Unele observații cu privire la validitatea și conservarea dependențelor funcționale în procesul de normalizare 3NF prin sinteză

Prof.dr.Valer ROȘCA
Universitatea Sibiu

În proiectarea și realizarea bazelor de date relaționale, normalizarea schemei acestora este o etapă de optimizare. În normalizarea prin sinteză, o schemă relațională R se obține pe baza unei liste date de constrîngeri și se prezintă ca o mulțime Δ de subscheme relaționale. Δ se construiește astfel încât să asigure funcționalitatea bazei și să elimine redundanța nenecesară de date și anomaliile de actualizare.
Normalizarea este un proces care implică un volum mare de muncă și experiență din partea analistului. De aceea, el este, cel puțin parțial, asistat cu instrumente software. În acest sens, lucrarea face cîteva observații cu privire la sinteză, pentru cazul 3NF, ca procedeu de normalizare care trebuie să asigure validitatea și să conserve dependențele funcționale ale bazei de date
Cuvinte cheie: *schema de relație, axiomele lui Armstrong, normalizare, sinteză, algoritm*

1. Preliminarii

Pentru fixarea notațiilor și a sensului noțiunilor utilizate, se face o succintă prezentare a părții structurale a modelului relațional, relevantă pentru scopul propus.

D1: Fie $D=\{D_1,D_2,...,D_n\}$ o familie de domenii și $A=\{A_1,A_2,...,A_n\}$ o familie de atribute, cu A_i definit pe D_i. Se spune că:
- cuplul $S=<A,D>$ este o *schemă de relație n-ară (intensie)*;
- o submulțime $E\subseteq D_1xD_2x...xD_n$ este o *extensie* (instanțiere) a lui S;
- $(a_1,a_2,...,a_n)$ este un *tuplu* al extensiei E.

Domeniile și atributele sunt atomice, adică valorile lor sunt nedecomposabile. Pentru simplitate, familia de domenii se omite și o submulțime de atribute $X=\{A,..,B\}$ se desemnează ca atribut compus și se notează în forma X=(A,B). Acest mod de abordare impune o anumită ordine a atributelor, ceeace se va presupune în continuare. Dacă $t \in E$, atunci valorile lui X pentru t constituie un tuplu notat t(X) care se obține ca proiecție după atributele simple din care X este constituit.

D2: Un cuplu de atribute $<X,Y>$, $Y \not\subset X$ se spune că definește o *dependență funcțională* (DF) și se notează $X\rightarrow Y$, dacă pentru orice extensie E și oricare două tuple $t_1, t_2 \in E$, din $t_1(X)=t_2(X)$ rezultă $t_1(Y)=t_2(Y)$. Despre o extensie *E* se spune că *verifică dependența funcțională*, dacă pentru ea sunt adevărate relațiile din definiția anterioară.

Fie F o mulțime de DF și f o DF care nu aparține lui F. Se spune că f este *implicată* (derivată) de F, dacă orice extensie E a lui R care verifică dependențele lui F, verifică și dependența dată. Mulțimea F^+, a dependențelor implicate de F, este denumită *închidere* a lui F. Două mulțimi de DF sunt *echivalente*, dacă au aceeași închidere. Dacă, prin eliminarea dependenței f din F, se obține aceeași închidere pentru F, atunci se spune că f este *redundantă* sau *superfluă*. O DF, $X\rightarrow Y$ se zice *redusă*, dacă există atributul A în relația R și $(X,A)\rightarrow Y$ este în F^+.

Dependențele funcționale sunt noțiuni semantice, ceeace înseamnă că stabilirea lor este parte a procesului de înțelegere a semificației datelor și se realizează de către analist. Dependențele funcționale joacă un rol central în normalizare.

Dependențele funcționale au unele proprietăți care permit costruirea închiderii unei mulțimi F de DF-uri, denumite axiomele lui Armstrong:

a) *reflexivitatea*: dacă $Y\subseteq X$ atunci $X\rightarrow Y$;
b) *incrementarea:* $X\rightarrow(Y,Z)$ dacă și numai dacă $X\rightarrow Y$ și $X\rightarrow Z$;

c) *tranzitivitate:* dacă $X \rightarrow Y$ şi $Y \rightarrow Z$ rezultă $X \rightarrow Z$.

Axiomele lui Armstrong formează un set complet, pentru o mulţime F de dependenţe funcţionale, în sensul că ele generează toate dependenţele implicate de F şi numai pe acestea. De aceea, se spune că închiderea este mulţimea generată de axiomele lui Armstrong.

Se consideră mulţimea de DF ale lui F^+ care au membrul drept atribut simplu şi sunt reduse, din care se elimină dependenţele redundante; mulţimea F^{min} a tuturor DF astfel obţinută este echivalentă cu F şi este denumită *acoperirea minimală* a lui F. Acoperirea minimală nu este unică. Există algoritmi care permit determinarea directă a unei acoperiri minimale a lui F.

