi ciruară, deoarece nu este LR-minimă; ea este însă echivalentă cu mulțimea circulară $\{A \to B, B \to A, AC \to D, BD \to C\}$.

Algoritmul 81 (vezi figura 28) calculează acoperiri circulare LR-minime.

Algoritmul 81 (*calculul acoperirilor circulare LR-minime*)

Intrare: o mulțime de *FD* Γ

Ieşire:

o mulțime circulară LR-minimă de *FD* Δ, echivalentă cu Γ

Început:

Calculează cu algoritmul 74 o acoperire minimală Δ a Γ;

repetă pentru fiecare $(X \to X^+ \in \Delta)$

dacă $(\exists Y, Y \subset X$, a.î. $Y^+ = X^+)$

atunci $\Delta = \Delta - \{ X \to X^+ \} \cup \{Y \to Y^+\}$;

sfârşit dacă;

sfârşit repetă;

repetă pentru fiecare clasă de echivalență γ a părților stânga din Γ

(fie $\gamma = \{Y_0, Y_1, ..., Y_k\}$ şi $Z = Y_0^+ - Y_0 Y_1 ... Y_k$)

$\Delta = \Delta - \{Y_0 \to Y_0^+, Y_1 \to Y_1^+, ..., Y_k \to Y_k^+\} \cup \{Y_0 \to ZY_1, Y_1 \to Y_2, ..., Y_k \to Y_0\}$;

sfârşit repetă;

repetă pentru fiecare $X \to Y \in \Delta$

dacă ($\Gamma - \{X \to Y\} \cup \{X \to W\}$ este echivalent cu Δ, $W \subset Y$)

atunci $\Delta = \Delta - \{ X \to Y \} \cup \{X \to W\}$;

sfârşit dacă;

sfârşit repetă;

Sfârşit algoritm 81;

Figura 28: Un algoritm de calcul al acoperirilor LR-minime circulare

Exemplul 31

Aplicând algoritmul 81 asupra acoperirii minimale obținute în exemplul 29, se obține una dintre următoarele două acoperiri circulare:

$\Delta = \{ A \to NPCD, NP \to A, C \to S, D \to Ş, Ş \to D, AL \to T \}$ sau

$\Delta' = \{ A \to NP, NP \to ACD, C \to S, D \to Ş, Ş \to D, AL \to T \}$.

4.4 Tablouri şi algoritmi de vânare pentru funcțional dependențe

Tablourile sunt, în esență, relații ce pot conține şi *variabile* drept valori ale atributelor. Ca atare, un tablou abstractizează relații sau, dimpotrivă, posibile subrelații. Formalizarea abstractizărilor se face cu ajutorul *aplicațiilor de conținere* (vezi definiția 84).

Figura 29 prezintă un exemplu de tablou precum şi o relație obținută din acesta prin înlocuirea variabilelor cu constante.

T

Student	*LocNaştere*	*Țara*
Vlad	v_L	n_1
Miron	v_L	România
Raphaella	N_2	n_3
v_S	Bucureşti	n_4
Diana	București	n_5

r

Student	*LocNaştere*	*Țara*
Vlad	Bucureşti	România
Miron	Bucureşti	România
Raphaella	Auckland	Noua Zeelandă
Eric	Bucureşti	România
Diana	București	România

Figura 29: Un tablou şi o relație

Vânarea se bazează pe următoarea observație: dacă relația reprezentată de un tablou trebuie să satisfacă anumite constrângeri, atunci variabilele acestuia nu pot reprezenta constantele în mod liber. De exemplu, dacă relația din figura 29 trebuie să satisfacă *LocNaştere* $\rightarrow$ *Țara*, atunci putem deduce că variabila n_1 trebuie să aibă valoarea „România”, iar variabilele n_4 şi n_5 trebuie să reprezinte o aceeaşi valoare. Ca atare, vânarea este un algoritm ce transformă tablourile conform unei mulțimi de constrângeri. De exemplu, vânarea tabloului *T* din figura 29 în raport cu *FD* *LocNaştere* $\rightarrow$ *Țara* conduce la obținerea tabloului *T'* din figura 30.

T'

Student	*LocNaştere*	*Țara*
Vlad	v_L	România
Miron	v_L	România
Raphaella	n_2	n_3
v_S	Bucureşti	n_4
Diana	București	n_4

Figura 30: Un tablou vânat

Din motive ce vor deveni evidente pe parcurs, variabilele tablourilor se împart în următoarele două categorii:

1. *variabile distinse* (*vd*), câte una per atribut (notată v_A pentru atributul *A*)
2. *variabile nedistinse* (*vnd*): n_1, n_2, ..., n_k, ... (numărabile).

Definiția 82 Dat fiind o mulțime de atribute *X*, se zice *linie* peste *X* o funcție ce asociază fiecărui atribut $A \in X$:

(1) o constantă (din *dom*(*A*)) ori
(2) variabila distinsă v_A ori
(3) o *vnd*.

Evident că linia este o generalizare a tuplului: o linie fără variabile este un tuplu. Ca atare, notațiile adoptate pentru valorile tuplilor şi/sau subtuplilor se păstrează (*l*[*A*] etc.).

Definiția 83 Se zice *tablou* peste o mulțime de atribute *X* şi se notează *T*(*X*) o mulțime de linii peste *X* cu proprietatea că nici o *vnd* nu apare ca valoare a mai multor atribute.[35]

Peste diverşii simboli ce pot apare drept valori ale atributelor unui tablou se defineşte următoarea ordine parțială $\preccurlyeq$:

- Constantele nu sunt comparabile între ele şi preced toate variabilele: pentru orice constantă *a* şi variabilă *v* are loc $a \preccurlyeq v$
- Toate variabilele distinse preced toate *vnd*
- *vnd* sunt total ordonate conform ordinii indicilor: $n_i \preccurlyeq n_j$ dacă şi numai dacă $i \leq j$.

  Scriem $s_1 \prec s_2$ dacă $s_1 \preccurlyeq s_2$ şi $s_1 \neq s_2$.

[35] În literatură, tablourile definite cu această restricție se mai zic şi *tipate*.

Definiția 84 Fie T_1 si T_2 două tablouri peste aceeași mulțime de atribute X; se zice *aplicație de conținere* o funcție $\varphi : T_1 \to T_2$ cu următoarele două proprietăți:

1. pentru orice simbol s apărând în T_1, $\varphi(s) \preccurlyeq s$.
2. $\varphi(T_1) \subseteq T_2$ (i.e. $\forall t_1 \in T_1$, $\exists t_2 \in T_2$, a.î. $\forall A \in X$, $\varphi(t_1[A]) = t_2[A]$).

Evident, conform definiției ordinii parțiale, prima condiție din definiția de mai sus înseamnă că φ „duce":

- constantele în ele însele (i.e. φ este identitatea peste constante)
- variabilele distinse în ele însele sau în constante şi
- *vnd* ori într-o constantă, ori în variabila distinsă a atributului respectiv, ori într-o *vnd* cu indice inferior.

Reamintindu-ne noțiunea intuitivă a reprezentării relațiilor prin tablouri, se poate spune că un tablou T reprezintă o relație r (sau o submulțime a ei) dacă există o aplicație de conținere de la T la r. De exemplu, date fiind tabloul şi relația din figura 29, aplicația de conținere corespunzătoare este următoarea:

$\varphi(v_S)$ = Eric $\varphi(n_2)$ = Auckland $\varphi(n_5)$ = România
$\varphi(v_L)$ = Bucureşti $\varphi(n_3)$ = Noua Zeelandă
$\varphi(n_1)$ = România $\varphi(n_4)$ = România

Similar, o funcție care face să-i corespundă România lui n_1 şi pe n_4 lui n_5, iar pentru restul simbolilor tabloului T din figura 29 este identitatea, e o aplicație de conținere între acest tablou şi tabloul T' din figura 30. Evident că aplicațiile de conținere se pot compune: de exemplu, problema 22 se referă la tranzitivitatea lor.

Definiția 85 Figura 31 prezintă *algoritmul de vânare a unui tablou conform unor FD*; scopul său este transformarea unui tablou oarecare *T*, conform unei mulțimi *F* de *FD*, într-un tablou satisfăcând *F* (se mai zice că *T este vânat conform F* sau, simplu, dacă *F* este subînțeles sau irelevant, că *se aplică vânarea asupra lui T* sau că *T este vânat*).[36]

Algoritmul 86 (*vânarea unui tablou conform unei mulțimi de FD*)
Intrare: un tablou *T* şi o mulțime de *FD F* având câte un singur atribut în partea dreaptă
Ieşire:
un tablou $CHASE_F(T)$, satisfăcând *F*
Început:
$CHASE_F(T) = T$;
repetă pentru fiecare ($X \to A \in F$ şi $t_1, t_2 \in CHASE_F(T)$, cu $t_1[X] = t_2[X]$ şi $t_1[A] \prec t_2[A]$)
înlocuieşte toate aparițiile lui $t_2[A]$ din $CHASE_F(T)$ cu $t_1[A]$;
sfârşit repetă;
Sfârşit algoritm 86;

Figura 31: Algoritmul de vânare pentru FD

Definiția 87 Execuțiile instrucțiunii constituind corpul buclei algoritmului 86 se zic *paşii vânării*; dacă într-un asemenea pas sunt implicate *FD f* şi liniile t_1 şi t_2, se zice că *se aplică f* (asupra t_1 şi t_2). De remarcat că, atunci când într-un pas se modifică valoarea unei linii *t* pentru un atribut *A* din *v* în *w*, toate aparițiile lui *v* din tablou sunt schimbate în *w*; ca atare, se zice că *un pas schimbă v în w*, iar *o secvență de paşi schimbă pe* v_1 *în*

[36] Satisfacerea *FD* de către un tablou se definește, evident, similar ca pentru o relație.

w_1, v_2 *în* w_2, ... şi aşa mai departe. Se mai zice că o secvență de paşi *identifică* v_1 şi v_2, dacă îi schimbă pe amândoi într-o aceeaşi valoare w sau dacă schimbă pe v_1 în v_2 ori pe v_2 în v_1.

Exemplul 32

Să vânăm tabloul T din figura 29 în raport cu $F = \{Student \rightarrow LocNaştere, LocNaştere \rightarrow Țara\}$. Este posibilă aplicarea *FD LocNaştere* $\rightarrow$ *Țara* asupra primelor două linii, schimbând n_1 în "România" şi apoi asupra ultimelor două linii, schimbând n_5 în n_4; cum nici o altă aplicare nu mai este posibilă, algoritmul se termină calculând tabloul din figura 30. Secvența sa de paşi a identificat n_1 cu "România" şi pe n_5 cu n_4.

Evident că algoritmul 86 se termină întotdeauna pentru tablouri finite: numărul de simboli dintr-un asemenea tablou este finit, iar valoarea fiecărei linii pentru fiecare atribut în parte se poate modifica doar de un număr finit de ori (deoarece, pentru orice simbol v, există doar o mulțime finită de simboli care îl preced); rezultă trivial că instrucțiunea constituind corpul buclei nu se execută decât de un număr finit de ori.

Aplicarea vânării asupra anumitor tablouri poate doar detecta violarea *FD*, fără a putea însă modifica valorile implicate, deoarece acestea nu sunt comparabile.

Exemplul 33

Fie tabloul din figura 32 şi $F = \{Student \rightarrow LocNaştere, LocNaştere \rightarrow Țara\}$. Este posibilă aplicarea *Student*$\rightarrow$*LocNaştere* care va schimba n_1 în "Bucureşti". Cele două linii vor avea în consecință aceleaşi valori pentru *LocNaştere*, dar valori diferite pentru *Țara*; cum acestea nu sunt comparabile, tabloul rezultat în urma vânării (care este o relație!) violează *F*.

T

Student	*LocNaştere*	*Țara*
Vlad	Bucureşti	România
Vlad	n_1	Franța

Figura 32: Un tablou conținând o violare dură a unei FD

Este, de aceea, utilă diferențierea între două tipuri de violări ale *FD* de către tablouri:

Definiția 88 Dată fiind o *FD* $X \rightarrow A$, două linii având aceleaşi valori pentru X, dar valori diferite pentru A pot conduce la una din următoarele două tipuri de violări:

1. o *violare uşoară* a $X \rightarrow A$, dacă valorile corespunzătoare lui A sunt comparabile
2. o *violare dură* a $X \rightarrow A$, dacă valorile corespunzătoare lui A nu sunt comparabile.

Conform definiției $CHASE_F(T)$, rezultă că un asemenea tablou nu poate conține violări uşoare; rezultă că dacă $CHASE_F(T)$ violează F, atunci el conține una sau mai multe violări dure.

De notat şi că, în general, algoritmul 86 este nedeterminist, în sensul libertății în alegerea ordinii de considerare atât a *FD*, cât şi a liniilor vânate în predicatul buclei, ceea ce poate conduce la diverse modificări şi deci la diverse rezultate.

Exemplul 34

Fie tabloul din figura 33; vânarea sa în raport cu $F = \{Student \rightarrow LocNaştere, LocNaştere \rightarrow Țara\}$ poate începe cu aplicarea *FD Student*$\rightarrow$*LocNaştere* primei şi celei de-a treia linii, schimbând astfel v_L în „Bucureşti", atât în prima, cât şi în cea de-a doua linie. Apoi, se poate aplica *FD LocNaştere*$\rightarrow$*Țara* ori primei şi celei de-a doua linii, ori primei şi celei de-a treia. În primul caz, n_1 este schimbat în „Noua Zeelandă", în timp ce în al doilea el este schimbat în „România"; aceasta produce ca rezultat al vânării prima, respectiv a doua relație din figura 34. De notat însă că ambele violează *FD LocNaştere*$\rightarrow$*Țara* şi că, evident, în ambele cazuri vânarea nu mai poate continua.

Student	*LocNaştere*	*Ţara*
Vlad	v_L	n_1
Raphaella	v_L	Noua Zeelandă
Diana	Bucureşti	România

Figura 33: Tabloul iniţial pentru exemplul 34

Student	*LocNaştere*	*Ţara*
Vlad	Bucureşti	Noua Zeelandă
Raphaella	Bucureşti	Noua Zeelandă
Diana	Bucureşti	România

Student	*LocNaştere*	*Ţara*
Vlad	Bucureşti	România
Raphaella	Bucureşti	Noua Zeelandă
Diana	Bucureşti	România

Figura 34: Relaţiile obţinute prin vânarea tabloului din figura 33

Din perspectiva acestui exemplu, comportarea algoritmului 86 ar putea părea inacceptabilă. Din fericire, nedeterminismul apare doar în conjuncţie cu violări dure ale *FD*. În cele ce urmează, ca atare, formalizăm următoarea afirmaţie: dacă o execuţie a algoritmului 86 asupra *T*, conform *F*, produce un tablou ce satisface *F*, atunci orice altă execuţie a sa produce acelaşi tablou; dacă, dimpotrivă, o execuţie a sa produce un tablou ce violează *F*, atunci orice altă execuţie produce un tablou ce violează *F* (chiar dacă nu neapărat mereu acelaşi tablou). Pentru aceasta, noi concepte sunt însă necesare.

Definiţia 89 Două *valori v* şi *w* se zic *direct egalabile* în raport cu *F* într-un tablou *T*, dacă *T* conţine două linii t_1 şi t_2 cu proprietatea că $\exists\ X \rightarrow A \in F$ astfel încât $t_1[X] = t_2[X]$, $t_1[A] = v$ şi $t_2[A] = w$.

Din definiţia de mai sus, rezultă că dacă două linii sunt direct egalabile, atunci ele sunt ori identice, ori provoacă o violare (uşoară sau dură) a *F*.

Definiţia 90 Două *valori v* şi *w* se zic *egalabile* în raport cu *F* într-un tablou *T*, dacă:

1. ele sunt direct egalabile sau
2. $\exists u \in T$ astfel încât atât (u,v), cât şi (u,w) sunt direct egalabile sau
3. *T* conţine două linii t_1 şi t_2, cu proprietatea că $t_1[A] = v$, $t_2[A] = w$ şi $\exists\ X \rightarrow A \in F$ astfel încât $\forall B \in X$, $t_1[B]$ şi $t_2[B]$ sunt direct egalabile.

Din definiţia de mai sus, rezultă că oricare ar fi tabloul $T(U)$ şi atributul $A \in U$, *egalabilitatea* este o relaţie de echivalenţă pentru *A*-valorile din *T* (vezi problema 80). De asemenea, mai rezultă şi că paşii vânării păstrează egalabilitatea, adică, dacă *v* şi *w* sunt egalabile în *T* iar o secvenţă de paşi transformă *T* în *T'* schimbând *v* în *v'* şi *w* în *w'*, atunci şi *v'* şi *w'* sunt egalabile în *T'*. Mai mult, dacă două valori sunt egalabile în *T'*, atunci ele apar şi sunt egalabile şi în *T*. În sfârşit, dacă două valori sunt identificate de o succesiune de paşi ai unei vânări a *T*, atunci ele sunt egalabile în *T*.

Lema 91 Fie *T* un tablou şi *F* o mulţime de *FD*; dacă două valori sunt egalabile în *T* în raport cu *F* dar nu sunt egale, atunci *T* conţine o violare a *F* (uşoară sau dură).

Demonstraţie:

Asignăm fiecărei perechi (v,w) de simboli egalabili un grad de egalabilitate $grd(v,w)$ conform celor trei cazuri ale definiţiei 90, astfel:

1. $grd(v,w) = 1$
2. $grd(v,w) = grd(u,v) + grd(u,w)$
3. $grd(v,w) = 1 + \Sigma_{B \in X}\, grd(t_1[B], t_2[B])$.

Să considerăm perechea de simboli egalabili (v,w) având cel mai mic grad *g*; distingem trei cazuri posibile:

1. *v* şi *w* sunt direct egalabili şi distincţi: în mod trivial, lema este demonstrată
2. $\exists u \in T$ astfel încât atât (u,v), cât şi (u,w) sunt direct egalabile cu un grad mai mic decât *g*: ca atare, $u = v$, $u = w$ şi deci $v = w$, ceea ce contrazice ipoteza; rezultă că acest caz nu este posibil

3. există în T două linii t_1 şi t_2 astfel încât $t_1[A] = v$, $t_2[A] = w$ şi $\exists\ X \to A \in F$ cu proprietatea că $\forall B \in X$, $t_1[B]$ şi $t_2[B]$ sunt direct egalabile cu un grad mai mic decât g: ca atare, $t_1[X] = t_2[X]$ şi deci t_1 şi t_2 violează $X \to A$

q.e.d.

Teorema 92 Fie T un tablou, F o mulțime de *FD* şi execuțiile algoritmului 86 cu intrările T şi F;

1. dacă o execuție generează un tablou ce violează F, atunci orice execuție generează un tablou ce violează F
2. dacă o execuție generează un tablou ce satisface F, atunci orice execuție generează acelaşi tablou.

Demonstrație:

Egalabilitatea fiind o relație de echivalență (vezi problema 80), pentru orice atribut A putem împărți simbolii ce apar drept valori ale lui A în clase de echivalență. Arătăm că:

(i) orice execuție a algoritmului 86 produce un tablou ce nu satisface F dacă şi numai dacă există o clasă de echivalență ce conține doi simboli incomparabili (i.e. do-uă constante distincte); şi că

(ii) dacă o execuție produce un tablou ce satisface F, atunci ea identifică fiecare simbol cu simbolul minim al clasei sale de echivalență.

Evident, demonstrarea acestor două puncte este echivalentă cu demonstrarea teoremei.

(i) ($\Rightarrow$)(*dacă*)

Fie o clasă de echivalență conținând două constante distincte a şi b. Cum a şi b sunt egalabile în T dar nu pot fi identificate una cu cealaltă, conform păstrării egalabilității, rezultă că ele rămân egalabile şi în $CHASE_F(T)$, pentru orice execuție a algoritmului 86. Aceasta însă, conform lemei 91, înseamnă că $CHASE_F(T)$ violează F; cum $CHASE_F(T)$ nu poate conține violări uşoare, rezultă că $CHASE_F(T)$ conține o violare dură.

($\Leftarrow$)(*şi numai dacă*)

Dacă a şi b sunt constante distincte, ele sunt incomparabile; dacă ele conduc la o violare dură a $CHASE_F(T)$, atunci violarea are loc şi în T. Ca atare, ele aparțin aceleiaşi clase de echivalență.

(ii) Dacă $CHASE_F(T)$ satisface F, atunci el nu conține nici o pereche de simboli distincți egalabili. Ca atare, orice simbol al T a fost identificat cu un alt simbol tot din T, aparținând aceleiaşi clase de echivalență (căci dacă doi simboli au fost identificați, atunci ei erau egalabili în tabloul inițial); de asemenea, pentru fiecare clasă de echiva-lență, toți simbolii au fost identificați cu o aceeaşi valoare. Simbolii din fiecare clasă de echivalență sunt toți comparabili (în caz contrar, $CHASE_F(T)$ ar viola F conform (i) de mai sus!), deci există un minim între ei. Cum corpul buclei interne a algoritmului nu va schimba niciodată un simbol cu un altul care nu îl precede, rezultă că toți sim-bolii din fiecare clasă de echivalență în parte sunt identificați cu minimul clasei res-pective

q.e.d.

Teorema 92 este fundamentală în confirmarea utilității vânării: ea ne asigură că dacă rezultatul acesteia satisface *FD*, atunci el este unic.

Ne vom concentra atenția în cele ce urmează asupra conexiunilor dintre vânare şi aplicațiile de conținere. Întâi şi întâi, remarcăm că oricare ar fi T şi F, există o aplicație

de conținere între T și $CHASE_F(T)$ și anume funcția φ ce face să-i corespundă fiecărui simbol v al T, simbolul v' în care v este schimbat de vânarea T. De asemenea, deoarece aplicațiile de conținere sunt tranzitive (vezi problema 22), rezultă că dacă există o aplicație de conținere între $CHASE_F(T_1)$ și T_2, atunci există o asemenea aplicație și între T_1 și T_2.

Lema următoare prezintă o altă proprietate importantă a vânării, care este folosită în rezolvarea problemei implicației pentru *FD* în această abordare.

Lema 93 Fie T și T' tablouri, iar F o mulțime de *FD*; dacă există o aplicație de conținere φ de la T la T' și T' satisface F, atunci φ este, de asemenea, o aplicație de conținere de la $CHASE_F(T)$ la T', iar $CHASE_F(T)$ satisface F.

Demonstrație:

Fie $T_0, T_1, ..., T_k$, cu $T = T_0$ și $T_k = CHASE_F(T)$ tablourile generate succesiv în cursul execuției algoritmului 86; arătăm că, $\forall i$, $1 \le i \le k$:

1. există o aplicație de conținere de la T_i la T' și că
2. T_i nu conține nici o violare dură a F.

1. Pentru a demonstra această primă afirmație procedăm prin inducție după i. Baza inducției are loc în mod trivial. Presupunem apoi că există o aplicație de conținere de la T_{i-1} la T' și că pasul i schimbă o valoare v în w ca rezultat al aplicării unei *FD* $X \to A$ asupra liniilor t_1, t_2 (evident, $t_1[A] = u$, $t_2[A] = w$); fie $t_1' = \varphi(t_1)$ și $t_2' = \varphi(t_2)$; conform definiției aplicațiilor de conținere, $t_1, t_2 \in T'$ și $t_1'[X] = t_2'[X]$. Cum T' satisface F, rezultă că $t_1'[A] = t_2'[A]$ și deci că $\varphi(v) = = \varphi(w)$. Ca atare, deoarece singura diferență între T_{i-1} și T_i este aceea că toate aparițiile lui v au fost înlocuite cu w, urmează că φ este, de asemenea, o aplicație de conținere de la T_i la T'.
2. Să presupunem prin absurd că T conține o pereche de linii t_1 și t_2 astfel încât $t_1[A]$ și $t_2[A]$ sunt constante distincte, iar $t_1[X] = t_2[X]$; fie $t_1' = \varphi(t_1)$ și $t_2' = \varphi(t_2)$; conform definiției aplicațiilor de conținere, rezultă că $t_1'[X] = t_2'[X]$ și că $t_1'[A] = t_1[A] \ne t_2[A] = t_2'[A]$, ceea ce contrazice ipoteza că T' satisface F

q.e.d.

Definiția 94 Fie o schemă $R(U)$ și o *FD* $f = X \to Y$; se zice *tabloul lui* f și se notează cu T_f un tablou cu două linii, ambele conținând variabile distinse pentru X și *vnd* distincte pentru $U - X$.

Teorema 95 Fie o schemă $R(U)$, F o mulțime de *FD*, o *FD* $f = X \to Y$, iar T_f tabloul lui f; F implică f dacă și numai dacă cele două linii ale $CHASE_F(T_f)$ coincid pentru toate atributele din Y.

Demonstrație:

Întâi și întâi, remarcăm că, deoarece T_f nu conține constante, $CHASE_F(T_f)$ satisface F, oricare ar fi F și f.

($\Rightarrow$)(*dacă*)

Să presupunem că cele două linii ale $CHASE_F(T_f)$ coincid pentru Y; fie un conținut $r(U)$ satisfăcând F, iar t_1 și t_2 tupli ai r cu proprietatea că $t_1[X]=t_2[X]$. Susținem că $t_1[Y]= =t_2[Y]$, ceea ce confirmă că r satisface și f: fie ψ funcția care face să corespundă celor două linii ale T_f tuplul t_1, respectiv t_2; ψ este deci o aplicație de conținere de la T_f la $\{t_1, t_2\}$, deoarece este o funcție peste simboli (cele două linii coincid pentru X, iar $t_1[X] = = t_2[X]$) și face să corespundă constante variabilelor. Conform lemei 93, cum r satisface F și deci și $\{t_1, t_2\} \subseteq r$ satisface F, rezultă că ψ este o aplicație de conținere și de la $CHASE_F(T_f)$ la $\{t_1, t_2\}$. Ca atare, deoarece cele două linii ale $CHASE_F(T_f)$ coincid pentru Y, din definiția aplicațiilor de incluziune urmează că $t_1[Y] = t_2[Y]$.

($\Leftarrow$)(*și numai dacă*)

Dacă cele două linii ale $CHASE_F(T_f)$ nu coincid pentru Y, atunci orice conținut r cu doi tupli obținuți din $CHASE_F(T_f)$ făcând să corespundă fiecărei variabile o constantă distinctă violează f, deşi evident satisface F (deoarece $CHASE_F(T_f)$ satisface F) q.e.d.

Corolar 96 Cele două linii ale tabloului vânat $CHASE_F(T_{X \to Y})$ au aceeaşi valoare pentru un atribut $A \Leftrightarrow A \in X^+$ (demonstrația este propusă cititorului de problema 23).

Exemplul 35

Aplicăm teorema 95 pentru a verifica dacă mulțimea $F = \{BC \to DE, A \to B, AEG \to H\}$ implică $AC \to DE$ (vezi, pentru comparație, exemplul 22). Tablourile $T_{AC \to DE}$ şi $CHASE_F(T_{AC \to DE})$ sunt prezentate în figura 35 şi, evident, confirmă implicația. Aceleaşi tablouri pot fi însă, de exemplu, folosite şi pentru a demonstra că F nu implică $AC \to G$.

$T_{AC \to DE}$

A	B	C	D	E	G	H
v_A	N_1	v_C	n_2	n_3	n_4	n_5
v_A	N_6	v_C	n_7	n_8	n_9	n_{10}

$CHASE_F(T_{AC \to DE})$

A	B	C	D	E	G	H
v_A	n_1	v_C	n_2	n_3	n_4	n_5
v_A	n_1	v_C	n_2	n_3	n_9	n_{10}

Figura 35: Testarea implicației FD cu ajutorul vânării

Metoda sugerată de teorema 95 pentru a rezolva problema implicației se bazează pe următoarea idee intuitivă: date fiind F şi f, întâi se construieşte T_f, care este cel mai general tablou ce violează f, iar apoi se vânează T_f conform F; dacă rezultatul obținut violează f, atunci s-a obținut un contraexemplu ce dovedeşte că implicația este falsă; în caz contrar, conform definiției, rezultă că implicația este adevărată.

Evident, teorema 95 oferă posibilitatea de a decide în problema implicației pentru *FD* independent de existența regulilor de inferență pentru acestea: chiar omițând subsecțiunea 4.1, se pot demonstra cu ajutorul ei toate rezultatele importante cu privire la *FD*, inclusiv corectitudinea algoritmilor 56 şi 58.

4.5 Funcțional dependențe şi descompuneri fără pierderi

După cum am văzut deja în subsecțiunile 1.6, 2.7 şi 3.2, descompunerea fără pierderi este extrem de importantă în teoria relațională, deoarece metodele de proiectare a schemelor normalizate (vezi şi subsecțiunea 5.1) descompun relațiile cu ajutorul operatorului de proiecție. Relațiile $\pi_{Xi}(r)$, ..., $\pi_{Xm}(r)$ astfel obținute „reprezintă" relația inițială r dacă aceasta poate fi reobținută din joinul proiecțiilor, adică dacă descompunerea r conform acestor proiecții se face, conform definiției, fără pierderi (i.e. $\bowtie_{i=1}^{m}(\pi_{Xi}(r)) = r$).

Prima teoremă a acestei subsecțiuni caracterizează *descompunerile fără pierderi binare*; demonstrația ei rezultă imediat (vezi problema 81), drept corolar al teoremei 99 de mai jos (a cărei demonstrație, trivial, nu se bazează pe teorema 97).

Teorema 97 Fie o schemă $R(U)$, F o mulțime de *FD* peste U, iar $X, X_1, X_2 \in U$, astfel încât $X_1X_2 = U$ şi $X_1 \cap X_2 = X$; relațiile peste $R(U)$ au o descompunere fără pierderi în raport cu X_1, X_2 dacă şi numai dacă cel puțin una dintre *FD* $X \to X_1$ sau $X \to X_2$ aparțin F^+.

Exemplul 36

Fie schema *R*(*Student*, *LocNaştere*, *Țara*, *Disciplina*); toate conținuturile acestei relații ce satisfac $F = \{Student \to LocNaştere, LocNaştere \to Țara\}$ au o descompunere fără pierderi peste *Student*, *Disciplina* respectiv *Student*, *LocNaştere*, *Țara*, deoarece *Student*→*LocNaştere*,*Țara* este implicată de *F*. Un exemplu de asemenea relație este prezentat în figura 36. Pe de altă parte, nu toate conținuturile pot fi descompuse fără pierderi, de exemplu, peste *LocNaş-tere*, *Disciplina* respectiv *Student*, *LocNaştere*, *Țara*, deoarece nici

LocNaşte-re→Disciplina, nici *LocNaştere→Student,Ţara* nu sunt implicate de *F* (relaţia din figura 37 fiind un asemenea contraexemplu).

Student	*LocNaştere*	*Ţara*	*Disciplina*
Vlad	Bucureşti	România	Baze de date
Diana	Auckland	Noua Zeelandă	Baze de date
Vlad	Bucureşti	România	Limbaje formale
Miron	Bucureşti	România	Algoritmi

Figura 36: O relaţie decompozabilă binar fără pierderi

De remarcat că, evident, există relaţii ce admit descompuneri fără pierderi şi fără a satisface condiţiile necesare teoremei 97 (cele suficiente, desigur, fiind întotdeauna satisfăcute).

Exemplul 37

Figura 37 prezintă o relaţie ce admite o descompunere fără pierderi peste *LocNaştere*, *Disciplina* respectiv *Student*, *LocNaştere*, *Ţara*, deşi nu satisface nici *LocNaştere* → *Disciplina*, nici *LocNaştere* → *Student,Ţara*.

Student	*LocNaştere*	*Ţara*	*Disciplina*
Vlad	Bucureşti	România	Baze de date
Diana	Auckland	Noua Zeelandă	Algoritmi
Vlad	Bucureşti	România	Limbaje formale
Miron	Bucureşti	România	Limbaje formale
Miron	Bucureşti	România	Baze de date

Figura 37: O relaţie decompozabilă fără pierderi deşi nu satisface condiţiile necesare teoremei 97

Corolar 98 Fie $R(U)$ şi X, X_1, X_2 cu proprietăţile din teorema 97; un conţinut al $R(U)$ are o descompunere fără pierderi în raport cu X_1, X_2 dacă satisface cel puţin una dintre *FD* $X \to X_1$ sau $X \to X_2$ (demonstraţia este propusă cititorului de problema 24).

Să considerăm în continuare descompunerile *n*-are. Conform lemei 10, incluziunea $\bowtie_{i=1}^{m}(\pi_{Xi}(r)) \supseteq r$ are loc întotdeauna. În acest caz însă, nu există vreo caracterizare sintetică a descompunerilor fără pierderi, precum în cazul binar: cea mai simplă metodă necesită folosirea algoritmului de vânare.

Teorema 99 Fie o schemă $R(U)$, F o mulţime de *FD* peste U, iar $X_1, X_2, ..., X_n \in U$, astfel încât $X_1 ... Xn = U$. Fie T un tablou peste U cu n linii $l_1, l_2, ..., l_n$, cu proprietatea că $\forall i,\ 1 \le i \le n, \forall A \in U$, dacă $A \in X_i$, atunci $l_i[A] = v_A$, iar dacă $A \in U\text{-}X$, atunci $l_i[A]$ este o *vnd* distinctă. În aceste condiţii, toate relaţiile peste $R(U)$ ce satisfac F au o descompu-nere fără pierderi în raport cu $X_1, ..., X_n$ dacă şi numai dacă tabloul $CHASE_F(T)$ conţine o linie compusă numai din variabile distinse.

Demonstraţie:

(⇒)(*dacă*)

Să presupunem că $CHASE_F(T)$ conţine o linie doar cu variabile distinse şi fie un conţinut $r(U)$ satisfăcând F, iar $t \in \bowtie_{i=1}^{n}(\pi_{Xi}(r))$. Susţinem că $t \in r$ (ceea ce va demonstra această primă implicaţie, deoarece $\bowtie_{i=1}^{n}(\pi_{Xi}(r)) \supseteq r$ are loc întotdeauna).

Fie t_i, $\forall i,\ 1 \le i \le n$, tuplul din $\pi_{Xi}(r)$ ce contribuie la formarea t prin join, iar t_i' tuplul lui r din care t_i provine; fie φ funcţia care face să-i corespundă lui $l_i[A]$ pe $t_i'[A]$, $\forall i,\ 1 \le \le i \le n,\ \forall A \in U$. Evident, φ este o aplicaţie de conţinere de la T la $\{t_1', t_2', \ldots, t_n'\}$, deoarece dacă $l_i[A] = l_j[A]$, atunci $t_i'[A] = t_j'[A]$, iar dacă $A \ne B$, atunci $l_i[A] \ne l_j[B]$. De notat că variabilelor distinse le corespund valorile lui t şi deci, dacă s este o linie din $CHASE_F(T)$ compusă numai din asemenea variabile, atunci $\varphi(s) = t$. Cum $\{t_1', t_2', \ldots,$

t_n'} $\subseteq r$, iar r satisface F, conform lemei 93, rezultă că φ este o aplicație de conținere și de la $CHASE_F(T)$ la {t_1', t_2', …, t_n'}; ca atare, $\varphi(s) \in$ {t_1', t_2', …, t_n'}. Deoarece $\varphi(s)=t$, rezultă că $t \in$ {t_1', t_2', …, t_n'} și deci că $t \in r$.

(⇐)(*și numai dacă*)

Dacă $CHASE_F(T)$ nu conține nici o linie formată doar din variabile distinse, orice conținut r obținut din T prin înlocuirea fiecărei variabile cu o constantă distinctă are o descompunere cu pierderi peste X_1, X_2, ..., X_n: $\bowtie_{i=1}^{n}(\pi_{Xi}(r))$ conține tuplul ale cărui valori corespund variabilelor distinse din T, care tuplu nu aparține însă r. În același timp însă, deoarece T nu conține constante, atât $CHASE_F(T)$ cât și r satisfac F

q.e.d.

Exemplul 38

Fie schema R(*Student*, *LocNaştere*, *Disciplina*, *Profesor*) și mulțimea de *FD* F = {*Student*→*LocNaştere*, *Disciplina*→*Profesor*}. Cu ajutorul teoremei 99, putem demonstra că orice conținut r al R ce satisface F are o descompunere fără pierderi peste *Student*, *Disciplina*, respectiv *Student*, *LocNaştere*, *Profe-sor* și *LocNaştere*, *Disciplina*, *Profesor*. Figura 38 prezintă atât tabloul inițial, cât și cel obținut din acesta prin vânare.

T

Student	*LocNaştere*	*Profesor*	*Disciplina*
v_S	n_1	n_2	v_D
v_S	v_L	v_P	n_3
n_4	v_L	v_P	v_D

$CHASE_F(T)$

Student	*LocNaştere*	*Profesor*	*Disciplina*
v_S	v_L	n_2	v_D
v_S	v_L	v_P	v_D
n_4	v_L	v_P	v_D

Figura 38: O relație decompozabilă fără pierderi, deși nu satisface condițiile necesare teoremei 97

Din nou, ca și în cazul binar, condițiile necesare acestei teoreme (pentru cazul general) nu sunt întotdeauna obligatorii (vezi și problema 25).

4.6 Problema implicației pentru dependențele de incluziune

Am introdus în secțiunea 2.6 o versiune restrictivă a celor mai uzuale tipuri de constrângeri interrelaționale: dependențele de incluziune tipate. Vom discuta acum pe larg proprietățile generale ale acestei clase de constrângeri, precum și interacțiunea ei cu clasa funcțional dependențelor.

Definiția 100 Fie $\boldsymbol{R}$ = {$R_1(X_1)$, $R_2(X_2)$,..., $R_n(X_n)$} schema unei bd; $R_i[S_1] \subseteq R_j[S_2]$ se zice *dependență de incluziune* (*DIN*), unde $1 \le i, j \le n$, iar $S_1 = <A_{1,1}, ..., A_{1,p}>$ este o secvență de atribute distincte ale X_i și $S_2 = <A_{2,1}, ..., A_{2,p}>$ este o secvență de atribute distincte ale X_j[37]. Un conținut $\boldsymbol{r}$ al $\boldsymbol{R}$ satisface această *DIN* dacă $\forall t \in \boldsymbol{r}_i$, $\exists u \in \boldsymbol{r}_j$ astfel încât $t[A_{1,k}] = u[A_{2,k}]$, $\forall k$, $1 \le k \le p$.[38]

Exemplul 39

[37] De remarcat că cele două secvențe au aceeași lungime.

[38] O definiție echivalentă folosește *operatorul de redenumire* ρ: r satisface $R_i[S_1] \subseteq R_j[S_2]$, dacă $\pi_{S1}(r_i) \subseteq \subseteq \rho_{S1 \leftarrow S2}(\pi_{S2}(r_j))$.

Fie următoarea schemă simplificată pentru o agenție de rezervări de bilete aeriene: *ZBORURI*(*#Zbor*, *Origine*, *Destinație*)

ORAR(*#Orar*, *Zbor*, *Zi*, *Decolare*)

REZERVĂRI(*#Rezervare*, *Orar*, *Data*, *Pasager*)

Desigur că există următoarele restricții: oricărui zbor îi corespunde cel puțin un zbor invers (de întoarcere); orarul referă doar zboruri cunoscute; iar rezervările nu se pot face decât pentru zile în care sunt prevăzute în orar zboruri.

Formalizarea acestor restricții se poate evident face cu ajutorul următoarelor trei *DIN*: d_1: *ZBORURI*[*Origine*,*Destinație*] $\subseteq$ *ZBORURI* [*Destinație*,*Origine*]

d_2: *ORAR*[*Zbor*] $\subseteq$ *ZBORURI*[*#Zbor*]

d_3: *REZERVĂRI*[*Orar*] $\subseteq$ *ORAR*[*#Orar*].

Este uşor de verificat că bd din figura 39 (care are toate cheile în parantezele urmând numelor relațiilor) satisface toate aceste trei *DIN*; în schimb, dacă, de exemplu, s-ar elimina ultimul tuplu din *ZBORURI*, atunci ea ar viola atât d_1, cât şi d_2.

ZBORURI (*#Zbor*, *Origine•Destinație*)

#Zbor	*Origine*	*Destinație*
1	OTP	JFK
2	JFK	OTP
3	OTP	RCG
4	RCG	OTP

ORAR (*#Orar*, *Zbor•Zi•Decolare*) *Zbor* $\subseteq$ *#Zbor*

#Orar	*Zbor*	*Zi*	*Decolare*
1	3	Luni	06h50’
2	3	Joi	21h50’
3	4	Luni	12h20’
4	4	Vineri	05h30’

REZERVĂRI (*#Rezervare*, *Orar•Data•Pasager*) *Orar* $\subseteq$ *#Orar*

#Rezervare	*Orar*	*Data*	*Pasager*
1	3	2001.01.11	4586
2	3	2001.01.11	4592
3	3	2001.01.18	4973
4	3	2001.01.18	5002

Figura 39: O bază de date satisfăcând DIN din exemplul 39

Se poate observa din exemplul de mai sus că secvențele S_1 şi S_2 pot fi alcătuite din aceleaşi atribute (dar plasate în altă ordine, vezi d_1); în general, ele sunt interrelaționale

şi provin din propagarea cheilor, caz în care se zic *DIN tipate*[39] (vezi d_2 şi d_3); d_1 însă este un exemplu de *DIN* intrarelaţională netrivială[40].

Înainte de a vedea în ce măsură este posibilă proiectarea unui algoritm de vânare pentru *DIN*, este utilă, parametrizarea algoritmului similar pentru *FD*: dacă o constrângere este uşor violată, algoritmul 86 egalează valori; pentru a-l generaliza şi la alte tipuri de constrângeri (care nu impun egalarea unor valori, ci altceva, de exemplu generarea unei noi linii etc.), se poate defini, de exemplu, pentru fiecare asemenea tip în parte, o funcţie booleană *aplicabil* care să testeze dacă există un *k*-tuplu de linii implicate într-o violare uşoară a unei constrângeri date şi o procedură *aplică* care precizează acţiunile ce trebuie îndeplinite în caz de violare. Figura 40 prezintă noul algoritm 101 pentru vânarea *FD* astfel rezultat (şi care, fiind o pură rescriere sintactică a algoritmului 86, se bucură, evident, de aceleaşi proprietăţi cu acesta).

Boolean function aplicabil ($X \rightarrow A$, $<t_1, t_2>$, T) {
  if $t_1[X] = t_2[X] \wedge t_1[A] \prec t_2[A]$
    return true
  else
    return false;
};
procedure aplică ($X \rightarrow A$, $<t_1, t_2>$, T) {
  înlocuieşte toate aparițiile lui $t_2[A]$ din $CHASE_\Gamma(T)$ cu $t_1[A]$;
};
Algoritmul 101 (*vânarea unui tablou conform unei mulțimi de constrângeri* Γ)
Intrare: un tablou T şi o mulțime de constrângeri Γ
Ieşire:
un tablou $CHASE_\Gamma(T)$ satisfăcând Γ
Început:
$CHASE_\Gamma(T) = T$;
repetă pentru fiecare $t_1, \ldots, t_k \in T$, $\gamma \in \Gamma$ şi *aplicabil* (γ, $<t_1, \ldots, t_k>$, T)
aplică (γ, $<t_1, \ldots, t_k>$, T);
sfârşit repetă;
Sfârşit algoritm 101;

Figura 40: Algoritmul de vânare pentru FD rescris cu ajutorul funcţiei ***aplicabil*** *şi procedurii* ***aplică***

S-ar putea extinde la *DIN* algoritmul de vânare proiectat pentru *FD*, dar el nu s-ar mai bucura de aceleaşi proprietăţi. Întâi şi întâi, cum *DIN* sunt, în general, interrelaţionale, vânarea trebuie să considere mulţimi de tablouri şi nu doar un singur tablou; dar chiar şi în cazul *DIN* intrarelaţionale sunt diferenţe faţă de *FD*: pentru *FD*, procedura *aplică* schimbă tabloul pe cât de puţin posibil; pentru *DIN*, această strategie ar adăuga câte un tuplu per fiecare violare în care însă sunt cunoscute doar valorile atributelor din secvenţele de incluziune; cum despre valorile tuturor celorlalte atribute nu se ştie nimic, cea mai rezonabilă alegere este folosirea câte unei *vnd* distincte pentru fiecare asemenea atribut. Formalizând aceste observaţii, se obţin funcţia *aplicabil* şi procedura *aplică* din figura 41 (care se referă la un tablou T definit peste o schemă $R(X)$).

Boolean function aplicabil ($R[A_{1,1}, ..., A_{1,p}] \subseteq R[A_{2,1}, ..., A_{2,p}]$, $<t_1>$, T) {

[39] În literatura relaţională, tiparea este definită pur sintactic: *DIN* este tipată dacă S_1 şi S_2 sunt alcătuite din atribute având, în aceeaşi ordine, aceleaşi nume (deşi aparţin unor relaţii diferite).
[40] Orice *DIN* de forma $R[S] \subseteq R[S]$ este trivială!

if $\forall t_2 \in T$, $t_1 \neq t_2$, $\exists j$, $1 \leq j \leq p$, astfel încât $t_1[A_{1,j}] \neq t_2[A_{2,j}]$
return true
else
return false;
};
procedure aplică ($R[A_{1,1}, ..., A_{1,p}] \subseteq R[A_{2,1}, ..., A_{2,p}]$, $<t_1>$, T) {
adaugă la T un nou tuplu t, cu $t[A_{2,j}] = t_1[A_{1,j}]$, $\forall j$ $1 \leq j \leq p$ şi
cu $t[A]$ egal cu o *vnd* distinctă, $\forall A \notin \{A_{2,1}, ..., A_{2,p}\}$;
};

Figura 41: Funcția ***aplicabil*** *şi procedura* ***aplică*** *pentru DIN intrarelaționale*

Din nefericire, evident, nici măcar în acest caz simplu al *DIN* intrarelaționale, terminarea nu este garantată, deoarece la fiecare pas pot fi adăugate noi *vnd*. Fie, de exemplu, primul tablou din figura 42 şi vânarea corespunzătoare *DIN* $R[Părinte] \subseteq R[Copil]$. La fiecare pas e adăugat un nou tuplu, ceea ce introduce o nouă violare a acestei *DIN* şi aşa mai departe, la nesfârşit. Figura 42 prezintă în plus doar tablourile rezultate după primii doi paşi; se observă că toate violează *DIN*. Lăsăm cititorului drept exercițiu (vezi problema 27) găsirea unei condiții de terminare a vânării pentru *DIN*.

Există o variantă a procedurii *aplică* din figura 41 care garantează terminarea şi deci permite testarea implicației pentru *DIN*. Ea însă nu permite generalizarea pentru cazul în care sunt implicate şi alte tipuri de constrângeri. Ca atare, studiem problema *DIN* cu ajutorul regulilor de inferență, folosind idei din vânare doar în demonstrarea teoremei de completitudine.

R

Părinte	*Copil*
Basarab I	Nicolae Alexandru

T_1

Părinte	*Copil*
Basarab I	Nicolae Alexandru
n_1	Basarab I

T_2

Părinte	*Copil*
Basarab I	Nicolae Alexandru
n_1	Basarab I
n_2	n_1

Figura 42: O vânare ce nu se termină

Strategia de abordare este aceeaşi ca şi în cazul *FD*: prezentăm întâi câteva condiții suficiente pentru implicația *DIN*, le formalizăm apoi ca reguli de inferență şi, în final, demonstrăm completitudinea acestor reguli. Fie, din nou, o schemă de bd oarecare $\boldsymbol{R} = = \{R_1(X_1), R_2(X_2), ..., R_n(X_n)\}$; demonstrația următoarelor trei leme este lăsată cititorului drept exercițiu (vezi problemele 28, 29 şi 30).

Lema 102 O *DIN* $R_i[S_1] \subseteq R_j[S_2]$ este trivială dacă şi numai dacă $i = j$ şi $S_1 = S_2$.

Lema 103 Dacă o instanță $\boldsymbol{r}$ de bd satisface *DIN* $R_i[<A_{1,1}, ..., A_{1,p}>] \subseteq R_j[<A_{2,1}, ..., A_{2,p}>]$, atunci ea satisface şi *DIN* $R_i[<A_{1,k1}, ..., A_{1,kq}>] \subseteq R_j[<A_{2,k1}, ..., A_{2,kq}>]$, oricare ar fi secvența de întregi distincți $k_1, ..., k_q$ inclusă în $\{1,2, ..., q\}$, $q \leq p$.

Lema 104 Dacă o instanță $\boldsymbol{r}$ de bd satisface *DIN* $R_i[S_1] \subseteq R_j[S_2]$ şi $R_j[S_2] \subseteq R_k[S_3]$, atunci ea satisface şi *DIN* $R_i[S_1] \subseteq R_k[S_3]$.

Aceste trei leme asigură demonstrația corectitudinii următoarelor reguli de inferență:

DIN1. *Reflexivitatea DIN*:

dacă $\perp$, *atunci* $R_i[S] \subseteq R_i[S]$.

DIN2. *Proiecţia şi permutarea DIN*:

dacă $R_i[<A_{1,1}, ..., A_{1,p}>] \subseteq R_j[<A_{2,1}, ..., A_{2,p}>]$

cu $\{k_1, ..., k_q\} \subseteq \{1,2, ..., q\}$, $q \leq p$,

atunci $R_i[<A_{1,k1}, ..., A_{1,kq}>] \subseteq R_j[<A_{2,k1}, ..., A_{2,kq}>]$.

DIN3. *Tranzitivitatea DIN*:

dacă $R_i[S_1] \subseteq R_j[S_2]$, $R_j[S_2] \subseteq R_k[S_3]$, *atunci* $R_i[S_1] \subseteq R_k[S_3]$.

Figura 43: Regulile de inferenţă pentru DIN

Următoarea propoziţie este esenţială în demonstrarea completitudinii acestor reguli (demonstraţia ei este lăsată în seama cititorului, conform problemei 31):

Propoziţia 105 Fie *D* o mulţime oarecare de *DIN*; dacă o relaţie r_i conţine un tuplu *t* cu proprietatea că $t[A_{k,h}] = j_h > 0$, $\forall h$, $1 \leq h \leq q$, atunci este posibilă, cu ajutorul regulilor DIN1, DIN2, DIN3, derivarea din *D* a *DIN* $R_i[<A_{0,j1}, ..., A_{0,jq}>] \subseteq R_i[<A_{1,k1}, ..., A_{1,kq}>]$.

Teorema 106 Regulile DIN1, DIN2 şi DIN3 sunt complete pentru derivarea *DIN*.

Demonstraţie:

Arătăm că, date fiind o mulţime *D* de *DIN* şi o *DIN* $d = R_i[<A_{0,1}, ..., A_{0,p}>] \subseteq \subseteq R_j[<A_{1,1}, ..., A_{1,p}>]$ arbitrar fixate, dacă *D* implică *d*, atunci *d* este derivabilă din *D* cu ajutorul regulilor DIN1, DIN2 şi DIN3.

Fie *D* implică *d*. Pornim de la o instanţă iniţială minimală, în care r_i constă doar dintr-un tuplu t_0, iar r_j este vidă; presupunem că toate atributele au un codomeniu comun, numărabil, echipotent deci cu mulţimea numerelor naturale şi, ca atare, putem să identificăm valorile sale cu ajutorul naturalilor:

$t_0[A_{0,k}] = k$, $\forall k$, $1 \leq k \leq p$

$t_0[A] = 0$, $\forall A \notin \{A_{0,1}, ..., A_{0,p}\}$.

