Introducere în teoria proiectării bazelor de date relaționale. Dependențe funcționale. Chei. Axiome. Închideri. Acoperiri.

prof. dr. ing. Mircea Petrescu

Ne referim la proiectarea conceptuală şi logică. O problemă principală pentru o relație – alegerea schemei relației, din mai multe posibile. Ideea centrală în alegerea schemelor de relație: dependența datelor. Dependența datelor este o restricție asupra relațiilor r_i care pot constitui valoarea curentă a unei scheme de relație R. Exemplu de dependență: un atribut determină în mod unic un alt atribut, ca în cazul ADRESA=f(nume). Dependența funcțională va fi reprezentată prin notația specifică NUME→ADRESĂ.

Fie schema de relație: FURNIZORI(Numef, Adresaf, Articol, Pret). Probleme aici:

1. redundanță – repetarea adresei;
2. inconsistență potențială (anomalii de actualizare);
3. anomalii de inserare;
4. anomalii de eliminare (invers cu 3)) – dacă eliminăm toate articolele livrate de un furnizor, putem pierde urma adresei sale (întrucât perechea de atribute Articol, Numef formează o cheie a relației FURNIZORI).

Problemele dispar dacă relația unică FURNIZORI este descompusă în următoarele două relații:
FD(Numef, Adresaf)
FAP(Numef, Articol, Pret)
Deci: FURNIZORI→FD, FURNIZORI→FAP.

Prin descompunere nu mai există problema redundanțelor la adrese. Putem introduce adresa unui furnizor chiar dacă la un moment dat nu livrează nimic. Dezavantaj al descompunerii de mai sus: dacă dorim adresele furnizorilor unui anumit articol, trebuie efectuată mai întâi o joncțiune, ceea ce este scump. Cu relația unică FURNIZORI – o selecție, apoi o proiecție.

Analizând o schemă de relație putem înțelege dezavantajele ei (vezi restricțiile de mai sus). O întrebare firească: în ce mod efectuăm descompunerile, astfel încât acestea să fie eficiente (să conducă la scheme avantajoase).

Dependențele funcționale

Definiție. Fie schema de relație $R(A_1, A_2, ..., A_n)$ şi submulțimile de atribute X, Y astfel încât $X, Y \subseteq \{A_1, A_2, ..., A_n\}$. Atunci X determină funcțional Y, sau $X \rightarrow Y$, dacă oricare ar fi relația r care constituie o valoare curentă a schemei R, nu este posibil ca r să posede două tupluri care să coincidă prin valorile componentelor corespunzătoare atributelor din X, dar să nu coincidă prin valorile mai multor componente corespunzătoare atributelor din Y.

Singura cale de a găsi dependențele funcționale valabile pentru o schemă R este analiza atentă a înțelesului (semnificației) fiecărui atribut al acesteia şi a modului în care sunt atribuite valori atributelor.

Baza de date de lucru

Fie o bază de date simplă, cu relațiile Întreprinderi, Comenzi, Furnizori, având schemele:
Intreprinderi=(Nume, Adresa, Balplati)
Comenzi=(Nume, Articol, Cantitate)
Furnizori=(Numef, Adresaf, Articol, Pret)

Eventual, cu baza de date putem asocia numele CENIN (Centrală Industrială). Numele întreprinderilor, furnizorilor sau altor atribute pot fi simbolice, dar pot avea, de exemplu, forma Nume=Imet (Întreprindere Metalurgică). O realizare (instanță) a bazei de date poate fi următoarea:

Intreprinderi

Nume	Adresa	Balplati
N1	Ad1	+Bp1
N2	Ad2	0
N3	Ad3	-Bp3
N4	Ad4	-Bp4

Comenzi

Nume	Articol	Cantitate
N1	Ar1	Q1
N1	Ar2	Q2
N3	Ar3	Q3
N4	Ar4	Q4
N3	Ar5	Q5
N3	Ar6	Q6

Furnizori

Numef	Adresaf	Articol	Pret
Nf1	Adf1	Ar1	P1
Nf1	Adf1	Ar6	P2
Nf1	Adf1	Ar5	P3
Nf2	Adf2	Ar4	P4
Nf2	Adf2	Ar7	P5
Nf2	Adf2	Ar1	P6
Nf2	Adf2	Ar2	P7
Nf3	Adf3	Ar6	P8
Nf3	Adf3	Ar4	P9
Nf3	Adf3	Ar5	P10

În fond, dependențele funcționale sunt aserțiuni care reprezintă legături din lumea reală. Dependențele funcționale nu pot fi demonstrate, dar pot fi realizate de un SGBD proiectat în mod adecvat.

Multe sisteme existente realizează dependențe funcționale care rezultă din faptul că o cheie determină celelalte atribute ale unei relații.