D3: Un cuplu <S,F>, în care S este o schemă de relaţie *n*-ară, iar F o mulţime de DF pe atributele lui S, se numeşte *schemă relaţională* şi se notează R=<S,F> sau R(S), dacă nu este necesar să se pună în evidenţă F. Un atribut simplu sau compus K se spune că este *cheie* a lui R, dacă există DF, $K \rightarrow S$ şi K este mulţime minimală.

O schemă relaţională are cel puţin o cheie. În cazul că există mai multe chei (chei candidate), una se alege drept *cheie primară*, iar celelalte devin chei alternative. Atributele care compun o cheie se numesc *atribute primitive*. Cheia primară este mecanismul prin care se asigură identificarea unică a tuplelor în extensiile lui R.

D4: Fie R=<S,F>, Y, Z atribute ale lui R. Se spune că Y *depinde parţial* de Z, dacă există $X \subset Z$ şi $X \rightarrow Y$; altfel Y *depinde total* de Z. Y *depinde tranzitiv* de Z, dacă există X, $Y \rightarrow X$ şi $Y \not\subset X$, astfel încât $Z \rightarrow X$ şi $X \rightarrow Y$.

Dependenţele parţiale şi tranzitive modelează legături nedorite între atribute care impietează asupra schemei relaţionale, deoarece produc anomalii de actualizare. În raport cu ele se definesc formele normale care interesează aici.

D5: Fiind dată o schemă relaţională R=<S,F>, o cheie K a lui R şi X, Y atribute neprimitive, se spune că R este în:

- *prima formă normală* (1NF), dacă oricare ar fi X există $K \rightarrow X$;
- *a doua formă normală* (2NF), dacă R este în 1NF şi orice X depinde total K;
- *a treia formă normală* (3NF), dacă R este în 2NF şi orice X depinde netranzitiv de K.

Formele normale ameliorează succesiv schema relaţională R, din punctul de vedere al anomaliilor de actualizare, astfel încât forma 3NF este considerată acceptabilă pentru multe cazuri de baze de date.

2. Descompunerea unei scheme relaţionale

D6: Fie R=<S,F> o schemă relaţională. Se spune că $R_1=<S_1,F_1>$ este o *subschemă relaţională* a lui R, dacă $S_1 \subset S$ şi F_1 este mulţimea de DF $X \rightarrow Y$ din F cu $X \subset S_1$ si $Y \subset S_1$.

D7: Fiind dată schema relaţională R=<S,F>, o mulţime $\Delta=\{R_1, R_2, ..., R_p\}$ de subscheme ale lui R, cu $R_i=<S_i,F_i>$, este o *descompunere* a lui R, dacă $S=\cup S_i$.

Pentru uniformitate, subschemele sunt denumite scheme relaţionale, iar R este numită schemă globală. Interesează descompuneri ale lui R corecte, din anumite puncte de vedere, aşa cum rezultă din definiţia care urmează.

D8: O descompunere Δ a lui R se spune că:

- *păstrează conţinutul* bazei (este *validă*), dacă pentru orice extensie E a lui R, E este *join natural* al proiecţiilor sale după Δ, adică $E=JOINN(E(R_1),..,E(Rp))$, unde cu $E(R_i)$ s-a notat proiecţia lui E după atributele lui R_i;
- *păstrează (conservă) dependenţele funcţionale*, dacă $F^+=(\cup F_i)^+$.

3. Normalizare prin sinteză

In abordarea normalizării s-au conturat două metode care contruiesc o descompunere: normalizarea prin descompunere şi normalizarea prin sinteză. Normalizarea prin descompunere pleacă de la *relaţia universală* R şi o mulţime de DF-uri, considerată semificativă pentru aceasta şi utilizând, succesiv, dependenţele funcţionale şi apoi cele tranzitive, crează o descompunere Δ printr-un proces dihotomic, de divizare a

unei relații curente în două subrelații. Procesul este relativ ușor de aplicat și chiar de automatizat, dar are neajunsuri, dintre care se detașează, din punct de vedere semantic, faptul că algoritmii de descompunere nu pot asigura, în general, descompuneri 3NF care să fie valide și, în același timp, să conserve dependențele funcționale.

Normalizarea prin sinteză este o metodă alternativă la normalizarea prin descompunere și, în același timp, reciprocă. Ea pleacă de la mulțimea atributelor care vor intra în baza de date și mulțimea de DF-uri, puse în evidență în procesul de analiză și construiește relațiile elementare ale unei descompuneri, selectând atributele acestora într-un anumit mod.