Construim din aceasta o instanţă de bd ***r*** prin aplicarea unui soi de procedură de vânare inspirată de funcţia şi procedura din figura 41: funcţia *aplicabil* este doar o generalizare a celei din figura 41 la cazul interrelaţional; procedura *aplică* însă, pe lângă această generalizare obligatorie, mai diferă de cea din figura 41 şi prin aceea că foloseşte valoarea 0 în loc de *vnd* distincte asignate atributelor neimplicate în *DIN* curentă.

În mod evident, deoarece niciodată nu se introduc noi valori, algoritmul se termină, iar bd ***r*** astfel generată satisface *D* şi deci şi *d* (căci am presupus că *D* implică *d*). Ca atare, r_j conţine un tuplu *t* a.î. $t[A_{1,k}] = k$, $\forall k$, $1 \leq k \leq p$. Conform propoziţiei 105, aceasta înseamnă că *d* este derivabilă din *D* cu ajutorul regulilor DIN1, DIN2 şi DIN3 q.e.d.

Raţionând într-un mod similar cu cel folosit pentru *FD* (vezi teorema 57), am putea acum proiecta un algoritm de decizie pentru problema implicaţiei în clasa *DIN*: dată fiind schema unei bd, o mulţime *I* de *DIN*, schema unei relaţii R_i şi o secvenţă de atribute S_1 din X_i, acest algoritm găseşte toate schemele de relaţie R_j şi secvenţele corespunzătoare de atribute S_2 din X_j cu proprietatea că *I* implică *DIN* $R_i[S_1] \subseteq R_j[S_2]$ (vezi problema 32). Totuşi, în ciuda similarităţii sale cu algoritmul 56, această procedură nu poate fi, în general, implementată eficient, deoarece ea nu manipulează doar o mulţime de atribute (precum algoritmul 56), ci o mulţime de secvenţe (deci de mulţimi) de atribute.

S-a arătat în [60] că problema generală a implicaţiei pentru *DIN* este *PSPACE-completă*, i.e. că nu există un algoritm polinomial pentru ea decât în cazul particular în care *P=PSPACE*. În particular, problema devine polinomială pentru fiecare din următoarele două subclase restrictive de *DIN*:

(1) dacă toate *DIN* sunt tipate, i.e. de forma $R_i[S] \subseteq R_j[S]$, unde $S \subseteq X_i \cap X_j$ (vezi problemele 33 şi 34); sau

(2) dacă lungimea secvenţelor de atribute implicate în *DIN* este *limitată*, i.e. dacă $\exists k$ întreg a.î. $\forall R_i[S_1] \subseteq R_j[S_2]$, $1 \le i, j \le n$, $S_1 = <A_{1,1}, ..., A_{1,p}>$, $S_2 = <A_{2,1}, ..., A_{2,p}>$, $p \le k$.[41]

În sfârşit, deoarece studiul interacţiunii între *FD* şi *DIN* ridică probleme complicate, frizând nedecidabilitatea, tratăm acest subiect separat, în secţiunea următoare, dintr-un punct de vedere mai general.

4.7 Decidabilitatea implicaţiei finite şi a celei nerestricţionate

Încheiem studiul implicaţiei *DIN* furnizând un punct de vedere mai general asupra unei părţi a problematicii constrângerilor, ce are drept punct de plecare interacţiunea între *DIN* şi *FD*.

Exemplul 40

Fie schema de bd cu o singură relaţie şi două atribute $R(AB)$, peste care sunt definite două constrângeri: $\Gamma = \{A \rightarrow B, R[A] \subseteq R[B]\}$. Afirmăm că orice instanţă r finită ce satisface Γ, satisface şi *DIN* $R[B] \subseteq R[A]$.

Într-adevăr, deoarece r satisface *FD* $A \rightarrow B$, numărul valorilor distincte ale atributului A nu este mai mic decât cel similar pentru B; cum r satisface şi *DIN* $R[A] \subseteq R[B]$, mulţimea valorilor lui B conţine (posibil nu propriu) pe cea a lui A. Ca o consecinţă a acestor două observaţii, rezultă că cele două mulţimi de valori sunt identice şi deci r satisface şi *DIN* $R[B] \subseteq R[A]$.

Desigur că aşa ceva nu are loc pentru relaţii infinite (i.e. cu un număr infinit de tupli); figura 44 prezintă un contraexemplu: r satisface Γ, dar violează *DIN* $R[B] \subseteq R[A]$, deoarece 0 nu apare drept valoare a atributului A.

A	B
1	0
2	1
3	2
...	...
n	n-1
n+1	n
...	...

Figura 44: Relaţia infinită din exemplul 40

Exemplul 40 prezintă deci o mulţime Γ de *FD* şi de *DIN* şi o *DIN* δ astfel încât Γ nu implică δ, dar, totuşi, orice conţinut finit r ce satisface Γ, satisface şi δ, ceea ce nu este niciodată posibil dacă Γ e alcătuită doar din *FD* (vezi şi problema 35). Din această cauză, este necesar să distingem între două noţiuni de implicaţie:

Definiţia 107 Se zice că:

1. Γ *implică nerestrictiv* γ, dacă γ ţine în orice conţinut de bd (finit sau infinit) ce satisface Γ.
2. Γ *implică finit* γ, dacă γ ţine în orice conţinut finit de bd ce satisface Γ.

Definiţia 108 O *clasă de constrângeri* se zice *finit controlabilă* dacă cele două noţiuni ale implicaţiei (din definiţia 107) coincid.

Exemplul 40 arată deci că reuniunea claselor de *FD* şi *DIN* nu este finit controlabilă. Vom vedea în această subsecţiune că finit-controlabilitatea este strâns legată de decidabilitatea problemei implicaţiei.[42] În cele ce urmează, clasele de constrângeri considerate

[41] În cazul banal $k = 1$, *DIN* se zic *unare* (vezi şi problemele 33, 34).

sunt oarecari: unica condiție impusă este posibilitatea formalizării tuturor constrângerilor ce le alcătuiesc cu ajutorul propozițiilor logice de ordin întâi.

Lema 109 Fie C o clasă de constrângeri și $\Gamma \subset C$; mulțimea de constrângeri a clasei C care nu sunt finit implicate de Γ este recursiv numărabilă.

Demonstrație:

Putem enumera conținuturi de bd și constrângeri finite (e.g. diagonal) și să alegem doar acele constrângeri ale C care sunt violate în vreun conținut ce satisface Γ q.e.d.

Lema 110 Fie C o clasă de constrângeri și $\Gamma \subset C$; mulțimea de constrângeri a cla-sei C restricționat implicate de Γ este recursiv numărabilă.

Demonstrație:

Cum constrângerile sunt propoziții de ordin întâi, concluzia rezultă imediat din recursiv numărabilitatea propozițiilor în calculul predicatelor de ordin întâi q.e.d.

Teorema 111 Dacă o clasă de constrângeri C este finit controlabilă, atunci ambele probleme ale implicației pentru C sunt decidabile.

[42] Restul acestei subsecțiuni necesită cunoașterea noțiunilor fundamentale de calculabilitate, precum decidabilitatea sau recursiv numărabilitatea; el poate fi ignorat, fără ca înțelegerea restului lucrării să fie afectată.

Demonstrație:

Dacă clasa e finit controlabilă, atunci, conform definiției, cele două probleme ale implicației coincid; conform lemelor 109 şi 110, atât mulțimea constrângerilor implicate, cât şi cea a celor neimplicate sunt recursiv numărabile, deci ambele sunt decidabile.

q.e.d.

De remarcat că teorema 111 dovedeşte că proprietatea de controlabilitate finită este o condiție suficientă pentru decidabilitatea problemei implicației. Această condiție nu este totuşi şi necesară: de exemplu, în [83] se arată că ambele implicații sunt decidabile pentru reuniunea claselor de *FD* şi de *DIN* unare, deşi nici aceasta nu este finit controlabilă. În cele ce urmează, confirmăm parțial acest lucru prezentând o soluție la problema implicației nerestrictive pentru această reuniune. Pentru simplitate, ne referim doar la *DIN* unare intrarelaționale (care, am văzut deja din exemplul 40, nu sunt nici măcar ele finit controlabile).

Lema 112 Fie o schemă de relație $R(U)$ şi o mulțime de constrângeri $\Gamma = F \cup I$, unde F este o mulțime de *FD*, iar I una de *DIN* unare intrarelaționale peste R; au loc următoarele două duble implicații:

1. O *FD* este implicată de Γ dacă şi numai dacă ea este implicată de F.
2. O *DIN* este implicată de Γ dacă şi numai dacă ea este implicată de I.

Demonstrație:

1. Fie $f = X \to A$ o *FD* care nu este implicată de F; construim un conținut de relație (posibil infinit) care satisface Γ dar violează f, confirmând astfel că Γ nu implică f. Fie un conținut de relație cu doi tupli având aceleaşi valori pentru toate atributele din X^+, dar valori diferite pentru toate celelalte atribute (deci şi pentru A): contraexemplul tipic de relație ce satisface F dar violează f. Dacă vânăm această relație în raport cu I, numerotând violările *DIN* în ordinea în care apar şi aplicându-le apoi exact în această ordine, se obține un conținut infinit dar numărabil ce satisface I. Susținem că această relație satisface şi F; aceasta rezultă din faptul că relația inițială satisface F, iar vânarea în raport cu *DIN* unare nu poate introduce violări ale *FD*: tuplii adăugați de vânare au noi valori pentru toate atributele, cu excepția unuia, pentru care valoarea este deja printre cele existente în relație, dar pentru alte atribute.
2. Demonstrația acestui punct este lăsată cititorului (vezi problema 37).

q.e.d.

Următoarea teoremă este o consecință directă a lemei 112 (vezi problema 82):

Teorema 113 Problema implicației nerestricționate pentru *FD* şi *DIN* unare este decidabilă.

4.8 Constrângeri de integritate şi valori nule

Două sunt principalele probleme ridicate de existența valorilor nule în subuniversul constrângerilor şi anume:

1. extensia tipurilor de constrângeri „clasice” la acest cadru, mai general şi

2. introducerea şi studierea de noi tipuri de constrângeri ce apar necesare doar în prezența valorilor nule.

În legătură cu prima dintre ele, considerăm în următoarea subsecțiune doar funcțional dependențele şi valorile necunoscute; interacțiunea dintre *FD* şi celelalte două tipuri de valori nule este considerată separat, în subsecțiunea a doua. Ultima subsecțiune a acestei secțiuni adresează a doua problemă de mai sus. Întreaga secțiune curentă face apel la următoarea definiție:

Definiția 114 Un *tuplu t* al unei relații $r(X)$ se zice *A-total*, oricare ar fi $A \in X$, dacă $t[A]$ este o valoare specificată, respectiv *Y-total*, oricare ar fi $Y \subseteq X$, dacă el este *A*-total pentru orice $A \in Y$; dacă *t* este *X*-total, atunci el se mai zice simplu şi *total*.

4.8.1 Funcțional dependențe şi valori necunoscute

Dacă se interpretează valorile nule drept necunoscute, *FD* (fiind predicate definite peste relații) pot fi uşor extinse cu ajutorul unei logici ternare şi a principiului substituției. În această logică, date fiind o relație *r* şi o *FD* f oarecari, f poate avea una dintre următoarele trei valori:

- *adevărat*, dacă toate substituțiile posibile pentru valorile nule generează conținuturi fără valori nule ce satisfac f sau
- *fals*, dacă toate substituțiile posibile pentru valorile nule generează conținuturi fără valori nule ce violează f sau
- *necunoscut*, în toate celelalte cazuri, adică dacă unele dintre substituțiile posibile pentru valorile nule generează conținuturi (fără valori nule) ce violează f, în timp ce alte substituții generează conținuturi ce satisfac f.

Definiția 115 Fie *r* un conținut oarecare de *relație*; se zice că f este:

- *puternic satisfăcută* de *r* dacă f este *adevărat* pentru *r* sau
- *slab satisfăcută* de *r* dacă f este *adevărat* sau *necunoscut* pentru *r* sau
- *nesatisfăcută* (*violată*) de *r* dacă f este *fals* pentru *r*.

Exemplul 41

Fie relațiile din figura 45: *FD* $f: AB \rightarrow C$ este evident puternic satisfăcută de r_1, deoarece toate substituțiile valorilor nule generează conținuturi lipsite de nuluri ce satisfac f. r_2 satisface însă doar slab pe f, deoarece dacă valoarea nulă este înlocuită cu 1, atunci f este satisfăcută de noul conținut, în timp ce conținuturile obținute prin înlocuirea ei cu orice altă valoare violează f. În sfârşit, r_3 violează în mod trivial f, din cauza primilor doi tupli (care sunt totali şi deci invarianți în raport cu orice substituție).

r_1

A	*B*	*C*
1	1	1
2	1	ϕ
1	ϕ	1

r_2

A	*B*	*C*
1	1	1
1	1	ϕ

r_3

A	*B*	*C*
1	1	1
1	1	2
ϕ	1	ϕ

Figura 45: Relații ilustrând violarea, satisfacerea slabă şi cea puternică a FD

Dacă domeniile sunt infinite, cele două forme de satisfacere a *FD* pot fi caracterizate respectiv de următoarele două leme (a căror demonstrație este propusă cititorului de problemele 38, respectiv 39).

Lema 116 O relație $r(XYZ)$ satisface puternic o *FD* $X \rightarrow Y$ dacă şi numai dacă, oricare ar fi perechea de tupli $t_1, t_2 \in r$, este adevărată una din următoarele două condiții:

1. există un atribut $A \in X$, astfel încât t_1 şi t_2 sunt distincți şi *A*-totali sau
2. pentru orice atribut $A \in Y$, t_1 şi t_2 sunt egali şi *A*-totali.

Lema 117 O relație $r(XYZ)$ satisface slab o *FD* $X \rightarrow Y$ dacă şi numai dacă, oricare ar fi perechea de tupli $t_1, t_2 \in r$, este adevărată una din următoarele două condiții:

1. există un atribut $A \in X$, astfel încât t_1 şi t_2 ori sunt distincți, ori nu sunt A-totali sau
2. pentru orice atribut $A \in Y$, t_1 şi t_2 ori sunt egali, ori unul dintre ei nu este A-total.

În condițiile satisfacerii puternice, are loc următorul rezultat interesant şi de aşteptat în legătură cu problema implicației pentru *FD*:

Teorema 118 Fie F o mulțime de *FD* şi f o *FD* peste o schemă de relație $R(U)$; următoarele două afirmații sunt echivalente:

1. F implică f în raport cu noțiunea uzuală de satisfacție în relații fără valori nule.
2. F implică f în raport cu noțiunea de satisfacție puternică în relații cu valori nule.

Demonstrație:

(1. $\Rightarrow$ 2.)

Procedăm prin reducere la absurd: arătăm că dacă 2. este fals, atunci şi 1. e fals, i.e. că dacă F nu implică f în raport cu noțiunea de satisfacție puternică, atunci F nu implică f nici în raport cu noțiunea uzuală de satisfacție. Dacă presupunem prin absurd că F nu implică f în raport cu noțiunea de satisfacție puternică, atunci există o relație r (posibil conținând nuluri) care satisface puternic F dar nu şi f; în acest caz însă, există şi cel puțin o relație r' fără valori nule, obținută din r prin substituirea acestora cu constante, care satisface F dar nu şi f; dar aceasta ar însemna că F nu implică f, ceea ce contrazice ipoteza.

(2. $\Rightarrow$ 1.)

Fie r o relație fără valori nule peste $R(U)$ satisfăcând F; dacă considerăm r ca pe o relație (posibil) conținând nuluri, ea satisface puternic F prin definiție şi deci, cum 2. e adevărat, ea satisface puternic şi f, ceea ce, trivial, înseamnă că r satisface f q.e.d.

De remarcat întâi şi întâi că, de fapt, în cursul demonstrației acestei teoreme nu se face practic uz de nici de definiția şi nici de proprietățile *FD*; ca atare, desigur, se poate demonstra un rezultat analog pentru orice alt tip de constrângere căruia i se asociază noțiunea de satisfacere puternică.

Apoi, desigur, cea mai importantă consecință a teoremei 118 este aceea că toate rezultatele obținute anterior în acest capitol privitoare la relații fără valori nule au loc de asemenea în cazul satisfacerii puternice şi pentru cele conținând valori nule.

Din nefericire, aşa cum este de aşteptat, acest lucru nu este adevărat şi în cazul satisfacerii slabe, aşa cum ne arată următorul contraexemplu.

Exemplul 42

Fie relația din figura 46, care satisface slab atât $A \rightarrow B$, cât şi $B \rightarrow C$, dar violează $A \rightarrow C$. În esență, aceasta se datorează faptului că orice relație fără valori nule obținută din ea prin substituție satisface $A \rightarrow B \Leftrightarrow$ violează $B \rightarrow C$

A	B	C
1	ϕ	1
1	ϕ	2

Figura 46: Un contraexemplu de tranzitivitate pentru satisfacerea slabă a FD

Exemplul de mai sus scoate de asemenea în evidență o proprietate a satisfacerii slabe ne mai întâlnită până acum în nici un alt context: pot exista două *FD* care considerate separat sunt slab satisfăcute, dar care considerate împreună nu mai sunt nici măcar slab satisfăcute (i.e. nu există nici o substituție care să conducă la conținuturi fără valori nule satisfacând ambele *FD*).

Definiția 119 Noțiunea de *satisfacere* se zice *aditivă* pentru o clasă oarecare de constrângeri, dacă satisfacerea oricăror două mulțimi de constrângeri din acea clasă implică satisfacerea reuniunii acestor mulțimi.

Evident că satisfacerea uzuală studiată până aici (i.e. cea pentru relații lipsite de valori nule) este aditivă, în timp ce satisfacerea slabă nu este. Ca atare, înainte de a studia problema implicației pentru acest nou tip de satisfacere, este necesară rezolvarea următoarei probleme ridicată de satisfacerea slabă: date fiind o relație *r* şi o mulțime de *FD F, fiecare în parte* slab satisfăcute de *r*, în ce condiții satisface *r* slab *toate FD* din *F* (i.e. dacă există, este şi unică sau nu şi cum se poate obține din *r* prin substituții o relație *r'* care să satisfacă slab *F*)?

Presupunând infinite domeniile atributelor, se poate rezolva această problemă cu ajutorul tablourilor şi algoritmului de vânare: date fiind o relație *r* cu valori nule şi o mulțime *F* de *FD*, întâi se construieşte din *r* un tablou T_r, înlocuind fiecare valoare nulă în parte cu o *vnd* distinctă; conform teoremei următoare, caracterizarea satisfacerii slabe se bazează apoi pe rezultatul vânării acestui tablou.

Teorema 120 O relație *r* satisface slab o mulțime de *FD F* dacă şi numai dacă tabloul $CHASE_F(T_r)$ satisface *F*.

Demonstrație:

($\Rightarrow$)(*dacă*)

Fie $CHASE_F(T_r)$ satisfăcând *F*; înlocuind fiecare variabilă a sa cu o constantă distinctă (lucru oricând posibil căci domeniile sunt considerate infinite!), se obține în mod trivial o relație satisfăcând *F*.

($\Leftarrow$)(*şi numai dacă*)

Dacă *r* satisface slab *F*, atunci există o relație r_s fără valori nule, obținută din *r* prin substituții, ce satisface *F*. Cum toate variabilele tabloului sunt distincte, rezultă că există o aplicație de conținere de la T_r la r_s şi deci, conform lemei 93, urmează că şi $CHASE_F(T_r)$ satisface *F*

q.e.d.

De remarcat de asemenea faptul că, în legătură cu problema implicației, exemplul din figura 46 arată şi că regula FD3 (a tranzitivității extinse) nu mai este solidă în contextul considerării valorilor nule.[43] În schimb, se poate demonstra (vezi problema 42) că următoarea regulă este solidă:

FD4 (*Reuniune extinsă*): *dacă* $XZ \rightarrow Y$, $X \rightarrow W$, *atunci* $XZ \rightarrow YW$

Teorema următoare (a cărei demonstrație este propusă cititorului de problema 43) garantează completitudinea acestei noi mulțimi de reguli de inferență în contextul satisfacerii slabe:

Teorema 121 Mulțimea de reguli de inferență {FD1, FD2, FD4} este solidă şi completă în raport cu implicația *FD* în contextul satisfacerii slabe.

Mai mult, procedând similar ca pentru cazul absenței valorilor nule, se pot proiecta şi în acest context algoritmi eficienți pentru problema implicației: de exemplu, se poate folosi pentru testarea implicației o variantă a algoritmului 58 (figura 26) de calcul a închiderii unei mulțimi de atribute (vezi problema 44).

4.8.2 Funcțional dependențe şi valori inexistente sau lipsite de informație

Evident că noțiunea de substituție a valorilor nule nu are sens în raport cu nulurile inexistente sau lipsite de informație. Fie oricare ar fi o relație $r(U)$, o *FD* $X \rightarrow Y$ şi doi

[43] FD1 şi FD2 rămân însă, evident, solide (vezi problema 42).

tupli t_1, $t_2 \in r$; orice mod de a trata satisfacerea *FD* în acest context trebuie să țină cont cel puțin de următoarele patru considerente:

- ✓ Noile definiții trebuie să fie o extensie a celor uzuale (fără valori nule); ca atare, un conținut de relație *r* trebuie să violeze $X \rightarrow Y$ dacă t_1 şi t_2 sunt ambii *XY*-totali, egali pe *X* şi distincți pe *Y* (i.e. $t_1[X] = t_2[X]$ şi $t_1[Y] \neq t_2[Y]$).
- ✓ Dacă $t_1[X] = t_2[X]$, dar t_1 şi t_2 nu sunt ambii *X*-totali (i.e. sunt implicate şi valori nule), atunci nu pare necesar ca $t_1[Y] = t_2[Y]$.
- ✓ Dacă $t_1[X] = t_2[X]$, t_1 şi t_2 sunt ambii *X*-totali, dar există $A \in Y$ astfel încât unul dintre $t_1[A]$ sau $t_2[A]$ este nul iar celălalt nu, atunci *FD* trebuie să fie violată (deoarece pentru *A* ar fi posibile cel puțin două informații diferite, ceea ce contrazice condiția de funcționalitate).
- ✓ Dacă $t_1[X] = t_2[X]$, t_1 şi t_2 sunt ambii *X*-totali, dar, oricare ar fi $A \in Y$, dacă unul dintre $t_1[A]$ sau $t_2[A]$ este nul, atunci şi celălalt este nul, atunci *FD* trebuie să fie satisfăcută (deoarece, pentru orice *A*, este mereu posibilă o singură informație).

Definiția 122 Se zice *funcțional dependență cu nuluri (FDN*) orice *FD* satisfăcută de o relație *r* (posibil conținând şi valori nule inexistente şi/sau lipsite de informație) în următorul sens: oricare ar fi doi tupli t_1, $t_2 \in r$, ambii *X*-totali şi cu proprietatea că $t_1[X] = = t_2[X]$, atunci şi $t_1[Y] = t_2[Y]$.

Exemplul 43

Fie relațiile din figura 47; evident, r_1 nu satisface *FDN Persoana→Telefon*, deoarece, pentru aceeaşi valoare a *Persoana*, cei doi tupli au valori diferite pentru *Telefon* (chiar dacă una e nulă); în schimb, r_2 satisface *FDN Persoana, TipTelefon→Telefon*, fiindca nu există nici o pereche de tupli care să aibă pentru perechea *Persoana,TipTelefon* aceleaşi valori non-nule (primii doi au aceleaşi valori pentru aceste prime două atribute, dar una din ele este nulă).

r_1

Persoana	*TipTelefon*	*Telefon*
Ioan	servici	887.87.89
Ioan	acasă	ϕ

r_2

Persoana	*TipTelefon*	*Telefon*
Ioan	ϕ	887.87.89
Ioan	ϕ	095.87.87.89
Radu	servici	ϕ
Radu	acasă	ϕ

Figura 47: Exemple de relații pentru FDN din exemplul 43

Considerând din nou exemplul 42, se poate constata că el constituie un contraexemplu şi în cazul problemei implicației pentru *FDN*: relația din figura 46 satisface *FDN* $A \rightarrow B$ şi $B \rightarrow C$, dar violează *FDN* $A \rightarrow C$; deducem de aici că regula tranzitivității extinse nu este solidă nici în cazul *FDN*!

Pe de altă parte, este evident că regulile FD1, FD2 şi FD4 sunt solide şi în acest context (vezi problema 45). Mai mult, următoarea teoremă le garantează şi completitudinea:

Teorema 123 Mulțimea de reguli de inferență {FD1,FD2,FD4} este solidă şi completă în raport cu implicația *FDN*.

Demonstrația acestei teoreme este propusă cititorului de problema 46; de notat însă că aceasta nu se poate baza pe teorema 121, deoarece noțiunile de satisfacere slabă pentru *FD* cu nuluri necunoscute, respectiv *FDN* nu sunt echivalente nici măcar sintactic, aşa cum se poate vedea considerând *FD* $A \rightarrow C$ şi relația din figura 48.

A	*B*	*C*	*D*
1	ϕ	1	ϕ
1	ϕ	ϕ	2

Figura 48: Un contraexemplu dovedind neechivalența FDN şi FD slab satisfăcute

Din punct de vedere al problemei implicației, cum regulile de inferență pentru *FDN* sunt identice cu cele pentru *FD* în contextul satisfacerii slabe, este evident că algoritmii pentru acestea din urmă sunt aplicabili şi *FDN*.

4.8.3 Constrângeri de existență

În contextul relațiilor admițând şi valori nule, pe lângă extensiile claselor de constrângeri uzuale (i.e. definite în absența nulurilor), dintre care am studiat câteva în cele două subsecțiuni precedente, au fost introduse şi noi clase de constrângeri, care ori sunt triviale, ori nu au nici un sens în absența valorilor nule.

În acest sens, am văzut deja în subsecțiunea 2.3 că este de dorit controlul asupra permiterii de valori nule: de exemplu, accepția uzuală în domeniu le interzice pentru cheile primare. Există însă şi situații în care, deşi atributele respective nu participă la formarea cheii primare, valorile nule se doresc totuşi interzise în anumite cazuri.

Interzicerea apariției valorilor nule, indiferent de context, este formalizată cu ajutorul aşa numitelor *constrângeri de existență*.

Definiția 124 Fie o schemă de relație $R(U)$ şi $XY \subseteq U$; o relație r peste R satisface *constrângerea de existență* (*CE*) $e : X \vdash Y$, dacă orice tuplu X-total $t \in r$ este şi Y-total.

De remarcat că dacă Y este mulțimea vidă, atunci *CE* este trivial satisfăcută de orice relație, în timp ce dacă X este mulțimea vidă, atunci *CE* impune ca toți tuplii să aibă valori non-nule pentru toate atributele din Y.

Exemplul 44

Fie relația din figura 49, în care *#Persoana* este cheie primară; restricția ca valorile sale să nu poată include nuluri se poate evident formaliza cu ajutorul *CE* $\varnothing \vdash$ *#Persoana*. Dacă, de exemplu, domeniul modelat impune suplimentar restricția ca numele unei persoane să nu poată fi nul dacă prenumele său nu este nul, aceasta se poate evident formaliza cu ajutorul *CE Prenume* $\vdash$ $\vdash$*Nume*. Trivial, relația din figura 49 satisface ambele aceste *CE*, dar nu satisface, de exemplu, nici $\varnothing \vdash$ *Nume*, nici *Nume* $\vdash$ *Prenume*.

#Persoana	*Prenume*	*Nume*
1	Ioan	Ionescu
2	ϕ	Popescu
3	ϕ	ϕ

Figura 49: Relația suport pentru CE din exemplul 44

Următoarea teoremă (a cărei demonstrație este propusă cititorului de problema 49) arată că *CE* au, în esență, aceleaşi reguli de inferență ca şi *FD* uzuale (definite în absența valorilor nule) sau, altfel spus, că şi echivalentul regulii tranzitivității extinse rămâne solidă chiar şi în prezența nulurilor.

Teorema 125 Regulile de inferență obținute din FD1, FD2 şi FD3 prin înlocuirea simbolului „$\rightarrow$” cu simbolul „$\vdash$” sunt solide şi complete pentru derivarea *CE*.

Această teoremă are o consecință evidentă extrem de importantă: majoritatea teoriei dezvoltate pentru *FD* (inclusiv algoritmii de calcul a închiderilor şi de rezolvare a problemei implicației) se poate extinde imediat, natural şi fără efort şi la clasa *CE*.

5. Formele normale 4, 5, (3,3) și domenii-chei

Conceptul de descompunere fără pierderi este foarte important în modelul relațional şi de aceea s-a dorit caracterizarea sa formală. Cercetările în această direcție au condus repede la concluzia că nu este posibilă formularea de condiții necesare şi suficiente pentru ca o relație să poată fi descompusă fără pierderi bazându-ne doar pe *FD*, deoarece există relații ce au această proprietate fără ca ele să satisfacă vreo *FD* ne-trivială.

5.1 Dependențe join

De exemplu, relația $r(M,P,S,F)$ din figura 50 (*Magazii*, *Piese*, *Subansansamble*, *Furnizori*) nu satisface nici o *FD* netrivială şi totuşi admite descompunerea fără pierderi peste schema $\boldsymbol{S} = \{MP, PS, SF\}$. Ca atare, a fost introdus în modelul relațional un nou tip de constrângere, numit *dependență join* (*DJ*), ce este satisfăcut de o relație dacă şi numai dacă ea admite o descompunere fără pierderi.

Formal, dată fiind o schemă de relație $R(U)$, o *DJ* se notează prin $\bowtie[X_1,X_2, ..., X_n]$, unde $X_1,X_2, ..., X_n \subset U$ şi $X_1X_2 ...X_n = U$.

Definiția 2.126 O relație r *satisface DJ* $\bowtie[X_1,X_2, ..., X_n]$, dacă $r = \bowtie_{i=1}^{n}\pi_{Xi}(r)$, adică dacă descompunerea ei peste $X_1,X_2, ..., X_n$ se face fără pierderi.

Exemplul 45

Relația r din figura 50 satisface $DJ \bowtie[MP,PS,SF]$ în timp ce relația r din figura 51 nu o satisface (lucru ce se poate observa analizând cea de-a doua tabelă a figurii, în care este reprezentat joinul proiecțiilor ei peste cele trei perechi de atribute ale *DJ*).

Cel mai uzual caz de descompunere este cel binar, corespunzător unei *DJ* de tip $\bowtie[X_1,X_2]$. Acest caz particular de *DJ* a fost de fapt introdus în literatură, sub numele de *dependență multivaluată*, anterior generalizării sale *n*-are.

5.2 Dependențe multivaluate

Definiția 127 Dată fiind o schemă de relație $R(U)$, o *dependență multivaluată* (*DMV*) are sintaxa $X \rightarrow\rightarrow Y$, unde $X,Y \subseteq U$. Fie $Z = U - XY$; o relație r satisface *DMV* $X \rightarrow\rightarrow Y$ dacă, pentru orice pereche de tupli $t_1,t_2 \in r$ cu proprietatea că $t_1[X] = t_2[X]$, există un tuplu $t \in r$ astfel încât $t[XY] = t_1[XY]$ şi $t[XZ] = t_2[XZ]$.

Intuitiv, am putea spune că *DMV* $X \rightarrow\rightarrow Y$ cere ca, pentru orice X-valoare în r, Y-valorile asociate să fie independente de Z-valori.

Cum Y şi Z pot fi interschimbate în definiția de mai sus fără a îi altera înțelesul, rezultă că orice relație $R(U)$ satisface $X \rightarrow\rightarrow Y$ dacă şi numai dacă ea satisface $X \rightarrow\rightarrow Z$ (ceea ce se numeşte *proprietatea de complementaritate a DMV*).

Exemplul 46

Fie o schemă de relație definită peste atributele (*R*)*estaurant*, (*C*)*hefliu* şi (*B*)*ere*. Un tuplu al relației memorează faptul că acel chefliu bea acel tip de bere în restaurantul respectiv; restaurantele şi chefliii nu au neapărat nume distincte. Dacă toți chefliii ar bea în fiecare restaurant pe care îl frecventează din toate tipurile de bere oferite de acel restaurant, atunci conținutul relației trebuie să satisfacă *DMV* $R \rightarrow\rightarrow C$: pentru orice pereche de tupli $<r_1,c_1,b_1>$ şi $<r_1,c_2,b_2>$ există şi tuplul $<r_1,c_1,b_2>$, deoarece, dacă chefliul c_2 bea bere b_2 în r_1, atunci c_1 va bea şi el bere b_2 în r_1, fiindcă restaurantul r_1 oferă nu doar bere b_1 ci şi bere b_2 (şi, desigur, din motive similare, relația conține obligatoriu şi tuplul $<r_1,c_2,b_1>$).

r

M	*P*	*S*	*F*
m_1	p_1	s_1	f_1
m_1	p_1	s_1	f_2
m_1	p_1	s_2	f_2
m_1	p_2	s_2	f_2
m_2	p_2	s_2	f_2

Figura 50: O relație decompozabilă fără pierderi peste MP, PS, SF

r

M	*P*	*S*	*F*
m_1	p_1	s_1	f_1
m_1	p_1	s_1	f_2
m_2	p_1	s_2	f_2

$\pi_{MP}(r) \bowtie \pi_{PS}(r) \bowtie \pi_{SF}(r)$

M	*P*	*S*	*F*
m_1	p_1	s_1	f_1
m_1	p_1	s_1	f_2
m_1	p_1	s_2	f_2
m_2	p_1	s_1	f_1
m_2	p_1	s_1	f_2
m_2	p_1	s_2	f_2

Figura 51: O violare a DJ $\bowtie$*[MP,PS,SF]*

Analizând relațiile din figura 51, se poate observa că prima dintre ele satisface *DMV* $R \rightarrow\rightarrow C$, în timp ce cea de-a doua nu o satisface (pentru că îi lipseşte tuplul care să afirme că 'Nicolae' bea la 'Select' şi bere 'Vidraru').

De remarcat şi că prima satisface, iar a doua violează şi *DMV* $R \rightarrow\rightarrow B$.

Teorema 128 Fie $r(U)$ o relație şi $X_1,X_2,X \subset U$ astfel încât $X_1X_2 = U$ şi $X = X_1 \cap X_2$;
Descompunerea $r(U)$ peste X_1,X_2 este fără pierderi dacă şi numai dacă $r(U)$ satisface *DMV* $X \rightarrow\rightarrow X_1$ (sau, echivalent, *DMV* $X \rightarrow\rightarrow X_2$).

Demonstrație:

(*dacă*)($\Rightarrow$)

Fie r satisfăcând $X \rightarrow\rightarrow X_1$ şi $t \in \pi_{X1}(r) \bowtie \pi_{X2}(r)$; trebuie arătat că $t \in r$, ceea ce este suficient pentru demonstrarea acestei implicații (deoarece, întotdeauna, $r \subseteq \subseteq \pi_{X1}(r) \bowtie \pi_{X2}(r)$). Conform definiției operatorilor join şi proiecție, există $t_1, t_2 \in r$ astfel încât $t[X_1] = t_1[X_1]$ şi $t[X_2] = t_2[X_2]$; din definiția *DMV*, deoarece $X = X_1 \cap X_2$ şi deci $t_1[X] = t_2[X]$, $\exists\ t' \in r$ astfel încât $t'\ [X_1] = t_1[X_1]$ şi $t'\ [X_2] = t_2[X_2]$. Aceasta înseamnă însă că $t = t'$, ceea ce demonstrează că $t \in r$.

(*şi numai dacă*)($\Leftarrow$)

Fie $r = \pi_{X1}(r) \bowtie \pi_{X2}(r)$ şi $\forall\ t_1, t_2 \in r$ cu proprietatea că $t_1[X] = t_2[X]$; trebuie arătat că $\exists\ t \in r$ cu proprietatea că $t[X_1] = t_1[X_1]$ şi $t[X_2] = t_2[X_2]$, ceea ce ar confirma faptul că r satisface $X \rightarrow\rightarrow X_1$. Fie $t_1' = t[X_1]$ şi $t_2' = t[X_2]$; rezultă că $t_1' \in \pi_{X1}(r)$, iar $t_2' \in \pi_{X2}(r)$. Din definiția joinului însă, t este un tuplu al $\pi_{X1}(r) \bowtie \pi_{X2}(r)$ ce provine din t_1' şi t_2'. În fine, deoarece $r = \pi_{X1}(r) \bowtie \pi_{X2}(r)$, rezultă că $t \in r$

q.e.d.

Prima relație din figura 52 se descompune fără pierderi peste *RC* şi *RB*, în timp ce cea de-a doua nu are această proprietate. Merită observat faptul că o relație are o descompunere fără pierderi *dacă şi numai dacă* ea satisface o *DMV*, ceea ce, ne reamintim,

nu este cazul pentru *FD* (unde, conform propoziției 22 şi exemplului din figura 50, satisfacerea unei *FD* este doar o condiție suficientă, nu şi necesară).

Restaurant	*Chefliu*	*Bere*
Select	Vadim	Rahova
Select	Nicolae	Rahova
Select	Vadim	Vidraru
Select	Nicolae	Vidraru
Ferentari	Florin	Radeberger
Melody	Nicolae	Grivița
Cotroceni	Nicolae	Grivița

Restaurant	*Chefliu*	*Bere*
Select	Vadim	Rahova
Select	Nicolae	Rahova
Select	Vadim	Vidraru

Figura 52: O relație ce satisface DMV $R \rightarrow\rightarrow C$ şi una ce o violează

Este interesantă şi compararea definițiilor *FD* şi *DMV*. O *FD* $X \rightarrow Y$ afirmă o legătură funcțională între *X*-valori şi *Y*-valori, în timp ce *DMV* $X \rightarrow\rightarrow Y$ afirmă că pentru orice *X*-valoare *x*, *Y*-valorile asociate cu *x* sunt independente de *Z*-valorile corespunzătoare; altfel spus, dat fiind orice *x*, există o mulțime de *Y*-valori şi una de *Z*-valori ce apar asociate cu *x* în toate combinațiile posibile. Aceasta justifică numele dat *DMV* şi sugerează că *FD* sunt un caz particular de *DMV* (lucru precizat de următoarea teoremă):

Teorema 129 Fie $r(U)$ o relație şi $XY \subseteq U$. Dacă *r* satisface *FD* $X \rightarrow Y$, atunci ea satisface şi *DMV* $X \rightarrow\rightarrow Y$.

Demonstrație

Dacă *r* satisface $X \rightarrow Y$, atunci, conform propoziției 22, *r* admite o descompunere fără pierderi peste *XY* şi *XZ* (unde $Z = U - XY$); ca atare, conform teoremei 128, *r* satisface şi $X \rightarrow\rightarrow Y$

q.e.d.

Reciproca acestei teoreme este falsă; se poate observa din figura 52, de exemplu, că deşi prima relație satisface *DMV* $R \rightarrow\rightarrow C$, ea nu satisface *FD* $R \rightarrow C$. În consecință, se poate afirma că *DMV* este o constrângere strict mai slabă decât *FD*: pentru orice mulțime *M* de *DMV* există o mulțime *F* de *FD* astfel încât, dacă o relație satisface *F*, atunci ea satisface şi *M*.

Invers, ar fi însă incorect să afirmăm că *DMV* generalizează *FD*, deoarece nu este posibilă exprimarea *FD* cu ajutorul *DMV*: dată fiind o mulțime *F* de *FD*, nu este posibil, în general, să se găsească o mulțime *M* de *DMV* astfel încât o relație să satisfacă *F* dacă şi numai dacă ea satisface *M*.

5.3 Formele normale 4, 5 şi (3,3)

Extensia naturală a *FNBC* în cazul luării în considerare şi a *DMV* o constituie cea de-a patra formă normală (în raport cu mulțimi de *FD* şi *DMV*):

Definiția 130 O schemă de relație se zice a fi în *forma normală 4* (*FN4*), dacă partea stângă a oricărei *DMV* netriviale este o supercheie.

Demonstrația teoremei următoare este propusă cititorului drept exercițiu în problema 64.

Teorema 131 $FN4 \Rightarrow FNBC$ (o schemă în *FN4* este şi în *FNBC*).

Similar, dacă se consideră *DJ* împreună cu *FD*, se defineşte *forma normală 5* (*FN5*), mai degrabă cunoscută sub numele de *forma normală proiecţie-join* (*FNPJ*):

Definiţia 132 O schemă de relaţie se zice a fi în *FNPJ*, dacă orice *DJ* este implicată de mulţimea dependenţelor de cheie.

Teorema 133 $FNPJ \Rightarrow FN4$ (o schemă în *FN5* este şi în *FN4*).

Demonstraţia teoremei 133 este propusă cititorului drept exerciţiu în problema 65. Ilustrăm *FN4* şi *FN5* cu două contraexemple clasice din literatură:

Exemplul 47

Fie schema de relaţie *F*(*urnizor*)*P*(*rodus*)*A*(*gent vânzări*) fără nici o *FD*; ca atare, singura cheie fiind *FPA*, schema este în *FNBC*; dacă considerăm *DMV* $F \rightarrow\rightarrow P$ şi $F \rightarrow\rightarrow A$ (i.e. dacă un agent vinde un produs al unui furnizor, atunci el vinde orice produs al acelui furnizor), schema rezultată rămâne în *FNBC* dar nu este în *FN4*. Într-adevăr, se poate constata uşor că schema prezintă o anomalie de inserţie (de exemplu, dacă se încearcă adăugarea unui tuplu <Vasile, ventilator, Philips> la conţinutul tabelei din figura 53, pentru a nu viola cele două *DMV* ar mai trebui introduşi obligatoriu încă trei tupli: <Ion, ventilator, Philips>, <Vasile, mixer, Philips> şi <Vasile, aspirator, Philips>); descompunerea schemei în *FP* şi *FA* este însă în *FN4*.

Dacă considerăm în schimb $DJ \bowtie [AP,PF,AF]$ (i.e. dacă un agent vinde un produs, acest produs este fabricat de un furnizor, iar agentul vinde produse ale acestui furnizor, atunci agentul trebuie obligatoriu să vândă produsul respectiv şi de la acest furnizor şi nu doar de la alţi furnizori care ar fabrica şi ei respectivul produs), schema rezultată este în *FN4* dar nu este în *FNPJ*; într-adevăr, în absenţa oricărei *FD* sau *DMV*, *DJ* nu implică nici o *DMV* netrivială şi este deci în *FN4*; dar ea prezintă cel puţin o anomalie de ştergere (aşa cum se poate uşor observa din conţinutul prezentat în figura 53: de exemplu, ştergerea ultimei linii a tabelei violează *DJ*); descompunerea schemei în *AP*, *FP* şi *FA* este însă în *FNPJ*.

Agent	*Produs*	*Furnizor*
Ion	aspirator	Moulinex
Ion	mixer	Philips
Gheorghe	mixer	Moulinex
Gheorghe	aspirator	Philips
Ion	aspirator	Philips

Figura 53: O relaţie în FNBC/FN4 dar nu şi în FN4/FNPJ

Un corolar extrem de puternic al rezultatelor teoretice prezentate până acum este următorul (demonstraţia sa este propusă cititorului de problema 64):

Corolarul 134 O schemă este în *FNBC* (respectiv *FN4*) dacă oricare ar fi *d FD* (respectiv *DMV*) a schemei, există cel puţin o dependenţă cheie ce implică *d*.[44]

O altă extensie a *FNBC*, pentru cazul mulţimilor de constrângeri conţinând doar *FD*, a apărut necesară datorită relaţiilor în care sunt posibile anomalii de actualizare numai într-o anumită submulţime a tuplilor (iar în restul lor nu). Formalizarea acestor scheme „parţial violate" se face astfel:

Definiţia 135 O schemă *R* având o mulţime de constrângeri Γ se zice a fi în *forma normală* (*3,3*) (*FN*(*3,3*)) dacă, oricare ar fi propoziţia *F*, o *FD* netrivială $X \rightarrow A$ ţine în $\sigma_F(r)$ pentru orice conţinut valid *r* al *R* dacă şi numai dacă *X* este o supercheie a lui *R*.

Se poate uşor demonstra următoarea teoremă (vezi problema 68):

Teorema 136 $FN(3,3) \Rightarrow FNBC$ (o schemă în *FN*(*3,3*) este şi în *FNBC*).

[44] Se poate demonstra uşor că acest lucru nu este adevărat pentru *FNPJ* şi *DJ* (vezi problema 84).

5.4 Forma normală domenii-chei

Definiția 137 Fie o schemă de bd $R(U)$ peste mulțimea de atribute U și Γ o *mulțime de constrângeri* oarecare peste $R(U)$; Γ se zice *coerentă* dacă există cel puțin un conținut al R care să o satisfacă.

Pentru a putea defini riguros cea mai înaltă formă normală relațională cunoscută în literatură (domenii-chei) este nevoie întâi de a remarca posibila interacțiune, în anumite cazuri, a constrângerilor de domenii cu alte tipuri de constrângeri.

Astfel, de exemplu, dacă un domeniu este vid, atunci doar conținutul vid poate fi valid pentru relația în cauză! Într-un asemenea caz limită, cum atât *FD* cât și *DMV* țin în orice conținut vid (și nu doar ele!), domeniul vid implică satisfacerea oricărei *FD* și *DMV* (ca și a multor altor tipuri de dependențe) în orice conținut valid al relației!!

Similar, dacă un domeniu D_A conține doar o singură valoare, atunci atributul A corespunzător este funcțional dependent de orice (submulțime de) atribut(e) al(e) relației, adică orice *FD* $X \to A$ ține în orice conținut valid r al R, oricare ar fi $X \subseteq U$.

Mai subtil încă, nu este greu de remarcat totuși că nici o relație nu poate satisface o *DJ* având n componente dacă vreunul din domeniile implicate de *DJ* nu are minim n valori distincte!

Tocmai din asemenea motive se presupune întotdeauna că domeniile sunt infinite, deoarece acestea, conform lemei următoare, nu interacționează cu nici una dintre dependențele cunoscute (*FD*, *DMV*, *DJ*, *DIN* etc.):

Lema 138 Fie $R(U)$ o schemă de relație, D o mulțime de constrângeri de domeniu, K o mulțime de dependențe cheie, iar M o mulțime de *FD* și *DMV*;

a. dacă pentru orice atribut $A \in U$, $card(D_A) \geq 2$, atunci pentru orice *FD* sau *DMV* d peste U, d este implicat de $D \cup M$ dacă și numai dacă d este implicat de M;
b. dacă pentru orice atribut $A \in U$, $card(D_A) \geq n$, atunci pentru orice *DJ* j peste U cu n componente (i.e. de forma $\bowtie[X_1, X_2, ..., X_n]$, unde $X_i \subseteq U$, $\forall i$, $1 \leq i \leq n$), j este implicat de $D \cup K$ dacă și numai dacă j este implicat de K.

Demonstrație:

Partea *dacă* a demonstrației punctului a. este ușoară și o propunem cititorului ca exercițiu. Demonstrăm doar partea *numai dacă*, însă doar pentru *FD* (extensia ei la *DMV* ca și demonstrația punctului b. rămânând tot ca exercițiu – vezi problema 69).

Fie F o mulțime de *FD* și D o mulțime de constrângeri de domeniu oarecari fixate; să presupunem prin absurd că $\exists f \in F$ astfel încât f este implicată de $F \cup D$ dar f nu este implicată de F. Ca atare, există o relație r care satisface F dar violează f și D. Fie t_1, $t_2 \in r$ doi tupli care violează f și să notăm cu r_0 relația care conține doar aceștia doi tupli. Evident, $r_0 \subseteq r$, iar r_0 satisface F și violează f; mai mult, r_0 trebuie să violeze și D (în caz contrar, deoarece $F \cup D$ implică f, ea ar satisface și f).

Fie r_0' relația obținută din r_0 prin înlocuirea fiecărei valori ilegale (adică a celor ce nu satisfac constrângerile de domeniu D) cu una legală, cu o singură condiție: valorile ilegale identice, respectiv distincte între ele, sunt înlocuite cu valori legale identice, respectiv distincte (adică există un izomorfism între r_0 și r_0'). De remarcat că o asemenea înlocuire este întotdeauna posibilă, deoarece r_0 conține doar doi tupli și deci sunt cel mult două valori pentru orice atribut $A \in U$, iar prin ipoteză există cel puțin două valori legale pentru orice atribut.

Datorită izomorfismului, este evident că r_0' satisface f dacă și numai dacă r_0 satisface f; deoarece însă, prin construcție, r_0 violează f, rezultă că și r_0' violează f. Aceasta conduce însă la o contradicție, deoarece r_0' satisface atât F (fiind o submulțime a r care satisface F), cât și D (deoarece, prin construcție, nu conține valori ilegale), iar $F \cup D$ implică f; rezultă că presupunerea făcută este absurdă

q.e.d.

De notat că al doilea punct al lemei de mai sus are nevoie de o ipoteză mai puternică decât primul, deoarece contraexemplele unei *DJ* pot conține tot atâția tupli ca și *DJ* însăși.

Putem acum, în fine, defini și caracteriza forma normală domenii-chei:

Definiția 139 O schemă de relație $[R, \Gamma]$ se zice a fi în *forma normală domenii-chei* (*FNDC*) dacă orice constrângere *c* din Γ este implicată de constrângerile primitive din Γ^+.

Teorema 140 (*caracterizarea FNDC*)

O relație este în *FNDC* dacă și numai dacă ea nu are anomalii de actualizare.

Demonstrație:

($\Rightarrow$) Fie *R* în *FNDC*, *r* un conținut valid al *R*, iar *t* un tuplu compatibil cu *r*; atunci $r \cup \{t\}$ satisface toate constrângerile peste *R* deoarece, din definiția compatibilității, constrângerile primitive sunt satisfăcute, iar din definiția *FNDC*, acestea implică toate celelalte constrângeri.

Similar, dacă $t \in r$, atunci $r - \{ t \}$ satisface toate constrângerile peste *R* deoarece, evident, ștergerea unui tuplu nu poate viola nici o constrângere primitivă, iar din definiția *FNDC*, acestea implică toate celelalte constrângeri; ca atare, *R* nu are anomalii de actualizare.

($\Leftarrow$) Știind acum că *R* nu are anomalii de actualizare, să presupunem prin absurd că, totuși, ea nu este în *FNDC*;

R nefiind în *FNDC*, înseamnă că există cel puțin o constrângere *c* care nu este implicată de constrângerile primitive; aceasta implică faptul că există cel puțin un conținut *r* al *R* în care *c* nu ține, deși toate constrângerile primitive țin; cum mulțimea de constrângeri asociată *R* este coerentă, rezultă însă că există cel puțin un conținut r^* al *R* care este valid (în care, deci, ține și *c*); să construim *r* din r^* în doi pași: întâi ștergând din r^* toți tuplii, unul câte unul și apoi inserând în r^*, tot unul câte unul, toți tuplii din *r* (acest lucru este evident posibil, deoarece ambele conținuturi sunt finite, iar constrângerile peste *R* sunt statice); cum r^* este valid iar *r* nu, trebuie să existe, în cursul acestui proces de construcție, un tuplu *t* și două conținuturi *r'* și *r"* astfel încât:

1. *r'* valid
2. *r"* invalid și
3. $r" = r' \cup \{ t \}$ sau $r" = r' - \{ t \}$.

