O metodă de construire a tuturor acoperirilor neredundante existente de dependenţe funcţionale

Dorian SARANCIUC
Universitatea Tehnică a Moldovei
Catedra "Automatică şi Tehnologii Informaţionale"
saranciuc@mail.utm.md

***Rezumat* — În articol este abordată problema construirii tuturor acoperirilor neredundante existente ale unei mulţimii de dependenţe funcţionale. Aceasta este posibil, utilizînd ca bază un algoritm procedural conventional, prin permutarea dependenţelor în mulţimea initială şi calcularea acoperirii pentru fiecare mulţime nou formată. Însă pentru o mulţime cu *n* dependenţe numărul permutărilor pesibile este *n!*, ceea ce ar duce la creşterea enormă a timpului de calcul. A fost propusă o metodă de a minimiza numărul permutărilor, care constă în eliminarea din lista permutărilor a dependenţelor neesenţiale. Au fost propuse criteriile pentru determinarea dependenţelor neesenţiale, formulate în baza noţiunilor de atribute neesenţiale şi recuperabile. A fost arătată eficienţa metodei propuse în baza unor exemple. A fost realizat algoritmul de căutare a tuturor acoperirilor neredundante existente într-un limbaj procedural şi estimat timpul de calcul.**

***Cuvinte cheie* — acoperire neredundantă, dependenţă funcţională neesenţială, atribute neesenţiale şi recuperabile.**

I. ÎNTRODUCERE

Bazele de date (BD) constituie nucleul tuturor sistemelor informaţionale (SI) moderne, marea majoritate a lor fiind de tip relaţional. Utilizarea efectivă a SI depinde, în mare măsură, de eficienţa şi corectitudinea funcţionării BD integrate în el. Complexitatea crescîndă a SI impune necesitatea automatizării proiectării BD la toate nivelele, în special la cel conceptual şi logic. Proiectarea conceptuală eficientă în prezent este posibilă utilizînd sisteme de modelare vizuală în baza UML. Însă, necătînd la faptul fundamentului ştiinţific dezvoltat al modelului relaţional, proiectarea logică a BD, inclusiv normalizarea, nu este realizată în măsura cuvenită de către sistemele comerciale existente Astfel, proiectarea logică a BD, în esenţă, rămîne manuală.

Dezvoltarea vertiginoasă a limbajelor de programare orientate pe obiect deschide noi perspective pentru realizarea algoritmilor utilizaţi în proiectarea logică a BD şi construirea în baza lor a unor sisteme care ar automatiza la maxim toate etapele de proiectare a BD, astfel uşurînd munca grea a proiectantului BD. Un algoritm de bază realizat într-un astfel de sistem trebuie să fie algoritmul de construire a acoperirii neredundante a unei milţimi de dependenţe funcţionale.

II. PARTEA TEORETICĂ

Dependenţele funcţionale au fost propuse de către E.Codd în 1970 [1] în calitate de constrîngeri de integritate cu scopul de a reflecta adecvat lumea reală în baza de date. Însă existenţa şi menţinerea lor în BD poate duce la anomalii şi redundanţă. Asigurarea integrităţii datelor şi minimizarea redundanţei constituie probleme importante ale proiectării logice a BD. În algoritmii de construire a schemei logice a BD dependenţele funcţionale au un rol primordial, deoarece sînt utilizate în calitate de date iniţiale. Una din sarcinile normalizării este construirea mulţimilor neredundante de dependenţe funcţionale.

Un algoritm pentru construirea acoperirii neredundante este prezentat în [2]. Rezultatul algoritmului depinde de ordinea examinării dependenţelor funcţionale din mulţimea iniţială. Astfel, pot exista diferite acoperiri neredundante pentru o mulţime de dependenţe funcţionale. Oricare din acoperirile neredundante obţinute poate fi utilizată în continuare pentru proiectarea logică a BD. Deaceea, un algoritm care ar construi toate acoperirile neredundante existente pentru mulţimea iniţială ar fi binevenit atît pentru proiectarea BD, cît şi pentru procesul de instruire.

