Dependențe multivaluate

prof. dr. ing. Mircea Petrescu

Fie schema de relație R, iar X şi Y submulțimi ale schemei R. Spunem atunci că există o dependență multivaluată a submulțimii Y de X, notată cu X–»Y, dacă fiind date valori pentru atributele din X, există o mulțime de zero sau mai multe valori asociate ale atributelor din Y, iar această mulțime de valori Y nu este legată în nici un fel cu valori ale atributelor din R-X-Y.

Într-o manieră formală, vom spune că dependența multiplă X–»Y este valabilă în R, dacă atunci când r este o valoare curentă pentru R, iar t şi s sunt două tupluri din r, astfel încât t[X]=s[X], atunci r conține, de asemenea, tuplurile u şi v, care îndeplinesc condițiile:

1. u[X]=v[X]=t[X]=s[X],
2. u[Y]=t[Y] şi u[R-X-Y]=s[R-X-Y],
3. v[Y]=s[Y] şi v[R-X-Y]=t[R-X-Y].

Cu alte cuvinte, putem schimba între ele valorile Y din t şi s, obținând două tupluri noi care trebuie, de asemenea, să facă parte din r.

Observăm că în definiția de mai sus nu am presupus că X şi Y sunt disjuncte. De asemenea, mai remarcăm că, mai sus, condiția 3 derivă din condițiile 1 şi 2.

Exemplu. O valoare curentă posibilă pentru schema R=CPOLSN este cea de mai jos; pentru simplitate, denumirile cursurilor au fost codificate numeric:

C	P	O	L	S	N
101	Stoian P.	L8	222	Bogdan M.	9
101	Stoian P.	M8	333	Bogdan M.	9
101	Stoian P.	V8	222	Bogdan M.	9
101	Stoian P.	L8	222	Andrei I.	8
101	Stoian P.	M8	333	Andrei I.	8
101	Stoian P.	V8	222	Andrei I.	8

În relație, L8 este „Luni de la ora 8”, etc.

Cazul luat – este simplu, numai un curs, cu doi studenți; totuşi, ies în evidență câteva proprietăți care pot fi valabile pentru orice relație derivată din schema P. Astfel, un curs se poate desfăsura la câteva ore (şi zile) diferite, de fiecare dată în săli diferite. Unui student îi corespunde un tuplu pentru fiecare curs urmat şi pentru fiecare şedință de curs. Nota obținută de student la acest curs este repetată în fiecare tuplu.

În general, ne vom aştepta ca dependența funcțională multiplă C–»OL să fie valabilă, adică există o mulțime de perechi oră-loc (clasă) asociate cu fiecare curs şi neasociate cu celelalte atribute. De exemplu, dacă în definiția formală a unei dependențe multiple introducem:

t = 101 Stoian P. L8 222 Bogdan M. 9
s = 101 Stoian P. M8 333 Andrei I. 8

atunci ne putem aştepta să reuşim să schimbăm (L8, 222) din tuplul t cu (M8, 333) în s, astfel încât să obținem următoarele două tupluri noi:

u = 101 Stoian P. L8 222 Andrei I. 8
v = 101 Stoian P. M8 333 Bogdan M. 9.

Examinând relația r, vedem că tuplurile u şi v fac parte din ea.

Trebuie să accentuăm faptul că dependența multiplă C–»OL este valabilă nu deoarece a fost valabilă în relația r, dată în forma tabulară. Această dependență este satisfăcută, deoarece pentru orice curs c, dacă aceasta are loc la ora o_1 în sala l_1, cu profesorul p_1 şi studentul s_1 cu nota n_1, şi dacă acest curs sa mai ține, de asemenea, la ora o_2 în sala l_2, cu profesorul p_2 şi studentul s_2 cu nota n_2, atunci ne aşteptăm, în virtutea modului în care înțelegem semnificația atributelor, ca acelaşi curs c să se desfăşoare la ora o_1 în sala l_1 cu profesorul p_2 şi studentul s_2 nu nota n_2.

Vom mai remarca, de asemenea, că nu sunt valabile dependențele multiple C–»O şi C–»L. Pentru a arăta acest lucru, să revenim la relația r cu tuplurile t şi s. Dacă C–»O ar fi satisfăcută, ne-am aştepta ca în r să găsim tuplul: 101 Stoian P. L8 333 Andrei I. 8, ceea ce nu este adevărat.