Să considerăm fiecare dintre cele două cazuri posibile în parte:

- dacă $r" = r' - \{ t \}$, atunci $r' \subset r^*$ și deci *r'* este valid; cum *r"* nu este valid, rezultă că *R* prezintă o anomalie de ștergere!
- dacă $r" = r' \cup \{ t \}$, atunci $r' \subset r$ și, ca atare, deoarece *r* satisface constrângerile primitive, rezultă că *t* este compatibil cu *r'*; cum *r"* este invalid, rezultă că *R* prezintă o anomalie de inserție!

Fiindcă în oricare dintre cele două cazuri posibile este contrazis faptul că *R* nu are anomalii de actualizare, rezultă că presupunerea făcută este absurdă și deci că *R* este în *FNDC*

q.e.d.

Exemplul 48

Fie relația *LOCALITĂȚI*(*Localitate*, *Primar*, *Prefect*) din exemplul 20 și figura 54, având următoarele constrângeri: *cheie*(*Localitate*)

dom(*Localitate*) = *şir*(*32*) *Localitate* → *Primar*,*Prefect*
dom(*Primar*) = *şir*(*32*)
dom(*Prefect*) = *şir*(*32*)

Este trivial că, sintactic, conform *FD* asertate, ea este în *FNBC*; semantic însă, evident, această schemă ar trebui să conțină *FD* suplimentară *Primar* → → *Prefect*, deoarece un primar conduce o unică localitate, fiecare localitate aparține unui unic județ, iar fiecare județ are un unic prefect (şi tocmai din cauza acestei *FD* "ascunse"[45], descompunerea din exemplul 20 este în *FNBC* dar se face cu pierderi!).

Considerând conținutul *r* şi tuplul *t* din figura 54, este evident că *t* este compatibil cu *r*, dar dă naştere la o anomalie de actualizare, deoarece $r \cup \{t\}$ violează *FD Primar* → *Prefect* şi, ca atare, această schemă nu este în *FNDC*.

În plus, desigur, aşa cum era de aşteptat, *FD Primar* → *Prefect* nu este implicată de constrângerile primitive.

r

Localitate	*Primar*	*Prefect*
Alba-Iulia	Ionescu	Văcărescu
Sebeş	Pop	Văcărescu
Lancrăm	Octavian	Văcărescu
Oltenița	Ilici	Serghei
Vaslui	Vasilescu	Văcărescu

t

Bucureşti	Ilici	Vladimir

Figura 54: O relație ce nu este în FNDC şi un tuplu producând o anomalie de inserție

Teorema 141 (*FNDC este cea mai înaltă formă normală relațională*)

Fie $[R, \Gamma]$ o schemă de relație cu proprietatea că toate constrângerile de domeniu permit cel puțin *k* valori distincte, unde $k \geq 1$ este numărul de componente al *DJ* având cel mai mare număr de componente dintre toate *DJ* ale unei acoperiri *J* a *DJ* din Γ^+, iar, în absența *DJ*, ele permit cel puțin două valori distincte pentru orice atribut;

1. (*FNDC* ⇒ *FNPJ*) dacă $[R, \Gamma]$ este în *FNDC*, atunci ea este şi în *FNPJ*
2. (*FNDC* ⇒ *FN4* ∧ *FNDC* ⇒ *FN*(*3*,*3*)) dacă $[R, \Gamma]$ este în *FNDC*, atunci ea este şi în *FN4* şi în *FN*(*3*,*3*).

Demonstrație:

1. Fie $d \in \Gamma^+$ o *DJ*; cum *R* este în *FNDC*, rezultă că orice *DJ* $j \in J$ este implicată de constrângerile primitive din Γ^+. Conform lemei 138.b, orice *j* este atunci implicat de dependențele cheie din Γ^+; ca atare, din definiția acoperirilor *d* fiind implicat de *J*, rezultă că *d* este implicat de dependențele cheie şi deci *R* este în *FNPJ*.

2. Demonstrația acestui subpunct este propusă cititorului de problema 85 q.e.d.

[45] Evident că, din nou, aceasta se datorează existenței următoarei diagrame de funcții (necomutative!): *Primar* : *LOCALITĂȚI* ↔ *PERSOANE*, *Prefect* : *JUDEȚE* ↔ *PERSOANE*, *Județ* : *LOCALITĂȚI* → →*JUDEȚ*

5.5 Problema implicației pentru dependențele multivaluate

Problema implicației pentru *DMV* poate fi abordată cu aceleaşi două tehnici folosite pentru *FD*. Lăsăm cititorului drept exercițiu tratarea ei cu ajutorul regulilor de inferență (vez problema

6. Câteva contraexemple de modelare demonstrând că proiectarea bazelor de date nu trebuie abordată relațional

Principalele trei neajunsuri ale abordării relaționale a modelării conceptuale a datelor sunt următoarele:

- MRD nu este întotdeauna capabil să ofere soluții corecte de proiectare nici măcar pentru baze de date (bd) extrem de simple;
- abordarea relațională a modelării datelor nu incită, în general, la formularea completă şi neambiguă a problemelor de modelat şi cu atât mai puțin la analiza lor aprofundată (pentru a vedea dacă şi în ce condiții ele au soluție); şi, în sfârşit, că
- înseşi exemplele din literatura relațională ilustrând metodologia de proiectare a bd constituie de prea multe ori adevărate contraexemple de modelare a datelor.

Articolul doreşte să contribuie la impunerea concluziei că MRD este total inadecvat modelării datelor; el ar trebui utilizat doar pentru implementarea „datalogică" [TL] a unor modele de date „infologice" [TL] („semantice" [McL]).

Abordarea relațională a modelării datelor presupune [Bre] existența: unei mulțimi finite „universale" Ω a tuturor obiectelor de modelat într-o bd; a unei colecții finite $\mathcal{V}$ de mulțimi de valori (zise „domenii"); a unei mulțimi finite de funcții $\mathcal{A} \subseteq \mathrm{Hom}(\Omega, \mathcal{V})$ (zise „atribute"); şi a unei mulțimi finite $\mathcal{D}$ de propoziții particulare din calculul predicatelor de ordinul întâi (zise „dependențe"). Precizăm că, simplificând, prin mo-delarea datelor înțelegem aici formularea problemei de proiectare a bd (în termenii unui model oarecare al datelor).

Indiferent de tipul de metodologie folosit (ascendent, prin „sintetizare", sau descendent, prin „descompunere") proiectarea „schemei" (structurii logice a) unei bd relaționale (bdr) pleacă, în general, de la $\mathcal{A}$ şi $\mathcal{D}$ cu scopul de a obține o colecție finită de submulțimi $\mathcal{R}_i \subseteq \mathcal{A}$ (zise „scheme de relații") care să memoreze mulțimea:

$\mathcal{C} = \cup_{\mathcal{R}i \subseteq \mathcal{A}} \Pi_{a \in \mathcal{R}i} \mathrm{Im}(a)$ zisă „conținutul" bd.

Bd astfel proiectată trebuie să îndeplinească minimum două condiții:

- $\cup_{i \in \{1,\ldots,n\}} \mathcal{R}_i = \mathcal{A}$ şi
- $\mathcal{C}$ să satisfacă simultan toate dependențele din $\mathcal{D}$.

În general, se caută obținerea de scheme care să includă „implicit" în structura relațiilor $\mathcal{R}_i$ $\forall d \in \mathcal{D}$ (sau o acoperire minimală [Ull] a lor). Eventualele funcționalități între atributele unei relații [Ma1] urmează a fi implementate specificând o mulțime de „chei" (submulțimi minimal injective de atribute ale schemei) asociate tabelei memo-rând conținutul relației respective. Foarte rar sunt considerate teoretic şi alte tipuri de constrângeri în afara dependențelor [Nic] şi şi mai rar este analizat impactul acestora asupra proiectării bd. Practic, implementările de modele relaționale [SB] oferă cel mult posibilitatea specificării unor „trăgaciuri" („triggers"), fiecare în parte impunând o asemenea constrângere particulară.

6.1 Problema codurilor poştale

(E.1.) [Ull]: Fie schema de relație *OAC*, unde *O* = Oraşe, *A* = Adrese, *C* = CoduriPoştale, cu dependențele funcționale (df) $OA \rightarrow C$ (orice adresă dintr-un oraş are un singur cod poştal) şi $C \rightarrow O$ (orice cod poştal este asignat unui singur

oraş, dar nu neapărat şi unei singure adrese din acel oraş – unde prin adresă se înțelege, de exemplu, strada şi numărul).

Schema are cheile *OA* şi *AC*, este în FN3, dar nu şi în FN Boyce-Codd (FNBC). Schema admite descompuneri în FNBC care au proprietatea de join fără pierderi (jfp) dar nici una care să păstreze dependența $OA \rightarrow C$. Descompunerile ei nefiind deci practic interesante, vom analiza doar schema originală. Fie următorul conținut al acesteia (perfect posibil în raport cu cheile tabelei !):

Oraşe	*Adrese*	*CoduriPoştale*
Sibiu	Carpaților, 10	2400
Iaşi	Ştefan cel Mare, 12	2400

Acest conținut este evident aberant (două orașe distincte nu pot avea același cod). Construirea unui asemenea contraexemplu este posibilă deoarece specificarea cheii *AC* este inutilă (ea neputând impune df $C \rightarrow O$), iar forțarea cheii *C* imposibilă (căci *C* nu este injectiv).

Am arătat în [Ma1] de ce *MRD nu poate oferi nici o soluție pentru acest exemplu*: este nevoie de o constrângere suplimentară, de tip comutativitate de diagrame de funcții, de neexprimat deci în termeni relaționali. Între timp, am rafinat şi mai mult [Ma2] soluția oferită cu acel prilej, distingând între orașele cărora li s-a asignat un singur cod (indiferent de adrese) şi cele care au fost „zonate" pe (porțiuni de) străzi sau chiar imobile (cărora li s-a asignat câte un cod în parte). E adevărat că această soluție e mai „complicată", căci conduce la o schemă datalogică cu 3 tabele şi 3 constrângeri explicite (în afara cheilor), în loc de 2, respectiv 1 (deoarece trebuie impusă corespunzător partiționării orașelor şi partiționarea codurilor poştale). Dar ea prezintă două avantaje suplimentare deloc neglijabile: pe lângă o reflectare corectă a semanticii modelate (ceea ce simplifică mult formularea de întrebări asupra conținutului bd) şi reprezentarea datalogică este mult mai naturală – căci nu se repetă fiecare oraş „nezonat" de sute de ori (pentru fiecare adresă a sa în parte) şi nici nu este obligatorie memorarea adreselor din aceste orașe (lucru care poate nici nu interesează!).

6.2 Problema orarelor universitare

(E.2.) [Ull]: Fie schema de relație *CPOASG* unde *C* = Cursuri, *P* = Profesori, *O* = Ore, *A* = Amfiteatre, *S* = Studenți, *G* = Grade, cu df $C \rightarrow P$ (fiecare curs este ținut de un singur profesor), $OA \rightarrow C$ (într-un amfiteatru nu se pot ține simultan mai multe cursuri), $OP \rightarrow A$ (un profesor nu se poate afla simultan în mai multe amfiteatre), $OS \rightarrow A$ (un student nu se poate afla simultan în mai multe amfiteatre) şi $CS \rightarrow G$ (absolvind un curs, fiecare student obține cel mult un grad).

Schema are doar cheia *OS* iar df listate constituie o acoperire minimală. Schema admite descompunerea „frumoasă" *CSG* (cu cheia *CS*), *CP* (cu cheia *C*), *COA* (cu cheile *CO* şi *OA*), *COS* (cu cheia *OS*) care este în FNBC, are proprietatea jfp, dar nu păstrează nici $OP \rightarrow A$, nici $OS \rightarrow A$. (Sunt posibile şi alte descompuneri echivalente cu aceasta, dar care „nu se suprapun atât de bine intuiției noastre despre ce anume informații ar trebui tabulate în bd" [Ull] sic!). Schema admite însă şi descompunerea *CSG*, *CP*, *COA*, *POA* (cu cheile *PO* şi *OA*), *SOA* (cu cheia *SO*) care este în FN3, păstrează toate dependențele şi are şi proprietatea jfp. Se afirmă că adăugarea la schemă a dependenței multivaluate (dmv) $C \twoheadrightarrow OA$ „permite, împreună cu df de mai sus, derivarea tuturor dmv pe care le-am simți intuitiv ca fiind valide" [Ull]. Astfel îmbogățită, schema admite o

descompunere cu proprietatea jfp în FN4 (şi anume *COA*, *CP*, *CSG*) dar care nu păstrează nici ea dependenţele $OP \rightarrow A$ şi $OS \rightarrow A$. (De remarcat în plus că, numai şi numai pentru a avea proprietatea jfp, descompunerea în FNBC necesită, suplimentar faţă de cea în FN4, şi tabela *COS* !).

Ne rămâne deci de analizat, din nou, doar descompunerea în FN3 (celelalte permit în mod evident conţinuturi invalide ale bd, din moment ce nu păstrează toate dependenţele). Fie următorul conţinut al acestei bd:

C	***S***	***G***
c_1	s_1	g_1
c_1	s_2	g_2

C	***P***
c_1	p_1
c_2	p_1

C	***O***	***A***
c_1	o_1	a_1
c_1	o_2	a_1
c_2	o_2	a_2

P	***O***	***A***
p_1	o_1	a_2
p_1	o_2	a_3

S	***O***	***A***
s_1	o_1	a_3
s_2	o_2	a_2

Invităm cititorul să verifice că fiecare tabelă în parte conţine doar tupli valizi în raport cu cheile ei. Totuşi, acest conţinut al bd este aberant: orarul memorat de *COA* obligă p_1 să se afle simultan în a_1 şi a_2, atât la o_1 cât şi la o_2 (şi cum, conform *POA*, la o_1 el s-a aflat în a_2, rezultă că c_1 a rămas descoperit; mai mult, atât c_1 cât şi c_2 au fost descoperite la o_2, căci p_1 figurează în a_3 la această oră !); similar, s_1 ar fi trebuit să se afle la o_1 simultan în a_1 şi a_2, dar el a ales -sau a nimerit, puţin importă- a_3, tot aşa pre-cum, la aceeaşi oră, s_2 ar fi trebuit să fie în a_1 dar a fost în a_2!).

Se poate observa că, în general:

- *COA* împiedică într-adevăr ca într-un amfiteatru să fie programate simultan mai multe cursuri şi ca un acelaşi profesor să fie programat simultan în mai multe amfiteatre; în schimb, *COA* permite memorarea informaţiei că un acelaşi profesor (student) ar trebui să se afle simultan în mai multe amfiteatre.
- *POA* împiedică într-adevăr ca mai mulţi profesori să figureze simultan într-un acelaşi amfiteatru şi ca un acelaşi profesor să figureze simultan în mai multe amfiteatre, dar *POA* nu poate obliga profesorii să fie, la orele prevăzute de conţinutul *COA*, în vreunul din amfiteatrele în care ar trebui să-şi ţină cursurile (eventual simultan – sic!).
- *SOA* împiedică într-adevăr ca vreun student să figureze simultan prezent în mai multe amfiteatre, dar nu îl poate obliga ca, la orele prevăzute de conţinutul *COA*, să se afle măcar într-unul din amfiteatrele în care sunt programate cursuri la care el este înscris.

În concluzie, ca şi în cazul (E.1.) de mai sus, nici un conţinut al vreunei bdr pentru acest exemplu nu va fi cu adevărat valid decât pur întâmplător. (Din punct de vedere relaţional, chiar şi conţinuturile analizate de noi sunt valide: joinul tuturor tabelelor elimină orice tuplu aberant ar conţine vreuna din ele, iar definiţia proprietăţii jfp nu porneşte de la conţinutul descompunerii ci de la cel al „relaţiei universale"). Deci nici pentru acest exemplu nu există vreo soluţie relaţională corectă.

Dar cel mai grav lucru în legătură cu acest exemplu este faptul că problema propusă nu este nici complet, nici precis formulată şi că, în general, ea nu are soluţie (matematică, darămite de modelare relaţională a datelor!) !!

Într-adevăr, să presupunem că orice student se poate înscrie simultan la orice cur-suri (aşa cum reiese implicit din context, în lipsa unei restricţii în acest sens). Este uşor de verificat că, în general, nu îi putem garanta că va putea participa la toate orele lor: îl putem cel mult informa despre ce posibilităţi de alegere are în orice moment. Ceea ce înseamnă că, în acest caz, *SOA* devine inutilă! E drept că, în principiu, ea ar putea memora unde

anume s-a aflat fiecare student la orice oră: dar, dacă impunem constrângerea explicită ca un student să nu poată participa decât la cursurile la care este înscris (şi anume: $(\forall o\in O)(\forall s\in S)(\forall a\in A)(\exists c\in C)(\exists g\in G)(\varphi_1(o,a)=c \wedge \varphi_2(c,s)=g)$, unde $\varphi_1 : G_1 \to C$, $\varphi_2 : G_2 \to G$ şi $G_1 \subseteq O \times A$, $G_2 \subseteq C \times S$), atunci nu vom putea me-mora, de exemplu faptul că în realitate s_1 a fost la o_1 în a_3; iar dacă impunem această constrângere, bd tot nu ne va putea spune ce cursuri a frecventat de fapt un student (căci joinul natural între *OSA* şi *COA* nu conține, de exemplu, nici o informație despre s_1)!

Probleme la fel de mari sunt şi cu alcătuirea orarului, chiar în ipoteza simplificatoare că nu ne interesează faptul că unui student anume i se oferă sau nu şansa de a putea frecventa regulat toate cursurile la care s-a înscris. Şi în acest caz este uşor de verificat că, dacă ne propunem ca obiectiv principal folosirea spațiului disponibil (amfiteatrele), atunci *POA* devine inutilă: dacă dorim concomitent ca orice oră de curs să fie acoperită cu un profesor (i.e. să impunem constrângerea explicită adițională: $(\forall o\in O)(\forall p\in P)(\forall a\in A)(\exists c\in C)(\varphi_3(c)=p \wedge \varphi_1(o,a)=c)$, unde $\varphi_3 : O \to P$), atunci tot nu vom putea memora, de exemplu, informația că, în realitate, p_1 a fost în a_3 la o_2; iar da-că nu avem această pretenție, atunci oricum bd nu ne poate spune ce cursuri a ținut de fapt un profesor (căci joinul natural între *COA* şi *POA* nu conține, de exemplu, nici o informație despre p_2)! Mai mult, în primul din aceste subcazuri vom fi probabil adese-ori obligați să angajăm de urgență noi profesori şi/sau să plătim degeaba salariul unora dintre ei!!

Similar, dacă ne stabilim ca obiectiv principal încărcarea profesorilor disponibili, atunci *COA* devine inutilă (căci putem obține orarul din *POA* şi *CP*). Mai mult, dacă dorim concomitent să putem efectiv ține toate cursurile, atunci –în general– sau vom fi obligați să construim (închiriem) noi amfiteatre (ziua neavând decât 24 de ore!) sau vom ține unele goale! Ca să nu mai vorbim despre faptul că este perfect posibil ca, în acest caz, să programăm cursuri la care nu s-a înscris nici un student!!

Spațiul nepermițându-ne dezvoltarea mai multor considerații pe marginea acestui exemplu, reținem doar concluzia acestei analize: *mai grav decât oferirea unei soluții greşite la o problemă, mai grav încă decât încercarea de a soluționa o problemă cu mijloace teoretice inadecvate, ni se pare formularea ambiguă, incompletă a problemei şi, mai ales, „repezirea cu capul înainte" în rezolvarea ei, fără a studia în prea-labil dacă şi în ce condiții ea are sau nu soluții.* Considerăm că morala acestui fiasco relațional trebuie să fie următoarea: nu orice manipulare sintactică, oricât de „solid" ar fi ea justificată de o teorie de nivel sintactic, nu poate produce decât întâmplător o re-prezentare corectă a semanticii de la care s-a pornit! Degeaba definiții de noi concep-te, degeaba algoritmi, degeaba teoreme –oricât de frumoase în sine!– atunci când premizele abordării sunt firave!! Aceste triste constatări, care ar trebui să pună pe gânduri în mod serios pe orice fan al modelării relaționale, sunt cu atât mai elocvente cu cât a-proximativ o jumătate din cap. 5 din [Ull] (dedicat proiectării bdr) gravitează tocmai în jurul acestui exemplu!

6.3 Problema premizelor universitare

(E.) [Ull]: Fie schema de relație *CSPA*, unde *C* = Cursuri, *S* = Studenți, *P* = Premize, *A* = Ani, cu df $SP \rightarrow A$ (un student a absolvit o premiză într-un singur an) şi dmv „încastrate" („embedded") *C*—>>*S* | *P* (un student se poate înscrie la mai multe cur-suri într-un singur an; un curs poate fi condiționat de absolvirea mai multor premize; iar absolvirea unui curs poate constitui o premiză a înscrierii la mai multe cursuri; menționăm că printr-o premiză se înțelege tot un curs). Schema are doar cheia *CSPA* şi admite descompunerea în FN4 *CS*, *CP*, *SPA* (cu cheia *SP*) care are proprietatea jfp şi şi păstrează toate dependențele.

Am arătat [Ma1,Ma2] că dmv încastrate nu se impun obiectiv: „necesitatea" lor este doar un rezultat al slăbiciunilor MRD, căci atât conceptul de dmv, cât şi presu-punerea existenței unor „relații universale" sunt nejustificat de tari. Acest lucru este trivial verificabil şi în cazul particular al exemplului de față. Trebuie recunoscut cu onestitate faptul că explicația ce am dat mai sus C—>>S | P modelează corect realitatea considerată şi că această explicație definește pur şi simplu două relații matematice, una peste C şi S, cealaltă peste C şi P. Lăsând la o parte modul „transcendent" în care se poate ajunge, în general, la concluzia că unei scheme relaționale trebuie să i se adauge o anumită dmv încastrată (singura indicație că ar putea fi eventual nevoie de ase-menea dependențe fiind furnizată doar sintactic, de algoritmul de testare a proprietății jfp!), să arătăm doar că şi această schemă permite conținuturi aberante. Fie, de exemplu:

C	*S*
c_1	s_1
c_1	c_3
c_2	c_5
c_5	c_1

C	*P*
c_1	c_2

S	*P*	*A*
s_1	c_4	a_1

Fiecare conținut în parte este valid în raport cu cheile tabelei respective. Se observă însă imediat că acest conținut al bd este aberant, căci s_1 nu ar trebui să aibă dreptul să se înscrie la c_1 (deoarece el nu a absolvit nici una din premizele c_2 şi c_3 necesare a-cestuia). Cauza este deci de aceeaşi natură cu cea semnalată pentru exemplele de mai sus: proprietatea jfp este prea slabă şi nu poate împiedica memorarea de tupli aberanți în tabelele descompunerii (ba, lucru şi mai grav, joinul natural ce intervine în defini-rea acestei proprietăți disimulează existența unor asemenea tupli!). Pe scurt, necazul cu această descompunere este că *CS* nu poate împiedica înscrierea la un curs studenți-lor care nu şi-au luat premizele necesare acelui curs.

Acesta nu este însă singurul necaz cu această schemă relațională: ea nu poate impune nici condiția ca mulțimea premizelor oricărui curs să fie inclusă în mulțimea cursurilor (de exemplu, c_4 din *SPA*, dar nu numai: de unde putem şti oare că c_3 din *CP* este la rândul lui curs?). Şi mai grav este însă faptul că o asemenea schemă nu poate împiedica condiționările circulare de cursuri! (Mai bine deci că MRD nu este capabil să exprime constrângerea ce ar interzice studenților înscrierea la cursurile pentru care nu şi-au luat premizele necesare căci, în acest caz, nici un student n-ar reuşi vreodată să se înscrie la c_1, c_2 sau c_5!!).

6.4 Problema contravenţiilor rutiere

(E.4.) [ZM]: Fie schema de relaţie *NFMAVPCID*, unde *N* = NrÎnmatriculare, *F* = Fabricant, *M* = Model, *A* = An, *V* = Valoare, *P* = Proprietar, *C* = Carnet, *I* = Încălcare-LegeCirculaţie, *D* = DataÎncălcare, cu dependenţele:

D_1: $N \rightarrow F$

D_2: $N \rightarrow A$ D_3: $N \rightarrow M$

D_4: $N \rightarrow V$

D_5: $N \rightarrow P$

D_6: $N \rightarrow C$ D_7: $N \twoheadrightarrow ID$

D_8: $FAM \rightarrow V$

D_9: $FAM \twoheadrightarrow NPCID$ D_{10}: $P \rightarrow C$ D_{11}: $P \twoheadrightarrow ID$

D_{12}: $P \twoheadrightarrow NFAMV$

D_{13}: $C \rightarrow P$

D_{14}: $C \twoheadrightarrow ID$ D_{15}: $C \twoheadrightarrow NFAMV$

Credem că cititorul este de aceeaşi părere cu noi: în faţa unei asemenea liste de dependenţe, înainte de a analiza soluţia relaţională a acestei probleme este preferabilă analizarea punerii ei! Numărul mare şi complexitatea unora dintre dependenţele asertate (de câtă imaginaţie, de câtă muncă o fi fost nevoie pentru abstractizarea D_9, D_{12} şi D_{15}?!) ne pun pe gânduri: ce vrea să spună fiecare în parte? sunt oare toate justificate? sunt ele consistente? modelează ele oare corect realitatea avută în vedere? Singurele informaţii suplimentare furnizate de autori [ZM] sunt doar că se doreşte ca bd să ofere date despre:

- fabricantul, modelul, anul de fabricaţie şi proprietarul fiecărei maşini înmatriculate
- valoarea curentă a fiecărui tip de maşină
- carnetul de conducere al oricărei persoane, precum şi posesorul oricărui carnet
- istoria tuturor încălcărilor legii circulaţiei (cod contravenţie, data contravenţiei) de către orice şofer.

Cu D_1 la D_5 putem fi de acord la prima vedere: orice maşină înmatriculată este de un unic model, a fost produsă într-un unic an de către un unic fabricant, are o unică valoare şi se află în posesia unui unic proprietar. D_6 ne stârneşte însă primele dubii: în ce sens îi asociem unei maşini un unic carnet de conducere? În general, ea poate fi condusă de mai mulţi şoferi (chiar la un acelaşi drum!). Este oare vorba despre carne-tul proprietarului ei (care este, într-adevăr unic)? Dacă da, de ce nu îl asociem proprie-tarului, urmând ca, la nevoie, să stabilim legătura între maşină şi carnetul acestuia prin tranzitivitate? (Lucru canonic şi în limbaj natural, căci întrebarea „Ce carnet de con-ducere are maşina *m*?” este lipsită de sens şi nimeni nu ar ghici că, de fapt, am vrut să întrebăm „Ce carnet de conducere are proprietarul maşinii *m*?”).

D_7 este şi mai ciudată: oare chiar se doreşte memorarea istoriei tuturor contravenţiilor săvârşite la volanul tuturor maşinilor? (căci nu ne putem închipui totuşi că maşinile încalcă singure vreo lege!). Şi dacă este aşa, nu ar fi mai normal să memorăm şi şoferii care le-au comis? (sau se doreşte cumva interzicerea dreptului de circulaţie al maşinilor la volanul cărora s-au comis prea multe contravenţii? sic!).

D_8 ne obligă la prima revenire asupra acceptului dat iniţial dependenţelor D_1 la D_7. În mod clar, D_4 a fost listată doar din dorinţa de a aserta cât mai multe dependenţe cu putinţă (toate?)! Ea este desigur redundantă (şi nu doar semantic: şi sintactic ea poate fi derivată din D_1, D_2, D_3 şi D_8!). Mai mult, simţim în acest moment nevoia abstractizării unui nou obiect de modelat în această bd: tipul unei maşini. Definim un tip de maşină ca fiind clasa de echivalenţă a tuturor maşinilor de un acelaşi model, produse într-un acelaşi an de către un acelaşi fabricant. Relaţional, putem modela aceasta con-siderând

atributul T = Tip şi înlocuind D_1 la D_4 şi D_8 cu: $N \to T$, $T \to V$ şi $T \leftrightarrow FAM$ (unde $X \leftrightarrow Y$ constituie o abreviere pentru $X \to Y \wedge Y \to X$).

Chiar înlocuind *FAM* cu T în partea stângă a D_9, vom fi totuşi obligaţi să respirăm adânc (şi îndelungi minute!) după terminarea „rostirii înţelesului" ei: oricare ar fi maşinile m_1 şi m_2 de un acelaşi tip t, există o maşină m_3 astfel încât ori ea are acelaşi număr de înmatriculare, acelaşi proprietar, care are acelaşi carnet (sic!) şi la volanul ei s-au săvârşit aceleaşi contravenţii, la aceleaşi date ca şi pentru m_1 şi, în acelaşi timp, ea este ce acelaşi tip şi a costat la fel cu m_2 (sic!), ori ea (m_3 pentru cei ce au uitat deja de unde am plecat!) are acelaşi număr de înmatriculare, acelaşi proprietar, care are ace-laşi carnet (re-sic!) şi la volanul ei s-au săvârşit aceleaşi contravenţii, la aceleaşi date ca şi pentru m_2 şi, în acelaşi timp, ea este de acelaşi tip şi a costat la fel (re-sic!) cu m_1. Mărturisim că, ajunşi prima oară în acest punct al analizei de faţă, nu ne-am putut abţine de la un hohot de râs moromeţian! Să fi fost de vină abordarea relaţională a modelării datelor în general? sau, în particular, acest nefericit concept zis dmv? sau a-ceastă utilizare a lui de către autori [ZM]? Probabil toate laolaltă!

Am pomenit deja (la (E.)) că dmv este inutil de „tare" şi că mult mai potrivită ni se pare utilizarea în loc a conceptului consacrat de relaţie matematică. Ridicându-ne corespunzător de la nivelul relaţional la cel semantic, matematic, fraza de mai sus ar vrea să ne spună că fiecărui tip de maşină i se asociază o mulţime de maşini de acel tip, fiecare dintre acestea aflându-se în posesia câte unui proprietar, care proprietari au fiecare câte un carnet de conducere şi au săvârşit contravenţii la diverse date. Trebuie recunoscut că, şi la acest nivel, asertarea D_9 este tot ridicolă: ea ne spune în fond doar că surjecţia canonică $N \to T$ nu este injectivă! (ceea ce, este drept, ne întăreşte încă o dată convingerea că am făcut bine să tipizăm maşinile şi că relaţia de echivalenţă considerată cu acel prilej nu aduce cu sine doar avantaje sintactice, de eleganţă a formalizării, ci chiar corespunde semanticii realităţii de modelat). Acest lucru fiind trivial în general, el nu merită explicitat (nu ne-ar ajunge o viaţă întreagă să caracterizăm un singur obiect dacă ne-am apuca să înşirăm tot ceea ce el ar putea fi dar nu este!).

D_{10} confirmă presupunerea ce am făcut cu ocazia analizei D_6 (şi cum nimic de aici încolo nu o va infirma, vom putea elimina liniştiţi şi D_6 din formularea problemei: atât semantic cât şi sintactic ea se deduce tranzitiv din D_5 şi D_{10}!). D_{11} ne spune că, în general, un proprietar de maşină poate încălca legea circulaţiei de mai multe ori, eventual în acelaşi mod, dar la diverse date. Să admitem ipoteza simplificatoare că nu ne interesează cazurile în care un acelaşi şofer a încălcat această lege în acelaşi mod de mai multe ori într-o aceeaşi zi (în caz contrar, n-am avea decât să memorăm asociat şi ora contravenţiei). Dar de ce oare „proprietar" şi nu „şofer", aşa cum este cazul nu nu-mai în realitate, ci şi în specificaţiile informale a problemei din [ZM]? Mai ales că, probabil, chiar şi la dânşii poţi fi şofer fără a fi proprietar de maşină şi viceversa (mă-car temporar!).

În consecinţă, noi am abstractiza şi submulţimea Şoferi (unde, evident, într-un model al datelor mai evoluat, atât Şoferi cât şi Proprietari ar fi incluse într-o mulţime de Oameni). Identificând elementele acestei noi submulţimi prin intermediul injecţiei S = Şoferi, putem înlocui D_{11} prin S—>>ID aşa cum este normal (sau, eventual, prin SN—>>ID dacă interesează realmente şi maşinile la al căror volan s-a comis fiecare contravenţie în parte). Similar, D_{10} se poate înlocui prin mult mai naturala $S \to C$. De remarcat în plus că oricum, chiar dacă rescriem astfel D_{10} şi D_{11} sau nu, ridicola D_7 devine inutilă semantic de îndată ce considerăm D_{11} şi vom renunţa deci şi la ea (sigur că şi sintactic ea se poate deduce din D_5 şi D_{11}!).

Invităm cititorul să analizeze D_{12} în detaliu singur (nouă ne-a ajuns D_9!!); semantic, ea ne spune că un proprietar poate poseda simultan mai multe maşini (nu neapărat de acelaşi tip). Şi această informaţie este însă trivială: dacă nu asertăm şi $P \to N$, urmează în

mod evident că $N \rightarrow P$ nu este injectivă (în general, relația inversă unei funcții nu este funcțională!). Deci nu este nevoie nici de D_{12}.

D_{13} trebuie înlocuită cu $C \rightarrow S$. Chiar dacă nu facem această înlocuire, având în vedere şi D_{10}, rezultă oricum că D_{14} şi D_{15} sunt la rândul lor inutile atât semantic cât şi sintactic (prima, în virtutea D_{11}; cea de a doua, în virtutea D_{12}, care tocmai am văzut că este inutilă în virtutea D_5!). Şi uite aşa, din inutilitate în inutilitate, s-au scris 15 dependențe din care merită să păstrăm doar 8!!

Recapitulând, din punct de vedere semantic (i.e. al modelării datelor) D_4, D_6, D_7, D_9, D_{12}, D_{14} şi D_{15} sunt inutile. Şi, în mod evident, nu MRD în sine este de blamat pentru aceasta. trebuie să ne punem atunci întrebarea dacă măcar din punct de vedere sintactic (i.e. al proiectării bd) ele nu sunt cumva necesare (ceea ce ar fi în defavoarea metodologiei de proiectare propusă de autori [ZM], dar i-ar scuza din punct de vedere al modelării). Cititorul poate însă verifica că algoritmul de proiectare propus în [ZM] nu are nici el nevoie de aceste dependențe (ci, aşa cum ne-am fi şi aşteptat, le elimină pe toate – cu un consum desigur inutil de timp şi resurse calculator!), lucru de altfel vizibil şi în descompunerea obținută ca soluție, descompunere în care nici una din aceste dependențe nu se reflectă.

Desigur că apare legitimă întrebarea la ce bun să complici lucrurile într-atât (şi într-un asemenea hal!) încât aproape să dublezi în mod inutil formularea unei probleme „de jucărie", atât de simplă în fond? Cum am mai putea stăpâni modele adevărate, cu mii de atribute şi sute de dependențe [Ma1] dacă am proceda la fel? Este oare indicată asertarea tuturor dependențelor ce ne „trec prin cap"? Sau, şi mai rău, trebuie oare să încercăm toate combinațiile posibile de atribute pentru a lista închiderea tranzitivă a dependențelor existente (sic!)? Desigur că răspunsul la aceste ultime două întrebări trebuie să fie (categoric!) negativ: *militățm pentru o analiză aprofundată a problemelor de modelat, pentru o formulare corectă, cât mai concisă şi elegantă cu putință a fiecăreia dintre ele.* Desigur că, din când în când, putem greşi inclusiv prin asertarea unor informații redundante. Dar de aici şi până la dublarea inutilă a dimensiunii modelului este un pas uriaş ce merită evitat! (mai ales că dacă reunim toate df cu aceeaşi parte stângă raportul util/superfluu devine şi mai dezastruos: 4/11, i.e. pentru fiecare două constrângeri necesare au fost adăugate în medie alte 5,5 inutile!).

În concluzie, chiar în termeni relaționali, acest adevărat contraexemplu de modelare a datelor ar trebui reformulat astfel: fie schema de relație *NTFMAVPSCID*, cu dependențele:

D_1': $N \rightarrow TP$ (o maşină aparține unui singur tip şi are un unic proprietar)

D_2': $T \rightarrow FMAV$ (un tip este caracterizat de tripletul fabricant, model, an de fabricație al maşinilor şi are un unic preț)

D_3': $S \leftrightarrow C$ (un şofer are un unic carnet de conducere, care este numai al său)

D_4': $S \text{—>>} ID$ (un şofer poate săvârşi o mulțime de încălcări ale legii circulației, la diverse date calendaristice).

Desigur că nici această formulare relațională nu modelează cu acuratețe realitatea avută în vedere: ea nu surprinde nici echivalența ce calculează elementele mulțimii identificate de *T* (reprezentate doar de D_1', care abstractizează surjecția canonică asociată, dar fără păstrarea informației despre această natură a semanticii sale, lucru imposibil de exprimat în MRD) şi nici faptul că mulțimile ale căror elemente sunt identificate de *P*, respectiv *S* trebuie să fie incluse ambele într-o aceeaşi mulțime (de oameni).

Soluția de proiectare oferită de autori [ZM] este descompunerea *VFMA*, *NF*, *NA*, *NM*, *PC*, *PID*, *NP*: ea este în FN4, păstrează toate dependențele şi are proprietatea jfp. Întâmplător, fie că analizăm această soluție, fie pe cea clasică echivalentă (i.e. *VFMA*, *NFMA*, *NPC*, *PID*), fie soluția reformulării date de noi mai sus (i.e. *NTP*, *TFMAV*, *SC*, *SID*) bd corespunzătoare nu pot memora conținuturi aberante. Dar acest lucru nu

constituie un merit al abordării relaționale ci se datorează doar trivialității problemei propuse!

6.5 Principalele limite ale modelării relaționale a datelor

Abordarea relațională a modelării datelor îşi leagănă utilizatorii în iluzia că, eventual ajutați de calculator, ar putea proiecta orice bd având doar o viziune simplistă şi superficială asupra ei. Sperăm să fi reuşit prin cele câteva contraexemple de modelare discutate în acest articol să demonstrăm că „soluțiile" astfel obținute nu pot fi decât la fel de superficiale şi simpliste.

Convingerea noastră este că, pentru a putea reprezenta satisfăcător realitatea într-o bd, modelele datelor trebuie să încorporeze mult mai multă matematică decât MRD. Pentru utilizările curente (gestiune economică, administrativă, tehnologică etc.) este suficientă teoria elementară a mulțimilor, relațiilor şi funcțiilor [Ma2]. (Însă, de îndată ce vom fi confruntați cu proiectarea de bd ştiințifice sau de reprezentarea cunoştințe-lor, deja apanaj al inteligenței artificiale, va trebui să apelăm şi la teoriile matematice modelând domeniile respective [Ma1].)

Considerăm că trecerea la utilizarea de astfel de modele ale datelor este posibilă şi, mai mult, constituie unica alternativă demnă de avut în vedere: pe lângă faptul (trivial!) că matematica este un limbaj universal, iar teoria elementară a mulțimilor, relațiilor şi funcțiilor este predată în şcoala generală azi obligatorie aproape pretutindeni, nici nu întrezărim vreo abordare în acelaşi timp corectă şi fructuoasă a modelării datelor care să-şi permită să ignore sau să nu folosească adecvat matematica cunoscută!

Se consideră (pe bună dreptate!) că, pentru a fi satisfăcătoare, soluția proiectării trebuie să „păstreze dependențele" (pd) [Ull]. De asemenea, schema obținută trebuie să satisfacă şi proprietatea jfp [Ull]. *Deşi teoria proiectării bdr demonstrează* [Ull] *că:*

- *nu întotdeauna există soluții care să satisfacă ambele aceste cerințe* (i.e. pd şi jfp)
- chiar în cazul existenței soluțiilor satisfăcând aceste cerințe, nu toate au vreun corespondent „intuitiv" („înțeles semantic"); şi că
- *proprietatea jfp nu previne, în general, memorarea de tupli aberanți în conținutul bd* (lucru considerat, e drept, ca un „avantaj al descompunerii" [Ull] sic!) şi
- *deşi MRD impune limitări severe de modelare* (ex.: nu sunt permise mai multe dependențe de acelaşi tip între aceleaşi două atribute!), *literatura evită în general recunoaşterea inadecvării acestei abordări a modelării datelor.* Dimpotrivă, prin proliferarea de noi şi noi tipuri de dependențe (fiecare reuşind să formalizeze re-zolvarea încă unui exemplu particular de modelare!), de FN asociate acestora şi de algoritmi de aducere a schemelor în aceste FN, se acreditează din ce în ce mai mult ideea că modelarea datelor se poate face în termeni relaționali (şi încă asistată de calculator!).

Iată şi câteva alte reflexii pe marginea acestor patru exemple celebre de modelare şi proiectare a bdr întâlnite în literatură: fiecare în parte ilustrează toate cele trei teze deja formulate în rezumatul articolului, însă operăm o distincție pur didactică între ele considerând că (E.1.), deşi extrem de simplu, nu are soluție relațională; (E.2.) şi (E.3) sugerează că problemele de modelat trebuiesc analizate mult mai în profunzime decât se obişnuieşte, de obicei, în abordarea relațională; în fine, (E.4) se vădeşte a fi un adevărat contraexemplu de modelare a datelor.

Nici unul din aceste exemple nu are soluții adecvate în MRD; pentru toate este necesară considerarea unei mulțimi de constrângeri mult mai largi decât dependențele (dar şi a unui cu totul alt tip de model al datelor!): avem în vedere, în special, comutativități de diagrame precum şi egalități (incluziuni) de imagini şi preimagini de funcții şi

relații [Ma1]. O analiză mai detaliată a acestor exemple (ca şi a altora), însoțită de soluționarea lor într-un model al datelor bazat pe teoria elementară a mulțimilor, relațiilor şi funcțiilor poate fi găsită în [Ma2].

Concluzia ce am dori să se impună de la sine în urma prezentării acestor contra-exemple este următoarea: modelarea datelor trebuie făcută la un nivel ierarhic superior MRD, în termenii unui model „infologic" („semantic"). Doar implementarea „datalogică" a acestuia va face uz (şi) de modelul relațional (căci cea mai naturală reprezentare a grafurilor de relații / funcții ce nu pot fi specificate analitic o constituie, într-adevăr, tabelele).

Nivelul „datalogic" al oricărei bd trebuie să rămână transparent atât proiectantului cât, mai ales, utilizatorului acesteia: nu numai pentru a-l scuti de „navigări" laborioase (şi necesitând cunoştințe de specialitate), ci, în special, pentru a-i interzice accesul direct la tabele [Ma1], [Ma2].

Modelul relațional a oscilat multă vreme între două strategii (în esență echivalente) de modelare a datelor, ambele pur sintactice:

- strategia *ascendentă* a "*sintetizării*" schemelor de relație dintr-o mulțime de atribute, respectiv
- strategia *descendentă* a "*descompunerii*" unei unice relații "*universale*", alcătuite din mulțimea tuturor atributelor, în mai multe scheme de relație.

Cel de-al doilea tip de strategie s-a impus în timp tot din considerente pur sintactice, legate în esență de descoperirea unor algoritmi de descompunere cu performanțe superioare celor de sinteză.

De remarcat că atributele sunt privite în abordarea relațională drept nume pentru mulțimi de valori ("*domeniile relațiilor*") şi că scopul comun al sintetizării şi descompunerii îl constituie obținerea "automatizabilă" a unei colecții de scheme relaționale într-o anumită formă normală. La rândul lor, formele normale sunt introduse ca garanți ai satisfacerii "automate" a unor mulțimi de constrângeri de anumite tipuri ("*dependențe*").

Modelarea relațională a datelor are meritul incontestabil că atrage pentru prima oară atenția asupra faptului că proiectarea "ad-hoc" a schemelor de fişiere conduce în general la apariția anomaliilor de actualizare. "Netratate", acestea induc pierderi şi/sau alterări de informații; tratamentul lor ulterior implică costisitoare tranzacții procedurale, ce înseamnă atât timp şi efort de programare, cât şi timp suplimentar de execuție a programelor aferente. În consecință, abordarea relațională propune prevenirea acestor anomalii printr-o proiectare riguroasă a schemelor de fişiere, care să ia în considerare şi constrângerile de integritate impuse de "realitatea" domeniilor modelate.

În plus, bazându-se pe o uniformizare simplificatoare ("*aplatizarea*" realizată de prima normalizare), pe formalizarea relațională a fişierelor şi pe dependențele abstractizând anumite tipuri de constrângeri de integritate, modelul relațional îşi propune inițial să ofere o proiectare algoritmică, efectiv şi eficient calculabilă, a schemelor de relații în diferitele *forme normale* aferente dependențelor.[46]

Această abordare a modelării datelor are însă limitele ei, atât din punct de vedere sintactic, cât şi semantic. Enumerăm în continuare doar pe cele de esență:

- întâi şi întâi, pe cât este de adevărat faptul că relațiile (cu tuplii şi atributele lor) constituie o abstractizare excelentă a fişierelor de date, pe atât de puțin evident este acela că fişierele ar constitui o abstractizare mulțumitoare a semanticii oricărui domeniu de interes al "realității", chiar însoțite fiind de constrângeri de integritate; *simplitatea relației ca unică "unealtă" de modelare constituie,*

[46] Deşi nu este circumscrisă "staticii" modelării, ci "dinamicii" tranzacționale asupra modelelor rezultate, se cuvine desigur a menționa şi aici contribuția suplimentară esențială a modelului relațional în formalizarea duală a interogărilor, atât algebrică cât şi logică de ordin întâi. Aceasta a permis nu doar obținerea de algoritmi optimizați de calcul al răspunsurilor pentru clase importante de întrebări, ci şi semnalarea eventualelor ambiguități în formularea lor.

desigur, un uriaş avantaj sintactic, dar induce o inevitabilă sărăcie semantică, insuficient atenuabilă prin adăugarea dependențelor;

- apoi, *s-au putut defini forme normale doar pentru câteva tipuri particulare de constrângeri*; în plus, *nici una dintre aceste forme normale nu este pe deplin mulțumitoare* (în sensul că nu asigură nici măcar garantarea satisfacerii tuturor tipurilor de constrângeri formalizate ca dependențe);
- mai mult, *nu se poate în general obține o schemă de bd în orice formă normală dorită*, pornind de la orice mulțime de dependențe, *iar în destule dintre cazurile particulare în care acest lucru este totuşi posibil, algoritmii de calcul aferenți sunt exponențiali sau polinomiali* (dar tot foarte lenți pentru aplicații "reale", chiar şi în al doilea caz, deoarece polinoamele au grad mare); mai mult, *majoritatea algoritmilor sunt nedeterminişti*, revenind tot proiectantului aplicației să aleagă dintre mai multele soluții calculate pe aceea care i se pare că ar avea cel mai mult "sens";
- în sfârşit, lucru deja semnalat în capitolul 3, considerăm că este extrem de grăitoare împrejurarea că *cea mai înaltă formă normală a MRD este definită doar în termeni de chei şi constrângeri de domeniu şi este caracterizabilă strict în termeni de anomalii şi nu în cei ai vreunui tip de dependență (fie el FD, DMV, DJ etc.)!*

7. Limbaje relaționale

Secțiunea 4 descrie rolul important jucat de limbajele de interogare în teoria şi practica bazelor de date relaționale. Această secțiune este dedicată unei analize aprofundate a principalelor tipuri de asemenea limbaje. Subsecțiunea 7.1 conține câteva definiții generale. Secțiunea 7.2 introduce restul operatorilor care, împreună cu cei trei deja definiți în Secțiunea 4, completează prezentarea *algebrei relaționale* (*AR*), principalul exponent al *clasei limbajelor procedurale* (i.e. *ale căror expresii descriu, pas cu pas, calculele necesare pentru obținerea răspunsului la o interogare*). Subsecțiunea 7.2.1 descrie puterea expresivă a algebrei relaționale, caracterizându-i capacitatea de extragere a informațiilor dintr-un conținut oarecare al unei baze de date. În Secțiunea 7.3 este introdusă *clasa limbajelor declarative: bazate pe calculul predicativ de ordin întâi, ele permit formularea interogărilor prin specificarea proprietăților pe care trebuie să le aibă rezultatele acestor.* Astfel, subsecțiunea 7.1 prezintă *calculul relațional al domeniilor* (*CRD*), al cărui strălucit reprezentant este limbajul *QBE*; subssecțiunea 7.1.1 atrage atenția asupra unei slăbiciuni a unui asemenea limbaj şi propune o versiune îmbunătățită ce o elimină. În subsecțiunea 7.1.2 se studiază puterea expresivă a limbajelor bazate pe calculul predicatelor; în esență, ea se dovedeşte a fi aceeaşi cu cea a *AR*, ceea ce conduce la definirea noțiunii de *completitudine* a limbajelor. În subsecțiunea 7.2 este prezentată cea de-a doua subclasă, ce stă la baza limbajelor de tip *Quel* şi *SQL*[47]: *calculul relațional al tuplilor* (*CRT*).

Secțiunea 7.4 discută limitările puterii expresive a limbajelor de interogare relaționale, prezentând exemple de tipuri de interogări importante ce nu pot fi specificate cu ajutorul acestor limbaje, în timp ce 7.5 prezintă implicațiile considerării valorilor nule în interogări.

7.1 Noțiuni preliminare

Pe parcursul acestui capitol presupunem o mulțime numărabilă infinită de atribute U_∞ ale cărei elemente au acelaşi domeniu D, ce poate fi finit sau infinit. Presupunem, pentru simplitate, că se pot introduce noi atribute oricând este nevoie (desigur că toate relațiile, fie ele operanzi sau rezultat, au scheme finite; este însă convenabil să nu existe nici o limitare a numărului lor de atribute). A doua presupunere simplificatoare poate părea limitativă, căci în realitate atributele sunt definite pe mai multe domenii distincte; pentru rezultatele pe care intenționăm să le discutăm în continuare însă, această distincție nu ar schimba nimic esențial, ci doar ar complica notațiile şi prezentarea.

Notăm cu $\mathcal{U} \subset \mathcal{P}(U_\infty)$ mulțimea tuturor relațiilor peste orice submulțime finită a U_∞. Dată fiind o schemă $\boldsymbol{R} \in \mathcal{U}$, notăm cu $I(\boldsymbol{R})$ mulțimea tuturor conținuturilor (instanțelor) sale posibile.

Definiția 142 Se zice *interogare* orice funcție parțial recursivă $Q : I(\boldsymbol{R}) \to \mathcal{U}$; dacă $\boldsymbol{r} \in I(\boldsymbol{R})$, $Q(\boldsymbol{r})$ se zice *rezultatul* lui Q pentru $\boldsymbol{r}$.

Observația 143 În majoritatea cazurilor, pentru orice interogare Q există o mulțime finită de atribute X astfel încât, dacă este definit, rezultatul lui Q este întotdeauna o relație peste X; notând cu $\mathcal{X}$ mulțimea tuturor relațiilor peste X, Q devine astfel o funcție $Q : I(\boldsymbol{R}) \to \mathcal{X}$.

