Descompunerea schemelor de relație. Joncțiuni fără pierderi. Descompuneri cu păstrarea dependențelor funcționale.

prof. dr. ing. Mircea Petrescu

Descompunerea schemelor de relație

Fie $R=\{A_1, A_2, ..., A_n\}$. Denumim descompunere a schemei R înlocuirea acesteia prin colecția $\rho=\{R_1, R_2, ..., R_n\}$, unde $R_i \subseteq R$, astfel încât $R_1 \cup R_2 \cup ... R_k = R$. Submulțimile R_i nu trebuie să fie neapărat disjuncte.

În general, se consideră că descompunerea schemelor de relație este de dorit atunci când trebuie rezolvate probleme ca eliminarea redundanțelor. Prudența față de descompunerea schemelor – determinată de faptul că în cazul schemelor descompuse, trebuie calculate mai multe joncțiuni decât atunci când schemele de relație nu sunt descompuse. Desigur, descompunerea schemelor are unele avantaje.

O problemă pe care ne-o punem: dacă am descompus o schemă de relație R în ρ, în ce mod vom obține, dacă dorim, o relație r care este valoarea curentă a schemei R? Întrebarea se pune, deoarece practic noi realizăm relațiile elementare în care se descompune r.

Relația r se poate obține relativ simplu (deşi nu „ieftin"), dacă schema R admite o „joncțiune fără pierderi" în raport cu o mulțime de dependențe funcționale F. Vom spune, în acest caz, că descompunerea este cu „joncțiune fără pierderi".

În această situație, r este dată de: $r = \prod_{R_1}(r) \bowtie \prod_{R_2}(r) \bowtie ... \bowtie \prod_{R_k}(r)$, unde $\bowtie$ este operatorul de joncțiune naturală.

Exemplu. Fie R=ABCD, iar $F=\{A \rightarrow C, BC \rightarrow D\}$. Fie o realizare (valoare curentă) a schemei R:

r=

A	B	C	D
a	p	b	m
m	q	c	p
b	a	a	q

Fie $r_{AC}=\Pi_{AC}(r)$, $r_{BCD}=\Pi_{BCD}(r)$.

Avem imediat:

r_{AC}=

A	C
a	b
m	c
b	a

r_{BCD}=

B	C	D
p	b	m
q	c	p
a	a	q

În continuare:

$r_{AC} \times r_{BCD}$=

A	C	B	C	D
a	b	p	b	m
a	b	q	c	p
a	b	a	a	q
m	c	p	b	m
m	c	q	c	p
m	c	a	a	q
b	a	p	b	m
b	a	q	c	p
b	a	a	a	q

Rezultă, aşadar:

A	B	C	D

$r_{AC} \triangleright\triangleleft r_{BCD}=$

a	p	b	m
m	q	c	p
b	a	a	q

Sau: $r = r_{AC} \triangleright\triangleleft r_{BCD} = \prod_{AC}(r) \triangleright\triangleleft \prod_{BCD}(r)$.

Deoarece descompunerea cu joncțiune fără pierderi prezintă aspecte importante pentru teoria relațiilor şi aduce după sine unele avantaje, vom releva unele proprietăți ale sale mai în profunzime.

Fie $\rho=(R_1, R_2, ...,R_k)$. Vom nota atunci cu $m_\rho(r)$ joncțiunea naturală a proiecțiilor realizării r pe schemele de relație din ρ, sau: $m_\rho(r) = \overset{k}{\underset{i=1}{\triangleright\triangleleft}} \prod_{R_i} r$ (deci o „aplicație" de tip proiecție-joncțiune).

Ținând seama de această notație, putem exprima condiția privind joncțiunea fără pierderi în raport cu mulțimea de dependențe funcționale F astfel: pentru ∀r care satisface F, avem $r=m_\rho(r)$.