Realizarea algoritmului NONREDUN(F,G) propus în [2] într-un limbaj procedural furnizează numai o singură soluţie. Obţinerea tuturor soluţiilor existente ar fi posibilă prin permutarea dependenţelor în mulţimea iniţială, calculînd acoperirea pentru fiecare variantă nouă a ordinii de plasare a dependenţelor. Însă, în acest caz numărul de permutări posibile creşte considerabil odată cu creşterea numărului de dependenţe în mulţimea iniţială (pentru *n* dependenţe avem *n!* permutări). Astfel, pentru o mulţime cu un număr mare de dependenţe timpul de calcul poate deveni extrem de mare.

Este evident, că nu toate permutările dependenţelor în mulţimea iniţială vor aduce la acoperiri diferite. În acest caz ar fi binevenită o tehnică de minimizare a numărului de permutări.

În lucrarea [3] a fost propusă o metodă de minimizare a numărului de permutări. Ideea metodei este următoarea: se formează o listă a permutărilor, care iniţial coincide cu mulţimea iniţială de dependenţe. Mai apoi, se exclud din listă "dependenţele neesenţiale", permutarea cărora nu aduce la acoperiri noi. Pentru identificarea "dependenţelor neesenţiale" au fost propuse cîteva criterii.

Mai tîrziu, în [4], au fost formalizate noţiunile de atribute *esenţiale* şi *recuperabile*. Mai jos sînt expuse criteriile de identificare a dependenţelor neesenţiale

propuse în [3], formulate în baza noţiunilor de atribute neesenţiale şi recuperabile:

1) Nu are rost să permutăm dependenţele în mulţimea iniţială, dacă ea este neredundantă.

Întradevăr, conform definiţiei acoperirii neredundante [5], din ea nu poate fi exclusă nici o dependenţă, prin urmare este imposibil de a obţine cu ajutorul algoritmului NONREDUN(F,G) dintr-o mulţime nereduntantă altă mulţime nereduntantă, care să difere de prima mulţime (a nu se confunda cu construirea altor tipuri de acoperiri cu ajutorul altor algoritmi).

2) Nu vor participa la permutare dependenţele care au în partea dreaptă minimum un atribut neesenţial (care nu se conţine în partea dreaptă a altor dependenţe).

De exemplu, în mulţimea F={A→BC, A→D, A→E, E→C} dependenta A→D nu poate fi redundantă, deoarece excluderea ei ar aduce la pierderea atributului neesenţial D. Prin urmare, această dependentă este neesenţială şi permutarea ei nu poate genera o acoperire nouă.

3) Nu are rost să permutăm dependenţa funcţională X→Y dacă ea nu conţine în partea dreaptă atribute recuperabile (închiderea lui X în raport cu mulţimea F\X→Y va fi egală cu X: $X^{+}_{F\backslash X->Y}=X$)

De exemplu, în mulţimea F={A→BC, A→D, A→B, E→C} în dependenta E→C atributul C nu este recuperabil, deoarece $E^{+}_{F\backslash E->C}$ = E, şi deci, dependenţa E->C nu este esenţială.

Astfel analiza prealabilă a dependenţelor permite de a spori productivitatea calculelor, deoarece fiecare dependenţă exclusă din lista permutărilor minimizează timpul de calcul de zeci sau chiar sute de ori (fig.1).

III. PARTEA EXPERIMENTALĂ

Pentru a ilustra eficienta metodei propuse pentru micşorarea numărului de permutări sînt aduse exemple de calcul a tuturor acoperirilor neredundante pentru două mulţimi de dependenţe funcţionale.

Exemplul 1.

Fie G={AB->C, BC->D, BE->C, **CD->B**, CE->AF, CF->BD, C->A, D->EF, F->E}.