Interogările sunt formulate în diverse limbaje cu ajutorul *expresiilor*; fiecare limbaj are, desigur, propria sa sintaxă (caracterizând expresiile corecte) şi semantică (precizând valoarea $E(\boldsymbol{r})$, definită sau nu, a fiecărei expresii corecte E pentru orice conținut al bd $\boldsymbol{r}$). Ca atare, expresiile definesc funcții şi se zice că o expresie E *reprezintă* o interogare Q

[47] De fapt, aceste limbaje moştenesc şi caracteristici algebrice.

dacă funcția definită de E este exact Q. De notat că, deşi interogările sunt definite pentru o anumită schemă fixată, este adesea posibilă definirea expresiilor peste mai multe (oricâte!) scheme: de exemplu, o expresie algebrică relațională se poate referi la orice schemă ce include toate schemele de relație menționate în expresie, fiecare din acestea trebuind doar să includă toate atributele menționate.

Definiția 144 Două *expresii* E_1 şi E_2 (nu neapărat ale unui acelaşi limbaj!) se zic *echivalente în raport cu o schemă de baze de date* $\boldsymbol{R}$ (notație: $E_1 \equiv_{\boldsymbol{R}} E_2$), dacă ele reprezintă aceeaşi interogare, adică dacă $E_1(\boldsymbol{r}) = E_2(\boldsymbol{r})$, $\forall \boldsymbol{r} \in I(\boldsymbol{R})$.

Observația 145 În majoritatea cazurilor, definiția de mai sus poate fi relaxată prin renunțarea la raportare la o anumită schemă: E_1 şi E_2 se zic simplu *echivalente* (notație: $E_1 \equiv E_2$), dacă ele sunt definite pentru aceeaşi mulțime de scheme şi echivalente în raport cu fiecare dintre acestea.

Definiția 146 Date fiind două *limbaje de interogare* L_1 şi L_2, se zice că L_1 este *cel puțin la fel de expresiv* ca L_2 dacă pentru orice expresie $E_1 \in L_1$ există cel puțin o expresie echivalentă $E_2 \in L_2$. L_1 şi L_2 se zic *la fel de expresive* sau *echivalente* dacă fiecare din ele este cel puțin la fel de expresiv ca celălalt.

7.2 Algebra relațională

Un limbaj se zice *procedural* dacă expresiile sale descriu pas cu pas modul de calcul al rezultatului (în cazul limbajelor de interogare, pornind de la conținutul bazei de date). Un exemplu de asemenea tip de limbaj îl constituie *algebra relațională* (*AR*) introdusă în secțiunea 4. Expresiile ei sunt compuse din operatori asupra relațiilor, iar rezultatul se construieşte aplicând operatorii în ordinea fixată de compunerea lor.

Mulțimea operatorilor relaționali include variante ale operatorilor matematici clasici asupra mulțimilor (reuniune, intersecție, diferență), precum şi operatori specifici asupra relațiilor de bd (proiecție, selecție, redenumire, join).

Relațiile sunt mulțimi de tupli şi deci li se pot aplica operatorii teoriei mulțimilor. Totuşi, pentru a construi expresii complexe ale căror operatori sunt definiți pe relații, trebuie să ne asigurăm că rezultatul fiecărei operații este de asemenea o relație, adică o mulțime de tupli definiți pe un acelaşi set de atribute fixat (şi nu pe diverse seturi de atribute diferite de la un tuplu la altul). De aceea, operatorii relaționali de *reuniune*, *intersecție* şi *diferență* sunt definiți pe baza celor din teoria mulțimilor dar cu restricția că operanzii lor sunt definiți peste un acelaşi set de atribute.[48]

Pe lângă *selecție*, *proiecție* şi *join* (definite în secțiunea 4), operatorii relaționali specifici includ *redenumirea*, plus alți câțiva operatori derivați din aceştia. Redenumirea este un operator unar, ce îşi datorează numele faptului că, intuitiv, schimbă numele atributelor lăsând neschimbat corpul operandului.

Definiția 147 Fie o relație r peste atributele X şi o funcție injectivă f definită pe şi cu valori în X, care atribuie un nou nume fiecărui atribut; se zice *redenumirea lui r în raport cu* f relația: $\rho_f(r) = \{t \mid \exists t' \in r \text{ a.î. } t'[A] = t[f(A)], \forall A \in X\}$.

Pentru a simplifica notația, funcția f este scrisă de obicei sub forma:
$f(A_1), \ldots, f(A_k) \leftarrow A_1, \ldots, A_k$, unde $A_1, \ldots, A_k$ este lista atributelor din X pentru care f nu este identitatea (adică doar atributele ale căror nume sunt modificate).

Operatorul de redenumire compensează rigiditatea operatorilor relaționali depinzând de numele atributelor (precum joinul natural şi operatorii proveniți din teoria mulțimilor): după o redenumire corespunzătoare, reuniunea, intersecția şi diferența pot fi aplicate asupra oricăror perechi de relații având aceeaşi aritate; similar, joinurile pot fi

[48] Unele abordări propun o restricție mai slabă, cerând operanzilor să aibă doar aceeaşi aritate; atributele rezultatului sunt în acest caz ori nedefinite, ori coincid cu cele ale primului operand. Abordarea de față garantează însă uniformitatea şi simetria fără a limita puterea expresivă (lucru datorat introducerii operatorului de redenumire).

aplicate asupra unei submulțimi sau supermulțimi a atributelor comune celor două relații operand.

Exemplul 49 Fie o bd referitoare la arborele genealogic al unei familii de domnitori, cu următoarea schemă:

DOMNII(*Domnitor*, *DeLa*, *La*), *cheie*(*La*) *TATA*(*Tata*, *Copil*)
PERSOANE(*Nume*, *Moarte*, *Necropolă*) *MAMA*(*Mama*, *Copil*)

Figura 54 prezintă un posibil conținut al acestei bd pentru familia Basarabilor. Se observă că prima tabelă are două chei şi cheia străină *Domnitor*, iar în ultimele două tabele fiecare atribut este cheie străină.

O expresie reunind ultimele două relații (pentru a obține astfel părinții copiilor) trebuie să facă uz de redenumiri preliminare ca în exemplul următor: (E1) $\rho_{Părinte \leftarrow Tata}(TATA) \cup \rho_{Părinte \leftarrow Mama}(MAMA)$

Următoarele două exemple ilustrează redenumiri necesare în cazul joinurilor naturale: primul calculează perechea părinților fiecărui domnitor, în timp ce al doilea selectează doar domnitorii ce au domnit până la moarte (şi ultima lor domnie):

(E2) $\pi_{Domnitor,Tata,Mama}(DOMNII \bowtie \rho_{Domnitor \leftarrow Copil}(TATA) \bowtie \rho_{Domnitor \leftarrow Copil}(MAMA))$

(E3) $\pi_{Nume,DeLa,Moarte}(\rho_{Nume,\ Moarte \leftarrow Domnitor,\ La}(DOMNII) \bowtie PERSOANE)$

Rezultatele aplicării acestor expresii asupra bd din figura 54 sunt prezentate în figura 55.

Cel mai important operator derivat inclus în mod uzual în algebra relațională este *theta-joinul* (cu specializarea sa, *echi-joinul*), care poate fi exprimat cu ajutorul selecției şi joinului natural. Importanța sa rezidă în capacitatea de a formula mai elegant (sintetic şi expresiv) anumite interogări.

Definiția 148 Fie două relații $r_1(X_1)$ şi $r_2(X_2)$ având mulțimi de atribute disjuncte; se zice *theta-join* şi se notează cu $r_1 \bowtie_{A1\theta A2} r_2$, unde $A_1 \in X_1$ şi $A_2 \in X_2$, iar θ este un operator de comparație, o relație peste mulțimea de atribute X_1X_2 al cărei conținut este alcătuit din tuplii t formați prin combinarea tuturor tuplilor $t_1 \in r_1$ şi $t_2 \in r_2$ cu proprietatea: $t_1[A_1]\ \theta\ t_2[A_2]$.

Evident că rezultatul theta-joinului de mai sus poate fi obținut printr-un join natural (care în acest caz este de fapt un produs cartezian!) urmat de o selecție: $\sigma_{A1\theta A2}(r_1 \bowtie r_2)$.

Definiția 149 Un theta-join pentru care θ = '=' se zice *echi-join*.

Operatorul theta-join definit mai sus admite următoarea generalizare (inspirată din definiția operatorului de selecție) şi anume: $r_1 \bowtie_F r_2$ este echivalent cu $\sigma_F(r_1 \bowtie r_2)$.

Exemplul 50 Figura 56 prezintă rezultatul următorului theta-join asupra bd din figura 54, care calculează copiii domnitorilor Basarabi decedați înaintea taților lor sau în acelaşi an cu aceştia:

(E4) $\pi_{\text{Domnitor,MoarteDomn,Copil,Moarte}}((DOMNII \bowtie \rho_{\text{Domnitor,MoarteDomn} \leftarrow \text{Nume,Moarte}}(PERSOANE)) \bowtie_{(\text{Moarte} <= \text{MoarteDomn})} (\rho_{\text{NumeDomn} \leftarrow \text{Tata}}(TATA) \bowtie \rho_{\text{Copil} \leftarrow \text{Nume}}(PERSOANE)))$.

Evident că această expresie este echivalentă cu următoarea:

(E5) $\sigma_{(Moarte <= MoarteDomn)}(\pi_{NumeDomn,MoarteDomn,Copil,Moarte}((DOMNII \bowtie \rho_{Domnitor,MoarteDomn \leftarrow Nume,Moarte}(PERSOANE)) \bowtie (\rho_{NumeDomn \leftarrow Tata}(TATA) \bowtie \rho_{Copil \leftarrow Nume}(PERSOANE))))$.

PERSOANE(Nume)

Nume	*Moarte*	*Necropolă*
Alexandru Aldea	1436	
Ana		
Anca		
Basarab I	1352	Câmpulung
Basarab Țepeluș		
Clara		
Dan I	1386	
Dan II	1431	
Elena		
Elisabeta		
Laiotă Basarab	1477	
Maria		
Maria Voichița	1511	
Mătușă Ștefan cel Mare		
Mihail I	1420	
Mihnea cel Rău	1509	
Mircea	1447	
Mircea cel Bătrân	1418	Cozia
Mircea Ciobanul	1559	
Neagoe Basarab	1521	Curtea Argeş
Nicolae Alexandru	1364	Câmpulung
Radu cel Frumos		
Radu cel Mare	1508	Dealu
Radu de la Afumați	1529	Curtea Argeş
Radu I	1384	
Radu II Pleşuvul	1427	
Radu Paisie		
Ruxandra		
Stanca	1530	Curtea Argeş
Teodosie		
Vlad		
Vlad Călugărul	1495	
Vlad cel Tânăr	1512	
Vlad Dracul	1447	
Vlad Țepeş	1477	Snagov
Vlad Uzurpatorul	1396	
Vladislav II	1456	
Vladislav Vlaicu	1377	Curtea Argeş

MAMA(Copil) Mama ⊆ Nume, Copil ⊆ Nume, Φ⊢Mama

Mama	*Copil*
Clara	Ana
Clara	Anca
Elena	Vlad
Maria	Vladislav Vlaicu
Maria	Radu I
Maria	Elisabeta
Mătușă Ștefan cel Mare	Vlad Țepeş
Mătușă Ștefan cel Mare	Radu cel Frumos
Mătușă Ștefan cel Mare	Mircea
Mătușă Ștefan cel Mare	Vlad Călugărul

- Dan II, Radu II Pleşuvul, Laiotă Basarab, Vlad Călugărul au domnit, de fapt, de mai multe ori

- au mai domnit, însă extrem de scurt timp fiecare, Mircea şi Basarab II.

- Elena, soția lui Vlad Țepeş, a fost fiica lui Iancu de Hunedoara şi sora lui Matei Corvin

- Maria Voichița a fost soția lui Ştefan cel Mare; Stanca a fost soția lui Ştefăniță, domn al Moldovei, fiu al lui Bogdan III

DOMNII(DeLa, La)Domnitor ⊆ Nume, Φ⊢Domnitor, Φ⊢La

Domnitor	*DeLa*	*La*
Basarab I	1310	1352
Nicolae Alexandru	1352	1364
Vladislav Vlaicu	1364	1377
Radu I	1377	1384
Dan I	1384	1386
Mircea cel Bătrân	1386	1394
Vlad Uzurpatorul	1394	1396
Mircea cel Bătrân	1396	1418
Mihail I	1418	1420
Dan II	1420	1424
Radu II Pleşuvul	1424	1426
Dan II	1426	1431
Alexandru Aldea	1431	1436
Vlad Dracul	1436	1442
Vladislav II	1442	1443
Vlad Dracul	1443	1447
Vladislav II	1447	1456
Vlad Țepeş	1456	1462
Radu cel Frumos	1462	1473
Laiotă Basarab	1473	1476
Vlad Țepeş	1476	1477
Basarab Țepeluș	1477	1481
Vlad Călugărul	1481	1495
Radu cel Mare	1495	1508

TATA(Copil) Tata ⊆ Nume, Copil ⊆ Nume, Φ⊢Tata

Tata	*Copil*
Basarab I	Nicolae
Basarab Țepeluș	Neagoe Basarab
Dan I	Dan II
Dan I	Vlad
Dan II	Vladislav II
Dan II	Laiotă Basarab
Laiotă Basarab	Basarab Țepeluș
Mircea cel Bătrân	Mihail I
Mircea cel Bătrân	Radu II Pleşuvul
Mircea cel Bătrân	Alexandru Aldea
Mircea cel Bătrân	Vlad Dracul
Neagoe Basarab	Teodosie
Neagoe Basarab	Stanca
Neagoe Basarab	Ruxandra
Nicolae Alexandru	Vladislav Vlaicu
Nicolae Alexandru	Radu I
Nicolae Alexandru	Elisabeta
Nicolae Alexandru	Ana
Nicolae Alexandru	Anca
Radu cel Frumos	Maria Voichița
Radu cel Mare	Radu de la
Radu cel Mare	Radu Paisie
Radu cel Mare	Mircea Ciobanul
Radu I	Dan I
Radu I	Mircea cel
Vlad Călugărul	Radu cel Mare
Vlad Călugărul	Vlad cel Tânăr
Vlad Dracul	Vlad Țepeş
Vlad Dracul	Radu cel Frumos
Vlad Dracul	Mircea
Vlad Dracul	Vlad Călugărul
Vlad Țepeş	Mihnea cel Rău
Vlad Țepeş	Vlad

Figura 54 Bd "Domnitori" şi un conținut al ei pentru începutul dinastiei Basarabilor

(E1)

Părinte	*Copil*
Mircea cel Bătrân	Alexandru Aldea
Nicolae Alexandru	Ana
Clara	Ana
Nicolae Alexandru	Anca
Clara	Anca
Laiotă Basarab	Basarab Țepeluș
Radu I	Dan I
Dan I	Dan II
Nicolae Alexandru	Elisabeta
Maria	Elisabeta
Dan II	Laiotă Basarab
Radu cel Frumos	Maria Voichița
Mircea cel Bătrân	Mihail I
Vlad Țepeș	Mihnea cel Rău
Vlad Dracul	Mircea
Mătușă Ștefan cel Mare	Mircea
Radu I	Mircea cel Bătrân
Radu cel Mare	Mircea Ciobanul
Basarab Țepeluș	Neagoe Basarab
Basarab I	Nicolae Alexandru
Vlad Dracul	Radu cel Frumos
Mătușă Ștefan cel Mare	Radu cel Frumos
Vlad Călugărul	Radu cel Mare
Radu cel Mare	Radu de la Afumați
Nicolae Alexandru	Radu I
Maria	Radu I
Mircea cel Bătrân	Radu II Pleșuvul
Radu cel Mare	Radu Paisie
Neagoe Basarab	Ruxandra
Neagoe Basarab	Stanca
Neagoe Basarab	Teodosie
Vlad Țepeș	Vlad
Elena	Vlad
Vlad Dracul	Vlad Călugărul
Mătușă Ștefan cel Mare	Vlad Călugărul
Vlad Călugărul	Vlad cel Tânăr
Mircea cel Bătrân	Vlad Dracul
Vlad Dracul	Vlad Țepeș
Mătușă Ștefan cel Mare	Vlad Țepeș
Dan I	Vlad Uzurpatorul
Dan II	Vladislav II
Nicolae Alexandru	Vladislav Vlaicu
Maria	Vladislav Vlaicu

(E3)

Nume	*DeLa*	*Moarte*
Basarab I	1310	1352
Nicolae Alexandru	1352	1364
Vladislav Vlaicu	1364	1377
Radu I	1377	1384
Dan I	1384	1386
Vlad Uzurpatorul	1394	1396
Mircea cel Bătrân	1396	1418
Mihail I	1418	1420
Dan II	1426	1431
Alexandru Aldea	1431	1436
Vlad Dracul	1443	1447
Vladislav II	1447	1456
Vlad Țepeș	1476	1477
Vlad Călugărul	1481	1495
Radu cel Mare	1495	1508

(E2)

Domnitor	*Tata*	*Mama*
Vladislav Vlaicu	Nicolae Alexandru	Maria
Radu I	Nicolae Alexandru	Maria
Vlad Țepeș	Vlad Dracul	Mătușă Ștefan cel Mare
Radu cel Frumos	Vlad Dracul	Mătușă Ștefan cel Mare
Vlad Călugărul	Vlad Dracul	Mătușă Ștefan cel Mare

Figura 55 Rezultatele expresiilor (E1), (E2) și (E3) asupra bd din figura 54

Domnitor	*MoarteDomn*	*Copil*	*Moarte*
Vlad Dracul	1447	Mircea	1447

Figura 56 Rezultatul theta-joinului (E4) asupra bd din figura 54

Interogările sunt formulate în algebra relațională cu ajutorul expresiilor construite cu operatori algebrici ai căror operanzi sunt scheme de relații şi/sau relații constante.

Exemplul 51 Referindu-ne din nou la relațiile din exemplul 142, următoarea expresie calculează copiii domnitorilor (vezi şi rezultatul ei pentru bd din figura 54 în figura 58):

(E6) $\pi_{Domnitor,\ Copil}\ (DOMNII \bowtie \rho_{Domnitor \leftarrow Tata}(TATA))$.

Considerând relația constantă din figura 57, următoarea expresie calculează relația cu două atribute din figura 58, în care se asociază fiecărei capitale domnitorul care a stabilit capitala în oraşul respectiv:

(E7) $\pi_{Domnitor,\ Capitala}\ (CAPITALE \bowtie_{DeLa \le An \wedge An \le La} DOMNII)$

În sfârşit, următoarea expresie calculează domnitorii care au abdicat sau au fost detronați:

(E8) $\pi_{Domnitor,La,Moarte}(\sigma_{Moarte>La}(DOMNII \bowtie \rho_{Nume \leftarrow Domnitor}(PERSOANE)))$.

Figura 58 prezintă rezultatul acestor expresii pentru bd din figura 54. Figura 59 prezintă o bd similară celei din figura 4, dar pentru dinastia Muşatinilor, figura 60 prezintă capitalele Moldovei, iar figura 61 rezultatele aceloraşi expresii aplicate însă acestor noi date (din figura 59).

7.2.1 Puterea expresivă a algebrei relaționale

Este rezonabil să ne întrebăm deja dacă algebra relațională este suficient de puternică pentru a exprima toate interogările cu putință. Vom încerca să răspundem la această întrebare spre sfârşitul capitolului; în această secțiune caracterizăm relațiile ce pot fi obținute drept rezultat al expresiilor algebrice relaționale aplicate asupra conținutului unei bd oarecare fixate. Se va dovedi că se pot astfel genera toate relațiile "cu sens", ceea ce va constitui o confirmare a importanței algebrei relaționale privite ca limbaj de interogare.

Se impune de la început următoarea precizare (care va fi aprofundată în subsecțiunea 7.2): în contextul curent, caracterizăm ceea ce se poate extrage din conținutul unei bd şi *nu* interogările ce pot fi exprimate în algebra relațională.

Pentru a simplifica discuția, presupunem că nu este definită nici o ordine peste domeniul comun D al tuturor atributelor; ca atare, selecțiile implică doar operatorii de egalitate şi neegalitate.

O primă caracteristică a operatorilor algebrei relaționale este aceea că ei nu pot <u>*crea*</u>, ci doar <u>*selecta*</u> valori: o valoare apare în rezultatul unei expresii numai dacă ea apare măcar într-unul dintre operanzi.

Definiția 150 Fie un conținut fixat $\boldsymbol{r} = \{r_1, \ldots, r_n\}$, $n \ge 1$, al unei bd oarecare; $D_{\boldsymbol{r}} \subseteq D$ se zice *domeniul activ* al $\boldsymbol{r}$ dacă el conține toate şi numai valorile ce apar în $\boldsymbol{r}$ ca valori ale unor tupli dintr-o relație.

Rezultatul unei expresii asupra conținutului $\boldsymbol{r}$ conține doar valori din domeniul activ şi eventual valori din relațiile constante. Pentru simplitate, în tot restul acestei secțiuni considerăm doar expresii ce nu implică relații constante. Operatorul de redenumire poate introduce noi nume de atribute; singura restricție este aceea ca numele acestora să fie distincte în cadrul fiecărei relații. Recapitulând, relațiile ce pot fi generate cu ajutorul expresiilor relaționale implicând doar relații peste un conținut de bd $\boldsymbol{r}$ pot avea orice schemă, dar pot conține doar valori din domeniul activ $D_{\boldsymbol{r}}$.

CAPITALE (*An*) Φ├*Capitala*

Capitala	*An*
Câmpulung Muscel	1310
Curtea de Argeş	1369
Târgovişte	1419
Bucureşti	1558

Figura 57 Capitalele Țării Româneşti şi anii aproximativi de stabilire a lor

(E6)

Domnitor	*Copil*
Mircea cel Bătrân	Alexandru Aldea
Nicolae Alexandru	Ana
Nicolae Alexandru	Anca
Laiotă Basarab	Basarab Țepeluş
Radu I	Dan I
Dan I	Dan II
Nicolae Alexandru	Elisabeta
Dan II	Laiotă Basarab
Radu cel Frumos	Maria Voichița
Mircea cel Bătrân	Mihail I
Vlad Țepeş	Mihnea cel Rău
Vlad Dracul	Mircea
Radu I	Mircea cel Bătrân
Radu cel Mare	Mircea Ciobanul
Basarab Țepeluş	Neagoe Basarab
Basarab I	Nicolae Alexandru
Vlad Dracul	Radu cel Frumos
Vlad Călugărul	Radu cel Mare
Radu cel Mare	Radu de la Afumați
Nicolae Alexandru	Radu I
Mircea cel Bătrân	Radu II Pleşuvul
Radu cel Mare	Radu Paisie
Vlad Țepeş	Vlad
Vlad Dracul	Vlad Călugărul
Vlad Călugărul	Vlad cel Tânăr
Mircea cel Bătrân	Vlad Dracul
Vlad Dracul	Vlad Țepeş
Dan I	Vlad Uzurpatorul
Dan II	Vladislav II
Nicolae Alexandru	Vladislav Vlaicu

(E7)

Domnitor	*Capitala*
Basarab I	Câmpulung Muscel
Vladislav Vlaicu	Curtea de Argeş
Mihail I	Târgovişte

(E8)

Domnitor	*La*	*Moarte*
Mircea cel Bătrân	1394	1418
Dan II	1420	1424
Radu II Pleşuvul	1424	1426
Vlad Dracul	1442	1447
Vladislav II	1443	1456
Vlad Țepeş	1462	1477
Laiotă Basarab	1476	1477

Figura 58 Rezultatele expresiilor (E6), (E7) şi (E8) asupra bd din figura 54

Se dovedeşte că oricare dintre aceste relații se poate obține folosind cu grijă selecțiile. Să considerăm, de exemplu, relațiile r_0 şi r_1 din figura 62. Este evident că relația r_1 se poate obține din r_0 cu ajutorul următoarei expresii:

(E9) $\sigma_{(F = \text{“SC”}) \wedge ((A = \text{“Ioan”} \wedge S = \text{“Andrea”}) \vee (A = \text{“Călin”} \wedge S = \text{“Monica”}))} (r_0)$.

Principala caracteristică a expresiei (E9) este aceea că ea construieşte rezultatul prin menționarea explicită în predicatul selecției a tuturor valorilor necesare. Vom arăta spre sfârşitul secțiunii că nu există nici un alt mod conceptual diferit de obținere a acestui rezultat. Motivul informal este acela că în relația inițială r_0 nu se pot distinge valorile atributelor între ele cu ajutorul tuplului cărora le aparțin: fiecare din tupli este

PERSOANE(Nume)

Nume	*Moarte*	*Necropolă*
Alexandru	1496	Bistrița
Alexandru cel Bun	1432	Bistrița
Alexăndrel		
Ana Neacșa		Bistrița
Anastasia		Rădăuți
Bogdan	1407	Rădăuți
Bogdan I	1365	Rădăuți
Bogdan II	1451	Rădăuți
Bogdan III	1517	Putna
Bogdănel	1477	Putna
Elena		
Evdochia	1467	
Iliaș	1445	
Ilie	1472	Putna
Iuga Vodă	1400	
Lațcu	1374	Rădăuți
Margareta Losoncz	1410	Baia
Margareta Mușata	1393	Siret
Maria	1518	Putna
Maria de Mangop	1477	Putna
Maria Oltea Basarab		
Maria Rareș din Hârlău		
Maria Voichița	1511	Putna
Mariana		
Mătușă Ștefan cel Mare		
Petru Aron	1470	
Petru II		
Petru Mușat	1391	Rădăuți
Petru Pătrașcu	1480	Putna
Petru Rareș	1546	Probota
Ringala Maria		
Roman I	1394	Rădăuți
Roman II		
Soră Petru Mușat?		
Ștefan cel Mare	1504	Putna
Ștefan I	1399	Rădăuți
Ștefan II	1447	

MAMA(Copil) Mama ⊆ Nume, Copil ⊆ Nume,
Φ ⊢ Mama

Mama	*Copil*
Ana Neacșa	Iliaș
Anastasia	Alexandru cel Bun
Evdochia	Alexandru
Evdochia	Elena
Evdochia	Petru Pătrașcu
Margareta Mușata	Petru Mușat
Margareta Mușata	Roman I
Margareta Mușata	Soră Petru Mușat?
Maria de Mangop	Ilie
Maria de Mangop	Bogdănel
Maria Oltea Basarab	Ștefan cel Mare
Maria Rareș din Hârlău	Petru Rareș
Maria Voichița	Bogdan III
Maria Voichița	Maria
Mariana	Petru Aron
Soră Petru Mușat?	Ștefan I
Soră Petru Mușat?	Iuga Vodă

DOMNII(DeLa, La)Domnitor ⊆ Nume,
Φ ⊢ Domnitor, Φ ⊢ La

Domnitor	*DeLa*	*La*
Bogdan I	1363	1365
Lațcu	1365	1374
Petru Mușat	1374	1391
Roman I	1391	1394
Ștefan I	1394	1399
Iuga Vodă	1399	1400
Alexandru cel Bun	1400	1432
Iliaș	1432	1433
Ștefan II	1433	1435
Iliaș	1435	1442
Ștefan II	1442	1447
Roman II	1447	1448
Petru II	1448	1449
Bogdan II	1449	1451
Petru Aron	1451	1452
Alexăndrel	1452	1455
Petru Aron	1455	1457
Ștefan cel Mare	1457	1504

TATA(Copil) Tata ⊆ Nume, Copil ⊆ Nume,
Φ ⊢ Tata

Tata	*Copil*
Alexandru cel Bun	Iliaș
Alexandru cel Bun	Ștefan II
Alexandru cel Bun	Bogdan II
Alexandru cel Bun	Mătușă Ștefan cel Mare
Alexandru cel Bun	Petru Aron
Alexandru cel Bun	Petru II
Bogdan I	Lațcu
Bogdan I	Margareta Mușata
Bogdan II	Ștefan cel Mare
Iliaș	Roman II
Iliaș	Alexăndrel
Roman I	Alexandru cel Bun
Roman I	Bogdan
Ștefan cel Mare	Alexandru
Ștefan cel Mare	Elena
Ștefan cel Mare	Petru Pătrașcu
Ștefan cel Mare	Ilie
Ștefan cel Mare	Bogdănel
Ștefan cel Mare	Bogdan III
Ștefan cel Mare	Maria
Ștefan cel Mare	Petru Rareș

- Iliaș, Ștefan II, Alexăndrel și Petru Aron au domnit, de fapt, de mai multe ori

- Iliaș și Ștefan II au domnit împreună între 1436-1442, primul în jumătatea nordică, al doilea în cea sudică, Ștefan II fiind considerat vasal al lui Iliaș

- numele surorii lui Petru Mușat care se presupune că era mama domnilor Ștefan I și Iuga Vodă nu ne este cunoscut

- Mătușa lui Ștefan cel Mare (al cărei nume nu ne este cunoscut) a fost mama lui Vlad Țepeș

- Maria Voichița, ultima soție a lui Ștefan cel Mare, a fost fiica domnului muntean Radu cel Frumos

- Elena, fiica lui Ștefan cel Mare, s-a căsătorit în 1483 cu Ivan cel Tânăr, fiul Marelui Cneaz al Moscovei Ivan III

Figura 59 Bd "Domnitori" și un conținut al ei pentru începutul dinastiei Mușatinilor

CAPITALE (*An*) *Φ*├*Capitala*

Capitala	*An*
Siret	1363
Suceava	1375
Iaşi	1650

Figura 60 Capitalele Moldovei şi anii aproximativi de stabilire a lor

(E6)

Domnitor	*Copil*
Ştefan cel Mare	Alexandru
Roman I	Alexandru cel Bun
Iliaş	Alexăndrel
Roman I	Bogdan
Alexandru cel Bun	Bogdan II
Ştefan cel Mare	Bogdan III
Ştefan cel Mare	Bogdănel
Ştefan cel Mare	Elena
Alexandru cel Bun	Iliaş
Ştefan cel Mare	Ilie
Bogdan I	Lațcu
Bogdan I	Margareta Muşata
Ştefan cel Mare	Maria
Alexandru cel Bun	Mătuşă Ştefan cel Mare
Alexandru cel Bun	Petru Aron
Alexandru cel Bun	Petru II
Ştefan cel Mare	Petru Pătraşcu
Ştefan cel Mare	Petru Rareş
Iliaş	Roman II
Bogdan II	Ştefan cel Mare
Alexandru cel Bun	Ştefan II

(E7)

Domnitor	*Capitala*
Bogdan I	Siret
Petru Muşat	Suceava

(E8)

Domnitor	*La*	*Moarte*
Iliaş	1433	1445
Iliaş	1442	1445
Ştefan II	1435	1447
Petru Aron	1452	1470
Petru Aron	1457	1470

Figura 61 Rezultatele expresiilor (E6), (E7) şi (E8) asupra bd din figura 59

r_0

Angajat	*Filială*	*Secretară*
Călin	SC	Monica
Ioan	SC	Monica
Călin	IE	Monica
Ioan	IE	Monica
Călin	SC	Andreea
Ioan	SC	Andreea
Călin	IE	Andreea
Ioan	IE	Andreea

r_1

Angajat	*Filială*	*Secretară*
Călin	SC	Monica
Ioan	SC	Andreea

Figura 62 O relație cu valori ce nu pot fi distinse şi o relație construită valoare cu valoare

legat în acelaşi mod de valorile pentru celelalte atribute. Acest lucru se poate formaliza spunând că valorile fiecărui atribut sunt interschimbabile: dacă schimbăm toate aparițiile lui ‘Ioan’ cu ‘Călin’ şi viceversa se obține exact aceeaşi relație[49].

[49] De notat că această relație satisface *DJ* ⋈(*Angajat, Filială, Secretară*)!

La celălalt capăt al spectrului, există cazul în care toate valorile pot fi distinse cu ajutorul legăturilor dintre ele. De exemplu, în relația din figura 63 (care memorează un fragment din succesiunea la domnie a primilor Basarabi), fiecare valoare poate fi identi-

r_0

Predecesor	*Succesor*
Basarab I Întemeietorul	Nicolae Alexandru
Nicolae Alexandru	Vladislav Vlaicu
Vladislav Vlaicu	Radu I

$$r_1 = \rho_{\text{Fondator} \leftarrow \text{Predecesor}}(\pi_{\text{Predecesor}}(r_0)) - \rho_{\text{Fondator} \leftarrow \text{Succcesor}}(\pi_{\text{Succesor}}(r_0))$$

Fondator
Basarab I Întemeietorul

$$\pi_{\text{AlDoilea}}(r_1 \bowtie \rho_{\text{Fondator, AlDoilea} \leftarrow \text{Predecesor, Succesor}}(r_0))$$

AlDoilea
Nicolae Alexandru

Figura 63 O relație cu valori ce pot fi distinse şi două expresii ce nu referă nici o valoare

$h(r_1)$

Angajat	*Filială*	*Secretară*
Ioan	IE	Monica
Călin	IE	Andreea

Figura 64 Rezultatul aplicării unui automorfism

$r_2 = h(r_2)$

Angajat	*Filială*
Călin	SC
Ioan	SC
Călin	IE
Ioan	IE

Figura 65 O relație invariantă la orice automorfism al relației r_1 din figura 62

ficată cu ajutorul proprietăților sale: fondatorul dinastiei, Basarab I, nu a urmat nici unui predecesor din familia sa; Nicolae Alexandru a fost succesorul direct al lui Basarab I ş.a.m.d.

În mod corespunzător, se pot scrie expresii de algebră relațională ce calculează valori fără a menționa vreuna în mod explicit; două dintre acestea sunt de asemenea prezentate în figura 6

Recapitulând, posibilitatea de a distinge între valorile unui conținut de bd este crucială; ca atare, ea este formalizată în cele ce urmează.

Definiția 151 O funcție parțială $h : D \to D$ se zice *automorfism* al unui conținut $\boldsymbol{r}$ de bd dacă:

- ea este o permutare a domeniului activ $D_{\boldsymbol{r}}$ şi dacă
- extinzându-i definiția la tupli, relații şi conținuturi de bd se obține funcția identitate a conținutului $\boldsymbol{r} : h(\boldsymbol{r}) = \boldsymbol{r} = \mathbf{1}_{\boldsymbol{r}}(\boldsymbol{r})$.

Cu alte cuvinte, un automorfism este o redenumire a valorilor din domeniul activ ce lasă invariant conținutul bd. În legătură cu discuția de mai sus, o valoare este considerată ca fiind posibil de distins într-un conținut de bd dacă toate automorfismele conținutului sunt egale cu identitatea asupra ei. Două valori nu pot fi distinse dacă funcția care:

(1) le schimbă între ele şi

(2) este egală cu identitatea asupra tuturor celorlalte valori

este un automorfism.

Exemplul 52 Singurul automorfism asupra relației r_0 din figura 63 este identitatea, ceea ce înseamnă că, într-adevăr, valorile acesteia se pot distinge unele de celelalte. Dimpotrivă, relația r_0 din figura 62 admite multe automorfisme (şi anume opt), deoarece pentru fiecare atribut cele două valori ale domeniului activ sunt interschimbabile; de exemplu, unul dintre acestea este următorul: h(Ioan) = Călin, h (Călin) = Ioan, h(SC) = IE, h(IE) = SC, h(Monica) = Andreea, h(Andreea) = Monica.

Automorfismele au fost definite cu ajutorul invarianței unui conținut al bd; este interesant de văzut însă şi cum se comportă ele în raport cu alte conținuturi.

Exemplul 53 Relația r_1 din figura 62 este transformată de automorfismul h din exemplul 52 în relația din figura 64. Pe de altă parte, relația r_2 din figura 65 rămâne neschimbată indiferent ce automorfism i s-ar aplica.

Fie $C \subseteq D_r$; pentru a putea formaliza selecțiile implicând constante sunt necesare următoarele două definiții.

Definiția 152 Orice automorfism care este egal pe C cu identitatea se zice *automorfism C-fix.*

Definiția 253 Dacă toate constantele apărând în predicatele selecțiilor unei expresii algebrice relaționale aparţin lui C, atunci se zice că *expresia menționează (cel mult) C*.

Cu ajutorul conceptelor introduse până acum se poate formula teorema principală a acestei secțiuni:

Teorema 154 Fie $\boldsymbol{r}$ un conținut al unei bd şi C o submulțime a domeniului activ D_r al acestuia; o relație r se poate obține ca rezultat al unei expresii algebrice relaționale asupra $\boldsymbol{r}$ menționând cel mult C dacă şi numai dacă r este invariant la orice automorfism C-fix peste $\boldsymbol{r}$.

Înainte de a o demonstra, merită să îi analizăm semnificația. Să considerăm întâi cazul $C = \varnothing$: o relație se poate obține ca rezultat al unei expresii ce nu menționează nici o constantă dacă şi numai dacă ea este invariantă la orice automorfism asupra conținutului bd. Dacă considerăm legăturile între valori ca fiind înțelesul semantic al conținutului bd, atunci automorfismele reprezintă gradul de "nesiguranță" inclus în conținut, adică distincțiile nedetectabile între valorile ce îl alcătuiesc. Relațiile invariante la toate automorfismele păstrează această nesiguranță, în timp ce celelalte conțin informații ce nu sunt memorate de conținutul bd. De aceea, în acest caz particular, teorema poate fi reformulată astfel: *Expresiile algebrice relaționale pot construi toate relațiile conținând doar informații ce sunt memorate de conținutul bd*.

Această idee poate fi extinsă la cazul general: folosind constantele în mod explicit, o expresie poate obține mai multe informații din conținutul bd, distingând între valori care altfel nu ar putea fi distinse între ele. La limită, dacă o expresie menționează toate valorile unei relații (precum definiția relației r_1 din figura 62), atunci ea o poate construi, indiferent de automorfismele asupra conținutului acesteia. În acest caz, notând cu C_r mulțimea tuturor valorilor din r, teorema afirmă că r poate fi obținut ca rezultat al unei expresii menționând C_r dacă şi numai dacă r este invariant la orice automorfism C_r-fix al r; cum orice automorfism C_r-fix este identitatea peste r, rezultă că r este în mod trivial invariant la orice asemenea automorfism.

Demonstrația teoremei 154 necesită însă alte câteva noțiuni şi rezultate preliminare.

Fie n valori $c_1, \ldots, c_n$ din D_r şi pentru $\forall C \subseteq D_r$ fie $h_1, \ldots, h_m$ unde $m \geq 1$, toate automorfismele C-fixe asupra $\boldsymbol{r}$.

Definiția 155 O relație cu n atribute $Bc_1, \ldots, Bc_n$ şi m tupli $t_1, \ldots, t_m$ cu $t_i[Bc_j] = h_i(c_j)$ (adică o relație în care fiecare tuplu descrie valoare cu valoare câte un automorfism) se zice *relația automorfismelor C-fixe* şi se notează cu r_C.

De exemplu, figura 66 prezintă relația automorfismelor {SC,IE}-fixe pentru relația r_0 din figura 62.

Lema 156 $\forall \boldsymbol{r}$ şi $\forall C \subseteq D_{\boldsymbol{r}}$, relaţia automorfismelor C-fixe r_C poate fi obţinută ca rezultat al unei expresii algebrice ai cărei operanzi sunt relaţii din $\boldsymbol{r}$.

$B_{Călin}$	B_{Ioan}	B_{SC}	B_{IE}	$B_{Andreea}$	B_{Monica}
Călin	Ioan	SC	IE	Andreea	Monica
Ioan	Călin	SC	IE	Andreea	Monica
Călin	Ioan	SC	IE	Monica	Andreea
Ioan	Călin	SC	IE	Monica	Andreea

Figura 66 O relaţie automorfism

Demonstraţie: Presupunem că $\boldsymbol{r}$ conţine doar o relaţie r_1; extensia demonstraţiei la oricâte relaţii este imediată (cu excepţia necesităţii de a introduce notaţii mai greoaie).

Presupunem că r_1 conţine p tupli t_1, …, t_p definiţi pe q atribute A_1, …, A_q. Întâi şi întâi construim o relaţie care, în esenţă, este produsul cartezian al lui r_1 cu el însuşi de p ori; sunt evident necesare pentru aceasta redenumiri adecvate, iar dacă presupunem că C_1, C_2, …, C_{pxq} sunt nume distincte de atribute, atunci această relaţie se obţine astfel:

$$r_P = \rho_{C1, \ldots, Cq \leftarrow A1, \ldots, Aq}(r_1) \bowtie \ldots \bowtie \rho_{C(p-1)xq+1, \ldots, Cpxq \leftarrow A1, \ldots, Aq}(r_1)$$

Considerăm un tuplu $t \in r_P$ care este juxtapunerea a p tupli distincţi: cu excepţia numelor atributelor, el conţine toţi tuplii din r_1; transformăm apoi r_P într-o altă relaţie r_W cu ajutorul următoarei proceduri, ce se referă explicit la tuplul t:

```
begin
  rW := rP;
  for i := 1 to p x q – 1
  do  for j := i + 1 to p x q
        do if t[Ci] = t[Cj]
```

then $r_W := \sigma_{Ci = Cj}(r_W)$
else $r_W := \sigma_{Ci \neq Cj}(r_W)$

```
end
```

Cum toate selecţiile se fac pe baza valorilor sale, tuplul t aparţine r_W. De asemenea, deoarece t conţine toţi tuplii din r_1, el conţine şi toate valorile din domeniul activ $D_{\boldsymbol{r}}$. Pentru orice valoare c_i, fie C_{ki} unul din atributele cu proprietatea că $t[C_{ki}] = c_i$; afirmăm că: $r_a = \rho_{Bc1, \ldots, Bcn \leftarrow Ck1, \ldots, Ckn}(\pi_{Ck1, \ldots, Ckn}(r_W))$ este relaţia automorfismelor $\varnothing$-fixe $r_\varnothing$. Demonstrăm această egalitate cu ajutorul dublei incluziuni:

1. $r_\varnothing \subseteq r_a$. Fie $t_\varnothing$ un tuplu oarecare al $r_\varnothing$; prin definiţie, el descrie un automorfism h cu proprietatea $t_\varnothing[Bc_j] = h(c_j)$, $\forall 1 \leq j \leq n$. Cum automorfismele transformă tuplii lui r_1 în tupli ai r_1, iar t este o combinaţie a tuplilor lui r_1, rezultă că tuplul $h(t)$ obţinut prin aplicarea lui h asupra lui t element cu element este de asemenea o combinaţie a tuplilor din r_1 şi deci aparţine lui r_P (care conţine toate aceste combinaţii posibile). De asemenea, automorfismele fiind bijective, ele păstrează egalităţile şi inegalităţile între valori: astfel, $h(t)$ are exact aceleaşi egalităţi de perechi ca t şi deci aparţine lui r_W. Având în vedere că tuplul obţinut prin restricţia lui $h(t)$ la C_{k1}, …, C_{kn} urmată de redenumirea atributelor sale cu Bc_1, …, Bc_n este exact $t_\varnothing$, care aparţine deci lui r_a, am demonstrat această primă incluziune.

2. $r_a \subseteq r_\varnothing$. Fie t_a un tuplu oarecare al r_a şi t_1 un tuplu al r_a obţinut dintr-un tuplu t al r_W; prin construcţie, $t_1[Bc_j] = c_j$, $\forall 1 \leq j \leq n$. Fie h o funcţie definită pe $D_{\boldsymbol{r}}$ astfel încât $h(t_1) = t_a$; deoarece toţi tuplii lui r_a au aceleaşi egalităţi de perechi, h este bijectivă. Fie t_W tuplul lui r_W din care provine t_a: rezultă că $t_W = h(t)$ şi deci, cum toţi tuplii lui r_W se obţin prin juxtapunerea tuplilor din r_1, rezultă că h este un automorfism. Ca atare, $t_a = h(t_1)$ aparţine relaţiei automorfismelor $r_\varnothing$. În sfârşit, pentru orice $C = \{c_{i1}, \ldots, c_{iC}\} \subseteq D_{\boldsymbol{r}}$, relaţia automorfismelor C-fixe r_C poate fi obţinută aplicând asupra $r_\varnothing$ o selecţie cu următorul predicat: $(Bc_{j1} = c_{i1}) \wedge \ldots \wedge (Bc_{jC} = c_{iC})$.

q.e.d.

Exemplul 54 Datorită complexității calculelor implicate, exemplificările procedurii folosite în demonstrația lemei de mai sus pot utiliza doar mici tabele. Figurile 67 şi 68 prezintă două asemenea exemple.

r_1 r_P

$r_W=\sigma_{C1\neq C2}(\sigma_{C1\neq C3}(\sigma_{C1\neq C4}(\sigma_{C2\neq C3}(\sigma_{C2\neq C4}(\sigma_{C3\neq C4}(r_P))))))$

A_1	A_2 $r_\varnothing$
1	$t\rightarrow$
3	4

C_1	C_2	C_3	C_4
1	2	3	4
3	4	1	2

B_1	B_2	B_3	B_4
1	2	3	4
3	4	1	2

C_1	C_2	C_3	C_4
1	2	1	2
1	2	3	4
3	4	1	2
3	4	3	4

Figura 67 Construcția unei relații a unui automorfism

r_1 r_P

$r_W=\sigma_{C1\neq C2}(\sigma_{C1=C3}(\sigma_{C1\neq C4}(\sigma_{C2\neq C3}(\sigma_{C2\neq C4}(\sigma_{C3\neq C4}(r_P))))))$

A_1	A_2
1	$t\rightarrow$
3	4

C_1	C_2	C_3	C_4 $r_\varnothing$
1	2	1	3
1	3	1	2

B_1	B_2	B_3
1	2	3
1	3	2

C_1	C_2	C_3	C_4
1	2	1	2
1	2	1	3
1	3	1	2
1	3	1	3

Figura 68 Construcția unei alte relații a unui automorfism

Demonstrația Teoremei 154

(⇐) (*şi numai dacă*)

Să presupunem că există o expresie E menționând C care produce r, adică $r = E(\boldsymbol{r})$. Fără a pierde nimic din generalitate (vezi problema 93), se poate presupune că E implică doar reuniuni, redenumiri, diferențe, produse carteziene şi selecții care au numai condiții atomice (adică de forma $A_1 = A_2$ sau $A = c$, unde $c \in C$). Pentru a demonstra această primă implicație, vom folosi inducția după numărul de operatori din E, arătând că rezultatul lui E este invariant în raport cu orice automorfism C-fix al lui $\boldsymbol{r}$.

Primul pas al inducției (corespunzând expresiilor fără nici un operator) este trivial: expresia este doar o relație $r_i \in \boldsymbol{r}$, care este invariantă la orice automorfism al lui $\boldsymbol{r}$, deoarece este element al $\boldsymbol{r}$.

Pentru al doilea pas al inducției, presupunem că operanzii celui mai din afară operator satisfac cerința (adică sunt invarianți în raport cu orice automorfism C-fix al lui $\boldsymbol{r}$) și arătăm că rezultatul satisface și el această cerință. Distingem între diversele cazuri, în funcție de cel mai din afară operator. În toate cazurile considerăm un automorfism C-fix oarecare h al lui $\boldsymbol{r}$ și arătăm că, $\forall t \in E(\boldsymbol{r})$, $h(t) \in E(\boldsymbol{r})$.

- *Reuniune*: $E=E_1 \cup E_2$. Dacă $t \in E(\boldsymbol{r})$, atunci $t \in E_1(\boldsymbol{r})$ sau $t \in E_2(\boldsymbol{r})$. Prin ipoteza inducției, în primul caz rezultă că $h(t) \in E_1(\boldsymbol{r})$, iar în al doilea că $h(t) \in E_2(\boldsymbol{r})$; prin urmare, $h(t) \in E_1(\boldsymbol{r}) \cup E_2(\boldsymbol{r})$=E($\boldsymbol{r}$).
- Pentru *redenumire*, *proiecție* și *produs cartezian* demonstrația este similară celei pentru reuniune.
- *Diferență*: $E=E_1-E_2$. Prin ipoteza de inducție, $h(t) \in E_1(\boldsymbol{r})$; pentru a demonstra acest subpunct trebuie deci arătat doar că $h(t) \notin E_2(\boldsymbol{r})$. Să presupunem prin absurd că $h(t) \in E_2(\boldsymbol{r})$. Deoarece $t \notin E_2(\boldsymbol{r})$, care este finit, iar h este o bijecție asupra tuplilor, $\exists t' \in E_2(\boldsymbol{r})$ astfel încât $h(t) \notin E_2(\boldsymbol{r})$, ceea ce contrazice ipoteza că $E_2(\boldsymbol{r})$ este invariant la automorfismele C-fixe ale lui $\boldsymbol{r}$.
- *Selecție* cu condiții atomice de egalitate: (1) $E=\sigma_{A1=A2}(E_1)$ sau (2) $E=\sigma_{A=c}(E_1)$ (unde $c \in C$). Dacă $t \in E(\boldsymbol{r})$, atunci $t \in E_1(\boldsymbol{r})$ și deci, prin ipoteza inducției, $h(t) \in E_1(\boldsymbol{r})$. Deoarece $t \in E(\boldsymbol{r})$, din definiția selecției rezultă ori că $t[A_1]= t[A_2]$ (în cazul (1)), ori că $t[A]=c$ (în cazul (2)). Cum h este un automorfism C-fix, urmează că $h(t)[A_1]= h(t)[A_2]$ (în cazul (1)), ori că $h(t)[A]=c$ (în cazul (2)); în ambele cazuri, rezultă deci că $h(t) \in E(\boldsymbol{r})$.

($\Rightarrow$) (*dacă*)

Fie r o relație invariantă la orice automorfism C-fix al lui $\boldsymbol{r}$ și fie un tuplu $t \in r$: deoarece orice tuplu al relației automorfismelor C-fixe r_C conține toate valorile din $\boldsymbol{r}$ (și deci toate cele din r, deci și toate cele din t), este posibil de obținut –aplicând asupra r_C redenumiri, produse carteziene (dacă există valori repetate în t) și o proiecție- o relație r_i astfel încât $t \in r_i$. Repetând aceeași construcție pentru fiecare $t \in r$, se poate construi o relație $r^* = \cup_{t \in r} r_t$. Cum r este finit și, conform lemei 156, r_C poate fi obținut cu ajutorul unei expresii algebrice relaționale menționând C, rezultă că r^* poate fi de asemenea obținut ca rezultat al unei expresii al acestei clase.

Pentru a termina demonstrația, mai trebuie arătat că $r^* = r$. Cum $\forall\ t \in r$, $t \in r_t$, rezultă că $r \subseteq r^*$. Pentru incluziunea opusă, $\forall t \in r$, relația r_t rezultă direct din relația automorfismelor C-fixe și deci, $\forall t' \in r_t$, $\exists h$ automorfism C-fix astfel încât $t'= h(t)$. Ca atare (deoarece r este invariant la orice automorfism C-fix), $t' \in r$; în consecință, $r_t \subseteq r$ $\forall t \in r$ și deci $r^* \subseteq r$.

q.e.d.