De asemenea, dacă t este un tuplu, iar X o listă de atribute, vom defini t[X] ca fiind lista componentelor din tuplu t pentru atributele din X. De exemplu, $\Pi_X(r)$ poate fi exprimată în forma $\{t[X] \mid t \in r\}$.

O lemă importantă: fie R o schemă de relație, $\rho=(R_1, R_2, ..., R_k)$ o descompunere a schemei R, r o relație pentru schema R, iar $r_i=\Pi_{Ri}(r)$. Atunci avem:

a) $r \subseteq m_\rho(r)$;
b) dacă $s=m_\rho(r)$, atunci $\Pi_{Ri}(s)=r_i$;
c) $m_\rho(m_\rho(r))=m_\rho(r)$.

Demonstrație:

a) fie $t \in r$. Atunci pentru fiecare i, $t_i=t[R_i] \in r_i$. Rezultă, prin definiția joncțiunii naturale, că $t \in m_\rho(r)$, deoarece t coincide cu t_i pe atributele din R_i pentru ∀i.

b) Din a) rezultă că $r \subseteq s$; prin urmare $\prod_{R_i}(r) \subseteq \prod_{R_i}(s)$; sau, cu alte cuvinte, $r_i \subseteq \prod_{R_i}(s)$. Pentru a arăta că $\prod_{R_i}(s) \subseteq r_i$ (punctul b) din enunțul lemei), vom admite, pentru o valoare i particulară, că $t_i \in \prod_{R_i}(s)$. Atunci, ∃ un tuplu t în s, astfel încât $t[R_i]=t_i$. Dar, întrucât $t \in s$, va exista un tuplu $u_j \in r_j$, pentru fiecare j, astfel încât $t[R_j]=u_j$. Însă $t[R_i]=t_i$, aşadar $t_i \in r_i$ şi, prin urmare, $\prod_{R_i}(s) \subseteq r_i$. Rezultă deci că $r_i = \prod_{R_i}(s)$.

c) Dacă $s=m_\rho(r)$, rezultă din b) că $\prod_{R_i}(s) = r_i$. Aşadar, $m_\rho(s) = \overset{k}{\underset{i=1}{\triangleright\triangleleft}} r_i = m_\rho(r)$.

Observăm că dacă pentru fiecare valoare i, r_i este o valoare curentă (relație) a unei scheme R_i, iar $s = \overset{k}{\underset{i=1}{\triangleright\triangleleft}} r_i$, atunci nu trebuie îndeplinită în mod necesar egalitatea: $\prod_{R_i}(s) = r_i$. Cauza pentru care putem face această afirmație: relația r_i poate conține tupluri „slab legate". Pentru a ilustra această idee, fie schemele $R_1=AB$, $R_2=BC$, precum şi relațiile $r_1=\{a_1b_1\}$ şi $r_2=\{b_1c_1, b_2,c_2\}$. Rezultă atunci imediat $s=\{a_1b_1c_1\}$, iar $\prod_{R_2}(s) = \prod_{BC}(s) = \{b_1c_1\}$, deci $\prod_{R_2}(s) \neq r_2$. Putem afirma că tuplul $\{b_1c_1\}$ din r_2 este „slab legat" în această relație, întrucât el constituie $\prod_{R_2}(s)$ fără „a antrena după sine" tuplul $\{b_2c_2\}$ şi a determina astfel respectarea egalității $\prod_{R_2}(s) = r_2$.

Memorarea tuplurilor „slab legate" este un avantaj al descompunerii relațiilor. Pe de altă parte, însă, descompunerea conduce la necesitatea de a efectua mai multe joncțiuni pentru a răspunde la mesajele de interogare, decât în cazul relațiilor nedescompuse. Decizia de a descompune relații →

trebuie atent analizată. Caz în care descompunerea se recomandă → pentru eliminarea – sau reducerea – redundanțelor.

Verificarea posibilității de descompunere cu joncțiune „fără pierderi”

Algoritmul care urmează → pentru a verifica dacă o descompunere are o joncțiune fără pierderi în raport cu o mulțime de dependențe funcționale.