În rezultatul aplicării algoritmului NONREDUN(G,H) dependenţa CD->B s-a dovedit a fi redundantă.

Acoperirea neredundantă obţinută este H={AB->C, BC->D, BE->C, CE->AF, CF->BD, C->A, D->EF, F->E}.

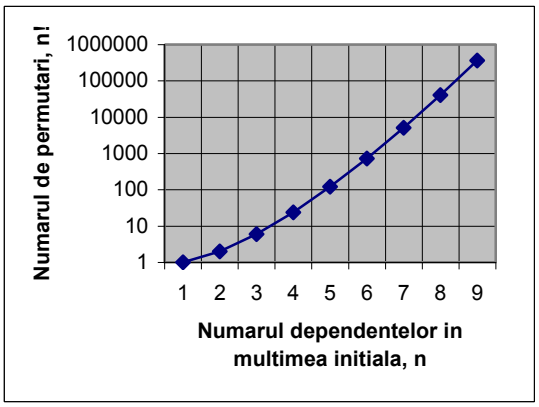
Vom aplica acuma asupra mulţimii iniţiale G criteriile propuse mai sus.

1) Conform primului criteriu, cînd mulţimea iniţială G este redundantă, are sens de a aplica permutările pentru a găsi alte acoperiri neredundante existente.
2) În mulţima G nu există dependenţe neesenţiale care ar conţine în partea dreaptă măcar un atribut care nu se conţine în părţile drepte ale altor dependenţe.
3) Dependenţele AB->C, BE->C, C->A, D->EF, F->E vor fi excluse din lista permutărilor deoarece părţile drepte a lor nu contin atribute recuperabile (închiderile determinanţilor lor $X^{+}_{F\backslash X->Y}=X$ vor fi egale cu X.

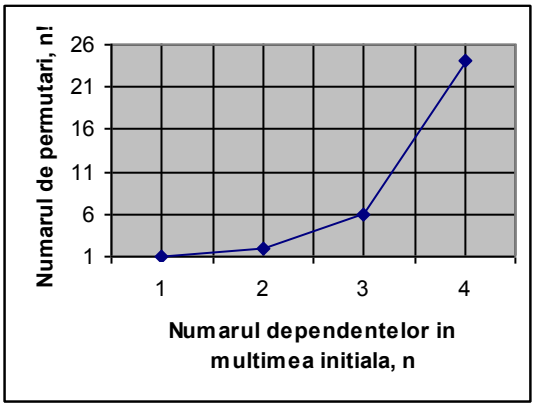
Cele patru dependenţe rămase (BC->D, CD->B, CE->AF, CF->BD) nu satisfac criteriilor propuse pentru a fi excluse din lista de permutări.

Fără analiza preventivă şi excluderea dependenţelor neesenţiale numărul de permutări pentru toate dependenţele din mulţimea iniţială ve fi |G|!=9!=362880 (fig.1a).

După excluderea dependenţelor neesenţiale numărul permutărilor pentru dependenţele rămase |G|!=4!=24 (fig.1b). Însă, după efectuarea tuturor permutărilor nu au fost găsite alte acoperiri neredundante diferite de dependenţe pentru mulţimea iniţială G.



a)



b)

Fig.1. Relaţia între numărul de permutări şi numărul de dependenţe funcţionale în mulţimea iniţială
a) pînă şi *b)* după excluderea dependenţelor neesenţiale

Exemplul 2.

Fie G={A->B, **BE->A**, BC->D, BE->C, CD->B, AB->C, CE->AF, DE->BF}. În rezultatul aplicării NONREDUN(G,H) dependenţa BE->A s-a dovedit a fi redundantă: $(BE)^{+}_{F\backslash BE->A}$ = ABCDEF, A⊆ABCDEF Acoperirea neredundantă obţinută este H={A->B, BC->D, BE->C, CD->B, AB->C, CE->AF, DE->BF. Vom aplica la mulţimea iniţială G criteriile propuse:

1) Mulţimea iniţială este redundantă, respectiv vom aplica metoda permutărilor.
2) Dependenţa BC->D va fi exclusă din lista permutărilor. Vom plasa dependenţa pe ultimul loc în mulţimea G.