In această lucrare, analiza procesului de sinteză 3NF se bazează pe o adaptare a algoritmul lui Berstein, caracteristic pentru acest caz, care utilizează o acoperire minimală a unei mulțimi de DF-uri. Analiza nu pierde din generalitate, dacă se presupune că din acoperirea minimală se elimină dependențele de forma $Y \rightarrow X$, atunci când există și dependența $X \rightarrow Y$, deoarece prezența lor în algoritm asigură numai reducerea numărului de relații ale descompunerii sintetizate. Astfel, algoritmul poate fi redat, în ceea ce este esențial prin următorii pași:

1) Se determină o acoperire minimal F^{min} a lui F;

2) Se împarte acoperirea minimală în grupuri de dependențe: un grup F_i este format din toate dependențele acoperirii minimale care au același membru stâng;

3) Dacă *n* este numărul de grupuri construite în pasul anterior, se definește descompunerea lui R ca fiind formată din relații $R_i=<S_i,F_i>$ i=1...n, unde S_i este mulțimea atributelor pe care le conțin dependențele din F_i.

Se pot face observațiile care urmează.

Observația A. Algoritmul produce o descompunere care conservă dependențele funcționale ale acoperirii minimale, prin însăși modul de construcție. În ceeace privește validitatea, se pot aduce exemple din care să rezulte că algoritmul nu o poate asigura în toate cazurile.

Exemplul 1: Se consideră atributele și dependențele funcționale care urmează:

S={CodReper, CodMașină, CodOper, CategOper, NrOper, TimpPreg, TimpExec}
F={(CodReper, CodMașină, CodOper,) → CategOper,
(CodReper, CodMașină, CodOper,)→TimpPreg,
(CodReper, CodMașină, CodOper,)→TimpExec,
(CodReper, CodMașină, CodOper,)→NrOper,
(CodReper, CodMașină, NrOper,)→CategOper,
(CodReper, CodMașină, NrOper,)→TimpPerg,
(CodReper, CodMașină, NrOper,)→TimpExec,
(CodReper, CodMașină, NrOper,)→CodOper,
CodOper→CategOper, NrOper→CategOper}

Se găsește că o acoperire minimală este:

F^{min}={(CodReper, CodMașină, CodOper)→TimpPreg,
(CodReper, CodMașină, CodOper)→TimpExec,
(CodReper, CodMașină, CodOper)→NrOper,
(CodReper, CodMașină, NrOper)→CodOper,
CodOper→CategOper, NrOper→CategOper}.

Aplicând algoritmul, rezultă următoarea descompunere:

R_1:S_1= {CodReper, CodMașină, CodOper, TimpPreg, TimpExec, NrOper}
F_1= {(CodReper, CodMașină, CodOper,)→TimpPreg,
(CodReper, CodMașină, CodOper,)→TimpExec,
(CodReper, CodMașină, CodOper,)→NrOper}
R_2: S_2= {CodReper, CodMașină, CodOper, NrOper}
F_2= {(CodReper, CodMașină, NrOper)→CodOper}
R_3: S_3= {CodOper, CategOper}
F_3= {CodOper→CategOper }
R_4: S_4= {NrOper, CategOper}
F_4= {NrOper→CategOper }.

Descompunerea este validă, asigurîndu-se posibilitatea de join a proiecțiilor și rezultă:
T_1: din R_1 cu R_2 după (CodReper, CodMașină, CodOper);
T_2: din T_1 cu R_3 după (CodOper);
R: din T_2 cu R_4 după (NrOper, CategOper).

Exemplul 2: Dacă se consideră o unitate de cercetare și S={Marca, NumeCercetător, CodProiect}, atunci F={Marca→ NumeCercetător}, deoarece un cercetător poate lucra simultan la mai multe proiecte și un proiect poate fi realizat de mai mulți cercetători. Rezultă că F^{min}=F și R=R_1=<S_1,F_1>, unde S_1={Marca, NumeCercetator} și F_1=F, adică un rezultat invalid.

Dacă se urmăreşte exemplul, se constată că se pierde atributul CodProiect. Dacă alături de R_1 se introduce relaţia R_2, cu S_2=âMarca, CodProiect} şi F_2=∅, se constată că se poate reconstrui R prin join natural din proiecţiile lui R_1 şi R_2. Din semantica relaţiei globale rezultă că atributul (Marca, CodProiect) este cheie pentru aceasta.

Analizând comparativ cele două exemple, rezultă o diferenţă formală în rezultatul descompunerii sintetizate, referitoare la cheia relaţiei universale aferente.

În exemplul 1 există două chei alternative:

K_1= (CodReper, CodMaşină, CodOper),

K_2= (CodReper, CodMaşină, NrOper)

şi după sinteză, ambele se regăsesc în relaţiile descompunerii, prima în R_1, cealaltă în R_2.

În exemplul al doilea, există o singură cheie şi anume K=(Marca, CodProiect), care, după sinteză, nu se mai regăseşte în descompunere. Dacă se introduce artificial R_2, cu această cheie, atunci descompunerea devine validă.

