Normalizarea bazelor de date prin descompunere

Prof.dr.Valer ROŞCA
Catedra de Matematică-Informatică, Universitatea *Lucian Blaga*, Sibiu

In proiectarea şi realizarea bazelor de date relaţionale, o etapă necesară o repezintă normalizarea schemei relaţionale a acestora. In principiu, prin normalizare, o schemă relaţională R este înlocuită, pe baza unei liste date de constrîngeri, cu o mulţime Δ de subscheme relaţionale, echivalentă într-un anumit sens cu R şi mai bună decît aceasta, din anumite puncte de vedere. In esenţă, Δ conservă funcţionalitatea lui R şi elimină redundanţa nenecesară de date şi anomaliile de actualizare pe care R le generează.
*Normalizarea este un proces dificil care implică un volum mare de muncă şi experienţă din partea analistului. De aceea, se pune problema asistării acestui proces cu instrumente CASE, bazate pe algoritmi corespunzători. In acest sens, plecînd de la cerinţa de conservare a datelor şi a dependenţelor funcţionale în procesul normalizării, lucrarea îşi propune să studieze descompunerea, pentru cazul **3NF** şi **BCNF**, ca bază pentru algoritmizare, scoţînd în evidenţă limitele acestei metode.*
***Cuvinte cheie :** baze de date relationale, dependenţe funcţionale, normalizarea structurii.*

1. Preliminarii

Pentru fixarea notaţiilor şi a sensului noţiunilor utilizate, se face o succintă prezentare a părţii structurale a modelului relaţional, relevantă pentru scopul propus.

D1: Fie $D=\{D_1,D_2,...,D_n\}$ o familie de domenii şi $A=\{A_1,A_2,...,A_n\}$ o familie de atribute, cu A_i definit pe D_i. Se spune că:

- cuplul $S=\{A, D\}$ este o **schemă de relaţie** n-ară (**intensie**);
- o submulţime $E\subseteq D_1xD_2x...xD_n$ este o **extensie** (instanţiere) a lui S;
- $(a_1,a_2,...,a_n)$ este un **tuplu** al extensiei E.

Domeniile şi atributele sunt atomice, adică valorile lor sunt nedecomposabile. Pentru simplitate, familia de domenii se omite şi o submulţime de atribute $X=\{A,..,B\}$ se desemnează ca atribut compus şi se notează prin juxtapunere, în forma X=A...B. Dacă $t \in E$, atunci valorile lui X pentru t constituie un tuplu notat t(X) care se obţine ca proiecţie dupa atributele simple din care X este constituit.

D2: Un cuplu de atribute $\{X,Y\}$, $Y\not\subset X$ se spune că defineşte o **dependenţă funcţională** (DF) netrivială şi se notează $X\rightarrow Y$, dacă pentru orice extensie E şi 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 dependenţa funcţională**, dacă pentru ea sunt adevărate relaţiile din definiţia anterioară. O DF, X->Y, se zice **elementară**, dacă Y este o mulţime formată dintr-un singur atribut şi X este o mulţime minimală.

Fie F o mulţime de DF şi f o DF care nu aparţine lui F. Se spune că f este **implicată** (derivată) de F, dacă orice extensie E a lui R care verifică dependenţele lui F, verifică şi dependenţa dată. Mulţimea F^+ a dependenţelor implicate de F este denumită **închidere** a lui F. Două mulţimi de DF sunt **echivalente**, dacă ele au aceeaşi închidere.

Dependenţele funcţionale sunt noţiuni semantice, ceea ce înseamnă că stabili-

rea lor este parte a procesului de înțelegere a semificației datelor și se realizează de către analist. Dependențele funcționale joacă un rol central in normalizare.

Dependențele funcționale au unele proprietăți care permit costruirea închiderii unei mulțimi F de DF-uri, dintre care utile sunt:

a) **incrementarea:** dacă X->Y și A este un atribut rezultă X|A->Y|A;

b) **tranzitivitate**: dacă X->Y și Y->Z rezultă X->Z.

D3: Un cuplu <S, F>, în care S este o schemă de relație n-ară, iar F o mulțime de DF pe atributele lui S, se numește **schemă relațională** și se notează R=<S, F> sau R(S), dacă nu este necesar să se pună în evidență F.

Un atribut simplu sau compus K se spune că este **cheie** a lui R, dacă există DF K->S și K este mulțime minimală.