3) Dependenţele (A->B, CD->B, AB->C, CE->AF, DE->BF) sînt neesenţiale, vor fi excluse din lista permutărilor şi vor fi plasate la sfîrşitul mulţimii iniţiale.

În rezultatul analizei complete a mulţimii iniţiale în lista permutărilor au rămas doar două dependenţe, care vor fi permutate, formînd mulţimile G1 şi G2:

G1={**BE->A, BE->C**, A->B, BC->D, CD->B, AB->C, CE->AF, DE->BF}

G2={**BE->C, BE->A**, A->B, BC->D, CD->B, AB->C, CE->AF, DE->BF}

La construirea acoperirii neredundante pentru mulţimea G1 obţinem o mulţime identică cu mulţimea H, construită din mulţimea iniţială G, dependenţa BE->A fiind identificată redundantă.

Construirea acoperirrii neredundante pentru G2 va da o altă mulţime, redundantă fiind dependenţa BE->C.

Astfel, în loc de 40320 permutări pentru mulţimea iniţială G (pentru opt dependenţe numărul de permutări este 8!=40320) au fost făcute doar 2 permutări – 2!=2.

Timpul experimental de calcul al unei acoperirii neredundante prin algoritmul NONREDUN(F,G) la calculator constituie circa 1ms. Pentru 7 dependenţe în mulţimea iniţială, timpul de calcul, aplicînd pemutările fără a exclude dependenţele neesenţiale constituie 5,04 sec, pentru 8 dependenţe – 40,32 sec, pentru 10 dependenţe – 60,48 min, iar pentru 15 dependenţe – 357288,08 ore (mai mult de 40 ani!).

IV. CONCLUZII

1. A fost realizat algoritmul de construire a tuturor acoperirilor neredundante existente pentru o mulţime de dependenţe funcţionale într-un limbaj de proramare procedural, aplicînd permutarea dependenţelor în mulţimea iniţială.
2. Pentru o mulţime cu *n* dependenţe sînt posibile *n!* mulţimi cu o ordine diferită a dependenţelor. Construirea acoperirilor pentru fiecare dintre aceste mulţimi duce la creşterea enormă a timpului de calcul.
3. Pentru reducerea timpului de calcul a fost propusă o metodă bazată pe excluderea din lista permutărilor a dependenţelor neesenţiale. Astfel, excluderea unei dependenţe poate minimiza numărul de permutări de zeci, sute sau chiar mii de ori.
4. Au fost reformate criteriile de identificare a dependenţelor ne esenţiale, utilizînd noţiunele de atribute neesenţiale şi recuperabile.
5. Au fost aduse exemple de construire a tuturor acoperirilor neredundante pentru mulţimi de dependenţe funcţionale utilizînd metoda propusă. A fost analizată eficienţa metodei.

BIBLIOGRAFIE

[1]. Codd E.F. A Relational Model of Data for Large Shared Data Banks. Comm. ACM, 1970, V.13, N6, p.337-387.

[2]. V. Cotelea. Baze de date relaţionale. Proectarea logică. Chişinău, ASEM, 1997.

[3]. Д. Саранчук, А. Голубев, Д. Киселев. К вопросу построения неизбыточных покрытий. CNTEI-2006, UTM, Chişinău, 2006.

[4]. Vitalie Cotelea.Modele şi algoritmi de proiectare logică a bazelor de date. Chişinău, ASEM, 2011.

[5]. Paredaens, J. About Fuctional Dependencies in a Database Structure and their Coverings. Phillips MBLE Lab. Report 342, 1977.