Din aceste exemple şi din altele cercetate, rezultă că o condiţie posibilă, necesară şi suficientă, trebuie să fie legată de prezenţa cheii (cheilor) relaţiei universale în descompunerea sintetizată.

Suficienţa. Problema suficienţei condiţiei de prezenţă a cheii în descompunere se poate rezolva, mai uşor, observând că există descompuneri ale unei relaţii unviersale în care cheia K a ei este conţinută într-o relaţie a acesteia şi totuşi descompunerea nu este validă şi/sau nu păstrează dependenţele funcţionale. Adică, rezultă constatarea că *condiţia nu este suficientă.*

• Dacă se reia exemplul 1, se poate considera o descompunere de forma:

R_1: S_1= {CodOper, CategOper}

F_1= {CodOper → CategOper}

R_2: S_2= {CodReper, CodMaşină, CodOper, NrOper, TimpPreg, TimpExec}

F_2= {(CodReper, CodMaşină, CodOper,)→TimpPreg,
(CodReper, CodMaşină, CodOper,)→TimpExec,
(CodReper, CodMaşină, CodOper,)→NrOper,
(CodReper, CodMaşină, NrOper,)→CodOper }

Aceasta conţine cheile în R_2, este validă, dar nu posedă toate dependenţele funcţionale (lipseşte dependenţa NrOper→CategOper).

• Pentru exemplul 2, se poate considera o descompunere de forma:

R_1= <(Marca, CodProiect); ∅>

R_2 =<(Nume, CodProiect); ∅>

care, de asemenea, conţine cheia relaţiei globale în R_1, este invalidă şi nu are dependenţa funcţională iniţială. Faptul că descompunerea este invalidă se poate proba considerînd următoarea extensie globală:

Marca	NumeCercetător	CodProiect
1	A	p1
2	B	p1

care va da extensiile proiecţii ale descompunerii de forma:

R_1		R_2	
Marca	CodProiect	NumeCercetator	Cod Proiect
1	p1	A	p1
2	p1	B	p1

Prin join, nu se asigură refacerea extensiei iniţiale şi se introduce imposibilitatea selectării numelui după marcă, aşa după cum rezultă mai jos:

Join (R_1, R_2)

Marca	NumeCercetător	CodProiect
1	A	p1
1	B	p1
2	A	p1

Necesitatea. După investigarea mai multor exemple de acest fel, se conturează ideea unei formulări de forma:

Dacă cheia (cheile) relaţiei universale este (sunt) conţinută (conţinute) în cel puţin o relaţie a descompunerii şi dacă descompunerea păstrează dependenţele funcţionale ale acoperirii minimale, atunci descompunerea este validă.

Atunci, rezultă că adăugarea la descompunere, a unei relaţii de forma <cheie, ∅>, dacă nici una din relaţii nu conţine cheia, asigură validitatea, având în vedere că sinteza păstrează dependenţele funcţionale.

Reciproca. De asemenea, se poate pune întrebarea dacă există următoarea proprietate:

Dacă descompunerea este validă, atunci cel puţin o relaţie din descompunere conţine cheia relaţiei universale.

Observația B. Acoperirea minimală nu este unică. Este suficient să se considere o altă ordine de tratare a dependențelor funcționale și se obține o acoperire minimală echivalentă. Pentru exemplul 1 se obține:

F^{min}={(CodReper, CodMașină, NrOper)→TimpPreg,
(CodReper, CodMașină, NrOper)→TimpExec,
(CodReper, CodMașină, NrOper)→CodOper,
(CodReper, CodMașină, CodOper)→NrOper,
CodOper→CategOper, NrOper→CategOper}.

Apare întrebarea: ce legătură există între acoperirea utilizată și descompunerea sintetizată din ea?

Din alte cazuri considerate, rezultă că acoperirile pot diferi nu numai din punctul de vederea al grupării atributelor, dar și în ceea ce privește numărul de relații în descompunere. Această situație generează o altă întrebare: există o descompunere astfel încât relațiile sale să fie distincte (două câte două) și să aibă cel mai mic cardinal (descompunere minimală)? Cum s-ar putea sintetiza această descompunere, dacă există?

5. Concluzie

Din cele de mai sus, se poate trage concluzia că algoritmii de normalizare prin sinteză, dacă se demonstrează corectitudinea observațiilor de mai sus, pot asigura construirea automatizată de descompuneri 3NF care să fie valide și, în același timp, să conserve dependențele funcționale. Mai mult, ar exista posibilitatea construirii de descompuneri minimale.

Bibliografie

1. Date C. J - An Introduction to Database Systems. Adison Wesley Publishing Co, 1982.
2. Lungu I, ș.a - Baze de date: organizare, proiectare și implementare, Colecția Informatica, Grupul ALL, 1995