Intrarea algoritmului: schema $R=A_1A_2...A_n$, mulțimea de dependențe F, descompunerea $\rho=(R_1R_2...R_k)$.

Ieşirea algoritmului: decizia dacă ρ admite o joncțiune fără pierderi.

Metoda: se construieşte o tabelă cu n coloane şi k linii; coloana j corespunde atributului A_j, linia i – schemei R_i. În linia i şi coloana j introducem simbolul a_j dacă $A_j \in R_i$; dacă nu, introducem simbolul b_{ij}.

În continuare, se examinează fiecare din dependențele $X \to Y$ care $\in F$, operând, de fiecare dată, modificări în tabel. Când „examinăm” o dependență $X \to Y$, căutăm liniile care coincid – ca valori – în toate coloanele ce corespund atributelor din X. Dacă sunt găsite două astfel de linii, se „egalează” simbolurile din aceste linii, în câmpurile corespunzătoare atributelor din Y. Dacă unul din simboluri este a_j, ambele simboluri (care se „egalează”) devin a_j; dacă simbolurile sunt b_{ij} şi b_{ji}, devin ambele b_{ij} sau b_{ji} – în mod arbitrar. Dacă modificând astfel liniile tabloului, una din linii devine $a_1...a_k$, descomunerea este cu joncțiune fără pierderi.

Exemplu – descompunerea schemei de relație FURNIZORI în FD şi FAP.

Dependențele sunt $F \to D$, $FA \to P$; tabelul inițial este:

	F	D	A	P
FD→	a_1	a_2	b_{13}	b_{14}
FAP→	a1	b_{22}	a_3	a_4

F – NUMEF
D – ADRESAF
A – ARTICOL
P – PRET

Deoarece $F \to D$ iar cele două linii coincid pe coloana F, vom egala simbolurile pentru D, făcând (notând!) $b_{22}=a_2$. Rezultă:

F	D	A	P
a_1	a_2	b_{13}	b_{14}
a_1	a_2	a_3	a_4

Deoarece o linie nu conține decât simboluri a, joncțiunea este fără pierderi.

Un alt exemplu: R=ABCDE, R_1=AD, R_2=AB, R_3=BE, R_4=CDE, R_5=AE, iar mulțimea dependențelor funcționale: $A \to C$, $B \to C$, $C \to D$, $DE \to C$, $CE \to A$. Ca exercițiu!

Algoritmul de mai sus permite verificarea joncțiunii fără pierderi pentru descompuneri în orice număr de subscheme. Pentru descompunerea în două subscheme, se poate folosi următoarea teoremă:

Teoremă. Fie R o schemă, $\rho=(R_1, R_2)$ o descompunere, F o mulțime de dependențe funcționale. Atunci ρ admite joncțiune fără pierderi în raport cu F dacă şi numai dacă: $R_1 \cap R_2 \to R_1 - R_2$ sau $R_1 \cap R_2 \to R_2 - R_1$. Ultimele două dependențe nu trebuie să $\in F$, ci este suficient să $\in F^+$.

Exemplu (fără demonstrația teoremei): fie R=ABC şi F={A→B}. Atunci descompunerea schemei R în AB şi AC admite o joncțiune fără pierderi, deoarece $AB \cap AC = A$, $AB - AC = B$, iar $A \to B$ este valabilă (notă: reamintim că AB este o notație pentru $A \cup B$, etc).

Dacă însă vom descompune R în subschemele R_1=AB şi R_2=BC, rezultă că $R_1 \cap R_2 = B$, iar B nu determină funcțional nici $R_1 - R_2 = A$, nici $R_2 - R_1 = C$. Aşadar, descompunerea în AB şi BC nu admite o joncțiune fără pierderi pentru dependența F={A→B}; acest lucru poate fi constatat luând relația:

r={$a_1b_1c_1$, a_2,b_2,c_2} pentru schema R. Atunci, $\Pi_{AB}(r)$={a_1b_1, a_2b_1}, $\Pi_{BC}(r)$={b_1c_1, b_1c_2}, iar $\prod_{AB}(r) \rhd\lhd \prod_{BC}(r) = \{a_1b_1c_1, a_1b_1c_2, a_2b_1c_1, a_2b_1c_2\}$.