Exemplu – să găsim dependențele funcționale în cazul BD de lucru CENIN. Presupunere – fiecare întreprindere are o adresă unică, o balanță de plăți unică. Nu exită două întreprinderi cu acelaşi nume (dacă apare o astfel de problemă se rezolvă prin introducerea unui element „de distingere").

Admitem atunci că: NUME→ADRESA, BALPLATI – în relația INTREPRINDERI.

De asemenea, NUME ARTICOL→CANTITATE – în relația COMENZI.

Notă – notații – XY pentru X∪Y, $A_1A_2...A_n$ pentru $\{A_1A_2...A_n\}$, unde A_i=atribute.

Ultima dependență – are la bază ipoteza că o întreprindere nu face două comenzi pentru acelaşi articol. O comandă suplimentară se realizează adăugând cantitatea la comanda veche, în loc de a insera un tuplu nou. Altfel, de exemplu, în relația COMENZI, IMET ar putea apare cu un tuplu nou, corespunzând unei comenzi de Tablă, cu o cantitate de 35Kg, dar nu ar putea apare cu un al doilea tuplu, pentru acelaşi articol, dar Cantitate = 25Kg, deoarece un astfel de tuplu există, de la prima comandă (în o relație nu putem avea două tupluri identice, relația fiind de mulțime).

Dependențe în relația FURNIZORI:

NUMEF→ADRESAF

NUMEF ARTICOL→PRET

Dependențe triviale: NUME→NUME, NUME ARTICOL→ARTICOL.

Dependențe netriviale, care derivă din cele găsite, de ex.: NUMEF ARTICOL→ADRESAF PRET.

Implicații logice ale dependențelor

Fie R o schemă de relație, iar A, B, C atribute în R. Ştim că, de exemplu, A→B şi B→C valabile în R. Se poate arăta că, de asemenea, A→C valabilă în R (tranzitivitate).

Fie F o mulțime de dependențe funcționale pentru R şi fie X→Y o dependență funcțională, valabilă tot pentru schema R. Atunci, F implică logic X→Y, dacă orice relație r pentru R, care satisface dependențele din F, satisface şi X→Y.

Închiderea mulțimii de dependențe F, notată cu F^+, se definește ca mulțimea dependențelor funcționale implicate logic de F. Când $F^+=F$, F este o familie completă de dependențe.

Exemplu: fie R=ABC, cu F={A, B, C}. Atunci, F^+ conține toate dependențele X→Y, astfel încât:

1. X conține pe A, de exemplu ABC→AB, AB→BC sau A→C.
2. X conține B dar nu A, iar Y nu conține A, de exemplu, BC→B, B→C, B→∅.
3. X→Y este una din cele două dependențe C→C sau C→∅.

Chei

Cheia = mulțime de atribute care determină în mod unic o entitate (deci concept similar cu cel de dependență funcțională).

Fie R o schemă, cu $A_1A_2...A_n$ mulțimea de dependențe F, iar X o submulțime a $A_1A_2...A_n$. X este *cheie unică* dacă:

1. $X→A_1A_2...A_n$ este în F^+;
2. pentru nicio submulțime $Y⊆X$, dependența funcțională $Y→A_1A_2...A_n$ nu face parte din F^+.

Condiția 2 – de minimalitate. Pentru o relație pot exista mai multe chei. Una din ele – cheie primară.

Exemplu: pentru R=ABC din exemplul anterior, singura cheie este A. Cauza: A→ABC face parte din F^+, dar nici un X care nu conține A nu determină funcțional ABC.

Exemplu: schema (ORAŞ, STR, COD); dependențe netriviale: ORAŞ STR→COD, COD→ORAŞ. Se verifică uşor că {ORAŞ, STR} şi {STR, COD} sunt, ambele, chei.

Axiome pentru dependențele funcționale

Pentru a găsi chei şi pentru a înțelege dependențele funcționale, în general, trebuie ca:

a) să putem calcula F^+ din F, sau, cel puțin
b) dacă sunt date F şi X→Y, să stabilim dacă X→Y face parte din F^+.

Pentru a rezolva a) şi b) ne sunt necesare reguli de inferență, care să ne învețe cum una sau mai multe dependențe funcționale implică alte dependențe.

Fie U – mulțimea universală de atribute, iar F – mulțimea de dependențe care include numai atribute din U.

Un sistem complet şi corect de reguli de inferență (axiomele lui Armstrong):

A1) Dacă $Y⊆X⊆U$, atunci X→Y este implicată logic de F (reflexibilitate). De aici obținem dependențe triviale.

A2) Dacă X→Y ține şi dacă $Z⊆U$, atunci XZ→YZ ține (augmentare, amplificare).

A3) Dacă X→Y şi Y→Z țin, atunci X→Z ține (tranzitivitate).