Exemplul 55 Relația r_1 din figura 62 se poate obține doar cu ajutorul unei expresii menționând cel puțin una dintre valorile {'Călin','Ioan'}, una dintre valorile {'IE','SC'} și una dintre valorile {'Andreea','Monica'} (un singur membru al fiecăreia dintre aceste perechi fiind suficient deoarece automorfismele sunt bijective și deci fixarea uneia dintre valori este suficientă pentru a o fixa și pe cealaltă). Dacă C = {'Călin','SC','Andreea'}, atunci r_C (care conține un singur tuplu) poate fi obținută cu ajutorul unei selecții având condiția C_1='Călin'$\wedge C_2$='SC'$\wedge C_3$='Andreea'$\wedge C_4 \neq$'Călin'$\wedge C_5 \neq$'SC'$\wedge C_6 \neq$'Andreea' aplicată asupra produsului cartezian $\rho_{C1C2C3 \leftarrow AFS}(r_0) \times \rho_{C4C5C6 \leftarrow AFS}(r_0)$. Apoi, urmând demonstrația părții *dacă* a teoremei 154, se poate obține r_1 din următoarea expresie:

$$\rho_{AFS \leftarrow C1C2C3}(\pi_{C1C2C3}(r_C)) \cup \rho_{AFS \leftarrow C4C5C6}(\pi_{C4C5C6}(r_C)).$$

De notat că, aşa cum este de obicei cazul construcțiilor generale, demonstrația teoremei 154 generează adesea o expresie complexă, deşi există şi expresii echivalente mult mai simple; de exemplu, relația r_2 din figura 65, care este invariantă la orice automorfism al r_0, poate fi obținută cu ajutorul unei singure proiecții: $r_2 = \pi_{AF}(r_0)$.

7.3 Calculul relațional

Termenul de *calcul relațional* (*CR*) se referă la o familie de limbaje care sunt bazate pe calculul predicatelor de ordin întâi. Cum operanzii relaționali sunt tabele (deci obiecte bidimensionale), este de aşteptat faptul să existe două subfamilii de asemenea limbaje (a căror denumire foloseşte terminologia modelului relațional): *calcul relațional al domeniilor* (în care variabilele sunt coloane, i.e. domeniile atributelor relaționale) şi *calcul relațional al tuplilor* (în care variabilele sunt linii, i.e. tuplii relațiilor).

Provenind din calculul predicativ, principala caracteristică a acestor limbaje este evident aceea că expresiile lor statuează doar proprietățile pe care trebuie să le satisfacă rezultatul, fără a preciza cum anume se calculează acesta (i.e. sunt declarative, neprocedurale).

Pentru completitudine, prezentarea de față a acestor limbaje nu presupune cunoştințe despre calculul predicatelor de ordin întâi. Pentru cei familiarizați însă cu acesta din urmă şi pentru justificarea unor alegeri tehnice, prezentăm în continuare principalele diferențe între calculul relațional şi cel predicativ de ordin întâi.

Prima simplificare majoră este lipsa simbolilor de funcții (inutili în context relațional, cu excepția celor booleni standard). În plus, simbolii de predicate (interpretați în calculul predicativ ca relații matematice) sunt interpretați (cu excepția operatorilor standard, precum cei de comparație „=”, „≠”, „<”, „≤”, „>”, „≥”) ca fiind conținuturi de relații ale unei bd.

A doua simplificare majoră constă în lipsa formulelor închise. În calculul predicativ, formulele se împart în două clase: *formule deschise* (care conțin variabile libere) şi *formule închise*, numite şi *propoziții* (care conțin doar variabile legate de cuantificatorii universal şi/sau existențial). În orice interpretare fixată, formulele închise au o unică valoare de adevăr, în timp ce valoarea de adevăr a formulelor deschise depinde de valorile asociate variabilelor libere. Cum calculul relațional este folosit doar ca limbaj de interogare, adică un limbaj ce defineşte funcții de la conținuturi de bd la relații, formulele închise nu interesează în acest context. În plus, înțelesul unei formule deschise este considerat a fi mulțimea tuplilor care face formula adevărată.

În al treilea rând, deoarece relațiile sunt definite diferit față de teoria mulțimilor (adică folosind mulțimi neordonate de nume unice pentru suporți, în loc de mulțimi ordonate de nume nu neapărat unice), notația folosită de calculul relațional diferă uşor de cea standard a celui predicativ (lucru care nu afectează însă cu nimic semantica sa).

7.3.1 Calculul relațional al domeniilor

Subsecțiunea de față include o primă prezentare simplificată a acestui limbaj de interogare.

Definiția 157 Expresiile CRD au forma $\{A_1: x_1, \ldots, A_k: x_k \mid f\}$, unde *f* este o *formulă*, $x_1, \ldots, x_k$ sunt *variabilele* ce apar în *f*, iar $A_1, \ldots, A_k$ sunt atribute. Lista perechilor $A_1: x_1, \ldots, A_k: x_k$, ce defineşte structura rezultatului expresiei (care este o relație peste atributele $A_1, \ldots, A_k$ conținând doar tuplii ale căror valori $c_1, \ldots, c_k$ substituite variabilelor $x_1, \ldots, x_k$ *satisfac*[50] *f*), se zice *lista țintă* a formulei.

[50] Conceptele de formulă, variabilă şi satisfacere sunt definite în detaliu în cele ce urmează

Exemplul 56 Pentru a furniza un suport intuitiv, prezentăm patru exemple de expresii *CRD* ale căror rezultate (vezi figura 69) sunt calculate pentru conținutul bd din figura 54; le vom aprofunda ulterior, după parcurgerea prezentării de detaliu a *CRD*.

Interogarea (E10) calculează numele și locul înhumării persoanelor despre care se știe că au decedat în secolul XIV; (E11) calculează numele și anul morții tuturor domnitorilor; (E12) calculează numele domnitorilor a căror mamă nu se cunoaște; iar (E13) calculează fondatorul dinastiei. Pentru simplitate, sunt folosite abrevieri neambigue ale numelor atributelor.

(E10) $\{N: x_1, Nec: x_3 \mid PERSOANE(N: x_1, M: x_2, Nec: x_3) \wedge (x_2 \geq 1300) \wedge (x_2 \leq 1399)\}$

(E11) $\{D: x_1, M: x_2 \mid PERSOANE(N: x_1, M: x_2, Nec: x_3) \wedge DOMNII(D: x_1, DL: x_4, L: x_5)\}$

(E12) $\{D: x_1 \mid DOMNII(D: x_1, DL: x_2, L: x_3) \wedge \neg(\exists(x_4(MAMA(M: x_4, C: x_1))))\}$

(E13) $\{D: x_1, DL: x_2, L: x_3 \mid DOMNII(D: x_1, DL: x_2, L: x_3) \wedge$
$\wedge \forall x_4(\forall x_5(\forall x_6(\neg DOMNII(D: x_4, DL: x_5, L: x_6) \vee (x_5 \geq x_2))))\}$

Evident că toate aceste expresii confirmă natura declarativă a calculului: lista țintă conține numele atributelor alcătuind schema relației rezultat, în timp ce formula definește condiția ce trebuie satisfăcută de toți tuplii conținutului acesteia, fără nici o precizare asupra modului de calcul al lor. În același timp, ele vădesc unul din principalele dezavantaje ale acestei prime versiuni a *CRD* și anume complexitatea notațională: expresiile *CRD* sunt cel mai adesea mult mai lungi decât echivalentul lor algebric relațional.

Stăruim în cele ce urmează asupra definiției exacte a *CRD*, începând cu sintaxa și terminând cu semantica sa.

Definiția 158 Simbolii ce pot apărea în formulele *CRD* sunt următorii:

- *Constante* (valori ale domeniului *D*); e.g.: „Basarab I”, 1310, „Câmpulung”;
- *Variabile* (elemente ale unei mulțimi *V* infinită, numărabilă, disjunctă de *D*, inclusă în monoidul liber generat peste un alfabet oarecare); exemple: x_1, x_2, x_3;
- *Nume de relații și atribute ale acestora*[51]; e.g.: „*DOMNII*”, „r_1”, „*Nume*”, „*La*”;
- *Operatori de comparație* („=”, „≠” și, dacă *D* este măcar parțial ordonat, „<”, „≤”, „>”, „≥”);
- *Conectori logici*: ¬ (*nu, negația*), ∧ (*și, conjuncția*), ∨ (*sau, disjuncția*);
- *Cuantificatori logici*: ∀ (*oricare ar fi, universal*) și ∃ (*există, existențial*);
- *Paranteze*: (pentru precizarea ordinii de evaluare a subformulelor); de exemplu: $(a \vee b) \wedge c$.

(E10)

Nume	*Necropolă*
Basarab I	Câmpulung
Nicolae Alexandru	Câmpulung
Vladislav Vlaicu	Curtea Argeș
Radu I	
Dan I	
Dan II	
Radu II Pleșuvul	
Alexandru Aldea	
Vlad Dracul	
Vladislav II	
Laiotă Basarab	
Basarab Țepeluș	
Radu cel Mare	

(E12)

Domnitor
Basarab I
Nicolae Alexandru
Dan I
Mircea cel Bătrân
Vlad Uzurpatorul
Mihail I

[51] Evident că mulțimile numelor de relații și atribute, variabilelor, precum și submulțimea constantelor de tip șir de caractere sunt submulțimi ale monoidului liber generat peste un alfabet oarecare.

(E11)

Domnitor	*Moarte*
Basarab I	1352
Nicolae Alexandru	1364
Vladislav Vlaicu	1377
Radu I	1384
Dan I	1386
Mircea cel Bătrân	1418
Vlad Uzurpatorul	1396
Mihail I	1420
Dan II	1431
Radu II Pleşuvul	1427
Alexandru Aldea	1436
Vlad Dracul	1447
Vladislav II	1456
Vlad Ţepeş	1477
Radu cel Frumos	
Laiotă Basarab	1477
Basarab Ţepeluş	
Vlad Călugărul	1495
Radu cel Mare	1508

(E13)

Domnitor	*DeLa*	*La*
Basarab I	1310	1352

Figura 69 Rezultatele expresiilor din exemplul 56 aplicate asupra bd din exemplul 49

Definiția 159 Componentele fundamentale ale formulelor *CRD* se zic *atomi* şi pot avea doar una din următoarele două forme:

- $R(A_1: x_1, \ldots, A_k: x_k)$, unde $R(A_1, \ldots, A_k)$ este o schemă de relație R cu atributele $A_1, \ldots, A_k$, iar $x_1, \ldots, x_k$ sunt variabile distincte; pentru a simplifica notația, se consideră câteodată că atributele $A_1, \ldots, A_k$ sunt ordonate şi se poate deci omite numele lor, scriind atomul corespunzător sub forma $R(x_1, \ldots, x_k)$.[52]
- $x\ \theta\ a$, unde x este o variabilă, a o variabilă sau o constantă, iar θ un operator de comparație.

Formulele *CRD* se construiesc recursiv din atomi, paranteze, conectori şi cuantificatori logici. Din motive ce se vor lămuri mai târziu, este utilă distincția între două tipuri de apariții ale variabilelor în formule: *libere* (de cuantificări) şi *legate* (de cuantificatori). De notat că, într-o aceeaşi formulă, o aceeaşi variabilă poate apărea (o dată sau de mai multe ori) atât liberă, cât şi legată (la unul sau mai mulți cuantificatori).

Cele patru reguli din definiția următoare descriu complet formulele şi tipul corespunzător de apariție al variabilelor.

Definiția 160 Conceptele de *formulă* (*a CRD*) şi de *apariții de variabile libere sau legate în formule* se definesc recursiv astfel:

- orice atom este formulă; toate aparițiile variabilelor unui atom se zic libere;
- dacă f este o formulă iar x o variabilă, atunci $\exists x(f)$ şi $\forall x(f)$ sunt formule; toate aparițiile lui x în f sunt legate (de cuantificatorul respectiv); toate aparițiile celorlalte variabile în aceste tipuri de formule sunt libere sau legate, după cum sunt ele în f;
- dacă f_1 şi f_2 sunt formule, atunci şi $(f_1) \wedge (f_2)$, $(f_1) \vee (f_2)$ şi $\neg (f_1)$ sunt formule (unde parantezele pot fi omise dacă nu există riscul de ambiguități[53]); aparițiile variabilelor în aceste formule sunt libere, respectiv legate, exact după cum sunt ele libere sau legate în subformulele f_1 şi f_2 în care apar;
- nimic altceva nu este formulă.

Exemplul 57 Revenind la bd din exemplul 49 şi figura 54, x_1 = „Câmpulung”, $x_1 \le x_2$ şi *DOMNII*(*Domnitor:* x_1, *DeLa:* x_2, *La:* x_3) sunt exemple de atomi şi deci de formule; presupunând că atributele relației *DOMNII* sunt ordonate, ultimul atom se poate scrie simplificat *DOMNII*(x_1, x_2, x_3). Următoarele formule (scrise simplificat) sunt neatomice:

[52] Este posibilă o definiție mai generală a atomilor având această formă, care permite atât aparții multiple ale unei aceleiaşi variabile, cât şi constante în locul variabilelor; definiția 159 este însă numai aparent mai restrictivă, deoarece acelaşi lucru se poate obține folosind conjuncții de atomi şi atomi cu egalitate (a doua formă, cu θ=”=”).

[53] Reamintim precedența operatorilor logici: în absența parantezelor (care, desigur, pot impune orice ordine dorită) se evaluează întâi negația (echivalent algebric: schimbarea semnului), apoi conjuncția (echivalent: înmulțirea) şi la sfârşit disjuncția (echivalent: adunarea).

(E14) $\exists\, x_3\ (PERSOANE(x_1, x_2, x_3) \wedge (x_3 =$ „Cozia”))

(E15) $\forall\, x_2\ (\exists\, x_3\ (PERSOANE(x_1, x_2, x_3) \wedge (x_3 =$ „Cozia”)))

(E16) $\exists\, x_4\ (DOMNII(x_1, x_2, x_3) \wedge (\exists\, x_1\ (TATA(x_4, x_1))))$

În expresia (E14), aparițiile variabilelor x_1, x_2 sunt libere, în timp ce ambele apariții ale x_3 sunt legate (de cuantificatorul existențial); în (E15), apariția variabilei x_1 este liberă, în timp ce ambele apariții ale x_3 sunt legate (de cuantificatorul existențial), ca şi cea a variabilei x_2 (legată de cuantificatorul universal); în (E16), aparițiile variabilelor x_2, x_3 sunt libere, apariția lui x_1 din atomul *DOMNII* este liberă, pe când cea din atomul *TATA* este legată (de al doilea cuantificator existențial), în timp ce apariția lui x_4 este legată (de primul cuantificator existențial).

Semantica *CRD* se defineşte precizând valorile expresiilor sale în raport cu o schemă de baze de date $\boldsymbol{R} = \{R_1(X_1), \ldots, R_m(X_m)\}$ şi un conținut $\boldsymbol{r} = \{r_1, \ldots, r_m\}$ al acesteia. Prin similitudine cu definiția recursivă a formulelor, semantica expresiilor *CRD* se defineşte pe baza definiției recursive a valorilor de adevăr ale formulelor, valori ce sunt calculate în funcție de valorile substituite variabilelor libere ale acestora. Este deci nevoie întâi de o definire preliminară a substituțiilor.

Definiția 161 Se zice *substituție* orice funcție $s : V \to D$, care asociază câte o constantă fiecărei variabile.[54]

Definiția 162 Valoarea de adevăr a unei formule a CRD se defineşte recursiv astfel:

- Atomi
  - Valoarea $R(A_1: x_1, \ldots, A_k: x_k)$ pentru o substituție s este *adevărat* dacă relația r peste R conține un tuplu t cu proprietatea că $t[A_i] = s(x_i)$, $\forall 1 \le i \le k$ şi este *fals* altminteri.
  - $x\ \theta\ a$
    1) $a = x' \in V$ (variabilă): $x\ \theta\ x'$ este *adevărat* pentru s dacă $s(x)$ şi $s(x')$ sunt în relația θ între ele şi este *fals* în caz contrar (de exemplu, dacă θ este egalitatea, atomul $x = x'$ este *adevărat* pentru s dacă $s(x) = s(x')$ şi este *fals* altminteri);
    2) $a = c \in D$ (constantă): $x\ \theta\ c$ este *adevărat* pentru s dacă $s(x)$ şi c sunt în relația θ între ele şi este *fals* în caz contrar.
- Valorile $(f_1) \wedge (f_2)$, $(f_1) \vee (f_2)$ şi $\neg\ (f_1)$ pentru o substituție s sunt definite în conformitate cu semantica conectorilor logici corespunzători aplicați valorilor subformulelor f_1 şi f_2 pentru s (adică, pentru $\neg\ (f_1)$, *adevărat* dacă f_1 este *fals*, respectiv *fals* dacă f_1 este *adevărat*; pentru $(f_1) \wedge (f_2)$, *adevărat* dacă atât f_1 cât şi f_2 au valoarea *adevărat*, respectiv *fals* în caz contrar; iar pentru $(f_1) \vee (f_2)$, *adevărat* dacă măcar una dintre f_1 şi f_2 are valoarea *adevărat*, respectiv *fals* în caz contrar).
- Valoarea $\exists x(f)$ pentru o substituție s este *adevărat* dacă există o substituție s' pentru care f este *adevărat* şi care diferă de s cel mult asupra lui x; altminteri ea este *fals*. În mod simetric, valoarea $\forall x(f)$ pentru o substituție s este *adevărat* dacă f este *adevărat* pentru orice substituție s' care diferă de s cel mult asupra lui x; altminteri ea este *fals*.

Definiția 163 Valoarea unei expresii CRD $\{A_1: x_1, \ldots, A_k: x_k \mid f\}$ este o relație peste atributele $A_1, \ldots, A_k$ ai cărei tupli sunt definiți de substituțiile pentru care f este *adevărat*: $\{t \mid \forall i, 1 \le i \le k, \exists s\ (f(s) = adevărat \wedge t[A_i] = s(x_i))\}$.

Exemplul 58 Să reconsiderăm expresia (E13) din exemplul 56:

(E13) $\{D: x_1, DL: x_2, L: x_3 \mid DOMNII(D: x_1, DL: x_2, L: x_3) \wedge \forall x_4(\forall x_5(\forall x_6$
$(\neg DOMNII(D: x_4, DL: x_5, L: x_6) \vee (x_5 \ge x_2))))\}$

[54] Substituțiile sunt definite ca funcții totale doar din motive „tehnice”; din punct de vedere practic însă, interesează doar variabilele ce apar în formula respectivă. Se poate uşor arăta că două substituții distincte conduc la o aceeaşi valoare de adevăr a unei formule dacă ele diferă doar asupra variabilelor ce nu apar în aceea formulă.

Pentru a îi calcula valoarea, trebuie întâi să calculăm valoarea formulei sale (subformulă cu subformulă) pentru diverse substituții. Astfel, subformula:

(E17) $DOMNII(D: x_1, DL: x_2, L: x_3)$ are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă există un tuplu $t \in domnii$ astfel încât $t[D] = s(x_1)$, $t[DL] = s(x_2)$ şi $t[DL] = s(x_3)$, indiferent de valorile lui *s* pentru celelalte variabile. Subformula:

(E18) $(x_5 \geq x_2)$ are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă valoarea lui *s* pentru x_5 este mai mare sau egală decât cea pentru x_2 (indiferent de valorile lui *s* pentru celelalte variabile). Subformula:

(E19) $DOMNII(D: x_4, DL: x_5, L: x_6)$ este similară cu (E17), fiind diferite doar numele variabilelor. Subformula:

(E20) $\neg DOMNII(D: x_4, DL: x_5, L: x_6)$

are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă valoarea (E19) pentru *s* este *fals*, iar subformula:

(E21) $\neg DOMNII(D: x_4, DL: x_5, L: x_6) \vee (x_5 \geq x_2)$ are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă măcar una dintre valorile (E18) şi (E20) pentru *s* este *adevărat*. Subformula cuantificată:

(E22) $\forall x_6(\neg DOMNII(D: x_4, DL: x_5, L: x_6) \vee (x_5 \geq x_2))$ are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă subformula (E21) are valoarea *adevărat* pentru toate substituțiile care diferă de *s* cel mult asupra lui x_6. Raționând în acelaşi mod ca mai sus pentru restul de cuantificatori şi conectori logici ai formulei din expresia (E13) rezultă că aceasta are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă există un tuplu $t \in domnii$ astfel încât $t[D] = s(x_1)$, $t[DL] = s(x_2)$ şi $t[DL] = s(x_3)$ şi, pentru toate substituțiile care diferă de *s* cel mult peste x_4, x_5, x_6, subformula (E21) are valoarea *adevărat*. Cu puțin efort, se poate constata deci că valoarea expresiei (E13) este exact cea cerută, adică submulțimea domniilor care au început înaintea tuturor celorlalte domnii (i.e. tuplul corespunzând domniei fondatoare a dinastiei).

De observat faptul că valoarea unei formule cuantificate (existențial sau universal) pentru o substituție *s* oarecare nu depinde de valoarea lui *s* pentru variabila legată de cuantificatorul respectiv. Cu ajutorul inducției matematice se poate generaliza această observație: valorile unei formule pentru o substituție *s* depind numai de valorile lui *s* pentru variabilele libere ale formulei (adică, dacă *s* şi *s'* diferă doar asupra unor variabile care nu au nici o apariție liberă în cadrul formulei, atunci formula are aceeaşi valoare atât pentru *s*, cât şi pentru *s'*). Pe baza acestei şi altor observații similare, se poate demonstra următoarea lemă (a cărei demonstrație este cerută de problema 96; în cele ce urmează, se oferă în exemplul 59 doar suport intuitiv pentru rezolvarea ei):

Lema 164. În *CRD*, pentru orice expresie, există o expresie echivalentă cu proprietatea că variabilele libere din formula ei sunt exact cele din lista țintă.

Exemplul 59 Expresia din exemplul 56:

(E10) $\{N: x_1, Nec: x_3 \mid PERSOANE(N: x_1, M: x_2, Nec: x_3) \wedge (x_2 \geq 1300) \wedge (x_2 \leq 1399)\}$,

a cărei listă țintă nu conține toate variabilele ce apar liber în formula ei, este evident echivalentă cu:

(E23) $\{N:x_1, Nec:x_3 \mid \exists x_2(PERSOANE(N:x_1, M:x_2, Nec:x_3) \wedge (x_2 \geq 1300) \wedge (x_2 \leq 1399))\}$.

În general, orice expresie a cărei listă țintă conține variabile ce nu apar liber în formula ei nu prea are înțeles; este însă întotdeauna posibilă transformarea ei prin adăugarea unei subformule „inofensive" care să conțină variabila lipsă; de exemplu, expresia:

$$\{ A: x_1, Nume: x_2 \mid \exists x_3(TATA(Tata: x_3, Copil: x_2))\}$$

poate fi transformată în expresia echivalentă:

$$\{ A: x_1, Nume: x_2 \mid \exists x_3(TATA(Tata: x_3, Copil: x_2)) \wedge (x_1 = x_1)\}.$$

Ca atare, ori de câte ori acest lucru este necesar, putem presupune că toate variabilele libere apar în lista țintă şi viceversa. Câteodată, e drept, precum de exemplu în cazul (E10) sau (E23), versiunile restrictive sunt mai greu de citit decât cele „liberale”; această presupunere simplifică însă mult tehnica argumentației teoretice. Din motive similare, este convenabil să presupunem în plus că orice formulă are cel puțin o variabilă liberă.[55]

O altă simplificare în structura formulelor *CRD* se poate obține luând în considerare echivalențele binecunoscute în matematică între conectori şi cuantificatori logici: de exemplu, conform regulii DeMorgan corespunzătoare, $\forall(f_1, f_2)(f_1) \vee (f_2)$ are aceeaşi valoare, indiferent de substituție, cu $\neg(\neg(f_1) \wedge\neg (f_2))$; similar, prin definiție, $\forall x(f)$ are aceeaşi valoare cu $\neg(\exists x(\neg(f)))$. Formalizând toate acestea şi făcând uz de inducția matematică, se poate demonstra (vezi problema 97) şi următoarea lemă:

Lema 165 În *CRD*, pentru orice expresie, există o expresie echivalentă a cărei formulă conține doar negații, conectori de un singur tip (i.e. ori doar conjuncții, ori doar disjuncții) şi cuantificatori de un singur tip (i.e. ori doar existențiali, ori doar universali).

Exemplul 60 Expresia din exemplele 56 şi 58:

(E13) $\{D: x_1, DL: x_2, L: x_3 \mid DOMNII(D: x_1, DL: x_2, L: x_3) \wedge \forall x_4(\forall x_5(\forall x_6 (\neg DOMNII(D: x_4, DL: x_5, L: x_6) \vee (x_5 \geq x_2))))\}$

poate fi transformată într-o expresie echivalentă a cărei formulă conține doar negații, conjuncții şi cuantificatori existențiali:

$\{D:x_1, DL:x_2, L:x_3 \mid DOMNII(D:x_1, DL:x_2, L:x_3) \wedge \neg\exists x_4(\exists x_5(\exists x_6(DOMNII (D:x_4, DL:x_5, L:x_6) \wedge \neg (x_5 \geq x_2))))\}$.

Se observă, în plus, că ultima subformulă, $\neg (x_5 \geq x_2)$, se poate simplifica, fiind evident echivalentă cu formula $(x_5 \leq x_2)$.

7.3.1.1 Independența domeniilor

Aşa cum a fost definit în secțiunea precedentă, *CRD* are câteva proprietăți indezirabile. În secțiunea de față şi în următoarea îl vom modifica în mod corespunzător, pentru a elimina aceste neajunsuri.

Exemplul 61 Să considerăm următoarea expresie *CRD* referitoare la schema bd „Domnitori” din exemplul 49:

(E24) $\{ Nume: x_1 \mid \neg(\exists x_2(TATA(Tata: x_1, Copil: x_2))) \}$.

Formula acestei expresii are valoarea *adevărat* pentru o substituție *s* dacă nu există nici un tuplu *t* în *Tata* astfel încât $t[Tata] = s(x_1)$; adică, are valoarea *adevărat* pentru toate substituțiile ale căror valori asupra x_1 nu apar în coloana *Tata* a relației. Deoarece substituțiile pot asocia fiecărei variabile oricare dintre valorile domeniului *D*, rezultatul expresiei (E24) este o relație cu atributul *Nume* şi cu câte un tuplu pentru fiecare dintre elementele domeniului *D* care nu apar ca valoare în coloana *Tata* din relația inițială.

Expresia (E24) de mai sus are evident proprietăți indezirabile: ea nu depinde doar de valorile conținutului curent al bd, ci şi de domeniul acestor valori; dacă acesta se schimbă, este posibil să se schimbe şi rezultatul expresiei. De asemenea, dacă domeniul este infinit, rezultatul este o mulțime infinită de tupli, deci nu mai este o relație. Ca atare, deoarece interogările sunt definite ca funcții peste mulțimea conținuturilor unei scheme, rezultă că expresia (E24) defineşte câte o interogare diferită pentru fiecare domeniu distinct de valori. Evident că nu este deloc plăcut de lucrat cu rezultate care se schimbă în funcție de domeniul de valori şi care mai şi pot fi infinite!

[55] Valoarea de adevăr a formulelor care nu conțin nici o variabilă liberă nu depinde de substituții. De aceea, o expresie a cărei formulă nu conține variabile libere poate fi considerată ca fiind o interogare booleană: ea are doar valoarea *adevărat* sau *fals*, depinzând de conținutul bd în cauză, şi nu o relație (deci o mulțime de tupli) cum au interogările uzuale.

Definiția 166 O *expresie* se zice a fi *independentă de domeniu* dacă ea reprezintă o aceeaşi interogare, indiferent de domeniul subiacent de valori. Formal, deoarece mulțimea conținuturilor valide se poate modifica atunci când se schimbă domeniul, se zice că o expresie *E* este independentă de domeniu dacă există o interogare *Q* cu proprietatea că pentru orice domeniu *D* expresia *E* reprezintă restricția *Q* la $I_D(\boldsymbol{R})$, unde $I_D(\boldsymbol{R})$ este notația pentru mulțimea tuturor conținuturilor lui $\boldsymbol{R}$ în raport cu *D*.

Definiția 167 Un *limbaj* se zice *independent de domeniu* dacă toate expresiile sale sunt independente de domeniu.

Din exemplul 61 rezultă că *CRD* este dependent de domeniu; desigur că ne-ar interesa să definim o restricție a sa care să fie independentă de domeniu (pe care să o numim, de exemplu, *calculul relațional al domeniilor independent de domeniu*, prescurtat *CRD-ID*). Din nefericire, literatura de specialitate include din acest punct de vedere un rezultat negativ important: s-a demonstrat (vezi [130], [345]) următoarea teoremă (deloc surprinzătoare!):

Teorema 168 Problema independenței domeniilor pentru *CRD* este nedecidabilă (i.e., dată fiind o expresie *CRD* oarecare, nu se poate întotdeauna decide dacă ea este sau nu independentă de domeniu).

Ca atare, rezultă că *CRD-ID* nu este un limbaj recursiv; dimpotrivă, este uşor de demonstrat însă (vezi problema 98) un rezultat pozitiv în această privință pentru algebra relațională:

Teorema 169 Algebra relațională este independentă de domeniu.

Desigur că aceasta implică imediat faptul că *CRD* este strict mai puternic decât *AR*: nu există expresii de algebră relațională echivalente cu expresiile *CRD* dependente de domeniu.

Restul acestei secțiuni este dedicat prezentării unui limbaj similar cu *CRD*, dar având aceeaşi putere expresivă ca şi *CRD-ID*; pentru aceasta, sunt necesare însă câteva preliminarii.

Definiția 170 Fie *E* o expresie *CRD* şi $\boldsymbol{r}$ un conținut de bd; se zice *domeniul activ al E şi* $\boldsymbol{r}$ reuniunea $D_{E,\boldsymbol{r}}$ a domeniului activ $D_{\boldsymbol{r}}$ al $\boldsymbol{r}$ cu mulțimea constantelor ce apar în formula expresiei *E*.

Violarea independenței domeniilor este posibilă tocmai pentru că variabilele pot lua în mod liber orice valori ale domeniului; dacă variabilele ar fi forțate să ia valori doar din domeniul activ (sau dintr-o submulțime a acestuia), atunci problema nu ar mai apărea. De aceea, s-a propus o versiune restrictivă a *CRD*, zisă *calculul relațional al domeniilor cu declarații de valori* (*CRD-DV*), care impune, pentru fiecare variabilă în parte, o declarație specificând submulțimea domeniului activ relevant (i.e. în care variabila poate lua valori), cu ajutorul unei mulțimi ale cărei elemente sunt constante şi/sau atribute ale relațiilor. Aceste declarații trebuie specificate, pentru fiecare variabilă în parte, la cel mai înalt nivel cu putință la care acest lucru are sens şi anume:

1) global, dacă variabila este liber sau, în caz contrar,
2) împreună cu cuantificatorul de care ea este legată.

Înainte de a furniza definiția exactă a *CRD-DV*, demonstrăm două rezultate pregătitoare ce se vor dovedi fundamentale pentru această abordare.

Lema 171 Pentru orice expresie *CRD E*, există o formulă $d(x)$ cu o variabilă liberă x care, pentru orice conținut de bd $\boldsymbol{r}$, are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă $s(x) \in D_{E,\boldsymbol{r}}$.

Demonstrație: Fie $R_1(A_{1,1} \ldots A_{1,p1})$, …, $R_n(A_{n,1} \ldots A_{n,pn})$ schemele de relații de interes, iar c_1, …, c_q constantele din *E*. Pentru fiecare atribut $A_{i,j}$ al schemei de relație R_i se poate defini o formulă $d_{i,j}$ având variabila liberă x care, pentru orice conținut al bd, are valoarea *adevărat* pentru o substituție *s* dacă şi numai dacă $s(x)$ este una dintre valorile pentru $A_{i,j}$ în relația r_i; de exemplu, pentru atributul $A_{1,1}$ al R_1 formula $d_{1,1}$ este: $\exists x_2(\ldots(\exists$

$x_{p1}(R_1(x, x_2, \ldots, x_{p1})))\ldots)$. Ca atare, formula $d_{\boldsymbol{R}}$ definită ca disjuncția $d_{i,j}$-urilor are valoarea *adevărat* pentru *s* dacă şi numai dacă $s(x) \in D_{\boldsymbol{r}}$.

Similar, formula d_E definită ca $(x = c_1) \vee \ldots \vee (x = c_q)$ are valoarea *adevărat* pentru *s* dacă şi numai dacă $s(x) \in \{c_1, \ldots, c_q\}$.

Rezultă că formula $d = d_{\boldsymbol{R}} \vee d_E$ satisface enunțul lemei.

q.e.d.

De notat că formula *d* din lema de mai sus nu depinde de conținutul bd, ci produce domeniul activ pentru orice conținut.

Lema 172 Fie *E* o expresie *CRD* independentă de domeniu, cu variabilele libere $x_1, \ldots, x_k$ şi formula f şi fie *E'* o expresie *CRD* cu aceeaşi listă țintă ca şi *E* şi având formula $d(x_1)\wedge\ldots\wedge d(x_k)\wedge f'$, unde f' se obține din f prin substituirea recursivă a subformulelor cuantificate $\exists x(g)$ şi $\forall x(g)$ cu $\exists x(d(x)\wedge g)$, respectiv $\forall x(\neg d(x)\vee g)$. În aceste condiții, *E* şi *E'* sunt echivalente.

Demonstrație: Deoarece, prin ipoteză, *E* este independentă de domeniu, este evident (din construcția sa) că şi *E'* este independentă de domeniu; ca atare, pentru a demonstra echivalența acestor două expresii, este suficient de arătat că, pentru orice conținut de bd ***r***, ele produc rezultate identice pentru un domeniu. Acest domeniu poate să fie diferit pentru conținuturi de bd diferite, dar este mereu cu putință de ales, pentru fiecare conținut ***r*** în parte, domeniul activ al ***r*** şi *E* (care, evident, este acelaşi cu cel al ***r*** şi *E'*). Deoarece diferențe între *E* şi *E'* există numai în subformulele care forțează toate variabilele să ia valori doar din domeniul activ, rezultă că, pentru orice conținut ***r***, rezultatele celor două expresii sunt egale şi deci expresiile *E* şi *E'* sunt echivalente.

q.e.d.

O consecință directă a lemei 172 este aceea că rezultatul unei expresii *CRD* independente de domeniu conține doar valori din domeniul activ al expresiei şi al conținutului de bd. Se poate acum introduce formal *CRD-DV*, noua versiune a *CRD*:

Definiția 173 Se zice *componentă a domeniului de valori* ori o mulțime finită *C* de constante, ori submulțimea *R*[*A*] a domeniului atributului *A* al unei scheme de relație *R* (*X*). Se zice *declarație de domeniu al valorilor* o mulțime finită nevidă de componente ale domeniului de valori.

Definiția 174 O expresie *CRD* se zice *expresie CRD-DV* dacă conține în plus câte o declarație de domeniu al valorilor pentru fiecare variabilă liberă ca şi pentru fiecare subformulă cuantificată a formulei sale.

Sintactic, declarațiile pentru variabilele libere se scriu în lista țintă astfel:

$$\{A_1: x_1(G_1), \ldots, A_k: x_k(G_k) \mid f\},$$

iar cele asociate cuantificatorilor astfel: $\forall x(G)(f')$, respectiv $\exists x(G)(f')$.

Semantica interogărilor în *CRD-DV* se defineşte la fel cu cea pentru expresiile *CRD*, cu excepția faptului că, la definirea valorii fiecărei formule, substituțiile sunt funcții ce asociază fiecărei variabile o valoare din domeniul corespunzător de valori (şi nu din întreg domeniul *D*).

> *Exemplul 62* Considerând din nou bd dinastică din exemplul 49, interogarea definită de expresia (E23) din exemplul 59 poate fi formulată în *CRD-DV* astfel (folosind abrevieri neambigue pentru numele relației şi ale atributelor sale):

$$\{N{:}x_1(P[N]), Nec{:}x_3(P[Nec]) \mid \exists x_2(P[M])(P(N{:}x_1, M{:}x_2, Nec{:}x_3)\wedge(x_2\geq 1300)\wedge(x_2\leq 1399))\}.$$

Teorema 175 CRD-DV este independent de domeniu şi echivalent cu *CRD-ID*.

Demonstrație: Vom demonstra echivalența celor două limbaje, ceea ce, evident, va avea drept consecință imediată independența de domeniu a *CRD-DV*.

($\Rightarrow$)(*CRD-DV este cel puțin la fel de expresiv ca şi CRD-ID*)

Fie o schemă oarecare conținând *n* relații $R_1(A_{1,1} \ldots A_{1,p1}), \ldots, R_n(A_{n,1} \ldots A_{n,pn})$ şi fie *E* o expresie *CRD-ID* oarecare ce referă constantele $c_1, \ldots, c_q$; putem deci considera

expresia echivalentă *E'* definită de lema 172. Fie *G* declarația de domeniu al valorilor corespunzătoare domeniului activ, care include deci toți termenii $R_i[A_{i,j}]$, $\forall 1 \le i \le n$, $\forall 1 \le j \le p_i$, mulțimea constantelor $c_1, \ldots, c_q$ și nimic altceva. Fie E_d expresia *CRD-DV* obținută din *E* prin atașarea declarației *G* fiecărei variabile: se poate demonstra ușor (chiar dacă durează destul de mult) că E_d este echivalentă cu *E'* și deci, conform lemei 172, echivalentă cu *E*.

(⇐)(*CRD-ID este cel puțin la fel de expresiv ca și CRD-DV*)

Pentru orice expresie *CRD-DV* se poate construi o expresie echivalentă *CRD* transformând declarațiile de domeniu al valorilor în subformule combinate cu ajutorul conjuncției împreună cu formula inițială, în mod similar celui din lema 172. Cum cele două expresii sunt echivalente, rezultă că expresia *CRD* este independentă de domeniu și deci aparține *CRD-DI*.

q.e.d.

Transformarea folosită în teorema 175 este doar tehnică; în majoritatea cazurilor este posibilă folosirea unor declarații de domeniu al valorilor mult mai simple.

Exemplul 63 Aplicând teorema 175 expresiei (E23) din exemplele 59 și 62 rezultă următoarea expresie *CRD-DV*:

$\{N: x_1(G), Nec: x_3(G) \mid \exists x_2(G)(P(N: x_1, M: x_2, Nec: x_3) \wedge (x_2 \ge 1300) \wedge (x_2 \le 1399))\}$,

unde *G* este declarația de domeniu corespunzătoare domeniului activ:

$D[D], D[DL], D[L], P[N], P[M], P[Nec], T[T], T[C], M[M], M[C], 1300, 1397$.

Există însă expresii *CRD-DV* echivalente mult mai simple, precum (E25), de exemplu, ale cărei declarații de domeniu al valorilor sunt elementare:

(E25)$\{N:x_1(P[N]), Nec:x_3(P[Nec]) \mid \exists x_2(P[M])(P(N:x_1, M:x_2, Nec:x_3) \wedge (x_2 \ge 1300) \wedge (x_2 \le 1399))\}$.

Cele mai uzuale interogări necesită doar expresii cu declarații simple de domeniu al valorilor. Excepțiile notabile sunt constituite de interogările care, în algebra relațională, necesită operatorul de reuniune.

Exemplul 64 O interogare pentru calcularea relației *părinte* din *TATA* și *MAMA*, formulată în *AR* prin expresia (E1) din exemplul 49, poate fi formulată în *CRD-DV* astfel:

$\{N: x_1(T[T], M[M]), C: x_2(T[C], M[C]) \mid T(T: x_1, C: x_2) \vee M(M: x_1, C: x_2)\}$.

De notat că, deoarece valorile ambelor atribute ale rezultatului provin din două relații distincte, declarațiile de domeniu al valorilor trebuie să conțină fiecare câte două elemente.

Încheiem această secțiune cu discutarea posibilității de a folosi *CRD* (sau *CRD-DV*) ca bază pentru un limbaj de interogare implementabil pe calculator. Limbajele bazate pe calcul pot fi preferabile celor algebrice datorită caracterului lor neprocedural (și pentru că, așa cum se dovedește în subsecțiunea 7.1.2, puterea lor expresivă este în esență aceeași); totuși, așa cum am văzut deja în multe exemple, expresiile *CRD* și cu atât mai mult cele *CRD-DV*, implică o mulțime de detalii și sunt adesea greu de scris, deoarece pentru fiecare relație implicată în interogare ele trebuie în mod uzual să conțină câte o variabilă (plus declarația de domeniu al valorilor corespunzătoare) pentru fiecare atribut. Aceste considerații și altele asemănătoare au motivat definirea celeilalte familii de limbaje bazate pe calculul predicativ de ordinul întâi, ale cărui variabile denotă tupli, reducându-se astfel adesea complexitatea expresiilor (subsecțiunea 7.2 este dedicată studierii acestei familii).

Singurul limbaj real de succes bazat pe *CRD*, *Query-by-Example (QBE)* [369], folosește o interfață grafică care elimină necesitatea specificării majorității detaliilor, mai ales pentru interogările uzuale, simple.

Exemplul 65 Expresia (E10) din exemplul 56 s-ar formula în *QBE* cu ajutorul notației tabulare prezentată în figura 70. *Scheletul tabular*

corespunzând schemei oricărei relații este furnizat de sistem la cererea utilizatorului, care poate apoi specifica în el diverse elemente; de exemplu, „*operatorul* P.” indică coloanele ce contribuie la rezultat (deci elementele listei atributelor operatorului algebric relațional de proiecție). De remarcat uşurința exprimării declarațiilor de domeniu al valorilor, a apartenenței variabilelor la relații şi a condițiilor logice – datorată folosirii interfeței grafice.

PERSOANE

Nume	*Moarte*	*Necropolă*
P.x	≥ 1300 ≤ 1399	P.y

Figura 70 O expresie de tip *QBE*

7.3.1.2 Puterea expresivă a calculului relațional al domeniilor

În această secțiune comparăm diversele versiuni ale *CRD* cu *AR* în termenii puterii lor expresive; am văzut deja că expresiile *CRD* generale nu sunt toate independente de domeniu, iar cele dependente de domeniu nu au echivalent în algebra relațională. Se dovedeşte însă că *CRD-ID* şi *CRD-DV* sunt echivalente cu *AR*. Teorema principală dovedind echivalența se poate demonstra mai uşor cu ajutorul următoarelor două leme.

Lema 176 Pentru orice expresie *CRD-DV* E_c există o expresie *AR* echivalentă E_a. În plus, dacă toate domeniile de valori din E_c sunt atribute, atunci E_a nu implică relații constante.

Demonstrație: Fără nici un fel de pierdere a generalității, putem presupune (datorită unei variante pentru *CRD-DV* a lemei 165) că formula f a E_c nu conține nici conjuncții, nici cuantificatori universali, iar datorită unei variante a lemei 164, că variabila oricărui cuantificator existențial apare liberă în subformula respectivă.

Demonstrația se bazează pe inducția matematică după numărul de conectori şi cuantificatori logici ai formulei f şi construieşte o expresie *AR* al cărei rezultat este o relație peste o mulțime de atribute având exact numele $x_1, \ldots, x_p$ ale variabilelor libere din f; se arată apoi că E este echivalentă cu expresia *CRD-DV* $\{x_1: x_1, \ldots, x_p: x_p \mid f\}$. În sfârşit, expresia *AR* echivalentă cu E_c se obține apoi cu ajutorul unei redenumiri.

Înainte de a începe demonstrația propriu-zisă, trebuie remarcat un detaliu important în legătură cu atomii şi anume acela că variabilele lor pot lua valori doar în domeniile de valori corespunzătoare. Pentru orice variabilă x există o expresie E_x care defineşte domeniul corespunzător de valori: reuniunea redenumirilor proiecțiilor pe o singură coloană cu relațiile constante corespunzând elementelor din domeniul respectiv de valori.

- ✓ n=0 (*formula este atomică*).
  - ➢ $R(A_1: x_1, \ldots, A_p: x_p)$; expresia *AR* $\rho_{x1, \ldots, xp \leftarrow A1, \ldots, Ap}(R) \bowtie E_{x1} \bowtie \ldots \bowtie E_{xp}$ satisface enunțul. De remarcat că joinurile împreună cu declarațiile domeniilor de valori garantează faptul că valorile provin din aceste domenii de valori.
  - ➢ Expresia *AR* este $\sigma_{x1\theta x2}(E_{x1} \bowtie E_{x2})$ (unde joinul e de fapt un produs cartezian), dacă atomul este de forma $x_1\theta\ x_2$; respectiv $\sigma_{x\theta c}(E_x)$, dacă atomul este de forma $x\theta c$.
- ✓ *Inducție*. Conform lemei 165, este suficient să considerăm doar negația, conjuncția şi cuantificatorul existențial.
  - ➢ *Negație:* $f = \neg(f_1)$. Fie $x_1, \ldots, x_n$ variabilele libere din f_1 şi fie E_{f1} expresia *AR* corespunzătoare lui f_1. Din definiția negației, tuplii ce satisfac f sunt exact cei ce

nu satisfac f_1. Deoarece suntem interesați doar de tuplii ce provin din domeniile de valori ale diverselor variabile, expresia corespunzând lui f se poate obține cu ajutorul unei diferențe între produsul cartezian al domeniilor de valori şi expresia E_{f1}: $(E_{x1} \bowtie \ldots \bowtie E_{xn}) - E_{f1.}$

- *Conjuncție:* $f = f_1 \wedge f_2$. Fie E_{f1} şi E_{f2} expresiile corespunzătoare subformulelor f_1, respectiv f_2. Intuitiv, expresia corespunzătoare lui f este $E_{f1} \cap E_{f2}$; aceste două expresii însă nu sunt neapărat definite peste o aceeaşi mulțime de atribute (căci subformulele f_1 şi f_2 nu au în general aceleaşi variabile libere), deci lucrurile nu sunt chiar atât de simple. Fie V_1 şi V_2 mulțimile de variabile libere din f_1, respectiv f_2, iar $V_1 - V_2 = \{y_1, \ldots, y_h\}$, $V_2 - V_1 = \{z_1, \ldots, z_k\}$. Pentru a obține expresii compatibile, extindem E_{f1} şi E_{f2} cu ajutorul următoarelor produse carteziene:
  $E_1 = E_{f1} \bowtie E_{y1} \bowtie \ldots \bowtie E_{yh}$ şi $E_2 = E_{f2} \bowtie E_{z1} \bowtie \ldots \bowtie E_{zk}$; rezultă că expresia corespunzătoare lui f este $E_1 \cap E_2$, care, de fapt, este echivalentă cu $E_{f1} \bowtie E_{f2}$.
- *Cuantificare existențială:* $f = \exists x(f_1)$. Să presupunem că x apare liberă în f_1 (în caz contrar, este evident că expresia corespunzătoare lui f este exact cea pentru f_1); fie $x_1, \ldots, x_n$ celelalte variabile libere ale f_1 (şi deci ale f), iar E_{f1} expresia corespunzătoare lui f_1. Evident, expresia corespunzătoare lui f se obține eliminând x cu ajutorul proiecției din expresia corespunzătoare lui f_1: $E_f = \pi_{x1, \ldots, xn}(E_{f1})$.

Cu aceasta, inducția este terminată. Expresia *AR* corespunzătoare expresiei *CRD-DV* $\{A_1: x_1, \ldots, A_k: x_k \mid f\}$, unde E_f este expresia *AR* pentru f, se obține prin următoarea redenumire: $\rho_{A1, \ldots, Ak \leftarrow x1, \ldots, xk}(E_f)$.

A doua parte a enunțului rezultă imediat din faptul că am folosit relații constante doar în subexpresiile asociate domeniilor de valori şi numai pentru a construi domenii de valori conținând constante.

q.e.d.

De observat că, aşa cum se întâmplă şi în cazul altor transformări generale, cea sugerată de lema 176 produce adesea expresii redundante.

Exemplul 66 Revenind la bd propusă în exemplul 49, să aplicăm procedura din lema 176 următoarei expresii *CRD-DV*, ce reprezintă interogarea care calculează numele persoanelor al căror tată este cunoscut, dar a căror mamă nu ne este cunoscută:

(E26) $\{Nume: x_1(T[C]) \mid \exists x_2(T[T])(T: x_2, C: x_1) \wedge \forall x_3(M[C])(\neg(x_1 = x_3)))\}$.

Întâi şi întâi eliminăm cuantificatorul universal (şi apoi dubla negație):

$\{Nume: x_1(T[C]) \mid \exists x_2(T[T])(T: x_2, C: x_1) \wedge \neg(\exists x_3(M[C])(x_1 = x_3)))\}$.

Apoi, notând cu E_{xi} expresiile corespunzătoare domeniilor de valori:

$E_{x1} = \rho_{x1 \leftarrow C}(\pi_C(T))$, $E_{x2} = \rho_{x2 \leftarrow T}(\pi_T(T))$, $E_{x3} = \rho_{x3 \leftarrow C}(\pi_C(M))$

şi aplicând repetat procedura din demonstrația lemei 176, se obține următoarea expresie *AR*: $\rho_{Nume \leftarrow x1}(\pi_{x1}(\rho_{x2,x1 \leftarrow T,C}(T) \bowtie E_{x1} \bowtie E_{x2}) \bowtie (E_{x1} - \pi_{x1}(\sigma_{x1 = x3}(E_{x1} \bowtie E_{x3}))))$.

Evident însă că următoarea expresie echivalentă este cu mult mai simplă:

$$\rho_{Nume \leftarrow C}(\pi_C(T) - (\pi_C(M)).$$

Lema 177 Pentru orice expresie *AR* E_a, există o expresie *CRD* echivalentă E_c, independentă de domeniu.

Demonstrație: A doua parte a enunțului este evidentă din independența de domenii a algebrei relaționale: dacă o expresie *CRD* este echivalentă cu una *AR*, atunci ea este obligatoriu independentă de domeniu.

Esența demonstrației se bazează tot pe inducția matematică asupra structurii expresiei *AR* E_a, adică asupra numărului ei de operatori. Datorită echivalențelor existente între operatori şi rezultatelor ce se cer demonstrate în problemele 93 şi 94, putem presupune că:

- toate relațiile constante din E_a sunt definite peste un unic atribut şi conțin exact un tuplu;
- nu apar nici intersecții, nici theta-joinuri;
- toate joinurile naturale sunt de fapt produse carteziene;
- predicatele selecțiilor sunt alcătuite doar dintr-un atom; şi că
- pentru orice operator de proiecție, lista atributelor după care se face proiecția elimină câte un singur atribut.

✓ n=0 (*nici un operator*). Distingem două cazuri:

- *relație constantă:* $E_a = \{t\}$, cu $t\,[A] = c$; expresia *CRD* $E_c = \{ A : x \mid x = c \}$ este echivalentă cu E_a.
- *relație a bd:* $E_a = R$, unde R are atributele $A_1, \ldots, A_k$. Expresia *CRD* echivalentă este evident următoarea: $E_c = \{A_1: x_1, \ldots, A_k: x_k \mid R(A_1: x_1, \ldots, A_k: x_k)\}$.

✓ *Inducție.* Presupunând posibilă traducerea subexpresiilor *AR*, considerăm diversele cazuri ce ar putea apare corespunzător operatorului de cel mai înalt nivel curent. Pentru operatorii unari, presupunem că subexpresia E'_a este definită peste atributele $A_1, \ldots, A_n$, iar expresia *CRD* echivalentă acesteia este $E'_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi\}$.