Din punct de vedere semantic, o schemă relațională R este un model al unor obiecte din lumea reală (concrete sau abstracte), în care domeniile reprezintă mulțimile de fapte admisibile cu privire la aceste obiecte, iar atributele sunt proprietăți, adică fapte de un anumit tip. In acest sens atributele se asimilează cu variabilele.

O schemă relațională are cel puțin o cheie. In 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 parțial** de Z, dacă există $X \subset Z$ și X->Y; altfel Y **depinde total** de Z. Y **depinde tranzitiv** de Z, dacă există X, Y->X și $Y \not\subset X$, astfel încît Z->X și X->Y.

Dependențele parțiale și tranzitive modelează legături nedorite între atribute care impietează asupra schemei relaționale, deoarece produc anomalii de actualizare. In raport cu ele se definesc formele normale care interesează aici.

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

- **prima formă normală** (1NF), dacă oricare ar fi X există K->X;
- **a doua formă normală** (2NF), dacă R este în 1NF și orice X depinde total K;
- **a treia formă normală** (3NF), dacă R este în 2NF și orice X depinde netransitiv de K.
- **forma tare 3NF** (BCNF), dacă R este în 3NF și, pentru orice DF X->Y , X este cheie alternativă pentru R.

Formele normale ameliorează succesiv schema relatională R, din punctul de vedere al anomaliilor de actualizare, astfel încît formele 3NF și BCNF să fie considerate acceptabile pentru multe cazuri de baze de date.

2. Descompunerea unei scheme relaționale

D6: Fie R=<S, F> o schemă relațională. Se spune că R_1=<S_1, F_1> este o **subschemă relațională** a lui R, dacă $S_1 \subset S$ și F_1 este mulțimea de DF X->Y din F cu $X \subset S_1$ și $Y \subset S_1$.

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

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

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

- **păstrează conținutul** 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 $F^+=(\bigcup F_i)^+$.

Există algoritmi pentru testarea validităţii şi conservării dependenţelor funcţionale pentru o descompunere (algoritmul lui Loizou, respectiv Ullman), pe care însă nu îi reproducem aici, dar vor fi utilizaţi în continuare.

Teorema care urmează dă suportul teoretic pentru descompunerea unei scheme relaţionale globale în două scheme, astfel încît să se păstreze conţinutul de date. Descompunerea se face pe baza unei dependenţe funcţionale X->Y.

Teorema 1: O schemă relaţională R(X,Y,Z), pentru care există X->Y, iar Z este un atribut oarecare, simplu sau compus, are totdeauna o descompunere validă sub forma: R_1(X,Y), R_2(X,Z).

Demonstraţie :

Trebuie demonstrată egalitatea din D8, avînd în vedere un join natural după atributul comun X. Incluziunea lui E în JOINN(E(R_1),E(R_2)) rezultă din modul de definire a operatorului de join. Pentru a demostra incluziunea inversă, considerăm (x,y,z) un triplet care aparţine extensiei obţinute prin join natural. Datorită faptului că X->Y, există unic $y \in Y$ astfel încît $(x,y) \in E(R_1)$, pentru oricare $x \in X$. Atunci, pentru oricare $(x,z) \in E(R_2)$, rezultă că (x,y,z) este element unic în E. Astfel, se obţine egalitatea.

Teorema 1 are consecinţe imediate în ameliorarea unei scheme globale care se găseşte într-o anumită formă normală şi posedă o singură dependenţă funcţională nedorită. Prin descompunere, se construieşte R_1 ca relaţie în care această dependenţă este una normală (dorită) şi X devine cheie primară. Avînd în vedere R_2, conţinutul de date este păstrat şi se pot aplica interogări pe ambele relaţii, datorită lui X care este cheie externă în R_2. Se poate formula corolarul care urmează.

Corolar: Fiind dată o schemă relaţională R(K,X,Y,Z), cu cheia primară K, într-o anumită formă normală şi X->Y o unică dependenţă funcţională nedorită, subschemele R_1(X,Y) cu X cheie primară şi R_2(K,X,Z) cu K cheie primară sunt cel puţin în forma nomală imediat superioară.

In baza corolarului anterior, se pot formula algoritmi pentru eliminarea dependenţelor funcţionale nedorite, de un anumit tip, printr-un proces de descompunere succesivă. In final se obţine o descompunere validă, în care relaţiile sunt (cel puţin) în forma normală imediat superioară. In elaborarea algoritmilor trebuie sa se ia în considerare observaţiile care urmează.