Alte dependențe multiple valabile: C–»SN şi OL–»SN. Există şi dependențe multiple triviale, ca OL–»L.

Se mai poate arăta că, de asemenea, că fiecare dependență funcțională $X \rightarrow Y$ valabilă, implică dependența multiplă X–»Y valabilă.

Axiome pentru dependențele funcționale şi multiple

Dependențele – date pe o mulțime U de atribute. Vom adăuga şi primele trei axiome ale lui Armstrong pentru dependențele funcționale. Avem:

A1. Dacă $Y \subseteq X \subseteq U$, atunci $X \rightarrow Y$ (axioma de reflexibilitate pentru dependențe funcționale).

A2. Dacă $X \rightarrow Y$ este valabilă, iar $Z \subseteq U$, atunci $XZ \rightarrow YZ$ (amplificare).

A3. Dacă $X \rightarrow Y$ şi $Y \rightarrow Z$, atunci $X \rightarrow Z$ (tranzitivitate pentru dependențe multiple).

A4. Dacă X–»Y, atunci Y–»U-X-Y (complementare pentru dependențe multiple).

A5. Dacă X–»Y şi $V \subseteq W$, atunci WX–»VY (amplificare pentru dependențe funcționale).

A6. Dacă X–»Y şi Y–»Z, atunci X–»Z-Y (tranzitivitate pentru dependențe funcționale).

Să facem unele comparații între cele două categorii de axiome. Constatăm, de exemplu, că A4 nu are corespondent în grupul de axiome pentru dependențele funcționale simple. Pe de altă parte, A1 nu are corespondent în A4-A6; totuşi faptul că X–»Y oridecâte ori $Y \subseteq X$ rezultă din A1 şi din regula potrivit căreia, dacă $X \rightarrow Y$, atunci X–»Y (aceasta este axioma A7, care va urma).

Axioma A6 este mai restrictivă decât corespondența sa pentru tranzitivitate, A3. Aserțiunea este mai generală, conform căreia X–»Y şi Y–»Z ar implica X–»Z, este falsă. De exemplu, în cazul schemei R=CPOLSN, am văzut că dependențele C–»OL şi OL–»O sunt valabile, dar C–»O este falsă.

Pentru a compensa – parțial – faptul că A6 este mai slabă decât A3, folosim o versiune mai puternică a axiomei A5 (mai puternică decât axioma de amplificare A2, pentru dependențe funcționale simple). Am fi putu înlocui A2 prin $X \rightarrow Y$ şi $V \subseteq W$ implică $WX \rightarrow VY$, dar pentru dependențele funcționale această regulă se deduce uşor din A1, A2 şi A3.

Ultimele două axiome leagă dependențele funcționale simple şi multiple:

A7. Dacă $X \rightarrow Y$, atunci X–»Y.

A8. Dacă X–»Y, $Z \subseteq Y$, iar pentru o submulțime W de atribute, disjunctă față de Y, avem $W \rightarrow Z$, atunci dependența funcțională simplă $X \rightarrow Z$ este şi ea valabilă.

Se poate demonstra că ansamblul de axiome A1-A8 este sigur şi complet.

Reguli de inferență pentru dependențele multiple

Regulile care urmează sunt utile, împreună cu axiomele prezentate, pentru a efectua inferența în contextul unei mulțimi de dependențe simple sau multiple. Sunt în continuare valabile regulile privind reuniunea, pseudo-tranzitivitatea şi descompunerea, date în cazul dependențelor funcționale. Avem în plus:

1. Dacă X–»Y şi X–»Z, atunci X–»YZ (regula reuniunii pentru dependențe multiple).

2. Dacă X–»Y şi WY–»Z, atunci WX–»Z-WY (regula de pseudotranzitivitate pentru dependenţe multiple).
3. Dacă X–»Y şi XY→Z, atunci X→Z-Y (regula combinată de pseudotranzitivitate).
4. Dacă X–»Y şi X–»Z, atunci sunt, de asemenea, valabile şi X–»Y∩Z, X–»Y-Z, X–»Z-Y (regula de descompunere pentru dependenţe multiple).

Remarcăm că regula de descompunere pentru dependenţele multiple este mai slabă decât regula de descompunere pentru dependenţe simple. Ultima regulă ne permite să deducem imediat, din X→Y, că X→A pentru fiecare atribut A∈Y. Regula de descompunere pentru dependenţe multiple ne permite să deducem X–»A din X–»Y dacă putem găsi o submulţime Z astfel încât X–»Z şi este valabilă fie Z∩Y=A, fie Y-Z=A.