- *Proiecție:* $E_a = \pi_{A1, \ldots, An-1}(E'_a)$; adică presupunem că se elimină doar atributul A_n. Expresia *CRD* echivalentă cu E_a este:
$$E_c = \{A_1: x_1, \ldots, A_{n-1}: x_{n-1} \mid \exists x_n(\psi)\}.$$
- *Redenumire:* $E_a = \rho_f(E'_a)$. Trebuie modificată doar lista țintă:
$$E_c = \{f(A_1): x_1, \ldots, f(A_n): x_n \mid \psi\}.$$
- *Selecție:* $E_a = \sigma_{Aj\theta Ak}(E'_a)$ (celălalt caz, în care condiția este de forma $A\theta c$, poate fi tratat în mod similar). Este suficientă transformarea condiției atomice folosind variabilele corespunzând atributelor şi formarea unei conjuncții între atomul astfel construit şi formula ψ: $E_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi \wedge (x_j \theta x_k)\}$.
- *Reuniune:* $E_a = E'_a \cup E''_a$. Din definiția reuniunii, E'_a şi E''_a sunt definite peste aceeaşi mulțime de atribute, ceea ce se păstrează deci şi pentru expresiile *CRD* corespunzătoare:
$$E'_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi'\}$$
$$E''_c = \{A_1: y_1, \ldots, A_n: y_n \mid \psi''\}.$$
Desigur că variabilele lor nu sunt aceleaşi; totuşi, variabilele se pot redenumi: dacă ψ'' este formula obținută prin redenumirea aparițiilor libere ale $x_1, \ldots, x_n$ cu $y_1, \ldots, y_n$ (plus, eventual, alte redenumiri ce ar mai putea fi necesare), atunci expresia *CRD* pentru E_a este: $E_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi' \vee \psi''\}$.
- *Diferență:* similar cu reuniunea, înlocuind disjuncția cu conjuncție şi negând cea de-a doua subformulă: $E_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi' \wedge \neg\psi''\}$.
- *Produs cartezian:* $E_a = E'_a \bowtie E''_a$. Conform celei de-a treia presupuneri făcute la începutul demonstrației, cele două subexpresii *AR* generează rezultate definite pe mulțimi disjuncte de atribute şi putem deci presupune că expresiile lor *CRD* corespunzătoare au mulțimi de variabile libere disjuncte (în caz contrar, desigur, ele pot fi redenumite convenabil):
$$E'_c = \{A_1: x_1, \ldots, A_n: x_n \mid \psi'\}$$
$$E''_c = \{B_1: y_1, \ldots, B_m: y_m \mid \psi''\}.$$

În consecință, expresia *CRD* echivalentă cu E_a se obține concatenând cele două liste țintă și construind conjuncția celor două formule:

$$E_c = \{A_1: x_1, \ldots, A_n: x_n, B_1: y_1, \ldots, B_m: y_m \mid \psi' \wedge \psi''\}$$

q.e.d.

Exemplul 67 Revenind la exemplul 51, să aplicăm procedura din lema 177 următoarei expresii *AR* definită asupra bd din figurile 54 și 57:

(E7) $\pi_{Domnitor,\ Capitala}\ (CAPITALE \bowtie_{DeLa \le An \wedge An \le La} DOMNII)$.

Trebuie întâi să transformăm expresia pentru a satisface presupunerile inițiale ale demonstrației (în acest caz: joinul înlocuit cu produs cartezian, selecțiile se fac conform unui unic atom, proiecțiile elimină câte un singur atribut, iar relațiile constante au un singur atribut și un singur tuplu). Fie c_1 relația peste atributul *Capitala* conținând un singur tuplu cu valoarea „Câmpulung Muscel", iar c_2, c_3, c_4 relațiile constante similare conținând tuplii cu valorile „Curtea de Argeș", „Târgoviște", respectiv „București"; în mod similar, fie c_5 relația peste atributul *An* conținând un singur tuplu cu valoarea 1310, iar c_6, c_7, c_8 relațiile constante similare conținând tuplii cu valorile 1369, 1418, respectiv 1558. (E7) se transformă astfel în:

$$\pi_{Domnitor,\ Capitala}(\pi_{Domnitor,\ Capitala,\ An}(\pi_{Domnitor,\ Capitala,\ An,\ DeLa}(\sigma_{DeLa \le An}(\sigma_{La \ge An}(DOMNII \bowtie (c_1 \bowtie c_5) \cup (c_2 \bowtie c_6) \cup (c_3 \bowtie c_7) \cup (c_4 \bowtie c_8)))))).$$

Aplicând procedura din demonstrația lemei 177, se obține următoarea expresie *CRD*:

$$\{Domnitor: x_1, Capitala: x_4 \mid \exists x_2(\exists x_3(\exists x_5((x_5 \ge x_2) \wedge (x_3 \ge x_5) \wedge DOMNII(Domnitor: x_1, Dela: x_2, La: x_3) \wedge ((x_4 = \text{„Câmpulung Muscel"} \wedge x_5 = 1310) \vee (x_4 = \text{„Curtea de Argeș"} \wedge x_5 = 1369) \vee (x_4 = \text{„Târgoviște"} \wedge x_5 = 1418) \vee (x_4 = \text{„București"} \wedge x_5 = 1558)))))\}$$

Cu ajutorul teoremei 175 și a lemelor 176 și 177, se poate imediat demonstra (vezi problema 105) principalul rezultat privind echivalența limbajelor relaționale de interogare:

Teorema 178 Următoarele limbaje sunt echivalente:

- Algebra relațională.
- Calculul relațional al domeniilor independent de domenii.
- Calculul relațional al domeniilor cu declarații de domeniu al valorilor.

Teorema 178 afirmă că două clase de limbaje definite independent unul de celălalt au aceeași putere expresivă și deci pot implementa aceeași mulțime de interogări; aceasta înseamnă că ambele sunt membre ale unei clase *robuste*. Teorema 178 a condus la următoarea noțiune de completitudine:

Definiția 179 Un *limbaj de interogare* se zice *complet* dacă este cel puțin la fel de expresiv ca algebra relațională.

Reamintim că o caracterizare abstractă a completitudinii (i.e. independentă de orice limbaj particular s-ar alege drept etalon) este cea oferită de teorema 154. Limitările acestei noțiuni de completitudine sunt discutate în secțiunea 7.4.

7.3.2 Calculul relațional al tuplilor

Așa cum am văzut deja la sfârșitul subsecțiunii 7.1, expresiile *CRD* sunt adesea greu de scris, în special din cauza numărului mare de variabile necesare; situația se înrăutățește și mai tare în cazul subclasei *CRD-DV*, care, în plus, necesită și declarații de domeniu al valorilor pentru toate aceste variabile. Evident că expresiile ar fi mai ușor de scris și de citit dacă ar avea mai puține variabile; ca atare, în secțiunea de față prezentăm o altă variantă a calculului relațional, numită *calculul relațional al tuplilor* (*CRT*), în care variabilele denotă tupli în loc de domenii. După cum se poate observa și în exemplul următor, cel mai adesea *CRT* necesită un număr mai mic de variabile decât

CRD: în interogările simple, este suficientă câte o variabilă per relație, plus încă una, eventual, pentru relația rezultat. Există însă şi un preț ce trebuie plătit pentru această simplificare: fiecărei variabile trebuie să i se asocieze o structură, pentru a indica mulțimea atributelor pe care este definit tuplul respectiv;[56] în plus, comparațiile necesită calificări pentru a indica atributele implicate.

CRT are şi mai multe variante decât *CRD*; prima dintre cele pe care le prezentăm în continuare se va dovedi cea mai utilă în demonstrarea rezultatelor de echivalență cu *CRD*.

Definiția 180 Expresiile CRT au forma $\{x(X) \mid f\}$, unde f este o formulă, x este o *variabilă tuplu* şi este singura variabilă liberă a f, iar X este mulțimea de atribute peste care este definit x. Valoarea unei asemenea expresii este o relație peste X conținând toți şi numai acei tupli care satisfac f atunci când sunt substituiți variabilei x.

Exemplul 68 O expresie *CRD* pentru calculul unei relații, precum *DOMNII* aparținând bd din exemplul 49, necesită o variabilă per atribut:

$\{Domnitor{:}x_1, DeLa{:}x_2, La{:}x_3 \mid DOMNII(Domnitor{:}\ x_1, DeLa{:}\ x_2, La{:}\ x_3)\}$.

Echivalentul ei în *CRT* are nevoie de o singură variabilă :

$$\{x(Domnitor, DeLa, La) \mid DOMNII(x)\}.$$

Şi alte tipuri de expresii *CRT* sunt mai simple decât echivalentele lor *CRD*; următoarea expresie *CRT* este echivalentă cu (E13) din exemplul 56:

(E27) $\{x_1(Domnitor, DeLa, La) \mid DOMNII(x_1) \wedge \forall x_2(Domnitor, DeLa, La)\ (\neg (DOMNII(x_2) \vee (x_2.DeLa \geq x_1.DeLa)))\}$.

Alte expresii *CRT* sunt însă la fel de sau chiar şi mai complexe decât echivalentele lor *CRD*, deoarece egalitățile trebuie explicit specificate şi pentru că variabila tuplu corespunzătoare rezultatului este definită pe o mulțime de atribute distincte de oricare dintre cele ale schemelor de relații implicate în expresie. De exemplu, expresia *CRT* echivalentă (E11) din exemplul 56 este:

(E28) $\{ x_1(D, M) \mid \exists x_2(D, DL, L)(DOMNII(x_2) \wedge (x_2.D = x_1.D) \wedge \exists x_3(N, M, Nec)\ (PERSOANE(x_3) \wedge (xN = x_1.D) \wedge (xM = x_1.M)))\}$.

Atomii *CRT* diferă de cei ai *CRD* doar datorită structurii diferite a valorilor referite de variabile. Atomii de primul tip sunt mai simpli, deoarece în ei apare o singură variabilă, în timp ce atomii de al doilea tip necesită o notație care să permită referințe la componentele variabilelor tuplu, deoarece, în general, comparațiile se fac la acest nivel.

Definiția 181 Atomii formulelor CRT pot avea doar una din următoarele două forme:

- $R(x)$, unde $R(X)$ este o schemă de relație, iar x este o variabilă definită peste X.
- $x_1.A_1\ \theta\ x_2.A_2$ sau $x_1.A_1\ \theta\ c$, unde: x_1, x_2 sunt variabile definite peste X_1, respectiv X_2; $A_1 \in X_1$, $A_2 \in X_2$; c este o constantă; iar θ este un operator de comparație.

Formulele se definesc ca în *CRD*; singura diferență este aceea că sintaxa cuantificatorilor necesită specificarea structurii variabilei legate corespunzătoare. Astfel, structura este definită pentru orice variabilă a expresiei, deoarece există o singură variabilă liberă (a cărei structură este declarată în lista țintă).

Demonstrația teoremei următoare, care confirmă faptul că *CRT* şi *CRD* au aceeaşi putere expresivă, este propusă cititorului spre rezolvare de problema 106.

Teorema 182 CRT şi *CRD* sunt echivalente.

O consecință imediată a teoremei 182 este aceea că nici *CRT* nu este independent de domeniu; ea implică însă faptul că *CRT-ID* este echivalent cu *CRD-ID* şi deci că problema independenței de domeniu a *CRT* este nedecidabilă şi că *CRT-ID* este echivalent cu algebra relațională (vezi problema 107).

Corolar 183 Următoarele limbaje sunt echivalente:

- Algebra relațională.

[56] De notat însă că aceasta nu implică şi necesitatea declarării vreunui domeniu de valori.

- Calculul relațional al domeniilor independent de domenii.
- Calculul relațional al domeniilor cu declarații de domeniu al valorilor.
- Calculul relațional al tuplilor independent de domenii.

Aşa cum am procedat şi în cazul *CRD*, în acest moment este rezonabil să căutăm a defini un *CRT-DV* cu aceeaşi putere expresivă ca şi *CRT-ID*, pentru a garanta astfel independența de domenii. Cel mai natural mod de a defini domenii de valori pentru variabilele tuplu este acela de a le asocia relațiilor corespunzătoare: fiecare variabilă poate lua valori numai dintre tuplii relației. În acest mod, declarațiile de domeniu al valorilor ar defini şi structura variabilelor, ceea ce nu ar îngreuna mult notația. De asemenea, deoarece domeniile de valori specifică apartenența tuplilor la relații, atomii de prima formă ($R(x)$) nu ar mai fi necesari în multe cazuri.

Singura problemă în legătură cu această abordare priveşte tuplul rezultat, care nu are neapărat structura vreuneia din relațiile ce apar în formula expresiei. Soluția acestei probleme este definirea unei *liste țintă*, similară cu cea din *CRD*, dar cu elemente de forma *B:x.A*, unde *x* este variabila tuplu, *A* este un atribut al unei relații din care *B* ia valori, iar *B* este un atribut al relației rezultat. Dacă *B* este acelaşi cu *A*, el poate fi omis (pentru simplificarea notației). Se pot de asemenea folosi subliste prescurtate care referă aceleaşi variabile: de exemplu, se poate scrie $B_1 \ldots B_k: x.A_1 \ldots A_k$ în loc de $B_1: x.A_1 \ldots B_k: x.A_k$. O formă şi mai concisă se poate folosi dacă toate atributele unei variabile apar în lista țintă şi, în plus, fără vreo redenumire: în acest caz, după numele variabilei se scrie doar simbolul *.

Sintactic, declarațiile de domeniu al variabilelor legate se definesc ca şi în *CRD*, adică în momentul cuantificării, iar al celor libere – într-o secțiune specială a expresiilor, zisă *lista domeniilor*, care precede formula.

Definiția 184 Expresiile CRT-DV au următoarea formă: $\{T \mid L \mid f\}$, unde:

- T este *lista țintă*, ale cărei elemente au forma completă *Y: x.Z* (unde *x* este o variabilă tuplu, iar *Y* şi *Z* sunt liste de atribute având acelaşi cardinal) sau formele simplificate *x.Z* sau *x.**;
- L este *lista domeniilor*, ale cărei elemente au forma $x(R)$, unde *x* este o variabilă tuplu, iar *R* numele unei relații;
- f este o formulă, ale cărei variabile libere sunt prezente o dată şi numai o dată în lista domeniilor, ai cărei atomi au forma $x_1.A_1 \ \theta \ x_2.A_2$ sau $x_1.A_1 \ \theta \ c$ şi care are domenii de valori asociate tuturor variabilelor cuantificate.

Exemplul 69 Interogarea formulată de expresia *CRT* (E27) din exemplul 68 poate fi formulată în *CRT-DV* astfel:

(E29) $\{x_1.* \mid x_1(DOMNII) \mid \forall x_2(DOMNII)\ (x_2.DeLa \geq x_1.DeLa)\}$.

Similar, interogarea formulată de expresia (E28) poate fi formulată în *CRT-DV* astfel:

(E30) $\{ x_1.D, x_2.M \mid x_1(DOMNII), x_2(PERSOANE) \mid x_2.N = x_1.D \}$.

Expresiile (E29) şi (E30) sunt evident mai uşor de citit decât expresiile echivalente (E27) şi (E28). Acest lucru nu este întâmplător: pentru foarte multe interogări, formularea în *CRT-DV* este mai simplă decât în orice altă formă de calcul relațional. Din acest motiv, limbajele de interogare folosite în cele mai populare SGBD relaționale (şi nu numai), *SQL* şi *Quel*, se bazează pe *CRT-DV*: ambele au liste țintă, declarații explicite sau implicite de domeniu al valorilor şi formule cu atomi fără apartenență.

Totuşi, *CRT-DV* nu este la fel de puternic ca celelalte limbaje de interogare prezentate anterior, deoarece declarațiile de domeniu ale sale nu sunt suficiente pentru a reprezenta interogări ale căror rezultate provin din diverse relații sau includ valori ce nu apar în conținutul bd; de exemplu, nu se poate formula în *CRT-DV* o expresie echivalentă (E1) din exemplul 47. Problema 108 cere tocmai propuneri de extensii ce ar trebui aduse acestui limbaj pentru a îl face echivalent cu limbajele echivalente din

corolarul 183, în timp ce problema 109 cere demonstrarea următoarei teoreme de caracterizare a puterii expresive a *CRT-DV*:

Teorema 185 CRT-DV este echivalent cu un limbaj algebric relațional de tip *AR*, care are toți operatorii acestuia mai puțin reuniunea și care permite drept operanzi doar relații ale bd.

Teorema 185 are și o mare importanță practică: ea explică de ce *SQL* are nevoie de operatorul „UNION".

7.4 Limitări ale limbajelor de interogare relaționale

Am introdus la sfârșitul subsecțiunii 7.1.2 noțiunea de completitudine a limbajelor de interogare, motivată de echivalența algebrei relaționale cu două subclase ale calculului relațional. Completitudinea este și astăzi considerată ca un minim obligatoriu pentru puterea de calcul a limbajelor de interogare ale SGBD; totuși, s-a descoperit rapid faptul că ea nu este și suficientă, deoarece mai sunt necesare multe alte caracteristici suplimentare pentru a rezolva destule probleme practice uzuale.

Prima observație care trebuie făcută este aceea că, de fapt, completitudinea se referă doar la puterea *selectivă* a unui limbaj de interogare și nu la puterea sa *de calcul*: așa cum am văzut în subsecțiunea 7.2.1, expresiile *AR* (și deci și cele *CR*) nu pot crea noi valori.

De aceea, limbajelor complete nu li se cere să fie capabile să exprime interogări ce ar include calcule asupra valorilor memorate de bd. Dintre cele mai uzuale asemenea tipuri de calcule, care au fost adăugate (într-o formă sau alta) atât *SQL*, cât și *Quel*, menționăm:

- ✓ Calcule asupra valorilor componentelor de tupli, ce pot fi definite cu ajutorul expresiilor (e.g. aritmetice, șir, boolene etc.) asupra fiecărui tuplu al relațiilor bd (fundamentale sau derivate); exemple: interogări calculând câți ani a domnit fiecare domnitor în fiecare domnie în parte sau domnitorii ce și-au încheiat domniile cu cel puțin 3 ani înainte de moarte. Invităm cititorul să proiecteze extensiile necesare *AR* și diverselor forme de *CR* pentru ca acestea să poată exprima asemenea tipuri de calcule.
- ✓ Calcule ce implică mulțimi de tupli, precum cardinalitatea (sub)domeniilor, sume, minime, maxime sau medii (numite generic *calcule agregate*).

Ambele clase de calcule de mai sus pot fi introduse fără mare dificultate și cu păstrarea echivalențelor între limbaje, atât în *AR*, cât și în diversele forme de *CR*; ca atare, nu ne vom mai ocupa de ele în continuare.

A doua limitare majoră a noțiunii de completitudine este existența de interogări cu sens pe care limbajele complete nu trebuie să fie capabile să le exprime, chiar dacă rezultatele lor conțin doar valori memorate de bd și sunt invariante la orice automorfism al conținutului acesteia. Exemplul următor prezintă cel mai celebru reprezentant al acestei clase de interogări:

Exemplul 70 Expresia *AR* (E31) calculează relația de succesiune tranzitivă a generațiilor de urmași asupra relației *r* din figura 71:

(E31) $R \cup \pi_{S,U}(\rho_{I1 \leftarrow U}(R) \bowtie \rho_{I1 \leftarrow S}(R)) \cup \pi_{S,U}(\rho_{I1 \leftarrow U}(R) \bowtie \rho_{I1,I2 \leftarrow S,U}(R) \bowtie \rho_{I2 \leftarrow S}(R))$.

Totuși, (E31) nu calculează succesiunea tranzitivă pentru orice legătură de succesiune: de exemplu, dată fiind relația *r* din figura 72, ea generează relația (din aceeași figură) ce nu conține tuplul indicând faptul că Bogdan III a fost un succesor indirect al Roman I, așa cum ar trebui.

Formalizăm în continuare o noțiune utilă în contextul exemplelor de tipul celor de mai sus.

Definiția 186 Se zice *închidere tranzitivă a unei relații binare* (finite) $r(AB)$ relația $r^+(AB)$ cu proprietatea $\exists t \in r$, $\exists \{t_1, \ldots, t_k\} \subset r$, $k > 0$, $t, t_1, \ldots, t_k$ tupli, astfel încât $t[A] = t_1[A]$, $t_i[B] = t_{i+1}[A]$, $\forall i$, $1 \leq i \leq k\text{-}1$ şi $t_k[B] = t[B]$.

Pe baza acestei noțiuni, se poate formula şi demonstra principala teoremă a acestei secțiuni. Până atunci, mai sunt însă necesare câteva noțiuni şi rezultate preliminare. Aşa cum am procedat şi în subsecțiunea 7.2.1, presupunem că domeniul nu este ordonat; ca atare, expresiile conțin doar operatorii de comparație „=" şi „≠". (Invităm cititorul să demonstreze, ca exercițiu, că această presupunere simplificatoare nu are ca efect pierderea generalității rezultatelor ce urmează).

Fie $\forall n>0$ şi $a_1, ..., a_n$ n constante distincte; fie $r_n(A_1A_2)$ o relație binară cu n-1 tupli $t_1, ..., t_{n-1}$ astfel încât $t_i[A_1] = a_i$, iar $t_i[A_2] = a_{i+1}$. r_n poate fi privită ca reprezentând un graf cu n noduri $a_1, ..., a_n$ şi n-1 arce $a_1a_2, a_2a_3, ..., a_{n-1}a_n$. Evident că, în terminologia grafurilor, închiderea tranzitivă a r_n conține un arc a_ia_j dacă şi numai dacă $i < j$. În cele ce urmează, introducem pas cu pas noțiunea de *formulă propozițională de calcul al grafurilor în forma normală disjunctivă*:

Definiția 187 În calculul grafurilor, se zice *atom* oricare dintre următoarele două tipuri de obiecte:

- *Atom de egalitate*: $x = a$ sau $x \neq a$, unde x este o variabilă (peste mulțimea nodurilor grafului), iar a $\in \{ a_1, ..., a_n\}$ o constantă; înțelesul atomilor de acest tip este acelaşi cu cel din *CRD*;
- *Atom de distanță*: $dist(x,y) = k$ sau $dist(x,y) \neq k$, unde x şi y sunt variabile, iar k natural. Atomul $dist(x,y) = k$ este *adevărat* pentru o substituție s dacă $s(x) = a_i$, $s(y) = a_j$, iar $j - i = k$. Interpretarea acestui atom în contextul grafurilor este aceea că distanța (în graf) între valoarea lui x şi cea a lui y este egală cu k; în caz contrar, desigur, $dist(x,y) \neq k$.

Definiția 188 Atomii care conțin egalitatea se zic *pozitivi*, iar cei care conțin neegalitatea se zic *negativi*.

Definiția 189 Se zice *clauză* orice conjuncție de atomi.

Definiția 190 Se zice *formulă propozițională de calcul al grafurilor în forma normală disjunctivă* (*FPFND*) orice disjuncție de clauze.

Observația 191 Evident că orice formulă propozițională (construită, în general, cu negații, conjuncții şi disjuncții) bazată pe atomi definiți de 187 poate fi redusă la o *FPFND* echivalentă (folosind reducerea binecunoscută din calculul propozițional matematic, care produce o disjuncție de clauze, fiecare din ele fiind o conjuncție de literali, care, la rândul lor, sunt atomi sau atomi negați; cum, în contextul grafurilor, există atât atomi pozitivi, cât şi negativi, este evident că putem elimina toate negațiile).

Lema 192 $\exists N > 0$ natural, astfel încât $\forall E$ expresie *CRD* şi $\forall n > N$, rezultatul aplicării lui E asupra r_n poate fi descris de $E(r_n)=\{x_1, ..., x_k | \psi\}$, unde ψ este o *FPFND*.

r

(E31)

Strămoş	*Urmaş*
Basarab I	Nicolae Alexandru
Nicolae Alexandru	Radu I
Radu I	Dan I

Strămoş	*Urmaş*
Basarab I	Nicolae Alexandru
Basarab I	Radu I
Basarab I	Dan I
Nicolae Alexandru	Radu I
Nicolae Alexandru	Dan I
Radu I	Dan I

Figura 71 Fragment al succesiunii Basarabilor şi rezultatul expresiei (E31) asupra ei

r

(E31)

Strămoş	*Urmaş*
Roman I	Alexandru cel Bun
Alexandru cel Bun	Bogdan II
Bogdan II	Ştefan cel Mare
Ştefan cel Mare	Bogdan III

Strămoş	*Urmaş*
Roman I	Alexandru cel Bun
Roman I	Bogdan II
Roman I	Ştefan cel Mare
Alexandru cel Bun	Bogdan II
Alexandru cel Bun	Ştefan cel Mare
Alexandru cel Bun	Bogdan III
Bogdan II	Ştefan cel Mare
Bogdan II	Bogdan III
Ştefan cel Mare	Bogdan III

Figura 72 Fragment al succesiunii Muşatinilor şi rezultatul expresiei (E31) asupra ei

Demonstrație: Fie expresia *CRD* $\{ x_1, ..., x_k \mid f \}$, unde numele atributelor sunt omise, iar singurele constante care apar în f sunt $a_1, ..., a_n$. Demonstrația folosește inducția matematică după structura formulei f.

1. (*nici un conector sau cuantificator logic*). Rezultă că f este un atom; dacă el este de forma $x = a$ sau $x \neq a$, atunci el poate figura direct în ψ; dacă el este de forma $x = y$ sau $x \neq y$, atunci el poate fi înlocuit cu $dist(x,y) = 0$ (respectiv $dist(x,y) \neq 0$); în sfârşit, dacă el este de forma $R(x,y)$, atunci el poate fi înlocuit cu $dist(x,y) = 1$.
2. *Inducție*. Dacă cea mai exterioară structură este construită doar cu negații, disjuncții şi conjuncții, f poate fi redusă la o *FPFND* conform observației 191. Rămân deci de analizat doar cuantificatorii; cum cel universal poate fi redus la cel existențial, este suficientă tratarea doar a acestuia din urmă. Fie expresia $\{x_1, ..., x_k \mid \exists x(f)\}$; prin ipoteza inducției, f poate fi înlocuit cu o *FPFND* $\psi = \psi_1 \vee ... \vee \psi_p$; ca atare, singurul lucru ce ar mai trebui făcut este eliminarea cuantificatorului din formula $\exists x(\psi_1 \vee ... \vee \psi_p)$. Conform observației 191, disjuncțiile pot fi reduse la o *FPFND*, deci este

suficient de eliminat cuantificatorul din $\exists x(\psi)$, unde ψ este o clauză. Pentru aceasta, distingem următoarele două cazuri:

I. (ψ *nu conține nici un atom pozitiv referind* x). În acest caz, putem înlocui $\exists x(\psi)$ cu ψ', unde ψ' este o clauză ce are aceeaşi atomi ca şi ψ, cu excepția celor care referă x. Demonstrăm că $\exists N > 0$ natural, astfel încât $\{x_1, ..., x_e | \exists x(\psi)\}$ este egal cu $\{x_1, ..., x_e | \psi'\}$, în raport cu r_n, $\forall n > N$: dacă un tuplu satisface $\exists x(\psi)$, atunci el satisface toți atomii care nu referă x şi deci satisface ψ'. Reciproc, dacă un tuplu satisface ψ', fie N numărul de atomi din ψ care referă x (toți negativi în acest subcaz!) şi fie $\forall n > N$; evident că $\exists\ a_h \in \{a_1, ..., a_n\}$ astfel încât ψ este *adevărat* dacă a_h este substituit lui x, deoarece fiecare atom care referă x interzice una dintre valorile lui x, deci N atomi interzic N valori, în timp ce toate celelalte n-N sunt permise.

II. (ψ *conține un atom pozitiv* α *referind* x). Atomul α forțează x să ia o anume valoare sau să aibă o anumită distanță fixată față de o altă variabilă. Rezultă că $\exists x(\psi)$ poate fi înlocuit cu ψ', unde ψ' conține atomii lui ψ care nu referă x precum şi câte unul sau mai mulți atomi pentru fiecare dintre atomii referind x, după cum urmează:
   - dacă α are forma $x = a_i$, atunci (furnizând doar demonstrația pentru atomii pozitivi, deoarece pentru cei negativi ea este trivial similară!):
     - $dist(x,y) = k$ devine $y = a_{i+k}$ (care, la rândul ei, devine *fals*, dacă $i+k > n$; modul de tratare al atomilor *adevărat* şi *fals* este discutat mai jos, la sfârşitul demonstrației);
     - $dist(x,y) = k$ devine $y = a_{i-k}$ (*fals*, dacă $i-k \le 0$);
     - $x = a_j$ devine *adevărat* dacă $i = j$ şi *fals*, dacă $i \ne j$.
   - dacă α are forma $dist(x,y) = k$, atunci (furnizând din nou doar demonstrația pentru atomii pozitivi şi presupunând că nu există nici un atom de forma $x = a_i$, căci atunci el ar putea fi tratat ca mai sus):
     - $dist(x,z) = h$ devine $dist(y,z) = h-k$ dacă $k \le h$, respectiv $dist(z,y) = k-h$ dacă $k > h$;
     - $dist(z,x) = h$ devine $dist(z,y) = h+k$.
   - dacă α are forma $dist(y,x) = k$, atunci transformarea este evident similară.
   - de asemenea, în cazul în care α are forma $dist(x,y) = k$ (sau $dist(y,x) = k$), atunci sunt adăugați clauzei k-1 atomi de forma $y \ne a_1, ..., y \ne a_{k-1}$ (respectiv $y \ne a_{n-k+1}, ..., y \ne a_n$).
   - în sfârşit, atomii constanți sunt înlocuiți cu ceva care are mereu aceeaşi valoare: *fals* este înlocuit cu $dist(x,x) \ne 0$, iar *adevărat* cu $dist(x,x) = 0$. q.e.d.

Pe baza noțiunilor de până acum şi a acestei ultime leme, se poate formula şi demonstra principala teoremă a acestei secțiuni care, deşi se referă la *CRD*, se aplică evident şi tuturor celorlalte limbaje echivalente cu acesta.

Teorema 193 Nu se pot exprima în *CRD* interogările care calculează închiderea tranzitivă a oricărei relații binare.

Demonstrație: Să presupunem prin absurd că $\exists E$ expresie *CRD* care calculează închiderea tranzitivă a relațiilor binare. Dacă alegem $\forall n > N$, unde N este numărul natural din enunțul lemei 192, rezultă că $E(r_n)$ poate fi exprimat ca $\{x_1, x_2 \mid \psi(x_1, x_2)\}$, unde ψ este o formulă propozițională. Este important de remarcat faptul că formula ψ este independentă de n, cu excepția atomilor $y \ne a_{n-k+1}, ..., y \ne a_n$ adăugați de lema 192 pentru eliminarea cuantificatorilor. Mai mult, numărul de atomi ai ψ este independent de n.

Fie r_n, cu n mai mare decât maximul dintre următoarele patru valori:

- N;
- Numărul de atomi ai ψ;
- $k+p+2$, unde k este cea mai mare constantă apărând într-un atom pozitiv de distanță al ψ, iar p este cel mai mare indice al constantelor a_p care apar într-un atom pozitiv de egalitate;
- valoarea minimă p astfel încât există o secvență de cel puțin $k+2$ valori a_p, ..., a_{p+k+1} cu proprietatea că nici una dintre ele nu figurează în vreun atom negativ de egalitate, unde k este cea mai mare constantă apărând într-un atom negativ de distanță al ψ.

Arătăm că $E(r_n)$ nu este egal cu închiderea tranzitivă r_n^+ a r_n. Sunt posibile două cazuri:

1. (*orice clauză a ψ are un atom pozitiv*) Fiecare clauză ori forțează una dintre variabile (x_1 sau x_2) să fie egală cu o constantă, ori „fixează distanța" între x_1 și x_2. Fie k cea mai mare constantă apărând într-un atom pozitiv de distanță al ψ, iar p cel mai mare indice al constantelor a_p care apar într-un atom pozitiv de egalitate al ψ; ca atare, a_{p+1} și a_{p+k+2} nu apar în nici un atom pozitiv de egalitate, iar $k+1$ nu apare în nici un atom pozitiv de distanță. Rezultă că perechea a_{p+1}, a_{p+k+2}, care aparține lui r_n^+ (de remarcat că $n \geq p+k+2$), nu aparține lui $E(r_n)$, deoarece nu satisface nici o clauză. În consecință, în acest caz, $E(r_n) \neq r_n^+$.
2. (*o clauză ψ' a ψ are doar atomi negativi*) Fie a_p, a_{p+k+1} două valori care nu apar în atomii negativi de egalitate, unde k este cea mai mare constantă apărând într-un atom negativ de distanță al ψ (aceste valori există din ipoteza de mai sus asupra lui n); evident că perechea a_{p+k+1}, a_p satisface ψ', deoarece ea nu violează nici unul din atomii lui ψ' (care sunt toți negativi) și deci aparține $E(r_n)$, deși nu aparține lui r_n^+. În consecință, și în acest caz, $E(r_n) \neq r_n^+$.

q.e.d.

Exemplul 71 Pe lângă câțiva atomi negativi de egalitate (care nu schimbă cu nimic înțelesul formulei în cauză), lema 192 generează pentru expresia (E32):

$$\{x_1,x_2 \mid R(x_1,x_2) \vee \exists y_1(R(x_1,y_1) \wedge R(y_1,x_2)) \vee \exists\, y_1(\exists y_2(R(x_1,y_1) \wedge R(y_1,y_2) \wedge R(y_2,x_2)))\},$$

care este echivalentă cu (E31) din exemplul 70, următoarea expresie:

(E33) $\{ x_1, x_2 \mid dist(x_1, x_2) = 1 \vee dist(x_1, x_2) = 2 \vee dist(x_1, x_2) = 3 \}$.

Aceasta confirmă faptul că expresiile calculului relațional pot specifica doar un număr fix de distanțe ale închiderii tranzitive și deci nu pot specifica această operație în general, pentru orice relație binară.

La o primă privire superficială, ar putea părea că teorema 193 contrazice teorema 154, deoarece rezultatul unei închideri tranzitive este întotdeauna invariant în raport cu orice automorfism al conținutului bd. Reamintim însă (vezi sfârșitul subsecțiunii 7.2.1) că teorema 154 se referă la interogări definite peste *un* conținut fixat și *NU* peste *toate* conținuturile posibile ale unei scheme.

Definiția 194 O *interogare* Q se zice *generică* dacă, pentru $\forall \mathbf{r}$ conținut de bd și $\forall h$ automorfism al $\mathbf{r}$, $h(Q(\mathbf{r})) = Q(\mathbf{r})$ (i.e. Q păstrează automorfismele oricărui conținut).

Corolarul 195 Pentru orice schemă de bd, $\forall \mathbf{r}$ conținut al ei și $\forall Q$ generică există o expresie de algebră relațională E astfel încât $Q(\mathbf{r}) = E(\mathbf{r})$.

Demonstrație: Fie o schemă de bd, un conținut al ei $\mathbf{r}$ și o interogare Q generică oarecare fixate. Deoarece Q este în general parțială, $Q(\mathbf{r})$ ar putea fi nedefinită; în acest caz, este suficientă alegerea unei expresii *AR* a cărei valoare este și ea nedefinită (de exemplu, fiindcă referă schema unei relații ce nu există în schema de bd). În caz contrar, deoarece Q este generică, $\forall h$ automorfism al $\mathbf{r}$, are loc, prin definiție, $h(Q(\mathbf{r})) = Q(\mathbf{r})$, deci $Q(\mathbf{r})$ este invariant la orice automorfism al $\mathbf{r}$. Conform teoremei 154, există însă în acest caz

o expresie *E* cu proprietatea că $Q(\boldsymbol{r}) = E(\boldsymbol{r})$.

q.e.d.

Desigur că pentru orice relație binară fixată există o expresie *AR* care îi calculează închiderea tranzitivă (deoarece închiderea tranzitivă este o funcție generică); teorema 193 afirmă însă că nu există nici o expresie *AR* care să calculeze închiderea tranzitivă a *oricărei* relații binare. Aceasta înseamnă deci că putem oricând scrie o expresie *AR* care să calculeze închiderea tranzitivă a unui conținut oarecare fixat al unei relații binare; există oricând însă alte conținuturi valide (mai „bogate") ale aceleiași relații, pentru care expresia respectivă nu va mai calcula corect închiderea tranzitivă (iar pentru a obține închiderea tranzitivă a acestora, este nevoie de câte o nouă expresie *AR* pentru fiecare asemenea conținut, toate distincte între ele; mai mult, evident, pe măsură ce conținutul unei asemenea relații se „îmbogățește", expresia *AR* corespunzătoare devine din ce în ce mai complicată!).

Mai mult, de remarcat și că nu există nici expresii *AR* care să calculeze, măcar pentru un conținut fixat al unei asemenea relații, al *n*-lea pas din calculul închiderii tranzitive, unde $n \geq 1$ este un parametru.

O noțiune de completitudine mai generală a fost definită de Chandra și Harel, care au pus accentul nu pe rezultatele interogărilor pentru un conținut fixat de bd, ci pe interogări în sine, privite ca funcții:

Definiția 196 Un *limbaj L* se zice *CH-complet* dacă, pentru orice schemă de bd și $\forall Q$ generică, există o *L*-expresie *E* astfel încât, $\forall \boldsymbol{r}$ conținut al schemei, $Q(\boldsymbol{r}) = E(\boldsymbol{r})$.

Exemplul canonic de limbaj *CH*-complet, propus tot de Chandra și Harel, este numit *QL*; informal, el include algebra relațională extinsă cu o instrucțiune de atribuire (și deci variabile peste relații), plus repetiția cu un test de ajungere la mulțimea vidă.

Trebuie remarcat în acest context că, de fapt, limbajele comerciale clasice *SQL* și *Quel* sunt cel mai adesea ori extinse în sensul *CH*-completitudinii[57], ori sunt *încapsulate* într-un limbaj de programare gazdă de nivel înalt, care asigură construcțiile de programare ce permit puterea de calcul completă[58].

De notat că limbajele bazate pe logica de ordinul întâi ce sunt prezentate în capitolul 5 (e.g. *DATALOG*), care au puterea de exprimare superioară algebrei relaționale, permit specificarea închiderilor tranzitive.

7.5 Valori nule în limbajele de interogare

Definițiile semanticii expresiilor *AR* și *CR* devin foarte complicate atunci când trebuie considerate și valorile nule, indiferent de tipul acestora.

Exemplul 72

Fie relația din figura 73 (cu cheia primară *#Nota* și în care, evident, produsul *#Student•#Examen* este cheie, iar *#Student* și *#Examen* sunt chei străine) și următoarele două expresii echivalente:

$\sigma_{Nota > 4}(NOTE)$, respectiv

$\{x_1.* \mid x_1(NOTE) \mid x_1.Nota > 4\}$.

Rezultatul acestor expresii trebuie, evident, să includă primul și să excludă ultimul tuplu al relației din figura 73; trebuie însă oare inclus și al doilea tuplu sau nu?

Dacă nulul este necunoscut, atunci el ține temporar locul unei valori de fapt cunoscute sau în curs de a fi cunoscută, ceea ce însă, evident, nu ne oferă nici o indicație pentru răspunsul de care avem nevoie (pentru că nu se poate încă ști dacă valoarea corespunzătoare este mai mare decât 4 sau nu).

---

[57] A se vedea, de exemplu, *Transact-SQL* (al *MS SQL Server*) sau *PL/SQL* (al *Oracle*).

[58] A se vedea, de exemplu, *VBA* (al *Access*).

Dacă nulul este de tip inexistent (inaplicabil), atunci tuplul corespunzător nu trebuie probabil inclus în rezultat.

Dacă nulul este lipsit de informație, atunci, evident, dilema este încă şi mai puțin rezolvabilă.

Mai mult, dacă considerăm şi următoarele două expresii, care, intuitiv, sunt echivalente celor de mai sus, răspunsul la întrebarea privind al doilea tuplu nu devine deloc mai clar, ci dimpotrivă:

$\sigma_{\neg(Nota \leq 4)}(NOTE)$, respectiv

$\{x_1.* \mid x_1(NOTE) \mid \neg(x_1.Nota \leq 4)\}$.

NOTE (*#Nota*, *#Student•#Examen*)

#Nota	*#Student*	*#Examen*	*Nota*
1	1	1	10
2	1	2	ϕ
3	2	1	4

Figura 73 Relația citată de exemplul 72

Exemplul 72 sugerează că, în prezența valorilor nule, logica booleană uzuală (i.e. cu două valori) nu este capabilă să asigneze expresiilor relaționale valori de adevăr rezonabile, pe baza cărora să putem decide dacă un tuplu trebuie sau nu să facă parte din rezultatul unei interogări.

Ca atare, s-a încercat inițial extensia teoriei relaționale a interogărilor prin folosirea unei logici ternare şi a unui aşa numit *principiu al substituției nulurilor* aferent celei de-a treia valori logice adăugate celor boolene şi anume: *necunoscut*. Informal, efectul ei este furnizarea în asemenea cazuri a două răspunsuri la orice interogare: primul (aşa-numita *versiune adevărată*) conținând tuplii ce satisfac „cu siguranță" predicatele logicii extinse, iar al doilea (aşa-numita *versiune posibilă*) conținând tuplii care „le-ar putea" şi ei satisface (fără însă a putea fi siguri de acest lucru).

Această abordare a condus la definirea a două versiuni pentru operatorul de selecție şi la o tratare sintactică a nulurilor pentru toți ceilalți operatori relaționali. Evident, cele două versiuni ale selecției corespund versiunii adevărate, respectiv posibile menționate mai sus.

Aşa cum era de aşteptat însă (cel puțin privind retrospectiv), această abordare şi-a vădit rapid inconvenientele majore induse. De exemplu, în mod trivial, următoarele două expresii sunt echivalente în contextul absenței nulurilor, dar echivalența se pierde în cazul apariției valorilor nule, deoarece rezultatul celei de-a doua expresii este în continuare egal cu întreg conținutul tabelei *NOTE* (deci ea este echivalentă cu aplicația identitate), în timp ce rezultatul primei nu include nici un tuplu având valoare nulă pentru atributul *Nota*:

(E34) $\sigma_{Nota > 4}(NOTE) \cup \sigma_{Nota \leq 4}(NOTE)$, respectiv

(E35) $\{x_1.* \mid x_1(NOTE) \mid (x_1.Nota > 4) \vee (x_1.Nota \leq 4)\}$.

S-a dovedit că acest lucru se datorează faptului că principiul substituției nulurilor este solid numai în contextul *CR* şi nu şi în cel al *AR*. Mai mult, s-a observat imediat şi că aplicarea principiului substituției nulurilor în *CR* este extrem de laborios, întru-cât numărul substituțiilor necesare creşte exponențial cu numărul aparițiilor de nuluri.

În consecință, s-a încercat o altă abordare a acestei probleme, bazată în esență pe impunerea de constrângeri asupra nulurilor, pentru a restrânge posibilele substituții corespunzătoare. De exemplu, revenind la figura 73, proiectantul bd ar trebui să specifice pentru cel de-al doilea tuplu, într-o coloană separată dedicată acestui scop, numai una dintre următoarele valori: *adevărat*, *fals*, > 4 sau ≤ 4. Evident că această abordare permite surmontarea inconvenientelor celei anterioare, cu excepția faptului că şi ea tratează doar nulurile de tip necunoscut (i.e. nu şi nulurile de celelalte două tipuri).

La fel de evident este însă că şi această abordare prezintă propriile inconveniente, deloc neglijabile şi anume:

- numărul de constrângeri necesare poate fi enorm
- în timp, constrângerea asociată unei valori nule ar putea trebui să fie modificată
- unora dintre valorile nule s-ar putea să le fie necesare constrângeri ale căror expresii sunt foarte complicate (pentru a putea fi corect tratate din punctul de vedere al tuturor interogărilor implicate şi nu doar a uneia sau unora dintre ele)
- gestionarea acestor constrângeri ar putea fi foarte complicată (de exemplu, pentru a depista eventualele mulțimi de constrângeri incoerente sau constrângerile tautologice, precum, de exemplu, $\phi_1 > 4 \vee \phi_1 \leq 4$, problemă despre care se ştie că, în general, este computațional intratabilă).

Meritul ei incontestabil însă a fost acela că, atât comunitatea ştiințifică din domeniu, cât şi proiectanții SGBD relaționale s-au convins cu această ocazie că problema nulurilor este una a cărei semantică nu se pretează la pure manipulări sintactice şi că „maximum sintactic" ce poate fi oferit proiectantului de interogări relaționale este îmbogățirea *AR* şi *CR* cu un predicat unar atomic capabil a testa dacă o valoare este sau nu nulă.

În această abordare, se oferă un atom *NULL*(*A*), care pentru atributul *A* şi tuplul *t* are valoarea *adevărat* dacă *t*[*A*] este o valoare nulă, respectiv *fals* în caz contrar, iar valorile nule sunt considerate drept simboli speciali, care nu satisfac nici un alt predicat boolean.

De exemplu, înțelesul unui atom de tip $A\theta B$ este „*t*[*A*] şi *t*[*B*] nu sunt valori nule, iar *t*[*A*] este în relația θ cu *t*[*B*]", iar negația unui atom de tipul *A*=*B* este: $A \neq B \vee NULL(A) \vee NULL(B)$. Evident că, în cazul particular al expresiilor (E34) şi (E35) de mai sus, de fapt, sensul dorit este cel formalizat de expresiile (E36), respectiv (E37) (care, în plus, sunt astfel echivalente între ele atât în absența, cât şi în prezența nulurilor):

(E36) $\sigma_{Nota > 4}(NOTE) \cup \sigma_{Nota \leq 4}(NOTE) \cup \sigma_{NULL(Nota)}(NOTE)$, respectiv

(E37) $\{x_1.* \mid x_1(NOTE) \mid (x_1.Nota > 4) \vee (x_1.Nota \leq 4) \vee NULL(Nota)\}$.

Avantajele acestei abordări sunt evidente:

- nu este nevoie de înlocuirea logicii boolene cu una ternară
- nu este nevoie de vreun principiu de substituție a nulurilor
- nu este nevoie de constrângeri suplimentare atașate valorilor nule
- nu este nevoie de nici un fel de distincție între cele trei tipuri de valori nule.

Ca atare, toate SGBD relaționale implementează acest tip de soluție, iar standardul *ANSI SQL* [390] include şi funcția booleană corespunzătoare *IsNull*. De remarcat în plus că acest predicat oferă şi posibilitatea calculării mult mai elegante a operatorului algebric de diferență între mulțimi.

(E.1.) [Ull]: Fie schema de relație OAC, unde O = Oraşe, A = Adrese, C = Coduri-poştale, cu dependențele funcționale (df) OA→C (orice adresă dintr-un oraş are un singur cod poştal) şi C→O (orice cod poştal este asignat unui singur oraş, dar nu neapărat şi unei singure adrese din acle oraş – unde prin adresă se înțelege, de exemplu strada şi numărul).

Schema are cheile OA şi AC, este în FN3, dar nu şi în FN Boyce-Codd (FNBC). Schema admite descompuneri în FNBC care au proprietatea jfp dar nici una care să păstreze dependența OA→C. Descompunerile ei nefiind deci practic interesante, vom analiza doar schema originală. Fie următorul conținut al acesteia (perfect posibil în raport cu cheile tabelei !):

Oraşe	*Adrese (str., nr)*	*Cod*
Sibiu	Carpaților, 10	2400
Iaşi	Ştefan cel Mare, 12	2400

Acest conținut este evident aberant (două oraşe nu pot avea acelaşi cod). Construirea unui asemenea contraexemplu este posibilă deoarece specificarea cheii AC este inutilă (ea neputând impune df C→O) iar forțarea cheii C imposibilă (căci C nu este injectiv).

Am arătat în [Ma1] de ce *MRD nu poate oferi nici o soluție pentru acest exemplu*: este nevoie de o constrângere suplimentară, de tip comutativitate de diagrame de funcții, de neexprimat deci în termeni relaționali. Între timp am rafinat şi mai mult [Ma2] soluția oferită cu acel prilej, distingînd între oraşele cărora li s-a asignat un singur cod (indiferent de adrese) şi cele care au fost „zonate" pe (porțiuni de) străzi sau chiar imobile (cărora li s-a asignat cîte un cod în parte). E adevărat că această soluție e mai „complicată", căci conduce la o schemă datalogică cu 3 tabele şi 3 constrângeri explicite (în afara cheilor), în loc de 2, respectiv 1 (deoarece trebuie impusă corespunzător partiționării oraşelor, şi partiționarea codurilor poştale). Dar ea prezintă două avantaje suplimentare deloc neglijabile: pe lîngă o reflectare mult mai acurată a semanticii modelate (ceea ce simplifică mult formularea de întrebări asupra conținutului bd) şi reprezentarea datalogică este mult mai naturală – căci nu se repetă fiecare oraş „nezonat" de sute de ori (pentru fiecare adresă a sa în parte) şi nici nu este obligatorie memorarea adreselor din aceste oraşe (lucru care poate nici nu interesează!).

(E.2.) [Ull]: Fie schema de relație CPOASG unde C = Cursuri, P = Profesori, O = Ore, A = Amfiteatre, S = Studenți, G = Grade, cu df C→P (fiecare curs este ținut de un singur profesor), OA→C (într-un amfiteatru nu se pot ține simultan mai multe cursuri), OP→A (un profesor nu se poate afla simultan în mai multe amfiteatre), OS→A (un student nu se poate afla simultan în mai multe amfiteatre), şi CS→O (absolvind un curs, fiecare student obține cel mult un grad).

Schema are doar cheia OS iar df listate constituie o acoperire minimală. Schema admite descompunerea „frumoasă" CSG (cu cheia CS), CP (cu cheia C), COA (cu cheile CO şi OA), COS (cu cheia OS) care este în FNBC, are proprietatea jfp, dar nu păstrează nici OP→A, nici OS→A. (Sunt posibile şi alte descompuneri rchivalente cu aceasta, dar care „nu se suprapun atît de bine intuiției noastre despre ce anume informații ar trebui tabulate în bd" [Ull] sîc!). Schema admite însă şi descompunerea CSG, CP, COA, POA (cu cheile PO şi OA), SOA (cu cheia SO) care este în FN3, păstrează toate dependențele şi are şi proprietatea jfp. Se afirmă că adăugarea la schemă a dependenței multivaluate (dmv) C->>OA „permite, împreună cu df de mai sus, derivarea tuturor dmv pe care le-am simți intuitiv ca fiind valide" [Ull]. Astfel îmbogățită, schema admite o descompunere cu proprietatea jfp în FN4 (şi anume COA, CP, CSG) dar care nu păstrează nici ea dependențele OP→A şi OS→A. (De remarcat în

plus că, numai şi numai pentru a avea proprietatea jfp, descompunerea în FNBC necesită – suplimentar faţă de cea în FN4 – şi tabela COS !).