Descompuneri care conservă dependențele

Dacă o descompunere admite o joncțiune „fără pierderi", atunci orice relație care corespunde schemei descompuse poate fi obținută din proiecțiile sale; aceasta este o proprietate importantă.

O altă proprietate a descompunerii unei scheme R în ρ={R_1, R_2, ..., R_k}: mulțimea F a dependențelor funcționale pe R să fie implicată de proiecția lui F pe subschemele R_i.

Din punct de vedere formal, proiecția $\Pi_Z(F)$ a mulțimii de dependențe F pe mulțimea de atribute Z, este mulțimea dependențelor $X\rightarrow Y$ din F^+, astfel încât $XY\subseteq Z$ (vom observa că nu este necesar ca $X\rightarrow Y\in F$, ci numai în F^+).

O descompunere ρ conservă o mulțime de dependențe F dacă reuniunea tuturor dependențelor din $\prod_{R_i}(F)$, i=1, 2, ..., k, implică logic toate dependențele din F.

Exemplu. Ne reamintim că în cazul schemei (ORAŞ, STR, COD), pe care o vom prezenta prescurtat prin (O, S, C), aveam dependențele $OS\rightarrow C$ şi $C\rightarrow O$. Observăm că descompunerea OSC în subschemele SC şi OC admite o joncțiune fără pierderi, întrucât $(SC\cap OC)\rightarrow(OC–SC)$. În acelaşi timp, observăm că proiecția mulțimii F={$OS\rightarrow C$, $C\rightarrow O$} pe SC ne dă numai dependențe funcționale care decurg din reflexibilitate, în timp ce proiecția pe OC conduce la $C\rightarrow O$ şi dependențe triviale. Se poate verifica faptul că dependența $C\rightarrow O$ şi dependențele triviale nu implică $OS\rightarrow C$, prin urmare descompunerea nu conservă dependențele.

Ca exemplu, observăm că joncțiunea relațiilor din figurile a) şi b) este relația din figura c).

S	C
Bd. Nou 120	70500
Bd Nou 120	70501

a)

O	C
Bucureşti	70500
Bucureşti	70501

b)

O	S	C
Bucureşti	Bd. Nou 120	70500
Bucureşti	Bd. Nou 120	70501

c)

Relația din figura a) satisface (toate) dependențele triviale, ca orice relație. Relația din figura b) satisface dependențele triviale şi dependența $C\rightarrow O$. Cu toate acestea, relația obținută prin joncțiune, din figura c), nu satisface dependența $OS\rightarrow C$.

Vom mai observa că o descompunere poate avea o joncțiune fără pierderi față de o mulțime de dependențe F, dar fără a conserva F! În exemplul anterior am subliniat o astfel de situație. De asemenea, descompunerea poate conserva F, dar să nu admită o joncțiune fără pierderi. De exemplu, fie F={$A\rightarrow B$, $C\rightarrow D$}, R=ABCD, iar ρ=(AB, CD).

Cauza pentru care este de dorit ca o descompunere ρ să conserve F, constă în faptul că dependențele din F pot fi văzute ca *restricții de integritate* pentru relația R. Dacă dependențele proiectate nu implică F, atunci, dacă am reprezenta R prin ρ=(R_1, ..., R_k) am ajunge, eventual, la concluzia că valoarea curentă a subschemelor R_i a reprezentat o relație R care nu a satisfăcut F, chiar dacă ρ admitea o joncțiune „fără pierderi" față de F. De asemenea, oricare actualizare a unei subscheme R_i ar reclama o joncțiune pentru a verifica respectarea restricțiilor (de integritate).