Regula de descompunere pentru dependenţe funcţionale şi regula de reuniune, ne permite să enunţăm o teoremă privind mulţimile Y astfel încât X–»Y pentru X dat.

Teoremă. Dacă U este mulţimea tuturor atributelor, atunci putem partiţiona U-X în mulţimile de atribute Y_1, Y_2, ... Y_k, astfel încât dacă Z⊆U-X, atunci X–»Z dacă şi numai dacă Z este reuniunea unora din mulţimile Y.

Demonstraţie. Partiţionarea mulţimii U-X se începe considerând toate elementele din U-X ca făcând parte din acelaşi bloc. Să admitem acum că la un moment dat avem partiţia formată din blocurile W_1, W_2, ..., W_n, iar X–»W_i pentru i=1, 2, ..., n. Dacă X–»Z, iar Z nu este reuniunea unora din blocurile W_i, atunci se înlocuieşte fiecare W_i, respectând restricţia: W_i∩Z şi W_i-Z sunt ambele nevide, prin W_i∩Z şi W_i-Z. Potrivit regulei de descompunere, X–»W_i∩Z şi X–»W_i-Z. Întrucât nu putem partiţiona la nesfârşit o mulţime finită de atribute, vom ajunge în final la situaţia în care fiecare Z, astfel încât X–»Z, este reuniunea unora din blocurile partiţiei. Conform regulii de reuniune, X „multideterminã” reuniunea oricărei mulţimi de blocuri.

Mulţimile Y_1, Y_2, ..., Y_k, construite pentru o submulţime X⊆U, pornind de la o mulţime F de dependenţe fuuncţionale şi multiple, constituie baza de dependenţă pentru X în raport cu F.

Dacă X→Y_i, atunci Y_i trebuie să conţină numai un atribut, potrivit regulei de descompunere pentru dependenţele funcţionale şi axiomei A7.

Exemplu. Într-un exemplu anterior, am văzut că C–»OL. Prin urmare, folosind regula de complementare, C–»PSN. De asemenea, ştim că C–»P. Aplicând regula de descompunere, obţinem C–»SN. Se poate verifica că nici un atribut individual, în afară de P sau de însuşi C, nu este multideterminat de C. Aşadar, baza de dependenţă pentru C este {P, OL, SN}.

Închideri ale dependenţelor funcţionale şi multiple

Fie F o mulţime de dependenţe funcţionale şi multiple. Putem calcula mulţimea F^+ a dependenţelor funcţionale şi multiple implicate logic de F, folosind axiomele A1-A8. Acest procedeu de calcul consumă timp, durata de execuţie fiind direct proporţională cu exp(#F). Pe de altă parte, dorim adesea să aflăm numai o dependenţă particulară X→Y sau X–»Y rezultă din F, dacă, de exemplu, dorim să eliminăm dependenţele redundante.

Pentru a verifica existenţa unei dependenţe funcţionale multiple X–»Y, este suficient să determinăm baza de dependenţă pentru X şi să vedem dacă Y-X este reuniunea unor mulţimi din context. Pentru exemplul schemei CPOLSN, ştim că C–»CPSN, deoarece PSN este reuniunea lui P cu SN. De asemenea, C–»OLSN, dar C–»PO nu este valabilă, deoarece PO intersectează blocul OL al bazei de dependenţă, (însă) PO nu include în întregime OL.

Se poate arăta (Beeri), că pentru a calcula baza de dependenţă a lui X în raport cu F, este suficient să fie calculată baza în raport cu o mulţime de dependenţe multiple M; mulţimea M conţine:

- toate dependenţele multiple din F;

- pentru fiecare dependență funcțională $X \rightarrow Y$ din F, mulțimea dependențelor multiple $X \twoheadrightarrow A_1$, ..., $X \twoheadrightarrow A_n$, unde $Y=A_1...A_n$ pentru atributele $A_1,...,A_n$.

După ce a fost calculată baza de dependență, corespunzător mulțimii de dependențe multiple M, din această bază urmează a fi extrase dependențele funcționale netriviale. Se poate arăta că dacă X nu include A, atunci $X \rightarrow A$ este valabilă dacă şi numai dacă:

- A este o mulțime cu un singur element a bazei de dependență pentru X corespunzător mulțimii de dependențe M;
- există o mulțime de atribute Y, în afară de A, astfel încât $Y \rightarrow Z \in F$ iar $A \in Z$.