Ne rămâne deci de analizat, din nou, doar descompunerea în FN3 (celelalte permit în mod evident conţinuturi invalide ale bd, din moment ce nu păstrează toate dependenţele). Fie următorul conţinut al acestei bd:

C	*S*	*G*
C1	s1	g1
c1	s2	g2
c2	s1	g2

C	*P*
c1	p1
c2	p1

C	*O*	*A*
c1	o1	a1
c1	o2	a1
c2	o1	a2
c2	02	a2

P	*O*	*A*
p1	o1	a2
p1	o2	a3
p2	o1	a3

S	*O*	*A*
s1	o1	a3
s2	o2	a2

Invităm cititorul să verifice că fiecare tabelă în parte conţine doar tupli valizi în raport cu cheile ei. Totuşi, acest conţinut al bazei de date este aberant: „orarul” memorat de COA obligă p1 să se afle simultan în a1 şi a2, atît la o1 cât şi la o2 (şi cum, conform POA, la o1 el s-a aflat în a2, rezultă că c1 a rămas descoperit; mai mult, atît c1 cît şi c2 au fost descoperite la o2, căci p1 figurează în a3 la această oră !); similar, s1 ar fi trebuit să se afle la o1 simultan în a1 şi a2, dar el a ales – sau a nimerit, puţin importă – a3, tot aşa precum, la aceeaşi oră, s2 ar fi trebuit să fie în a1 dar a fost în a2!).

Se poate observa că, în general:

- COA împiedică într-adevăr ca într-un amfiteatru să fie programate simultan mai multe cursuri şi ca un acelaşi profesor să fie programat simultan în mai multe amfiteatre; în schimb, COA permite memorarea informaţiei că un acelaşi profesor (student) ar trebui să se afle simultan în mai multe amfiteatre.
- Poa împiedică într-adevăr ca mai mulţi profesori să figureze simultan într-un acelaşi amfiteatru şi ca un acelaşi profesor să figureze simultan în mai multe amfiteatre, dar POA nu poate obliga profesorii să fie – la orele prevăzute de conţinutul COA- în vreunul din amfiteatrele în care ar trebui să-şi ţină cursurile (eventual simultan – sîc!).
- SOA împiedică într-adevăr ca vreun student să figureze simultan prezent în mai multe amfiteatre, dar nu îl poate obliga ca – la orele prevăzute de conţinutul COA- să se afle măcar într-unul din amfiteatrele în care sunt programate cursuri la care el este înscris).

În concluzie, ca şi în cazul (E.1.) de mai sus, nici un conţinut al vreunei bdr pentru acest exemplu nu va fi cu adevărat valid decât pur întâmplător. (Din punct de vedere relaţional, chiar şi conţinuturile analizate de noi sunt valide: joinul tuturor tabelelor elimină orice tuplu aberant ar conţine vreuna din ele, iar deiniţia proprietăţii jfp nu porneşte de la conţinutul descompunerii ci de la cel al „relaţiei universale”). Deci nici pentru acest exemplu nu există soluţie relaţională corectă.

Dar cel mai grav lucru în legătură cu acest exemplu este faptul că problema propusă nu este nici complet, nici precis formulată şi că, în general, ea nu are soluţie (matematică, darmite de modelare!) !!

Într-adevăr, să presupunem că orice student se poate înscrie simultan la orice cursuri (aşa cum reiese implicit din context, în lipsa unei restricţii în acest sens). Este uşor de verificat că, în general, nu îi putem garanta că va putea participa la toate orele lor: îl puem cel mult informa despre ce posibilităţi de alegere are în orice moment. Ceea ce înseamnă că, în acest caz, SOA devine inutilă! E drept că, în principiu, ea ar putea memora unde anume s-a aflat fiecare strudent la orice oră: dar, dacă impunem constrângerea explicită ca un student să nu poată participa decât la cursurile la care este înscris (şi anume: $\forall\ o\in O\ \forall\ s\in S\ \forall\ a\in A\ \exists\ c\in C\ \exists\ g\in G\ (\varphi1(o,a)=c \wedge \varphi2(c,s)=g)$, unde $\varphi1: \mathcal{G}1\rightarrow C$, $\varphi2: \mathcal{G}2\rightarrow G$ şi $\mathcal{G}\ 1\subseteq O \times A$, $\mathcal{G}\ 2\subseteq C \times S$), atunci nu vom putea memora, de exemplu faptul că în realitate s1 a fost la o1 în a3; iar dacă nu impunem această

constrângere, bd tot nu ne va putea ce cursuri a frecventat de fapt un student (căci joinul natural între OSA şi COA nu conține, de exemplu nici o informație despre s1)!

Probleme la fel de mari sunt şi cu alcătuirea orarului, chiar în ipoteza simplificatoare că nu ne interesează faptul că unui student anume i se oferă sau nu şansa de a putea frecventa regulat toate cursurile la care s-a înscris. Şi în acest caz este uşor de verificat că, dacă ne propunem ca obiectiv principal folosirea spațiului disponibil (amfiteatrele), atunci POA devine inutilă: dacă dorim concomitent ca orice oră de curs să fie acoperită cu un porfesor (i.e. impunem constrângerea explicită ediționalâ: $\forall\ o\in O\ \forall\ p\in P\ \forall\ a\in A\ \exists\ c\in C\ (\varphi 3(c)=p \wedge \varphi 1(o,a)=c)$, unde $\varphi 3: O\rightarrow P$), atunci tot nu vom putea memora, de exemplu, informația că în realitate p1 a fost în a3 la o2; iar dacă nu avem această prtenție, atunci oricum bd nu ne poate spune ce cursuri a ținut de fapt un profesor (căci joinul natural între COA şi POA nu conține, de exemplu, nici o informație despre p2)! Mai mult, în primul din aceste subcazuri vom fi probabil adeseori obligați să angajăm de urgență noi profesori şi / sau să plătim degeaba salariul unora dintre ei!!

Similar, dacă ne stabilim ca obiectiv principal încărcarea profesorilor disponibili, atunci COA devine inutilă (căci putem obține „orarul" din POA şi CP). Mai mult, dacă dorim în concomitent să putem efectiv ține toate cursurile, atunci – în general – sau vom fi obligați să construim (închiriem) noi amfiteatre (ziua neavând decât 24 de ore!) sau vom ține unele goale! Ca să nu mai vorbim despre faptul că este perfect posibil ca, ăn acest caz, să programăm cursuri la care nu s-a încris nici un student!!

Spațiul nepermițându-ne dezvoltarea mai multor considerații pe marginea acestui exemplu, reținem doar concluzia acestei analize: *mai grav decât oferirea unei soluții greşite la o problemă, mai grav încă decât încercarea de a soluționa o problemă cu mijloace teoretice inadecvate, ni se pare formularea ambiguă, incompletă a problemei şi, mai ales, „ repezirea cu capul înainte" în rezolvarea ei, fără a studia în prealabil dacă şi în ce condiții ea are sau nu soluții.* Considerăm că morala acestui „fiasco" relațional trebuie să fie următoarea: nu orice manipulare sintactică, oricât de „solid" ar fi ea justificată de o teorie de nivel sintactic, nu poate produce decât întâmplător o reprezentare corectă a semanticii de la care s-a pornit! Degeaba definiții de noi concepte, degeaba algoritmi, degeaba teoreme – oricât de frumoase în sine! – atunci când premizele abordării sunt „firave"!! Aceste triste constatări, care ar trebui să pună pe gânduri în mod serios pe orice „fan" al modelării relaționale, sunt cu atât mai elocvente cu cît aproximativ o jumătate din cap. 5 din [Ull] (dedicat proiectării bdr) gravitează tocmai în jurul acestui exemplu!

(E.) [Ull]: Fie schema de relație CSPA, unde C = Cursuri, S = Studenți, P = Premize, A = Ani, cu df $SP\rightarrow A$ (un student a absolvit o premiză într-un singur an) şi dmv „încastrate" („embedded") C->>S|P (un student se poate înscrie la mai multe cursuri într-un singur an; un curs poate fi condiționat de absolvirea mai multor premize; iar absolvirea unui curs poate constitui o premiză a înscrierii la mai multe cursuri. Menționăm că printr-o premiză se înțelege tot un curs.). Schema are doar cheia CSPA şi admite descompunerea în FN4 CS, CP, SPA (cu cheia SP) care are proprietatea jfp şi şi păstrează toate dependențele.

Am arătat [Ma1,Ma2] că *dmv încastrate nu se impun obiectiv: „necesitatea" lor este un rezultat al slăbiciunilor MRD, atât conceptul de dmv, cît şi presupunerea existenței unor „relații universale" sunt nejustificat de „tari".* Acest lucru este trivial verificabil şi în cazul particular al exemplului de față. Trebuie recunoscut cu onestitate faptul că explicația ce am dat mai sus C->> S|P modelează acurat „realitatea" considerată şi că această explicație definește pur şi simplu două relații matematice – una peste C şi S, cealaltă peste C şi P. Lăsînd la o parte modul „transcendent" în care se poate ajunge, în general, la concluzia că unei scheme relaționale trebuie să i se adauge o anumită dmv încastrată (singura indicație că ar putea fi eventual nevoie de asemenea

dependențe fiind furnizată doar sintactic, de algoritmul de testare a proprietății jfp!), să arătăm doar că şi această schemă permite conținuturi aberante. Fie, de exemplu:

C	*S*
c1	s1

C	*P*
c1	c2
c1	c3
c2	c5
c5	c1

S	*P*	*A*
s1	c4	a1

Fiecare conținut în parte este valid în raport cu cheile tabelei respective. Se observă însă imediat că acest conținut al bd este aberant, căci s1 nu ar trebui să aibă dreptul să se înscrie la c1 (deoarece el nu a absolvit nici una din premizele c2 şi c3 necesare acestuia). Cauza este deci de aceeaşi natură cu cea semnalată pentru exemplele de mai sus: proprietatea jfp este prea „slabă" şi nu poate împiedica memorarea de tupli aberanți în tabelele descompunerii (ba, lucru şi mai grav, joinul natural ce intervine în definirea acestei proprietăți disimulează existența unor asemenea tupli!). Pe scurt, „necazul" cu această descompunere este că CS nu poate împiedica înscrierea la un curs studenților care nu şi-au luat premizele mecesare acelui curs.

Acesta nu este însă singurul necaz cu această schemă relațională: ea nu poate impune nici condiția ca mulțimea premizelor oricărui curs să fie inclusă în mulțimea cursurilor (de exemplu, c4 din SPA, dar nu numai, de unde putem şti oare că c3 din CP este la rândul lui curs?). Şi mai grav este însă faptul că o asemenea schemă nu poate împiedica condiționările circulare de cursuri! (Mai bine deci că MRD nu este capabil să exprime constrângerea ce ar interzice studenților înscrierea la cursurile pentru care nu şi-au luat premizele necesare căci, în acest caz, nici un student n-ar reuşi vreodată să se înscrie la c1, c2 sau c5!!).

(E.4.) [ZM]: Fie schema de relație NFMAVPCID, unde N = Număr înmtriculare, F = Fabricant, M = Model, A = An, V = Valoare, P = Proprietar, C = Carnet, I = Incalcare lege circulație, D = Data încălcare, cu dependențele:

D.1 N→F
D.2 N→A
D.3 N→M
D.4 N→V
D.5 N→P
D.6 N→C
D.7 N→>ID
D.8 FAM→V
D.9 FAM→>NPCID
D.10 P→C
D.11 P→>ID
D.12 P→>NFAMV
D.13 C→P
D.14 C→>ID
D.15 C→>NFAMV

Credem că cititorul este de aceiaşi părere cu noi: în fața unei asemenea lite de dependențe, înainte de a anailza soluție relațională a acestei probleme este preferabilă analizarea puterii ei! Numărul mare şi complexitatea unora dintre dependențele asertate (de câtă imaginație, de câtă muncă o fi fost nevoie pentru abstractizarea D9, D12 şi D15!) ne pun pe gânduri: ce vrea să spună fiecare în parte ? sunt oare toate justificate?

sunt ele consistente? modelează ele oare acurat „realitatea” dorită? Singurele informații suplimentare furnizate de autori [ZM] sunt următoarele: se doreşte ca bd să ofere date despre:

- fabricantul, modelul, anul de fabricație şi proprietarul fiecărei maşini înmatriculate
- valoarea curentă a fiecărui tip de maşină
- carnetul de conducere al oricărei persoane, precum şi posesorul oricărui carnet
- istoria tuturor încălcărilor legii circulației (cod contravenție, data contravenției) de către orice şofer.

Cu D1 la D5 putem fi de acord la prima vedere: orice maşină înmatriculată este de un unic model, a fost produsă într-un unic an de către un unic fabricant, are o unică valoare şi se află în posesia unui unic proprietar. D6 ne stârneşte însă primele dubii: în ce sens îi asociem unei maşini un unic carnet de conducere? În general, ea poate fi condusă de mai mulți şoferi (chiar la un acelaşi drum!). Este oare vorba despre carnetul proprietarului ei (care este, într-adevăr unic)? Dacă da, de ce nu îl asociem proprietarului – urmând ca, la nevoie, să stabilim legătura între maşină şi carnetul acestuia prin tranzitivitate? (Lucru „canonic” şi în limbaj natural, căci întrebarea „Ce carnet de conducere are maşina m?” este lipsită de sens, şi nimeni nu ar ghici că, de fapt, am vrut să întrebăm „Ce carnet de conducere are proprietarul maşinii ?).

D7 este şi mai ciudată: oare chiar se doreşte memorarea istoriei tuturor contravențiilor săvârşite la volanul tuturor maşinilor? (căci nu n putem închipui totuşi că maşinile încalcă singure vreo lege!). Şi dacă este aşa, nu ar fi mai normal să memorăm şi şoferii care le-au comis? (sau se doreşte cumva interzicerea dreptului de circulație al maşinilor la volanul cărora s-au comis prea multe contravenții? sîc!).

D8 ne obligă la prima revenire asupra acceptului dat inițial dependențelor D1 la D7. În mod clar, D4 a fost listată doar din dorința de a aserta cât mai multe dependențe (toate?)! Ea este desigur redundantă (şi nu doar semantic: şi sintactic ea poate fi derivată din D1, D2, D3 şi D8!). Mai mult, simțim în acest moment nevoia abstractizării unui nou obiect de modelat în această bd: tipul unei maşini. Definim un tip de maşină ca fiind clasa de echivalență a tutror maşinilor de un acelaşi model, produse într-un acelaşi an de către un acelaşi fabricant. Relațional, putem modela aceasta considerând atributul T = Tip şi înlocuind D1 la D4 şi D8 cu: $N \rightarrow T$, $T \rightarrow V$ şi $T \leftrightarrow FAM$ (unde $X \leftrightarrow Y$ constituie o abreviere pentru $X \rightarrow Y$ şi $Y \rightarrow X$).

Chiar înlocuind FAM cu T în partea stângă a D9, vom fi totuşi obligați să respirăm adânc (îndelungi minute!) după terminarea „rostirii înțelesului” ei: oricare ar fi maşinile m1 şi m2 de un acelaşi tip t, există o maşină m3 astfel încât ori ea are acelaşi număr de înmatriculare, acelaşi proprietar, care are acelaşi carnet (sîc!) şi la volanul ei s-au săvârşit aceleaşi contravenții, la aceleaşi date ca şi pentru m1 şi, în acelaşi timp, ea este ce acelaşi tip şi a costat la fel cu m2 (sîc!), ori ea (m3 pentru cei ce au uitat deja de unde am plecat!) are acelaşi număr de înmatriculare, acelaşi proprietar, care are acelaşi carnet (re-sîc!) şi la volanul ei s-au săvârşit aceleaşi contravenții, la aceleaşi date ca şi pentru m2 şi, în acelaşi timp, ea este de acelaşi tip şi a costat la fel (re-sîc!) cu m1. Mărturisim că, ajunşi prima oară în acest punct al analizei de față, nu ne-am putut abține un hohot de râs moromețian! Să fi fost de vină abordarea relațională a modelării datelor în general? sau, în particular, acest nefericit concept zis dmv? sau această utilizare a lui de către autori [ZM]? Probabil toate laolaltă!

Am pomenit deja (la (E.)) că dmv este inutil de „tare” şi că mult mai potrivită ni se pare utilizarea în loc a conceptului consacrat de relație matematică. Ridicându-ne corespunzător de la nivelul relațional la cel semantic, matematic, fraza de mai sus ar vrea să ne spună că fiecărui tip de maşină i se asociază o mulțime de maşini de acel tip, fiecare dintre acestea aflându-se în posesia câte unui proprietar, care proprietari au fiecare câte un carnet de conducere şi au săvârşit contravenții la diverse date. Trebuie

recunoscut că, şi la acest nivel, asertarea D9 este tot ridicolă: ea ne spune în fond doar că surjecţia canonică N→T nu este injectivă! (ceea ce, este drept, ne întăreşte încă o dată convingerea că am făcut bine să tipizăm maşinile şi că relaţia de echivalenţă considerată cu acel prilej nu aduce cu sine doar avantaje sintactice, de eleganţă a formalizării, ci chiar corespunde semanticii „realităţii" de modelat). Acest lucru fiind trivial în general, el nu merită explicitat (nu ne-ar ajunge o viaţă întreagă să caracterizăm un singur obiect dacă ne-am apuca să înşirăm tot ceea ce el ar putea fi dar nu este!).

D10 confirmă presupunerea ce am făcut cu ocazia analizei D6 (şi cum nimic de aici încolo nu o va infirma, vom putea elimina liniştiţi şi D6 din formularea problemei: atât semantic cât şi sintactic ea se deduce tranzitiv din D5 şi D10!). D11 ne spune că, în general, un proprietar de maşină poate încălca legea circulaţiei de mai multe ori, eventual în acelaşi mod, dar la diverse date. Să admitem ipoteza simplificatoare că nu ne interesează cazurile în care un acelaşi şofer a încălcat această lege în acelaşi mod de mai multe ori într-o aceeai zi (în caz contrar, n-am avea decât să memorăm asociat şi ora contravenţiei). Dar de ce oare „proprietar" şi nu „şofer", aşa cum este cazul nu numai în realitate, ci şi în specificaţiile informale a problemei din [ZM]? Mai ales că, probabil, chiar şi la dânşii poţi fi şofer fără a fi proprietar de maşină, şi viceversa (măcar temporar!).

În consecinţă, noi am abstractiza şi submulţimea ŞOFERI (unde, evident, într-un model al datelor mai evoluat, atât ŞOFERI cât şi PROPRIETARI ar fi incluse într-o mulţime de OAMENI). Identificând elementele acestei noi submulţimi prin intermediul injecţiei S = Şoferi, putem înlocui D11 prin S→>ID aşa cum este normal (sau, eventual, prin SN→>ID dacă interesează realmente şi maşinile la al căror volan s-a comis fiecare contravenţie în parte). Similar, D10 se poate înlocui prin mult mai naturala S→C. De remarcat în plus că oricum, chiar dacă rescriem astfel D10 şi D11 sau nu, ridicola D7 devine inutilă semantic de îndată ce considerăm D11- şi vom renunţa deci şi la ea (sigur că sintactic ea se poate deduce din D5 şi D11!).

Invităm cititorul să analizeze D12 în detaliu singur (nouă ne-a ajuns D9!!); semantic, ea ne spune că un proprietar poate poseda simultan mai multe maşini (nu neapărat de acelaşi tip). Şi această informaţie este însă trivială: dacă nu asertăm şi P→N, urmează în mod evident că N→P nu este injectivă (în general, relaţia inversă unei funcţii nu este funcţională!). Deci nu este nevoie nici de D12.

D13 trebuie înlocuită cu C→S. Chiar dacă nu facem această înlocuire, având în vedere şi D10, rezultă oricum că D14 şi D15 sunt la rândul lor inutile atât semantic cât şi sintactic (prima, în virtutea D11; cea de a doua, în virtutea D12 – care tocmai am văzut că este inutilă în virtutea D5!). Şi uite aşa, din inutilitate în inutilitate, s-au scris 15 dependenţe din care am putut „alege2 doar 8!!

Recapitulând, din punct de vedere semantic (i.e. al modelării) D4, D6, D7, D9, D12, D14 şi D15 sunt inutile. Şi, în mod evident, nu MRD în sine este de blamat pentru aceasta. trebuie să ne punem atunci întrebarea dacă măcar din punct de vedere sintactic (i.e. al proiectării bd) ele nu sunt cumva necesare (ceea ce ar fi în defavoarea metodologiei de proiectare propusă de autori [ZM], dar i-ar scuza din punct de vedere al modelării). Cititorul poate însă verifica că algoritmul de proiectare propus în [ZM] nu are nici el nevoie de aceste dependenţe (ci, aşa cum ne-am fi şi aşteptat, le elimină pe toate – cu un consum desigur inutil de timp şi resurse calculator!). Lucru de altfel vizibil şi în descompunerea obţinută ca soluţie, descompunere în care nici una din aceste dependenţe nu se reflectă.

Desigur că apare legitimă întrebarea la ce bun să complici lucrurile într-atît (şi într-un asemenea hal!) încât aproape să dublezi în mod inutil formularea unei probleme „de jucărie", atât de simplă în fond? Cum am mai putea stăpâni modele „adevărate" cu mii de atribute şi sute de dependenţe [Ma1] dacă am proceda la fel? Este oare indicată asertarea tuturor dependenţelor ce ne „trec prin cap"? Sau, şi mai rău, trebuie oare să

încercăm toate combinațiile posibile de atribute pentru a lista închiderea tranzitivă a dependenşelor existente (sîc!)? Desigur că răspunsul la aceste ultime două întrebări trebuie să fie (categoric!) negativ: *militäm pentru o analiză aprofundată a problemelor de modelat, pentru o formulare corectă, cât mai concisă şi elegantă cu putință a fiecăreia dintre ele.* Desigur că, din când în când, putem greşi inclusiv prin asertarea unor informații redundante. Dar de aici şi până la dublarea inutilă a dimensiunii modelului este un pas „uriaş" ce merită evitat! (mai ales că dacă reunim toate df cu aceiaşi parte stângă raportul util/superfluu devine şi mai dezastruos: 5/6!).

În concluzie, chiar în termeni relaționali, acest adevărat contraexemplu de modelare a datelor ar trebui reformulat astfel: fie schema de relație NTFMAVPSCID, cu dependențele:

D1' : N→T (o maşină aparține unui singur tip)

D2' : T→FMA (un tip este caracterizat biunivoc de tripletul fabricant, model, an de fabricație al maşinilor)

D3' : T→V (un tip de maşină are un singur preț)

D4' : N→P (o maşină are un unic proprietar)

D5' : S↔C (un şofer are un unic carnet de conducere, carnet care este numai al său)

D6' : S→> ID (un şofer poate săvârți o mulțime de încălcări ale legii circulației, la diverse date calendaristice)

Desigur că nici această formulare relațională nu modelează acurat „realitatea" avută în vedere: ea nu surprinde nici echivalența ce „calculează" elementele mulțimii identificate de T (reprezentate doar de D1' - abstractizând surjecția canonică asociată, dar fără păstrarea informației despre această natură a semanticii sale) şi nici faptul că mulțimile ale căror elemente sunt identificate de P, respectiv S trebuie să fie incluse ambele într-+ aceiaşi mulțime (de oameni). Acestea sunt însă lucruri imposibil de exprimat în MRD.

Soluția de proiectare oferită de autori [ZM] este descompunerea VFMA, NF; NA; NM, PC; PID; NP: ea este în FN4, păstrează toate dependențele şi are proprietatea de jfp. Întâmplător, fie că analşizăm această soluție, fie pe cea „clasică" echivalentă (i.e. VFMA, NFMA, NPC, PID), fie soluția reformulării date de noi mai sus (i.e. NTP, TFMAV, SC; SID) bd corespunzătoare nu pot memora conținuturi aberante. Dar acest lucru nu constituie un merit al abordării relaționale ci se datoreşte trivialității problemei propuse!

8. Probleme propuse spre rezolvare

1. Formalizați diferența dintre definiția produsului cartezian uzuală în teoria mulți-milor şi cea din modelul relațional al datelor.
2. Să se demonstreze că propoziția corespunzând valorilor nule lipsite de informație ($R_{-k}(t[A_1],...,t[A_{k-1}],t[A_{k+1}],...,t[A_n])$) este echivalentă cu disjuncția formulelor asociate celorlalte două tipuri de valori nule: $\exists x(R(t[A_1],...,t[A_{k-1}], x, t[A_{k+1}],..., t[A_n])) \vee (\neg\exists x(R(t[A_1],...,t[A_{k-1}], x, t[A_{k+1}],...,t[A_n])) \wedge R_{-k}(t[A_1],...,t[A_{k-1}],t[A_{k+1}], ..., t[A_n]))$.
3. Să se demonstreze propoziția 25.
4. Să se demonstreze propoziția 27.
5. Calculați cardinalitatea minimă şi cea maximă a joinului a n relații, n natural, în termenii cardinalității operanzilor.
6. Să se demonstreze că operatorul join este comutativ şi asociativ.
7. Demonstrați că operatorii de selecție şi proiecție sunt monotoni.[59] Extindeți definiția monotoniei la operatori n-ari şi demonstrați că şi operatorul join este monoton.
8. Să se demonstreze lema 9.
9. Să se demonstreze lema 10.
10. Să se demonstreze propoziția 22.
11. Fie $n = cardU$; să se arate că numărul de *FD* triviale peste $R(U)$ este $3^n - 2^n$.
12. Ar creşte puterea expresivă a *FD* dacă am admite şi *FD* fără nici un atribut în partea stângă şi/sau în cea dreaptă? De ce?
13. Demonstrați lema 46 (*indicație*: folosiți inducția după lungimea derivării).
14. Demonstrați lema 48.
15. Demonstrați sau infirmați soliditatea următoarelor reguli de deducție pentru *FD*:
    a. *dacă* $X \supset Y$, *atunci* $X \rightarrow Y$
    b. *dacă* $X \supseteq Y$, *atunci* $Y \rightarrow X$
    c. *dacă* $X \supseteq Y$, *atunci* $X \rightarrow Y$
    d. *dacă* $X \rightarrow Y$ *şi* $Z \supset X$, *atunci* $Z \rightarrow Y$
    e. *dacă* $X \rightarrow Y$ *şi* $Z \supset X$ şi $Y \supset V$, *atunci* $Z \rightarrow V$.
16. O mulțime de reguli de inferență $\boldsymbol{S}$ se zice *independentă* dacă nici o submulțime proprie a sa $\boldsymbol{S'}$ nu permite derivarea tuturor constrângerilor derivabile cu ajutorul $\boldsymbol{S}$ pentru orice mulțime de constrângeri. Demonstrați că mulțimea {FD1, FD2, FD3} este independentă.
17. Găsiți legături între diversele reguli de derivare din text şi cele din problema 15. Găsiți, printre toate aceste reguli de derivare, mulțimi de reguli complete şi independente.
18. Demonstrați corolarul 51.
19. Demonstrați că $\forall X$, X^+ este cea mai mică închidere conținând X.
20. Găsiți o mulțime solidă şi completă de reguli de derivare pentru constrângerile tuplu (*indicație*: sunt necesare presupuneri suplimentare asupra domeniilor atributelor).
21. Demonstrați lema 67.
22. Demonstrați tranzitivitatea aplicațiilor de conținere (i.e. pentru orice pereche $\varphi : T_1 \rightarrow T_2$ şi $\eta : T_2 \rightarrow T_3$ există o aplicație de conținere de la T_1 la T_3).
23. Demonstrați corolarul 96.

[59] Reamintim că un operator unar f se zice *monoton* dacă $\forall\ r_1, r_2, (r_1 \subseteq r_2) \Rightarrow f(r_1) \subseteq f(r_2)$

24. Demonstrați corolarul 98.
25. Furnizați un exemplu de relație pentru care condițiile necesare din teorema 99 nu sunt obligatorii.
26. Formulați o caracterizare a problemei implicației pentru *DIN* bazată pe vânare (cu presupunerea că algoritmul se termină).
27. Găsiți condiții suficiente de terminare a vânării pentru *DIN*.
28. Demonstrați lema 102.
29. Demonstrați lema 103.
30. Demonstrați lema 104.
31. Demonstrați propoziția 105 (*indicație*: folosiți inducția după numărul de pași necesari în vânare pentru a adăuga tuplul t la r_i).
32. Proiectați un algoritm de rezolvare a problemei implicației pentru *DIN* cu urmă-toarele specificații:
    Intrare: o schemă de bd, o mulțime I de *DIN*, o schemă de relație R_i și o listă de atribute L_1 din X_i
    Ieșire: toate schemele de relații R_j și listele de atribute L_2 cu proprietatea că I implică $R_i[L_1] \subseteq R_j[L_2]$.
33. Simplificați regulile de inferență pentru subclasa *DIN* tipate; proiectați un algoritm eficient de rezolvare a problemei implicației pentru această subclasă.
34. Construiți o mulțime de reguli de inferență solidă și completă pentru *DIN* unare; proiectați un algoritm de complexitate liniară pentru problema implicației corespunzătoare.
35. Arătați că implicația finită și cea restricționată coincid pentru *FD*.
36. Găsiți o mulțime Γ de *FD* și de *DIN* și o *FD* f astfel încât Γ nu implică (nerestrictiv) f, dar implică finit f.
37. Demonstrați punctul 2 al lemei 112.
38. Demonstrați lema 116 (*indicație*: infinitatea domeniilor este esențială!).
39. Demonstrați lema 117 (*indicație*: aceeași ca la problema de mai sus!).
40. Definiți noțiunile de satisfacere puternică și slabă pentru *DIN* și demonstrați teorema similară 118.
41. Demonstrați că dacă o mulțime F de *FD* este slab satisfăcută de o relație r și dacă există o relație r' fără valori nule, obținută din r prin substituții, ce satisface F, atunci r' satisface (în contextul implicației uzuale, lipsite de valori nule) toate *FD* implicate de F.
42. Demonstrați că regulile de inferență FD1, FD2 și FD4 sunt solide pentru *FD* și în prezența valorilor nule necunoscute.
43. Demonstrați teorema 121.
44. Proiectați, pornind de la figura 26, un algoritm de calcul a închiderii unei mulțimi de atribute în raport cu o mulțime de *FD* în contextul satisfacerii slabe.
45. Demonstrați soliditatea regulilor de inferență FD1, FD2 și FD4 pentru *FDN*.
46. Demonstrați teorema 123.
47. Demonstrați că dacă F și G sunt mulțimi de *FDN* echivalente, atunci, oricare ar fi $Y \rightarrow A \in F$, există o *FDN* $Z \rightarrow A \in G$ astfel încât $Z \subseteq Y$.
48. Demonstrați că pentru orice mulțime F de *FDN* există și este unică o submulțime a sa ce constituie o acoperire neredundantă pentru F.
49. Demonstrați teorema 125.
50. Studiați problema implicației pentru reuniunea claselor *FDN* și *CE*.
51. Demonstrați că (*teorema de caracterizare a schemelor de relații cu FD având o singură cheie*): dată fiind o schemă $[R, F]$, cu $R(U)$ și $F = \{X_1 \rightarrow Y_1, \ldots, X_k \rightarrow Y_k\}$, R are o unică cheie $\Leftrightarrow U - Z_1 \ldots Z_k$ este o supercheie, unde $Z_i = Y_i - X_i$, $\forall 1 \leq i \leq k$.

52. O schemă de relație [*R,F*] se zice a fi în *forma normală 2* (*FN2*) dacă F^+ nu conține nici o *FD* parțială $K \to A$ ($X \to A \in F^+$ se zice *parțială* dacă $\exists X' \subset X$ astfel încât $X' \to A \in F^+$), unde *K* este o cheie, iar *A* un atribut *neprim* (i.e. un atribut care nu face parte din nici o cheie). Demonstrați că *FN2* este strict mai slabă decât *FN3*.
53. O schemă de relație [*R,F*] se zice a fi în *forma normală 2 specială* (*FN2S*) dacă $\forall K,A \in R$, *K* cheie, $\neg\exists X \subset K$, $A \notin X$, astfel încât $X \to A \in F^+$. Demonstrați că *FN2S* este strict mai slabă decât *FN3*, dar strict mai puternică decât *FN2*.
54. O schemă de relație [*R,F*] se zice a fi în *forma normală 3* (*FN3*) dacă pentru orice *FD* netrivială $X \to A \in F$, unde *A* este neprim, *X* este o supercheie (unde, vezi şi problema 51 de mai sus, un atribut *A* se zice *prim* dacă face parte din cel puțin o cheie şi *neprim* în caz contrar). Să se demonstreze că:
    a. [*R,F*] este în *FN3* dacă nici un atribut neprim nu este tranzitiv dependent de nici o cheie a *R* (unde un atribut *A* se zice *tranzitiv dependent* de o mulțime de atribute *X* dacă există o mulțime de atribute *Y* astfel încât $X \to Y$, $Y \to A \in F^+$, $Y \to X \notin F^+$, iar $A \notin X$; rezultă, evident, conform definiției cheilor, că *A* este tranzitiv dependent de o cheie *K* dacă există o mulțime de atribute *X* care nu este supercheie, nici nu conține *A*, dar $X \to A \in F^+$).
    b. *FN3* este strict mai slabă decât *FNBC*, dar strict mai puternică decât *FN2S*.
55. O schemă de relație [*R,F*] se zice a fi în *forma normală cu chei elementare* (*FNKE*) dacă oricare ar fi *FD* netrivială $X \to A \in F$, ori *X* este o cheie elementară, ori *A* face parte dintr-o asemenea cheie (unde o cheie *K* se zice *elementară* dacă există măcar un atribut $A \in R$ astfel încât $\neg\exists K' \subset K$ cu proprietatea că $K' \to A \in F^+$). Demonstrați că *FNKE* este strict mai slabă decât *FNBC*, dar strict mai puternică decât *FN3*.
56. Fie o schemă *R*(*U*); demonstrați că dacă *R* satisface *DMV* $X \to\to Y$, atunci ea satisface şi *DMV* $X \to\to U - XY$ (*proprietatea de complementaritate a DMV*).
57. Construiți şi demonstrați o teoremă de caracterizare a *DMV* triviale.
58. Demonstrați că o relație poate satisface $X \to\to YZ$, deşi violează $X \to\to Y$.
59. Arătați că, în general, dată fiind o mulțime *F* de *FD*, nu este posibilă găsirea unei mulțimi *M* de *DMV* astfel încât o relație satisface $F \Leftrightarrow$ ea satisface *M*.
60. Demonstrați că $X \to\to Y$ şi $X \to\to Z$ implică $X \to\to Y - Z$ (şi deci, datorită proprietății de complementaritate, $X \to\to Y \cap Z$).
61. Demonstrați tranzitivitatea *DMV*.
62. Fie schema *STUDENȚI*(*#Student*, *#Curs*, *Credit*, *Student*, *Sex*, *Vârsta*, *Curs*). Identificați *FD* şi cheile ei. Proiectați o schemă echivalentă în *FNBC*. Se poate obține şi o schemă în *FNDC*? Justificați răspunsul.
63. Fie schema *STUDENȚI*(*#Student*, *#Laborator*, *#PC*, *Student*, *Laborator*, *PC*) având următoarele *FD*:

    #Student $\to$ *Student* *#Student* $\to$ *Laborator* *#Student* $\to$ *#Laborator*

    #Student $\to$ *PC* *#Laborator* $\to$ *PC*

    #Laborator $\to$ *Laborator*

    #PC $\to$ *PC* *#PC* $\to$ *#Laborator*

    Laborator $\to$ *#Laborator*

    Analizați eventualele anomalii ale schemei şi proiectați o schemă echivalentă în *FNDC*.
64. a. Demonstrați teorema 131.
    b. Arătați că ea nu ar mai avea loc dacă definiția *FN4* s-ar referi doar la *DMV* din Γ şi nu la cele din Γ^+.
65. Demonstrați teorema 133.

66. a. Demonstrați (corolarul 134) că o schemă [*R*,*F*] este în *FNBC* (respectiv *FN4*) dacă oricare ar fi *FD* (respectiv *DMV*) $d \in F$ există o unică dependență de cheie care implică *d*.

    b. Arătați că un rezultat similar pentru *FNPJ* (deci *d* fiind *DJ*) nu este adevărat.

67. a. Stabiliți valoarea de adevăr a următoarei propoziții: "Orice relație cu două atribute este în forma normală proiecție join (*FNPJ*)".

    b. Construiți și demonstrați o propoziție care să furnizeze condiția necesară și suficientă pentru ca o relație cu două atribute să fie în *FNPJ*.

68. Demonstrați teorema 136.

69. a. Demonstrați că în absența ipotezei că domeniile conțin cel puțin două valori distincte pentru orice atribut, lema 138.a. nu mai este adevărată.

    b. Completați demonstrația lemei 138.a. luând în considerare și *DMV*.

    c. Demonstrați lema 138.b.

70. Fie următoarea schemă de bd:

    PERSOANA (#Cod, Prenume, Nume, Sex, DataNaşterii)
    TATA (Tată, Copil)
    MAMA (Mamă, Copil)

    Scrieți o expresie *SPJ* pentru calculul ambilor părinți ai tuturor persoanelor memorate de bd. Furnizați câteva conținuturi valide ale acestei scheme și calculați, pentru fiecare în parte, rezultatul expresiei *SPJ* corespunzătoare.

71. Fie o bd "tehnologică" pentru memorarea unor repere (piese, subansamble, module, produse etc.) compuse din alte repere. Structura tehnologică a unui reper poate fi vizualizată sub forma unui arbore în care, la fiecare nivel, se pot afla ori repere "atomice" (în frunzele arborelui), ori repere "compuse" (în nodurile non-frunză). Proiectați o bd în *FNPJ*, considerând codul, denumirea, prețul și compunerea reperelor. Furnizați un posibil conținut valid.

72. În contextul bd de la exercițiul anterior scrieți expresiile *SPJ* pentru calculul următoarelor interogări:
    a) Submulțimea compunerilor memorate (denumire reper, denumire reper compus imediat superior).
    b) Submulțimea reperelor (denumire, preț) pentru care prețul este numai cu 1.000 mai mic decât prețul reperului imediat superior în a cărui componență intră.

    Pentru fiecare din aceste expresii, calculați răspunsurile pentru conținutul propus la problema 71 de mai sus.

73. Fie o bd de "personal" în care interesează numele, prenumele, sexul, data și locul nașterii, adresa și localitatea domiciliului, data angajării, data încetării angajării și angajatorul (firma, organizația administrativă etc.) fiecărei persoane, precum și denumirea, localitatea și eventualele subordonări între angajatori (e.g.: *Microsoft România srl*, cu sediul în București, România, este subordonată *Microsoft Central and Eastern Europe GmbH*, cu sediul în Munchen, Germania, care este subordonata *Microsoft Corporation*, cu sediul în Redmond, S.U.A.). Specificați schema în *FNPJ* a acestei bd și prezentați un posibil conținut valid al ei.

74. În contextul bd de la exercițiul anterior scrieți expresiile *SPJ* pentru calculul următoarelor interogări:
    a) Submulțimea subordonărilor memorate (denumire organizație subordonată, denumire organizație imediat superioară).
    b) Submulțimea persoanelor (nume, prenume, data și locul nașterii) care au lucrat în decursul timpului în aceeași localitate în care s-au născut, la o organizație cu sediul în aceeași localitate cu organizația căreia îi este subordonată.

Pentru fiecare din aceste expresii, calculați răspunsurile pentru conținutul propus la problema 73 de mai sus.

75. Considerați mulțimea cursurilor oferite de o universitate, cu presupunerea că fiecare din ele are asociată o submulțime (posibil vidă) de cursuri (zise precondiții) ce trebuie absolvite înainte de a fi permisă înscrierea la ele. Date fiind două atribute, cheia surogat şi denumirea cursului, proiectați o bd "universitară" în *FNPJ* pentru a memora cursurile şi precondițiile lor. Exemplificați cu un posibil conținut valid. Extindeți bd (tot în *FNPJ*) şi conținutul de mai sus (menținând validitatea) cu următoarele informații: un student (caracterizat de cod, prenume, nume şi cursurile absolvite, precum şi cele la care s-a înscris) se poate înscrie la mai multe cursuri, dacă a absolvit toate precondițiile acestora (evident că la orice curs se pot înscrie mai mulți studenți); pentru fiecare curs pot fi recomandate mai multe cărți (iar o carte poate fi recomandată la mai multe cursuri); pentru cursuri mai interesează şi cărțile recomandate şi studenții înscrişi; pentru cărți interesează codul, titlul şi cursurile la care sunt recomandate. Ilustrați cu un conținut valid al acestei bd.
76. În contextul bd de la exercițiul anterior scrieți expresiile *SPJ* pentru calculul următoarelor interogări:
    a) Submulțimea cursurilor cu cel puțin o precondiție (denumire curs, denumire precondiție).
    b) Submulțimea cursurilor (denumire) la care s-a înscris măcar un student.
    c) Submulțimea studenților (prenume, nume) înscrişi la cursurile la care este recomandată cartea cu titlul 'Tratat de algebră'.
    d) Submulțimea studenților (prenume, nume şi denumire curs absolvit) care au absolvit cursuri a căror denumire este aceeaşi cu titlul uneia din cărțile recomandate la curs.

    Pentru fiecare din aceste expresii, calculați răspunsurile pentru conținutul propus la problema 75 de mai sus.
77. Proiectați o bază de date "administrativ-geografică" în *FNPJ* care să memoreze denumirea, codul poştal, numărul de locuitori şi subdiviziunea administrativă (județ, stat, canton, departament, land, provincie, voievodat etc.) de care aparțin localitățile, denumirea, codul automobilistic, prefixul telefonic, localitatea de reşedință şi țara de care aparțin subdiviziunile administrative ale țărilor, denumirea, codul automobilistic, prefixul telefonic, denumirea subdiviziunilor administrative şi capitala țărilor lumii. Ilustrați cu un conținut valid al acestei bd.
78. În contextul bd de la exercițiul anterior scrieți expresiile *SPJ* pentru calculul următoarelor interogări:
    a) Submulțimea subdiviziunilor administrative (denumire, denumire țara, denumire capitală) care includ capitale cu cel puțin 1.000.000 de locuitori.
    b) Submulțimea județelor din România (denumire, cod auto, prefix telefonic, denumire localitate reşedință) care au reşedința într-o localitate a cărei denumire este diferită de cea a județului.
    c) Submulțimea țărilor (denumire, cod auto, prefix telefonic, denumire capitala) care au aceeaşi denumire cu cea a capitalei lor.
    d) Submulțimea capitalelor (denumire țara, denumire capitala, denumire subdiviziune administrativă, denumire reşedință subdiviziune) care nu sunt şi reşedință a subdiviziunii administrative din care fac parte.

    Pentru fiecare din aceste expresii, calculați răspunsurile pentru conținutul propus la problema 77 de mai sus.
79. Arătați că algoritmul 74 are complexitate $O(|\Gamma|\times\|\Gamma\|)$.
80. Arătați că egalabilitatea (vezi definiția 90) este o relație de echivalență.
81. Demonstrați teorema 97.

82. Demonstrați teorema 113.
83. Arătați că în orice conținut al unei scheme $R(U)$, dacă $XY = U$, atunci $X \rightarrow Y \Leftrightarrow cheie(X)$.
84. Demonstrați că un rezultat similar corolarului 134 nu are loc pentru *FNPJ* şi *DJ*.
85. Demonstrați punctul 2 al teoremei 141.
86. Demonstrați că: $\forall Y \subset X$, $card(X) = card(\pi_Y(X)) \Leftrightarrow \exists K \subseteq Y$, K cheie.
87.
88. **de aratat ca dem. Sunt mai usoare folosind def. Cu nucleu a FD; eventual de mutat mai jos**
89. **Precizați cheile, dependențele de incluziune şi constrângerile de existență pentru fiecare dintre tabelele figurilor cu numere cuprinse între şi**
90. Fie un SGBD relațional care oferă scheme în *FNDC*. Proiectați pentru metadatele sale:
    a. o *DEA*
    b. o bd în *FNDC*.
91. Construiți şi reprezentați în convențiile DEA un graf în care noduri să fie toate conceptele MRD, iar arcele să reprezinte relația „se bazează pe” (*indicații*: „spargeți” schema în subscheme interconectate; definițiile sunt entități; propozițiile, lemele, teoremele şi corolarele sunt asociații). Care sunt mulțimile de elemente minimale, respectiv maximale ale grafului astfel obținut?
92. Arătați că joinul natural a două relații având scheme identice este egal cu intersecția acestora.
93. Date fiind două relații $r_1(X_1)$ şi $r_2(X_2)$, unde $X_2 \subset X_1$, se zice *împărțirea* lui r_1 la r_2 şi se notează cu $r_1 \div r_2$ relația peste $X_1 - X_2$ conținând tuplii t cu proprietatea că, $\forall t_2 \in r_2$, $\exists\ t_1 \in r_1$ astfel încât t_1 este o combinație a lui t cu t_2 (adică, astfel încât $t_1[X_2]=t_2$ şi $t_1[X_1 - X_2]=t$). Arătați că:
    - împărțirea este un operator derivat (i.e. care se poate obține prin compunerea altor operatori);
    - împărțirea este un fel de inversă a produsului cartezian, demonstrând următoarea egalitate: $(r_1 \times r_2) \div r_2 = r_1$.
94. Demonstrați că orice expresie *AR* poate fi transformată într-o expresie echivalentă ale cărei relații constante sunt toate definite peste un singur atribut şi conțin un singur tuplu.
95. Demonstrați că orice expresie *AR* poate fi transformată într-o expresie echivalentă în care selecțiile au doar condiții atomice, proiecțiile elimină doar câte un singur atribut, iar ceilalți operatori sunt doar reuniuni sau redenumiri sau diferențe sau produse carteziene.
96. Discutați ipoteza care cere ca orice formulă sau subformulă să aibă cel puțin o variabilă liberă; în particular, considerați implicațiile sale asupra semanticii expresiilor (valoarea acestora pentru substituții) şi asupra echivalenței *CRD* cu *AR*.
97. Formulați interogările din problema 92 în *CRD*.
98. Demonstrați lema 164.
99. Demonstrați lema 165.
100. Demonstrați teorema 169 (*AR* este independentă de domeniu).
101. Demonstrați că rezultatul unei expresii *CRD-ID* conține doar valori din domeniul activ al expresiei şi al bd curente (*indicație*: folosiți lema 172).
102. Formulați interogările din problema 92 în *CRD* cu declarații de domeniu al valorilor.

103. Pe baza cunoştinţelor Dvs. de *QBE*, discutaţi de ce acesta poate fi considerat o implementare a *CRD-DV*. Găsiţi, în măsura posibilităţilor, o metodă de transformare a expresiilor între aceste două limbaje.
104. Discutaţi posibilele extensii ale *AR* pentru a o face echivalentă cu *CRD*, permiţând astfel expresii dependente de domeniu.
105. Propuneţi extensii ale *AR* şi ale *CR* ce ar putea formaliza interogări de tipul „*Câţi ani a domnit fiecare domnitor în parte?*” sau „*Ce domnitori şi-au început domnia cu cel puţin 25 de ani înaintea morţii?*”.
106. Formulaţi interogările din problema 92 în *CRT*.
107. Demonstraţi teorema 178.
108. Oferiţi o demonstraţie completă a teoremei 182 (echivalenţa *CRT* şi *CRD*).
109. Demonstraţi corolarul 183.
110. Propuneţi extensii ale *CRT-DV* care l-ar face echivalent cu celelalte limbaje discutate în acest capitol.
111. Demonstraţi teorema 185.

9. *Bibliografie*

1. S. Abiteboul. Updates, a new frontier. In *ICDT'88 (Second International Conference on Data Base Theory), Bruges, Lecture Notes in Computer Science 326.* Berlin: Springer-Verlag, 1988, 1-18.
2. S. Abiteboul and C. Beeri. On the power of languages for the manipulation of complex objects. Technical Report 846. INRIA, 1988.
3. S. Abiteboul and N. Bidoit. Nonfirst normal form relations: An algebra allowing data restructuring. *Journal of Comp. and System Sc.* 33(1):361-39 1986.
4. S. Abiteboul and S. Grumbach. Base de donnés et objects structurés. *Technique et Science Informatiques* 6(5):383-404. 1987.
5. S. Abiteboul and R. Hull. IFO: A formal semantics database model. *ACM Trans. on Database Syst.* 12(4):297-314. 1987.
6. S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. In *ACM SIGMOD International Conf. on Manageement of Data*, 1989:159-17
7. S. Abiteboul , P Kanellakis, and G. Grahne. On the representation and querying of possible worlds. In *ACM SIGMOD International Conf. on Management of Data*, 1987:34-48.
8. S. Abiteboul and V. Vianu. Equivalence and optimization of relational transactions. *Journal of the ACM* 35(1):70-120. 1988.
9. S. Abiteboul and V. Vianu. Procedural and declarative database update languages. In *Seventh ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, 1988:240-250.
10. S. Abiteboul and V. Vianu. A transaction-based approach to relational database specification. *Journal of the ACM* 36(4):758-787. 1987.
11. S. Abiteboul and V. Vianu. Procedural languages for database queries and updates. *Journal of Comp. and Syst. Sc.* 41(2):181-227. 1990.
12. S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Comp. and Syst. Sc.* 43(1):62-124. 1991.
13. J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories.* John Wiley & Sons, 1990.
14. R. Ahad, and D. Dedo. OpenODB from Hewlett-Packard: A commercial object-oriented dbms. In *Journal of Object-Oriented Programming*, 4(9): 31-35, 1992.
15. Ahad, Rafiul and Cheng, Tu-Ting. HP OpenODB: An Object.Oriented Database Management Systems for Commercial Applications. *Hewlett-Packard Journal*, vol.44, no.3, June 199 p.20.
16. A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM Trans. on Database Syst.* 4(3):297-314. 1977.
17. A. V. Aho, Y. Sagiv, and J. D. Ullman. Efficient optimization of a class of relational expressions. *ACM Trans. on Database Syst.* 4(4):435-454. 1977.
18. A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalence of relational expressions. SIAM *Journal on Computing* 8(2):218-246. 1977.
19. A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Sixth ACM Symp. on Principles of Programming Languages,* 1979:110-117.
20. A.M. Alupului. *Implementarea MatBase*. Lucrare de Diplomă pentru titlul de inginer, Univ. Politehnică Buc., Fac. Automatică, 1994.
21. W. W. Amstrong. Dependency structure of database relationships. In *IFIP Congress*, 1974:580-58
22. W. W. Amstrong and C. Delobel. Decompositions and functional dependencies in relations. *ACM Trans. on Database Syst.* 5(4):404-430. 1980.