Observaţii:

a) Dacă se dă mulţimea de DF elementare {X->Y_1, X->Y_2, ..., X->Y_p}, atunci există DF X->Y_1Y_2...Y_p şi reciproc. In aceste condiţii, algoritmul poate considera numai DF - uri elementare. Mulţimea F^{min} a tuturor DF elementare ale unei mulţimi F este echivalentă cu F şi este denumită **acoperirea minimală** a lui F.

b) Dacă R_1(X,Y_1), R_2(X,Y_2),...,R_p(X, Y_p) este o descompunere, atunci ea poate fi înlocuită cu schema R(X, Y_1 Y_2... Y_p).

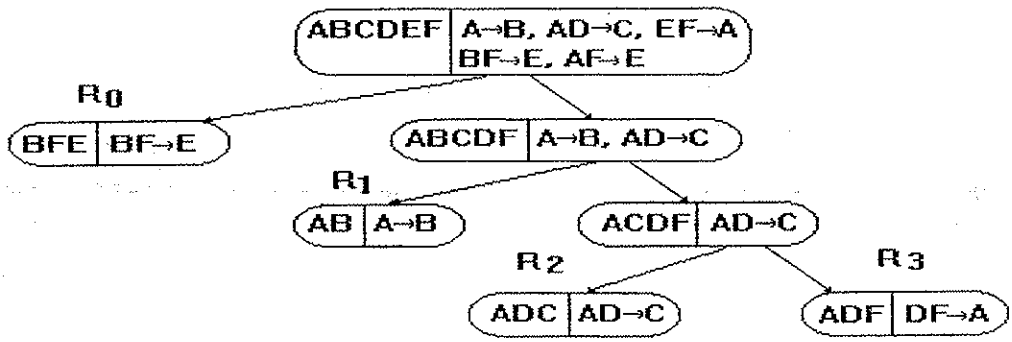
c) Orice schemă de forma R(X,Y), cu X cheie, este in forma BCNF.

In aceste condiţii, fiind dată o schemă globală R, într-o anumită formă normală, mulţimea dependenţelor funcţionale elementare F^{min} şi un anumit tip de dependenţe de transformat, se poate obţine o descompunere validă, în forma normală imediat superioară (cel puţin), aplicînd un algoritm general cu structura celui care urmează.

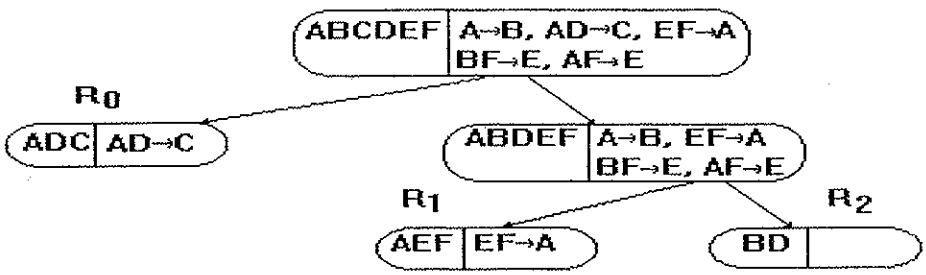
Algoritm:

0. Se consideră p=1, $F=F^{min}$.
1. Se caută în F o DF X->Y, de tipul dat. Dacă nu există, se trece la pasul 4.
2. Se realizează descompunerea: $R_p(X,Y)$, $R_{p+1}=S \setminus Y$ în care $F_p=\{X->Y\}$ şi $F_{p+1}=F \setminus F_p$.
3. Se actualizează F_{p+1}, eliminînd orice DF în care Y figurează în definiţia sa. Se considera $R=R_{p+1}$, $F=F_{p+1}$, se incrementează p şi se reia de la 1.
4. Daca p>2, se definitivează descompunerea conform observaţiei b, enunţată anterior.

Algoritmul poate fi amendat, astfel încît să se obţină eficienţa necesară aplicării lui practice. In acest sens, există unii algoritmi de descompunere, pentru trecere directă din 1NF în BCNF (Fischer) sau din 3NF în BCNF (Ullman). In mod logic, se pune întrebarea dacă un algoritm de tipul celui anterior produce o descompunere unică, în acelaşi timp, validă şi care păstrează DF - urile date. Exempul redat în figura 1, pentru cazul trecerii de la forma 3NF la BCNF şi alte exemple arată că descompunerea nu este unică, ci depinde de ordinea în care se consideră DF - urile tipului respectiv şi că, în general, se pierd dependenţe functionale. In aceste condiţii, apare o nouă întrebare şi anume: *ştiindu-se că o schemă relaţională admite o descompunere validă şi care păstrează DF - urile poate fi ea determinată printr-un proces de descompunere ?* Răspunsul se dă prin propoziţiile care urmează, pentru formele 3NF şi BCNF.



a) Ordinea DF-urilor: BF→E, A→B, AD→C



b) Ordinea DF - urilor: AD→C, EF→A

Fig. 1 - Influenţa ordinii DF asupra descompunerii

Propozitia 1: Există scheme relaţionale R în 2NF care posedă o descompunere 3NF, validă şi care conservă dependenţele funcţionale elementare, dar care nu poate fi obţinută printr-un algoritm de descompunere.