Algoritm pentru calculul bazei de dependență a unei submulțimi X în raport cu M (Beeri, timp polinomial).

Intrare: o mulțime de dependențe multiple M, pe o mulțime de atribute U, şi o mulțime $X \subseteq U$.

Ieşire: baza de dependență pentru X în raport cu M.

Metoda:

1. fie T mulțimea de mulțimi $Z \subseteq U$, astfel încât pentru o dependență $W \twoheadrightarrow Y$ din M, avem $W \subseteq X$, iar Z este fie Y-X, fie U-X-Y.
2. până când T va consta dintr-o colecție disjunctă de mulțimi, se va găsi o pereche de mulțimi Z_1 şi Z_2 în T care nu sunt disjuncte; aceste mulțimi se vor înlocui cu Z_1-Z_2, Z_2-Z_1 şi $Z_1 \cap Z_2$, eliminând mulțimea vidă, în caz că vreuna din Z_1 şi Z_2 este conținută în cealaltă. Fie S colecția finală de mulțimi.
3. până în situația în care nu se mai pot face schimbări în S, se caută dependențele $V \twoheadrightarrow W$ în M şi o mulțime Y în S astfel încât Y intersectează W dar nu V. Se înlocuieşte Y cu $Y \cap W$ şi Y-W în S.
4. colecția finală S de mulțimi este baza de dependență pentru X.

Joncțiuni fără pierderi

Am prezentat, anterior, un algoritm care ne arată când o descompunere (R_1, .., R_k) a unei scheme de relație R are o joncțiune fără pierderi, în ipoteza că numai dependențele satisfăcute de relațiile care derivă din schema R sunt funcționale. Algoritmul se poate generaliza pentru cazul dependențelor multiple şi va fi:

1. se construieşte tabloul elementelor a şi b, ca în forma inițială a algoritmului;
2. se construieşte o colecție de tablouri pornind de la un tablou T care a fost deja construit şi procedând în unul din următoarele două moduri:

a) identificând două simboluri ca urmare a unei dependențe funcționale, ca în cazul algoritmului „de bază" (pentru dependențe funcționale „simple"), sau

b) luând o dependență multiplă $X \twoheadrightarrow Y$ şi două linii t_1 şi t_2 ale unui tablou, astfel încât $t_1[X]=t_2[X]$, şi adăugând linia u, astfel încât $u[X]=t_1[X]$, $u[Y]=t_1[Y]$, $u[R-X-Y]=t_2[R-X-Y]$, dacă u nu se găseşte deja în tabloul T.

3. Deoarece în aplicarea procedeului nu se crează simboluri noi a şi b, numărul de tablouri este finit. Dacă oricare din tablouri conține într-o linie numai simboluri a, joncțiunea este fără pierderi.

Dacă însă schema R a bazei de date se descompune în două scheme, există un test mult mai simplu pentru proprietatea de joncțiune fără pierderi, potrivit teoremei următoare:

Teoremă. Fie R o schemă de relație şi $\rho=(R_1, R_2)$ o descompunere a lui R. De asemenea, fie F o mulțime de dependențe funcționale şi multiple pe atributele din R. Atunci ρ are o joncțiune fără

pierderi dacă şi numai dacă $R_1 \cap R_2 \twoheadrightarrow R_1 - R_2$ sau, în mod echivalent, $R_1 \cap R_2 \twoheadrightarrow R_2 - R_1$, prin regula de complementare.

Demonstrația – exercițiu.

A patra formă normală (FN4)

FN4 este o generalizare a Formei Normale Boyce-Codd, aplicată schemelor de relație cu dependențe multiple.

Fie R o schemă de relație şi F o mulțime de dependențe pe atributele din R. Spunem că schema R este în a patra formă normală dacă, oridecâte ori există o dependență multiplă $X \twoheadrightarrow Y$, unde Y nu este vidă şi nu este o submulțime a lui X, iar XY nu include toate atributele din R, atunci X conține o cheie pentru R.

Dacă F conține numai dependențele funcționale, atunci oridecâte ori R este în FN4, este şi în FNBC. Demonstrația – simplă (Ullman).