23. ANSI/X3/SPARC Study Group on Database Management Systems. Interim Report 75-0208. *FDT-Bulletin ACM SIGMOD* 7(2). 1975
24. M.A. Arbib, and E.G. Manes. *Arrows, Structures, and Functors. The Categorical Imperative*. Academic Press, New York, 1975.
25. H. Arisawa, K. Moriya, and T. Miura. Operations and the properties of non-first-normal-form databases. In *Ninth International Conf. on Very Large Data Bases, Florence*, 1983:197-204.
26. A. K. Arora and C. R. Carlson. The information preserving properties of certain relational database transformations. In *Fourth International Conf. on Very Large Data Bases, Berlin*, 1978:352-357.
27. Astrahan, M. M. et al. System R: A Relational Approach to Database Management. *ACM TODS*, vol. 1, no. 2, June 1976, pp. 97-137.
28. M.P. Atkinson, and K.G. Kulkarni. Experimenting with the functional data model. In *Databases - Role and Structure*, Stocker, Gray, and Atkinson eds., Cambridge Univ. Press, 1984.
29. Atwood, Thomas. ODMG-93: *The Object DBMS Standard, Part 2*. Object Magazine, Jan. 1994, p.32.
30. P. Atzeni, and V. de Antonellis. *Relational Database Theory*. The Benjamin/Cummings Publishing Company, Inc., 199
31. P. Atzeni, G. Ausiello, C. Batini, and M. Moscarini. Inclusion and equivalence between relational database schemata. *Theoretical Computer Science* 19(2):267-285. 1982.
32. P. Atzeni and E. P. F. Chan. Efficient query answering in the representative instance approach. In *Fourth ACM SIGACT SIGMOD Symp. on Principles of Database Systems*, 1985:181-188.
33. P. Atzeni and E. P. F. Chan. Efficient optimization of simple chase join expressions. *ACM Trans. on Database Syst.* 14(2):212-230. 1987.
34. P. Atzeni and E. P. F. Chan. Efficient and optimal query answering on independent schemes. *Theoretical Computer Science* 77(3):291-308. 1990.
35. P. Atzeni and M. C. De Bernardis. A new basis for the weak instance model. In *Sixth ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, 1987:79-86.
36. P. Atzeni and M. C. De Bernardis. A new interpretation for null values in the weak instance model. *Journal of Comp. and System Sc.* 41(1):25-4 1990.
37. P. Atzeni and N. M. Morfuni. Functional dependencies in relations with null values. *Information Processing Letters* 18(4):233-238. 1984.
38. P. Atzeni and N. M. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control* 70(1):1-31. 1986.
39. P. Atzeni and D. S. Parker Jr. Assumptions in relational database theory. In *ACM SIGACT SIGMOD Symposium on Principles of Database Systems*, 1982:1-7.
40. P. Atzeni and R. Torlone. Updating databases in the weak instance model. In *Eighth ACM SIGACT SIGMOD SIGACT Symposium on Principles of Database Systems*, 1989:101-107.
41. P. Atzeni and R. Torlone. Efficient updates to independent database schemes in the weak instance model. In *ACM SIGMOD International Conf. on Management of Data*, 1990:84-9
42. G. Ausiello, C. Batini, and M. Moscarini. On the equivalence among database schemata. In *Proc. International Conference on Databases.* Aberdeen, 1980.
43. J. Backus. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. In *Commun. ACM* 21(8): 613-641, 1978.
44. F. Bancilhon. On the completeness of query languages for relational databases. *Mathematical Foundations of Computer Science, LNCS 64 Berlin:Springer-Verlag*, 1978, 112-124.

45. F. Bancilhon, and P. Buneman, editors. *Advances in Database Programming Languages*. ACM Press, 1990.
46. F. Bancilhon, D. Maier, Y. Sagiv, and J. D. Ullman. Magic sets and other strange ways to implement logic programs. In *Fifth ACM SIGACT SIGMOD Symp. on Principles of Database Systems*, 1986:1-15.
47. F. Bancilhon and R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *ACM SIGMOD International Conf. on Management of Data*, 1986:16-52.
48. F. Bancilhon, P. Richard, and M. Scholl. VERSO: A relational back end database machine. In D. K. Hsiao, editor, *Advanced Database Machine Architecture.* Englewood Cliffs, N.J.:Prentice-Hall, 1983, 1-8.
49. D.W. Barnes and J.M. Mack. *An Algebraic Introduction to Mathematical Logic*. Springer Verlag, 1975.
50. C. Batini, S. Ceri, and S. B. Navathe. *Database Design with the Entity-Relationship Model.* Menlo Park, Calif.: Benjamin/Cummings, 1991.
51. Bayer, R. and McCreight, E.M. Organization and Maintenance of Large Ordered Indeces. *Acta Informatica*, vol. 1, no.3, 1972, pp.173-187.
52. Bayer, R. and Unterauer, K. Prefix B trees. *ACM TODS*, vol.2, no.1, March 1977, pp. 11-26.
53. C. Beeri. Data models and languages for databases. In *ICDT'88 (Second International Conference on Data Base Theory), Bruges, Lecture Notes in Computer Science 326.* Berlin: Springer-Verlag, 1988, 19-37.
54. C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Trans. on Database Syst.* 4(1):30-57.. 1977.
55. C. Beeri, P. A. Bernstein, and N. Goodmon. A sophisticate's introduction to database normalization theory. In *Fourth International Conf. on Very Large Data Bases, Berlin,* 1978:113-124.
56. C. Beeri, R. Fagin, and J. H. Howard. A complete axiomatization for functional and multivalued dependencies. In *ACM SIGMOD International Conf. on Management of Data*, 1978:47-61.
57. C. Beeri and P. Honeyman. Preserving functional dependencies. *SIAM Journal on Computing* 10(3):647-656. 1981.
58. C. Beeri and M. Kifer. An integrated approach to logical design of relational database schemes. *ACM Trans. on Database Syst.* 11(2):134-158. 1986.
59. C. Beeri, A. O. Mendelzon, Y. Sagiv, and J. D. Ullman. Equivalence of relational database schemas. *SIAM Journal on Computing* 10(2):352-370. 1981.
60. C. Beeri, and R. Ramakrishnan. On the power of magic. Proc. of the ACM Symp. on Principles of Db Systems, 1987.
61. C. Beeri and J. Rissanen. Faithful representation of relational database schemata. Report RJ 2722. IBM Research. San Jose, 1980.
62. C. Beeri and M. Y. Vardi. Formal systems for tuple and equality-generating dependencies. *SIAM Journal on Computing* 13(1):76-98. 1984.
63. C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM* 31(4):718-741. 1894.
64. P. A. Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM Trans. on Database Syst.* 1(4):277-298.1976.
65. Bernstein, P.A. *Middleware: An Architecture for Distributed System Services*. DEC Corp., Cambridge Res. Lab., March 199
66. Bernstein, P.A. and Goodman, Nathan. Concurrency Control in Distributed Database Systems. *ACM Computing Surveys*, vol.13, no.2, June 1981, pp. 185-222.
67. P. A. Bernstein and N. Goodman. What does Boyce-Codd form do? In *Sixth International Conf. on Very Large Data Bases, Montreal,* 1980:245-257.

68. J. Biskup. A formal approach to null values in database relations. In H.Gallaire, J. Minker, and J. M. Nicolas, editors, *Advances in Database Theory.* New York: Plenum, 1981, 299-341.
69. J. Biskup. A foundation of Codd's relational maybe-operations. *ACM Trans. on Database Syst.* 8(4):608-636. 198
70. J. Biskup, U. Dayal, and P.A. Bernstein. Synthesizing independent database schemes. In *ACM SIGMOD International Conf. on Management of Data,* 1979:143-151.
71. Bobrowski, Steve. Database Security in a Client/Server World. *DBMS*, vol. 7, no. 10, Sept. 1994, p. 48.
72. Bobrowski, Steve. Parallel Oracle 7.1. *DBMS*, vol. 6, no. 12, Dec. 1993, p. 87.
73. Bontempo, Charles J. and Saracco, Cynthia Maro. *Database Management Principles and Products*. Prentice Hall, New Jersey, 1995.
74. Borland Int. *Borland PARADOX 7.0 for Windows 95.* Borland, 1992.
75. V. Breazu. Semantics in Complete Lattices for Relational Database Functional Dependencies. *Analele ştiinţifice ale Univ. "Al.I. Cuza*", Iaşi, tom XXVIII, s.Ia, 1982, f.1
76. V. Breazu şi C. Mancaş. Asupra unui model funcţional în teoria matematică a bazelor de date. Comparaţie cu modelul relaţional. Comun. la al 6-lea Simpozion CONDINF'80, Cluj, 1980.
77. V. Breazu şi C. Mancaş. On the Description Power of a Functional Database Model. In *Fourth Int. Conf. on Control Syst. and Comp. Sci.,* 4: 223-226, May 1981.
78. V. Breazu şi C. Mancaş. Forme normale pentru scheme cu dependenţe funcţionale în modelul relaţional al bazelor de date. In *Conf. Naţ. de Electron., Telecom., Automat. şi Calc*.. I.P.B., S.12, 1982: 152-156.
79. M. L. Brodie and J. Mylopoulos. Knowledge bases and databases: Semantic vs computational theories of information. In G. Ariav and J. Clifford, editors, *New Directions for Database Systems.* New York: Ablex, 1986.
80. M. L. Brodie and J. Mylopoulos, editors. *On Knowledge Base Management Systems.* Berlin: Springer-Verlag, 1986.
81. M. L. Brodie, J. Mylopoulos, and J. Schmidt, editors. *On Conceptual Modelling.* Berlin: Springer-Verlag, 1984.
82. F. Bry. Query Evaluation in Recursive databases: Bottom-up and Top-down Reconciled. In *IEEE Trans. on Knowledge and Data Engineering*, 2, 1990.
83. P. Buneman, and R.E. Frankel. FQL - A functional query language. In *ACM SIGMOD Conf*., Boston, 1979: 52-58.
84. P. Butterworth, A. Otis, and J. Stein. The Gemstone object database management system. *Comm. of the ACM* 34(10):64-77. 1991.
85. F. Cacace, S. Ceri, S. Crespi-Reghizzi, L. Tanca, and R. Zicari. Integrating object-oriented data modelling with a rule-based programming paradigm. In *ACM SIGMOD International Conf. on Management of Data,* 1990:225-236.
86. M. A. Casanova. The theory of functional and subset dependencies over relational expressions. *Technical Report 3/81.* Rio de Janeiro: Dept. de Informatica, Pontificia Unversidade Catolica. 1981.
87. M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *Journal of Comp. and System Sc.* 28(1): 29-57. 1984.
88. S. Ceri, S. Crespi-Reghizzi, A. Di Maio, and L. A. Lavazza. ALGRES. *IEEE Trans. on Software Eng.* SE-14(11). 1988.
89. S. Ceri, S. Crespi-Reghizzi, G. Lamperti, L. A. Lavazza, and R. Zicari. ALGRES: An advanced database system for complex applications. *IEEE Software* 7(4): 68-78. 1990.

90. S. Ceri and G. Gottlob. Normalization of relations and Prolog. *Communications of the ACM* 29(6):524-544. 1986.
91. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Data Bases.* Berlin: Springer-Verlag, 1987.
92. D. D. Chamberlin et al. A history and evaluation of System R. *Communications of the ACM* 24(10). 1981.
93. E. P. F. Chan. Optimal computation of total projections with unions of simple chase join expressions. In *ACM SIGMOD International Conf. on Management of Data,* 1984:149-16
94. E. P. F. Chan. A desing theory for solving the anomalies problem. *SIAM Journal on Computating* 18(3):429-448. 1987.
95. E. P. F. Chan and H. Hernández. On the desirability of -acyclic BCNF database schemes. *Theoretical Computer Science* 62(1-2). 1988.
96. E. P. F. Chan and A. O. Mendelzon. Independent and separable database schemes. *SIAM Journal on Computing* 16(5):841-851. 1987.
97. A. K. Chandra. Theory of database queries. In *Seventh ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems,* 1988:1-7.
98. A. K. Chandra and D. Harel. Computable queries for relational databases. *Journal of Comp. and System Sc.* 21:333-347. 1980.
99. A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *Thirteenth ACM SIGACT Symp. on Theory of Computing,* 1981:342-354.
100. A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM Journal on Computating* 14(3):671-677. 1985.
101. C.C. Chang. On the Evaluation of Queries Containing Derived Relations in a Relational Database. In H. Gallaire, J. Minker, and J. Nicolas, eds., *Advances in Database Theory*, Vol. I, Plenum Press, 1981.
102. C.C. Chang and H.J. Keisler. *Model Theory*. North-Holland Publishing Company, 197
103. P. P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Trans. on Database Syst.* 1(1):9-36. 1976.
104. Cheng, J.M. et al. IBM DB2 Performance: Design, Implementation, and Tuning. *IBM Systems Journal*, vol. 23, no. 2, 1984.
105. D. Chimenti, R. Gamboa, R. Krishnamurti, S. Naqvi, S. Tsur, and C. Zaniolo. The LDL system prototype. *IEEE Trans. on Knowledge and Data Eng.* 2(1):76-90. 1990.
106. E. F. Codd. A relational model for large shared data banks. *Communications of the ACM* 13(6):377-387. 1970.
107. E. F. Codd. A database sublanguage founded on the relational calculus. In *ACM SIGFIDET Workshop on Data Description, Access and Control,* 1971:35-61.
108. E. F. Codd. Further normalization of the data base relational model. In R. Rustin, editor, Data Base Systems. Englewood Cliffs, N. J. : Prentice-Hall, 1972, 33-64.
109. E. F. Codd. Relational completeness of data base sublanguages. In R. Rustin, editor, *Data Base Systems*. Englewood Cliffs, N. J. : Prentice-Hall, 1972, 65-98.
110. E. F. Codd. Recent investigations into relational database systems. In *IFIP Congress,* 1974:1017-1021.
111. E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. on Database Syst.* 4(4):397-434. 1977.
112. E. F. Codd. 'Universal' relation fails to replace relational model. Letter to the editor. *IEEE Software* 5(4):4-6. 1988 .
113. Comer, D. The Ubiquitous B-tree. *Computing Surveys*, vol.11, no.2, June 1979, pp.121-137.

114. S. Cosmadakis, P. C. Kanellakis, and N. Spyratos. Partition Semantics for Relations. unpub. paper, 1985.
115. S. Cosmadakis, P. C. Kanellakis, and M. Vardi. Polymonial-time implication problems for unary inclusion dependencies. *Journal of the ACM* 37(1):15-46. 1990.
116. Crus, R.A. Data Recovery in IBM DB2. *IBM Systems Journal*, vol. 23, no. 2, 1984, pp.178-188.
117. P. Dadam et al. A DBMS prototype to support extended NF^2 relations: An integrated view of flat tables and hierarchies. In *ACM SIGMOD International Conf. on Management of Data*, 1986:356-367.
118. C. J. Date. *An Introduction to Database Systems.* Volume 2. Reading, Mass.: Addison Wesley, 198
119. C. J. Date. *A Guide to DB2.* Reading, Mass.: Addison Wesley, 1984.
120. C. J. Date. *An Introduction to Database Systems.* 4th ed. Volume 1. Reading, Mass.: Addison Wesley, 1986.
121. C. J. Date. *A Guide to INGRES.* Reading, Mass.: Addison Wesley, 1987.
122. Date, C.J. and White, Colin. *A Guide to DB2. Fourth Edition*. Addison Wesley, 1992.
123. Date, C.J. *An Introduction to Database Systems. Sixth Edition*. Addison-Wesley, 1994.
124. U. Dayal, and others. Simplifying complex objects: The PROBE approach to modelling and querying them. In Schek and Schlageter, eds., *Informatik Fachberichte* 136: 17-38, Springer Verlag, 1987,
125. P. DeBra and J. Paredaens. An algorithm for horizontal decompositions. *Information Processing Letters* 17(2):91-95. 198
126. P. DeBra and J. Paredaens. Horizontal decompositions for handling exceptions to functional dependencies. In H. Gallaire, J. Minker, and J.-M. Nicolas, editors, *Advances in Database Theory.* Volume 2. New York: Plenum, 1984, 123-144.
127. C. Delobel and R. C. Casey. Decomposition of a database and the theory of Boolean switching functions. *IBM Journal of Research and Development* 17(5):370-386. 1972.
128. R. Demolombe. Syntactical characterization of a subset of domain independent formulas. Report. Toulouse: ONERA-CERT, 1982.
129. O. Deux et al. The O_2 system. *Communications of the ACM* 34(10):34-47. 1991.
130. R. Di Paola. The recursive unsolvability of the decision problem for the class of definite formulas. *Journal of the ACM* 16(2):324-327. 1967.
131. A. Diller. *Z, An Introduction to Formal Methods*, 2nd ed. John Wiley & Sons, 1994.
132. H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer Verlag, 1984.
133. R. A. ElMasri and S. B. Navathe. *Fundamentals of Database Systems. Second edition.* Redwood City, Calif.: Benjamin/Cummings, 1994.
134. R. A. ElMasri and S. B. Navathe. *Fundamentals of Database Systems.* Menlo Park, Calif.: Benjamin/Cummings, 1988.
135. H. B. Enderton. *A Mathematical Introduction to Logic.* New York: Academic Press, 1972.
136. Fagin, R. A Normal Form for Relational Databases that Is Based on Domains and Keys. *ACM TODS*, vol.6, no.3, 1981, pp.387-415.
137. R. Fagin. The decomposition versus the synthetical approach to relational database design. In *Third International Conf. on Very Large Data Bases, Tokyo,* 1977:441-446.
138. R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. on Database Syst.* 2(3):262-278. 1977.

139. R. Fagin. Normal forms and relational database operators. In *ACM SIGMOD International Conf. on Management of Data*, 1979:123-134.
140. R. Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. on Database Syst.* 6(3):310-317. 1981.
141. R. Fagin. Horn clauses and database dependencies. *Journal of the ACM* 29(4): 952-98 1982.
142. R. Fagin, A. O. Mendelzon, and J. D. Ullman. A simplified universal relation assumption and its properties. *ACM Trans. on Database Syst.* 7(3):343-360. 1982.
143. P. C. Fischer, L. V. Saxton, S. J. Thomas, and D. Van Gucht. Interactions between dependencies and nested relational structures. *Journal of Comp. and System Sc.* 31(2):343-354. 1985.
144. P. C. Fischer and S. J. Thomas. Operators for non-first-normal-form relations. In *Proc. of IEEE Computer Software Applications*, 1983:464-475.
145. P. C. Fischer and D. Van Gucht. Weak multivalued dependencies. *In Third ACM Symposium on Principles of Database Systems*, 1984:266-274.
146. D.H. Fishman, and very many others. Iris: An Object-Oriented Database Management System. In *ACM Trans. Office Inf. Syst*., 5(1): 48-69, 1987.
147. A. L. Furtado and L. Kerschberg. An algebra of quotient relations. *In ACM SIGMOD International Conf. on Management of Data*, 1977:1-8.
148. H. Gallaire and J. Minker, editors. *Logic and Databases.* New York: Plenum, 1978.
149. H. Gallaire, J. Minker, and J. M. Nicolas, editors. *Advances in Database Theory*. Volume 1. New York: Plenum, 1981.
150. H. Gallaire, J. Minker, and J. M. Nicolas, editors. *Advances in Database Theory*. Volume 2. New York: Plenum, 1984.
151. H. Gallaire, J. Minker, and J. M. Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys* 16(2):153-185. 1984.
152. G. Gardarin and P. Valduriez. *Relational Databases and Knowledge Bases.* Reading, Mass.: Addison Wesley, 1990.
153. M. R. Garey and D. S. Johnson. *Computers and Intractability.* San Francisco: W. H. Freeman and Company, 1977.
154. Gifford, Dwayne et al. *Access 95 Unleashed*. SAMS Publishing, Indianapolis, 1996.
155. J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. *A Junction Between Computer Science and Category Theory, I: Basic Concepts and Examples (Part 1)*. IBM T.J. Watson Research Center Report RC 4526, 197
156. J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. *A Junction Between Computer Science and Category Theory, II: Basic Concepts and Examples (Part 2)*. IBM T.J. Watson Research Center Report RC 5908, 1976.
157. B. S. Goldstein. Constraints on null values in relational databases. *In Seventh Int'l Conf. on Very Large Data Bases*, 1981:101-111.
158. G. Goos and J. Hartmanis, editors. *Category Theory Applied to Computation and Control*. Proc. 1st Int. Symp., San Francisco, 1974. LNCS 25, Springer-Verlag, 1975.
159. Gotlieb, L. R. Computing join of relations. *ACM SIGMOD International Symposium on Management of Data*, 1975, pp. 55-6
160. M. Graham, A. O. Mendelzon, and M. Y. Vardi. Notions of dependency satisfaction. *Journal of the ACM* 33(1):105-127. 1986.
161. M. Graham and M. Yannakakis. Independent database schemas. *Journal of Comp. and System Sc.* 28(1):121-141. 1984.
162. G. Grahne. Horn tables - an efficient tool for handling incomplete information in databases. In *Eighth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems*, 1989:75-82.

163. J. Grant. Null values in a relational database. *Information Processing Letters* 6 (5):156-157. 1977.
164. J. Grant and J. Minker. The impact of logic programming on databases. *Communications of the ACM* 35(3):66-81. 1992.
165. G. Grätzer. *Lattice Theory. First Concepts and Distributive Lattices.* W.H. Freeman and Co., San Francisco, 1971.
166. A. Gray, James and Reuter, Andreas. *Transaction Processing: Concepts and Techniques*. Morgan Kaufman, U.S.A., 199
167. Haderle, D.J. and Jackson, R.D. IBM DB2 Overview. *IBM Systems Journal*, vol. 23, no. 2, 1984, pp.112-125.
168. Haerder, T. Implementing a Generalized Access Path Structure for A Relational Database System. *ACM TODS*, vol.3, no.3, Sept. 1978, pp. 285-298.
169. M. Hammer, and D. McLeod. The Semantic Data Model: a Modelling Mechanism for Database Applications. In *ACM SIGMOD Int. Conf. on the Manag. of Data*, 1978.
170. H. Hernández and E. P. F. Chan. Constant-time-maintainable BCNF database schemes. *ACM Trans. on Database Syst.* 16(4)571-597. 1991.
171. H. Herrlich, and G.E. Strecker. *Category Theory, An Introduction*, 2nd ed. Heldermann Verlag, Berlin, 1977.
172. P. Hilton, and Z-C. Wu. *A Course in Modern Algebra*. John Wiley & Sons, 1974.
173. P. Honeyman. Testing satisfaction of functional dependencies. *Journal of the ACM* 29(3):668-677. 1982.
174. Hsiao, Hui-I. *Parallel Database System Technology*. IBM Res. Rep. 18866, IBM T.J. Watson Res. Center, April 199
175. R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases.* New York: Academic Press, 1988, 193-256.
176. R. Hull and M. Yoshikawa. ILOG.: Declarative creation and manipulation of object identifiers. In *Sixteenth International Conference on Very Large Data Bases, Brisbane*, 1990:455-468.
177. R. B. Hull. Relative information capacity of simple relational schemata. *SIAM Journal on Computing* 15(3):856-886. 1986.
178. R. B. Hull and R. King. Semantic database modelling: Survey, applications and research issues. *ACM Comp. Surveys* 19(3):201-260. 1987.
179. IBM Corp. Distributed Relational Database Architecture Reference. SC26-4651, 1990.
180. IBM Corp. *Distributed Relational Database Architecture. Connectivity Guide*. Prentice Hall, New Jersey, Teresa Hopper, editor, 1995.
181. T. Imielinski and W. Lipski. On representing incomplete information in relational databases. In *Seventh International Conf. on Very Large Data Bases, Cannes*, 1981:388-397.
182. T. Imielinski and W. Lipski. Incomplete information and dependencies in relational databases. In *ACM SIGMOD International Conf. on Management of Data*, 1893:178-184.
183. T. Imielinski and W. Lipski. Incomplete information and dependencies in relational databases. *Journal of ACM* 31(4):761-791. 1984.
184. M. Ito, M. Iwasaki, and T. Kasami. Some results on the representative instance in relational databases. *SIAM Journal on Computing* 14(2):334-354. 1985.
185. V. Ivanov. Implementing FastBase. unpub. paper, subm. to 13th ISDBMS, Mamaia, 1990.
186. Jacobson, Ivar et al. *Object-Oriented Software Engineering. A Use Case Driven Approach. Revised Fourth Printing*. Addison-Wesley, ACM Press, 1994.

187. G. Jaeschke and H. -J. Schek. Remarks on the algebra for non first normal form relations. In *ACM SIGACT SIGMOD Symp. on Principles of Database Systems*, 1982:124-138.
188. James, Phil and Weingarten, Jan. *Internet Guide for Windows 95*. Ventana Communications Group, U.S.A., 1995.
189. J. H. Jou and P. C. Fischer. The complexity of recognizing 3NF relation schemes. *Information Processing Letters* 14(4):187-190. 198
190. Y. Kambayashi, K. Tanaka, and K. Takeda. Synthesis of unnormalized relations incorporating more meaning. *Inf. Sci.* 29:201-247. 198
191. P. Kandzia and H. Klein. On equivalence of relational databases in connection with normalization. In *Workshop on Formal Bases for Databases.* Toulouse: ONERA-CERT, 1977.
192. P. C. Kanellakis, S. Cosmadakis, and M. Vardi. Unary inclusion dependencies have polynomial-time inference problems. In *Fifteenth ACM SIGACT Symp. on Theory of Computing*, 1983:264-277.
193. A. Kato. An abstract Relational Model and Natural Join Functors. In *Bull. of Informatics and Cybern*., 20: 95-106, 198
194. Y. Kawahara. Categorical Relational Database Models. *Bulletin of Informatics and Cybern.* Vol. 21, No. 1-2, 1984.
195. A. M. Keller. Set-theoretic problems of null completion in relational databases. *Information Processing Letters* 22(5):261-265. 1986.
196. A. Kemper, and G. Moerkotte. *Object-Oriented Database Management*. Prentice-Hall, 1994.
197. W. Kent. Limitations of record-based information models. *ACM Trans. on Database Syst. 4(1): 107-131, 1977.*
198. W. Kent. Consequences of assuming a universal relation. *ACM Trans. on Database Syst.* 6(4):539-556. 1981.
199. W. Kent. The universal relation revisited. *ACM Trans. on Database Syst.* 8(4): 644-648. 198
200. L. Kerschberg, editor. *Proc. First Workshop on Expert Database Systems.* Menlo Park. Calif.: Benjamin/Cummings, 1986.
201. L. Kerschberg., editor. *Proc. First Conf. on Expert Database Systems.* Menlo Park, Calif.: Benjamin/Cummings, 1987.
202. L. Kerschberg., editor. *Proc. Second Conf. on Expert Database Systems.* Menlo Park, Calif.: Benjamin/Cummings, 1988.
203. L. Kerschberg., editor. *Proc. Third Conf. on Expert Database Systems.* Menlo Park, Calif.: Benjamin/Cummings, 1990.
204. L. Kerschberg, and J.E.S. Pacheco. A Functional Database Model. In *Comp. Sci. Monograph*, Pontificia Universidade Catolica, Rio de Janeiro, 1976.
205. M. Kifer, and E. Lozinskii. A Framework for an Efficient Implementation of Deductive Databases. In *Proc. Sixth Advanced Db Symp*., Tokyo, 1986.
206. W. Kim. Object-oriented databases: Definition and research directions. *IEEE Trans. on Knowledge and Data Eng.* 2(3):327-341. 1990.
207. W. Kim, J. F. Garza, N. Ballou, and D. Woelk. Architecture of the ORION next-generation database system. *IEEE Trans. on Knowledge and Data Eng.* 2(1):109-124. 1990.
208. S.C. Kleene. *Mathematical Logic*. John Wiley & Sons, Inc., 1967.
209. A. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *Journal of the ACM* 29(3):699-717. 1982.
210. Knuth, Donald E. *The Art of Computer Programming. Volume 3: Sorting and Searching*. Addison-Wesley, 197
211. H. F. Korth and A. Silberschatz. *Database Systems Concepts.* New York: McGraw-Hill, 1986.

212. H. F. Korth and J. D. Ullman. System/U: A system based on the universal relation assumption. XP1 Workschop on Relational Database Theory, State University of New York, Stonybrook, N. Y. 1980.
213. R. Kowalsky. Logic for Data Description. In *Logic and Databases*, Plenum Press, 1978:77-10
214. R. Kowalsky. Logic as a Database Language. In *Seminar on Theoretical Issues in Databases*, Cosenza, Italia, 1981.
215. K.G. Kulkarni. *Evaluation of Functional Data Models for DB Design and Use.* Ph.D. Thesis, Univ. of Edinburgh, 198
216. K.G. Kulkarni. *Extended Functional Data Model - User Manual.* Univ. of Edinburgh, 198
217. G. M. Kuper. *The Logical Data Model: A New Approach to Database Logic.* Ph. D. diss. Standford University, 1985.
218. G. M. Kuper and M. Y. Vardi. A new approach to database logic. In *Third ACM SIGACT SIGMOD Symp. on Principles of Database Systems,* 1984:86-96.
219. M. Lacroix and A. Pirotte. Generalized joins. *ACM SIGMOD Record* 8(3):14-15. 1976.
220. C. Lamb, G. Landis, J. Orenstein, and D Weinreb. The ObjectStore database System. *Communications of the ACM* 34(10)50-63, 1991.
221. C. H. LeDoux and D. S. Parker. Reflections on Boyce-Codd normal form. In *Eighth International Conf. on Very Large Data Bases, Mexico-City,* 1982:131-141.
222. T.T. Lee. An Algebraic Theory of Relational Databases. *The Bell System Tech. Journal*, Vol. 62, No.10, 198
223. R.M. Lee, and R. Gerritsen. Extended Semantics for Generalization Hierarchies. In *SIGMOD Int. Conf. on Manag. of Data,* 1978:18-25.
224. N. Lerat and W. Lipski, Jr. Nonaplicable nulls. *Theoretical Computer Science* 46(1):67-82. 1986.
225. Y. E. Lien. Multivalued dependencies with null values in relational databases. In *Fifth International Conf. on Very Large Data Bases,* 1979:61-66.
226. Y. E. Lien. On the equivalence of database model. *Journal of the ACM* 29(2): 333-362. 1992.
227. Linthicum, David. Client/Server Protocols: Choosing the Right Connection. *DBMS*, Jan. 1994, p.60.
228. Linthicum, David. Operating Systems for Database Servers. *DBMS*, Feb. 1994, p.62.
229. W. Lipski, Jr. Two NP-complete problems related to informational retrival. In *Fundamentals of Computation, Theory, Lectures Notes in Computer Science 56.* Berlin: Springer-Verlag, 1977, 452-458.
230. W. Lipski, Jr. On semantic issues connected with incomplete information databases. *ACM Trans. on Database Syst.* 4(3):262-296. 1977.
231. W. Lipski, Jr. On databases with incomplete information. *Journal of the ACM* 28(1):41-70. 1981.
232. J. W. Lloyd. *Foundations of Logic Programming*. 2nd ed. Berlin: Springer-Verlag, 1987.
233. G. M. Lohman, B. Lindsay, H. Pirahesh, and K. B. Schiefer. Extensions to Starburst: Objects, types, functions, rules. *Communications of the ACM* 34(10):94-107. 1991.
234. C. L. Lucchesi and S. L. Osborn. Candidate keys for relations. *Journal of Comp. and System Sc.* 17(2):270-277. 1978.
235. A. Lungu. Using Elementary Algebra for Querying Databases. unpub. paper, subm. to 13th ISDBMS, Mamaia 1990.
236. S. Mac Lane. *Categories for the Working Mathematicians*. Springer-Verlag, 1971.

237. D. Maier. Discarding the universal relation assumption: Preliminary report. XP1 Workshop on Relational Database Theory, State University of New York, Stonybrook. N. Y. 1980.
238. D. Maier. Minimum covers in the relational database model. *Journal of the ACM* 27(4):664-674. 1980.
239. D. Maier. *The Theory of Relational Databases. Potomac,* Md.: Computer Science Press, 198
240. D. Maier, A. O. Mendelzon, F.Sandri, and J. D. Ullman. Adequacy of decompositions in relational databases. *Journal of Comp. and System Sc*. 21(3):368-377. 1980.
241. D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. on Database Syst.* 4(4):455-468. 1977.
242. D. Maier, D. Rozenshtein, and D. S. Warren. Window functions. In P. C. Kanellakis and F. P. Preparata, editors, *Advances in Computing Research.* Volume Greenwich, Ct.: JAI Press, 1986, 213-246.
243. D. Maier and J. D. Ullman. Fragments of relations: First hack. XP2 Workshop on Relational Database Theory, Penn State University, State College, Pennsylvania, 1981.
244. D. Maier, J. D. Ullman, and M. Vardi. On the foundations of the universal relation model. *ACM Trans. on Database Syst.* 9(2):283-308. 1984.
245. A. Makinouchi. A consideration on normal form of not necessarly normalized relations. In *Third International Conf. on Very Large Data Base, Tokyo,* 1977:447-45
246. C. Mancaş. *Modelarea întreprinderilor cu ajutorul bazelor de date. Un prim model al bd I.I.R.U.C.* Raport tehnic IIRUC RTCC#5, martie 198
247. C. Mancaş. On using logic for relational database design. In *Fifth Int. Conf. on Control Syst. and Comp. Sci*., 3: 164-168, June 198
248. C. Mancaş. A Formal Viewpoint on the Entity-Relationship Data Model. In *Sixth Int. Conf. on Control Syst. and Comp. Sci*., 2: 174-178, May 1985.
249. C. Mancaş. Câteva contraexemple de proiectare a bd demonstrând că modelarea datelor nu trebuie abordată relaţional. In *INFO IAŞI'85*, 1985: 303-31
250. C. Mancaş. Introducere într-un model al datelor bazat pe teoria elementară a mulţimilor, relaţiilor şi funcţiilor. In *INFO IAŞI'85*, 1985: 314-320.
251. C. Mancaş. Introducere într-un model matematic al datelor. Comunic. la *INFOTEC'86*, ITCI Bucureşti, 1986.
252. C. Mancaş. Baze de date, sisteme de reprezentanţi şi surjecţii canonice asociate relaţiilor de echivalenţă. In *INFO IAŞI'87*, 1987: 99-110.
253. C. Mancaş. *Proiectul nucleului SGBD MatBase*. Raport tehnic IIRUC RTCC#20, nov. 1988.
254. C. Mancaş. A Deeper Insight into the Mathematical Data Model. In *13th Int. Seminar on DBMS, ISDBMS*. Mamaia, 1990: 122-134..
255. Matthews, Carole Boggs and Shepard, Patricia. *Paradox 4. The Complete Reference*. Osborne McGraw-Hill, California, 199
256. G.H. Mealy. Another Look at Data. In *AFIPS Fall Joint Comp. Conf*., (31): 525-534, 1967.
257. E. Mendelson. *Introduction to Mathematical Logic*. New York: Van Nostrand-Rehinold, 1978.
258. A. O. Mendelzon. Database states and their tableau. *ACM Trans. on Database Syst.* 9(2):264-282. 1984.
259. Microsoft Corp. *Getting Results with Microsoft Office for Windows 95*. Microsoft Press, Washington, 1995.
260. Microsoft Corp. *Microsoft Office for Windows 95 Resource Kit*. Microsoft Press, Washington, 1995.

261. Microsoft Corp. *Microsoft Visual FoxPro Resource Kit*. Microsoft Press, Washington, 1995.
262. Microsoft Corp. *Microsoft Windows 95 Resource Kit*. Microsoft Press, Washington, 1995.
263. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming.* Los Altos, Calif.: Morgan Kauffman, 1988.
264. J. C. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control* 56(1)154-17 198
265. Mohan, C.H. et al. Parallelism in Relational Database Management Systems. *IBM System Journal*, vol. 33, no.2, 1994, p. 347.
266. K. Morris, and many others. YAWN! (Yet Another Window on NAIL!). In *Proc. IEEE Int. Conf. on Data Engineering*, 1987.
267. I. Mumick, H. Pirahesh, and R. Ramakrishnan. The Magic of Duplicates and Aggregates. In *VLDB*, 1990.
268. S. Naqvi and S. Tsur. *A Logical Language for Data and Knowledge Bases.* Potomac, Md.: Computer Science Press, 1987.
269. J.-M. Nicolas. Logic for improving integrity checking in relational databases. *Acta Informatica* 18(3):227-25 1982.
270. Norton, Peter and Mueller, John. *Peter Norton's Complete Guide to Windows 95*. SAMS Pub., Indianapolis, 1995.
271. Obermarck, Ron. Distributed Deadlock Detection Algorithm. TODS, June 1982, pp. 187-208.
272. A. Ohori, P. Buneman, and V. Breazu-Tannen. Database programming in Machiavelli: A polymorphyc language with static type inference. In *ACM SIGMOD Conf. on Manag. of Data*, Portland, 1987.
273. Oracle Corp. *Oracle7 Relational Database Management System Feature Summary (including release 7.1)*. Oracle Corp. manual A14822, May 1994.
274. S. L. Osborn. Towards a universal relation interface. In *Fifth International Conf. on Very Large Data Bases, Rio de Janeiro*, 1979:52-60.
275. Z. M. Ozsoyoglu and L. Y. Yuan. A new normal form for nested relations. *ACM Trans. on Database Syst.* 12(1):111-136. 1987.
276. Papadimitriou, Christos. *The Theory of Database Concurency Control*. Computer Science Press, 1986.
277. J. Paredaens. About functional dependencies in a database structure and their coverings. Report 342. Philips MBLE Lab. 1977.
278. J. Paredaens. On the expressive power of the relational algebra. *Information Processing Letters* 7(2):107-111. 1978.
279. G. Phipps, M. Derr, and K. Ross. GLUE-NAIL!: A Deductive Database System. In *SIGMOD*, 1991.
280. A. Pirotte. High-level database languages. In H. Gallaire and J. Minker, editors, *Logic and Databases.* New York: Plenum, 1978, 409-435.
281. I. Purdea şi G. Pic. *Tratat de algebră modernă*, Vol. I. Ed. Acad. R.S.R., 1977.
282. I. Purdea. *Tratat de algebră modernă*, Vol. II. Ed. Acad. R.S.R., 1982.
283. H.R. Quillian. Semantic memory. In *Semantic Inf. Processing*, M. Minskz, ed., M.I.T. Press, 1968.
284. G. Radu. *Algebra categoriilor şi functorilor*. Junimea, Iaşi, 1988.
285. R. Ramakrishnan, D. Srivastava, and S. Sudarshan. CORAL: Control, Relations and Logic. In *VLDB*, 1992.
286. R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Sheshadri. Implementation of the ⟨CORAL⟩ deductive database system. In *SIGMOD*, 199
287. R. Ramakrishnan, and J. Ullman. Survey of Research in Deductive Database Systems. In *Journal of Logic Programming*, 1994.

288. R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Databases.* New York: Plenum, 1978, 55-76.
289. R. Reiter. Data Bases: A Logical Perspective. In *Workshop on Data Abstraction, Databases, and Conceptual Modelling*, Pingree Park, Colorado, 1980:174-176.
290. R. Reiter. Towards a logic reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On conceptual Modelling.* Berlin: Springer-Verlag, 1984.
291. R. Reiter. Foundations for knowledge-based systems. In IFIP Congress, 1986.
292. Relational Technology. *Distributed INGRES for the UNIX and VMS Operating Systems. Release 6.* Relational Technology, Nov. 1987.
293. J. Rissanen. Theory of joins for relational databases - a tutorial survey. In *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 64.* Berlin: Springer-Verlag, 1979, 537-551.
294. J. Rissanen. Independent components of relations. *ACM Trans. on Database Syst*. 2(4):314-325. 1982.
295. J. Rissanen. On equivalence of database schemes. In *ACM SIGACT SIGMOD Symp. on Principles of Database System,* 1982:23-26.
296. J. Rissanen and C. Delobel. Decomposition of files, a basis for data storage and retrieval. Report RJ 1220. IBM Research, San Jose, 197
297. I. Rival, ed. *Ordered Sets*. Proc. of NATO Adv. Study Inst., Banf, Canada, 1981. D. Reidel Pub. Company, 1982.
298. J.W. Robbin. *Mathematical Logic. A First Course.* W.A. Benjamin, Inc., 1967.
299. R.Rogers. *Mathematical Logic and Formalized Theories*. North-Holland Publishing Company, 2nd printing, 1974.
300. Rosenberg, A.L. and Snyder, L. Time- and Space-Optimality in B trees. *ACM TODS*, vol. 6, no.1, March 1981, pp.174-18
301. M. A. Roth and H. F. Korth. The design of ¬NF relational databases into nested normal form. In *ACM SIGMOD International Conf. on Management of Data,* 1987:143-157.
302. M. A. Roth, H. F. Korth, and D. S. Batory. SQL/NF: A query language for ¬1NF relational databases. *Information Systems* 12(1):99-114. 1987.
303. M. A. Roth, H. F. Korth, and A. Silberschatz. Extended algebra and calculus for ¬1NF relational databases. *ACM Trans. on Database Syst.* 13(4):389-417. 1988.
304. F. Sandri and J. D. Ullman. Template dependencies: A large class of dependencies in the relational model and its complete axiomatization. *Journal of the ACM* 29(2):363-372. 1982.
305. Y. Sagiv. Can we use the universal instance assumption without using nulls? In *ACM SIGMOD International Conf. on Management of Data,* 1981:108-120.
306. Y. Sagiv. A characterization of globally consistent databases and their correct access paths. *ACM Trans. on Database Systems* 8(2):266-286. 198
307. Y. Sagiv. Evaluation of queries in independent database schemes. *Journal of the ACM* 38(1):120-161. 1991.
308. H.-J. Schek. Towards a basic relational NF2 algebra processor. In *Eleventh International Conference on Very Large Data Bases,* Stockolm, 1985:173-182.
309. H.-J. Schek, H. B. Paul, M. H. Scholl, and G. Weikum. The DASDBS project: Objectives, experiences, and future prospects. *IEEE Trans. on Knowledge and Data Eng.* 2(1):25-4 1990.
310. H.-J. Schek and P. Pistor. Data structures for an integrated database management and information retrival system. *Eighth International Conf. on Very Large Data Bases,* Mexico City, 1982.

311. H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems,* 11(2). 1986.
312. E. Sciore. *The Universal Instance and Database Design.* Ph. D. diss. Princetown University (Dept. of EECS), 1980.
313. E. Sciore. A complete axiomatization of full join dependencies. *Journal of the ACM* 29(2):373-39 1982.
314. D.W. Shipman. The Functional Data Model and the Data Language DAPLEX. *ACM Trans. on Database Syst*. 6(1): 140-173, 1981.
315. A. Silberschatz, M. Stonebraker, and J. D. Ullman. Database systems: Achivements and opportunities. *Communications of the ACM* 34(10):110-120. 1991.
316. J. M. Smith. A normal form for abstract syntax. In *Fourth International Conf. on Very Large Data Bases,* Berlin, 1978:156-162.
317. J.M. Smith, S. Fox, and T. Landers. *Reference Manual for ADAPLEX*. Computer Corporation of America, 1981.
318. J. M. Smith and D. P. C. Smith. Database abstractions: Aggregation and generalization. *ACM Trans. on Database Syst.* 2(2):105-13 1977.
319. J. M. Smith and D. P. C. Smith. Principles of Database Conceptual Design. In *Database Design Techniques, Part I*. Lect. Notes in Comp. Sci., 132, Springer-Verlag, 1982.
320. N. Spyratos. The Partition Model: A Deductive Database Model. *ACM ToDS*, Vol. 12, No.1: 1-32, 1987.
321. D. Srivastava, R. Ramakrishnan, S. Sudarshan, and P. Sheshadri. CORAL++; Adding Object-orientation to a Logic Database Language. In *VLDB*, 199
322. P.M. Stocker, P.M.D. Gray, and M.P. Atkinson eds. *Databases - Role and Structure. An Advanced Course.* Cambridge University Press, 1984.
323. M. Stonebraker, editor. *The INGRES Papers.* Reading, Mass.: Addison Wesley, 1986.
324. M. Stonebraker. Future trends in database systems. *IEEE Trans. on Knowledge and Data Eng.* 1(1):33-44. 1987.
325. Stonebraker, Michael. *Object-Relational Database Systems*. Illustra Information Technologies, (nedatat).
326. M. Stonebraker and G. Kemnitz. The Postgres next-generation database management system. *Comm. of the ACM* 34(10)78-9 1991.
327. Sybase. *What's New in SYBASE SQL Server Release 10.0?*. Sybase, Doc. ID 36440-01-1000-04, May 1994.
328. Tandem Computers. *NonStop SQL, A Distributed, High-Performance, High-Availability Implementation of SQL*. Tech. Rep. 87.4, Tandem Computers, April 1994.
329. T. J. Teorey and J. P. Fry. *Database Design.* Englewood Cliffs, N. J.: Prentice-Hall, 1982.
330. J.W. Thatcher, E.G. Wagner, and J.B. Wright. *Notes on Algebraic Fundamentals for Theoretical Computer Science*. Reprint from J.W. de Bakker and J. van Leeuwen, eds., Found. of Comp. Sci. III, Part 2: Languages, Logic, Semantics, Mathematical Centre Tracts 109: 83-163, 1977.
331. S. J. Thomas. *A non-first-normal-form relational database model.* Ph. D. diss. Vanderbilt University, Nashville, Tenn. 198
332. S. J. Thomas and P. C. Fischer. Nested relational structures. In P. C. Kanellakis and F. P. Preparata, editors, *Advances in Computing Research.* Volume Greenwich, Ct.: JAI Press, 1986, 269-307.
333. R. Topor. Domain-independent formulas and databases. *Theoretical Computer Science* 52:281-306. 1986.
334. D. Tsichritzis and F. H. Lochovski. *Data Models.* Englewood Cliffs, N. J.: Prentice-Hall, 1982.

335. D.-M. Tsou and P. C. Fischer. Decomposition of a relation scheme into Boyce-Codd normal form. *SIGACT News* 14(3):23-27. 1982.
336. Ullman, Jeffrey D. *Principles of Database Systems*. Computer Science Press, 1980.
337. J. D. Ullman. *Principles of Database Systems.* 2nd ed. Potomac, Md.: Computer Science Press, 1982.
338. J. D. Ullman. *Principles of Database Systems.* Potomac, Md.: Computer Science Press, 1982.
339. J. D. Ullman. The U. R. strikes back. *In ACM SIGACT SIGMOD Symp. on Principles of Database Systems,* 1982:10-22.
340. J. D. Ullman. On Kent's 'Consequences of assuming a universal relation'. *ACM Trans. on Database Syst.* 8(4):637-64 198
341. J. D. Ullman. Implementation of logical query languages for databases. *ACM Trans. on Database Syst.* 10(3):289-321. 1985.
342. J. D. Ullman. *Principles of Database and Knowledge Base Systems.* Volume 1. Potomac, Md.: Computer Science Press, 1988.
343. J. D. Ullman. *Principles of Database and Knowledge Base Systems.* Volume 2. Potomac, Md.: Computer Science Press, 1987.
344. P. Valduriez and G. Gardarin. *Analysis and Comparison of Relational Database Systems.* Reading, Mass.: Addison Wesley, 1987.
345. M. Y. Vardi. The decision problem for database dependencies. *Information Processing Letters* 12(5):251-254. 1981.
346. M. Y. Vardi. The Implication and finite implication problems for typed template dependencies. *Journal of Comp. and System Sc.* 28(1):3-28. 1984.
347. M. Y. Vardi. Response to a letter to the editor. *IEEE Software* 5(4):4-6. 1988.
348. M. Y. Vardi. The universal relation data model for logical independence. *IEEE Software* 5(2):80-85. 1988.
349. Y. Vassiliou. Null values in database management: A denotational semantics approach. *ACM SIGMOD International Conf. on Management of Data,* 1979:162-167.
350. Y. Vassiliou. *A Formal Treatment of Imperfection in Database Management.* Ph. D. diss. University of Toronto, 1980.
351. Y. Vassiliou. Functional dependencies and incomplete information. In *Sixth International Conf. on Very Large Data Bases, Montreal,* 1980:260-267.
352. Verso, J. (pen name for Verso team). Verso: A database machine based on non-1NF relations. Technical Report 52 INRIA, 1980.
353. L. Vielle. Recursive Axioms in Deductive Databases: The Query-Subquery Approach. In *Proc. Int. Conf. on Expert DB Syst*., 1986.
354. L. Vielle. Database Complete Proof Production Based on SLD-resolution. *In Proc 4th Int. Conf. on Logic Programming,* 1987.
355. L. Vielle. From QSQ towards QoSaQ: Global Optimization of Recursive Queries. In *Proc. Int. Conf. on Expert DB Syst*., 1988.
356. K. Wang and M. H. Graham. Constant-time maintainability: A generalization of independence. *ACM Trans. on Database Syst.* 17(2):201-246. 1992.
357. D. Warren. Memoing for Logic Programs. *CACM*, 35:3, ACM, march 1992.
358. K. Whang, and S. Navathe. Integrating Expert Systems with DBMS - an Extended Disjunctive Normal Form Approach. In *Information Sciences*, 64, march 1992.
359. K. Wilkinson, P. Lyngbaek, and W. Hasan. The IRIS architecture and implementation. *IEEE Trans. on Knowledge and Data Eng.* 2(1):63-75. 1990.
360. E. Wong. A statistical approach to incomplete information in database systems. *ACM Trans. on Database Syst.* 7(3):470-488. 1982.
361. M. Yannakakis and C. Papadimitriou. Algebraic dependencies. *Journal of Comp. and System Sc.* 21(1):2-41. 1982.

362. C. Zaniolo. *Analysis and Design of Relational Schemata for Database Systems*. Ph. D. diss. UCLA, 1976.
363. C. Zaniolo. Relational views in a database system; support for queries. In *IEEE Int. Conference on Computer Software and Applications*, 1977:267-275.
364. C. Zaniolo. A new normal form for the design of relational database schemas. *ACM Trans. on Database Syst.* 7(3):489-497. 1982.
365. C. Zaniolo. Database relations with null values. *Journal of Comp. and System Sc.* 28(1):142-166. 1984.
366. C. Zaniolo, editor. Special Issues on Databases and Logic. *Data Engineering*, 10 (4). IEEE Computer Society, 1987.
367. C. Zaniolo and M. A. Melkanoff. On the design of relational database schemata. *ACM Trans. on Database Syst.* 6(1):1-47. 1981.
368. C. Zaniolo and M. A. Melkanoff. On the design of relational database schemata for database systems. *ACM Trans. on Database Syst.* 7(1):24-57. 1982.
369. M. M. Zloof. Query-by-example: A database language. *IBM Systems Journal* 16 (4):324-34 1977.
370. C. Mancaş. *Modelarea conceptuală a datelor*. Teză de doctorat, Universitatea Politehnică Bucureşti, Facultatea Automatică şi Calculatoare, 1997.
371. C. Mancaş. Normalizare versus corectitudinea modelării conceptuale a datelor. Constrângeri tuplu de neocolit nici în Modelul Matematic Elementar al Datelor. In Proc. Conferinţei de Informatică Teoretică şi Tehnologii Informatice CITTI'2000, 25-27 mai 2000, Constanța, pp.207-216.
372. V. Markovitz. *ERROL – An Entity-Relationship Role-Oriented Language*. Master thesis, Technion University, Haiffa, 1982.
373. R. Barker. *ORACLE CASE*Method – Entity Relationship Modelling*. Addison Wesley, 1990.
374. R. Barker, C. Longman. *ORACLE CASE*Method – Function and Process Modelling*. Addison Wesley, 1992.

- ✓ „*Foundations of Databases*", de S. Abiteboul, R. Hull şi V. Vianu [72] din 1995 (actuala „biblie" în domeniu)
- ✓ „*Data on the Web: From Relations to Semistructured Data and XML*" de S. Abiteboul, P. Buneman, D. Suciu din [73] 1999
- ✓ „*Database Design for Smarties: Using UML for Data Modeling*", de R.J. Muller [271] din 1997.

375. „Entity-Relationship Modeling. Foundations of Database technology", de B. Talheim [300] din 2000.