Demonstraţie:

Pentru demonstraţie, este suficient să se construiască o relaţie R în 2NF, cu o singură dependenţă transitivă care posedă o descompunere validă şi care păstrează DF şi să se arate că aceasta

nu poate fi obținută printr-un proces de descompunere.

Fie R(A,B,C,D), cu AD cheie globală şi F={AD->B, AD->C, BD->C, AB->C}. Se constată că $F=F^{min}$ şi R este în 2NF, deoarece atributele neprimitive B,C depind complet de cheia globala. R nu este in 3NF, deoarece există o dependență transitivă AD->BD->C care poate fi constatată aplicînd incrementarea cu D asupra lui AD->B.

R admite următoarea descompunere 3NF:

R_1=(A,B,C,D) cu F_1={AD->B, AD->C};

R_2=(B,C,D) cu F_2={BD->C};

R_3=(A,B,C) cu F_3={AB->C}.

Descompunerea este validă, după cum poate fi verificat cu ajutorul algoritmului lui Loizou, păstrează DF - urile elementare şi dependența transitivă este eliminată.

Deoarece F conține toate dependențele funcționale elementare, orice descompunere trebuie să se bazeze pe ele. Aplicînd algoritmul de descompunere, se obțin trei descompuneri distincte :

R_{11}=(A,D,B), R_{12}=(A,D,C) cu F_{11}={AD->B}, F_{12}={AD->C}

R_{21}=(B,D,C), R_{22}=(A,D,B) cu F_{21}={BD->C}, F_{22}={AD->B}

R_{31}=(A,B,C), R_{33}=(A,D,B) cu F_{31}={AB->C}, F_{32}={AD->B}.

Descompunerea dată nu figurează printre ele, ceea ce demonstrează propoziția.

O propoziție similară se poate da pentru cazul BCNF.

Propozitia 2: Există scheme relaționale R în 3NF care posedă o descompunere BCNF validă care conservă dependențele funcționale, dar care nu poate fi obținută printr-un algoritm de descompunere.

Demonstratie:

Fie R=(A,B,C,D), cu AD cheie globală, F^{min}= F={AD->B, AD->C, BC->D} şi R admite o descompunere de forma:

R_1=(A,D,B) cu F_1={AD->B}

R_2=(A,D,C) cu F_2={AD->C}

R_3=(B,C,D) cu F_3={BC->D}.

Descompunerea este BCNF, deoarece fiecare schemă are o singură dependență funcțională (observația c). Descompunerea păstrează DF - urile şi este validă, cum se poate constata prin verificare cu algoritmul lui Loizou.

Descompunerea nu poate fi obținută printr-un algoritm de descompunere. Intr-adevăr, F conține toate DF elementare şi în baza lor se obțin doar două descompuneri distincte:

R_{11}=(A,D,B), R_{12}=(A,D,C) cu F_{11}={AD->B}, F_{12}={AD->C}

R_{21}=(B,C,D), R_{22}=(B,C,A) cu F_{21}={BC->D}, F_{22}= Ć.

Descompunerile nu coincid cu cea dată şi, în plus, se pierd dependențe funcționale, ceea ce demonstrează propoziția.

3. Concluzie

Algoritmii de descompunere nu pot asigura, în general, descompuneri 3NF şi BCNF care să fie valide şi, în acelaşi timp, să conserve dependențele funcționale.

Se desprinde ideea că descompunerea, ca proces de ameliorare a structurilor relaționale, este utilă şi se justifică efortul de a construi instrumente software corespunzătoare, în măsura în care SGBD-ul nu implementează dependențele funcționale ca mecanism de asigurare a integrității bazei de date.

Bibliografie

1. Date C. J - An Introduction to Database Systems, Addison Wesley Publishing Co, 1982.

2. Ullman J.D - Principles of Database Systems, Computer Science Press, 1988.