Se poate găsi o descompunere a schemei R în $\rho=(R_1, R_2, ..., R_k)$, astfel încât ρ admite o joncțiune fără pierderi în raport cu F şi fiecare R_i este în FN4. Pentru a arăta acest lucru, se începe cu $\rho=R$, ca în algoritmul de descompunere în FNBC. Dacă în ρ există o schemă de relație, care nu este în FN4 în raport cu F proiectată pe S, atunci trebuie să existe în S o dependență $X \twoheadrightarrow Y$, unde X nu include o cheie pentru S, Y nu este vidă sau o submulțime a lui X, iar $XY \neq S$. Putem presupune că X şi Y sunt disjuncte, deoarece $X \twoheadrightarrow Y - X$ decurge din $X \twoheadrightarrow Y$ folosind A1, A7 şi regula de descompunere. Înlocuim apoi pe S cu $S_1=XY$ şi $S_2=S-X$, care trebuie să fie două scheme de relație cu mai puține atribute decât S. Atunci, potrivit teoremei care ne arată că algoritmul de descompunere în FN3 (dat anterior) cu conservarea dependențelor este corect, deoarece $S_1 \cap S_2 \twoheadrightarrow S_1 - S_2$, joncțiunea lui S_1 şi S_2 este fără pierderi în raport cu F proiectată pe S.

Se poate arăta că descompuneri repetate, ca cea de mai sus, generează o mulțime de scheme de relație, care are o joncțiune fără pierderi în raport cu F. Un detaliu important, care trebuie precizat, în ipoteza că sunt date R, F şi $S \subseteq R$, este modul de calcul al mulțimii de dependențe valabile în S, adică proiecția lui F pe S. Procedeul este următorul:

1. se calculează F^+;
2. pentru fiecare $X \rightarrow Y$ din F^+, dacă $X \subseteq S$, atunci $X \rightarrow Y \cap S$ este valabilă în S;
3. pentru fiecare $X \twoheadrightarrow Y$ din F^+, dacă $X \subseteq S$, atunci $X \twoheadrightarrow Y \cap S$ este valabilă în S;
4. nu se mai pot deduce alte dependențe pentru S din faptul că F este valabilă pentru schema de relație R.

Exemplu. Ne reamintim că în cazul schemei CPOLSN aveam acoperirea minimală: $C \rightarrow P$, $CS \rightarrow N$, $OL \rightarrow C$, $OS \rightarrow L$, $OP \rightarrow L$. Se poate arăta că pornind de la dependențele funcționale de mai sus şi de la dependența multiplă $C \twoheadrightarrow OL$, putem deduce toate dependențele multiple, despre care se poate constata – intuitiv – că sunt valabile.

De exemplu, am văzut că $C \twoheadrightarrow OL$ şi $C \rightarrow P$ implică $C \twoheadrightarrow SN$. De asemenea, ştim că $OL \rightarrow CP$, deci $OL \twoheadrightarrow CP$. Folosind regula de complementare, $OL \twoheadrightarrow SN$. Cu alte cuvinte, fiind date o oră şi o sală, există o mulțime asociată de perechi student-notă, respectiv studenții înscrişi la cursul care are loc în acea sală şi la acea oră, asociați cu notele obținute la acel curs.

Se mai pot deduce, în continuare, şi alte dependențe multiple; exercițiu.

Să aducem schema CPOLSN la FN4. Putem începe cu dependența multiplă $C \twoheadrightarrow OL$, care nu satisface condițiile FN4 deoarece atributul (mulțimea de atribute!) C nu conține o cheie (singura cheie pentru schema CPOLSN este SO). Vom descompune atunci CPOLSN în COL şi CPSN. Schema COL are cheia OL. Dependența multiplă $C \twoheadrightarrow OL$ nu încalcă restricțiile FN4 pentru schema

COL, deoarece părțile sale stângă şi dreaptă, împreună, conțin toate atributele din COL. Nicio altă dependență funcțională sau multiplă, proiectată pe COL, nu încalcă FN4, deci schema COL nu trebuie descompusă în continuare.

Nu la fel stau lucrurile cu CPSN. Singura cheie a acestei scheme este CS; mai observăm dependența multiplă C–»P, care decurge din C→P. Ca urmare, secționăm CPSN în CP şi CSN. Ultimele două scheme sunt ambele în FN4, în raport cu dependențele lor proiectate; aşadar, am obținut descompunerea schemei CPOLSN în COL, CP şi CSN. Această descompunere are o joncțiune fără pierderi şi este în FN4.