Exemplu: În schema {ORAŞ, STR, COD} mulțimea de atribute {STR, COD} era o cheie. Deci: STR COD→ORAŞ STR COD. Demonstrație:

1. COD→ORAŞ (prin ipoteză).
2. STR COD→ORAŞ STR (prin aplicare A2 pe 1).
3. ORAŞ STR→COD (prin ipoteză).
4. ORAŞ STR→ORAŞ STR COD (prin aplicare A2 pe 3).
5. STR COD→ORAŞ STR COD (prin aplicare A3 de la 2 la 4).

Alte reguli de inferență, care decurg din axiomele lui Armstrong:

a) dacă X→Y şi X→Z țin, atunci X→YZ ține (de fapt, X→Y∪Z!) (regula reuniunii);
b) dacă X→Y şi WY→Z țin, atunci ține şi XW→Z (regula de pseudotranzitivitate);
c) dacă X→Y ține şi dacă Z⊆Y, atunci X→Z ține (regula de descompunere).

Demonstrație.

a) X→Y este dată. Amplificăm cu X şi prin inferență obținem XX→XY, sau X→XY. De asemenea, X→Z este dată; amplificăm cu Y şi XY→ZY sau XY→YZ. Prin tranzitivitate (A3) rezultă X→YZ.

b) X→Y este dată. Amplificăm cu W şi obținem XW→YW. Dar WY→Z, sau YW→Y, deci prin A3 rezultă XW→Z.

c) X→Y este dată. Prin ipoteză Z⊆Y, deci din A1 rezultă Y→Z. Cu A3, din X→Y şi Y→Z obținem X→Z.

Consecință importantă a regulilor de reuniune şi descompunere: dacă A_1, A_2, ..., A_n sunt atribute, atunci $X \to A_1A_2...A_n$ ține (este valabilă) dacă şi numai dacă $X \to A_i$ ține pentru orice i.

Fie U o mulțime de atribute, X o submulțime a lui U, F o mulțime de dependențe pe U.

Definiție. Închidere a mulțimii X, în raport cu F, notată cu X^+, este mulțimea atributelor A astfel încât X→A poate fi dedusă din F folosind axiomele Armstrong.

Lemă. Dependența funcțională X→Y rezultă din axiomele Armstrong dacă şi numai dacă $Y \subseteq X^+$.

Demonstrație. Fie $Y=A_1A_2...A_n$. Facem ipoteza că $Y \subseteq X^+$.

Prin definiția lui X^+, $X \to A_i$ este implicat de axiomele Armstrong pentru toți i. Dar cum $Y=A_1A_2...A_n=\{A_1 \cup A_2 \cup \ldots \cup A_n\}$, prin regula de reuniune rezultă X→Y, deoarece $X \to A_i$ pentru orice i.

Se poate arăta (teoreme) că *sistemul axiomelor lui Armstrong este atât complet cât şi corect.*

Complet – dacă sunt date dependențele funcționale din F, prin axiome deducem toate dependențele funcționale din F^+.

Corect – folosind regulile de inferență reprezentate de axiomele Armstrong şi plecând din F, nu putem deduce dependențele funcționale care nu sunt în F^+.

Consecințele (demonstrabile, pe baza celor spuse mai sus):

a) X^+ a fost definit ca mulțimea de atribute A, astfel încât X→A decurge din F, folosind axiomele Armstrong. O definiție echivalentă pentru X^+ este: X^+ este mulțimea atributelor A astfel încât F implică logic pe X→A.
b) F^+ a fost introdusă ca mulțimea dependențelor implicate logic de F. Se poate însă defini F^+ ca mulțimea dependențelor care decurg din F prin axiomele Armstrong.

Calculul închiderilor

Algoritm. Intrare: U finit, F în U, X⊆U.

Ieşire: X^+, închiderea lui X în raport cu F.

Metoda: se calculează o succesiune de mulțimi de atribute, folosind axiomele; obținem $X^{(0)}$, $X^{(1)}$, ..., astfel încât:

1. $X^{(0)}=X$
2. $X^{(i+1)}=X^{(i)}\cup\{A\}$, astfel încât:
   a) $\exists\ Y\rightarrow Z$ în F, cu
   b) $A\in Z$
   c) $Y\subseteq X^{(i)}$

Deoarece $X=X^{(0)}\subseteq X^{(1)}\subseteq...\subseteq U$, iar U este finită, trebuie ca în final să se ajungă la un astfel de i, încât $X^{(i+1)}=X^{(i)}$. Urmează atunci că $X^{(i)}=X^{(i+1)}=X^{(i+2)}=...$ şi calculul se opreşte. Se poate arăta că, pentru această ultimă valoare a lui i: $X^+=X^{(i)}$.

Exemplu. Fie F: $AB\rightarrow C$, $D\rightarrow EG$, $C\rightarrow A$, $BE\rightarrow C$, $BC\rightarrow D$, $CG\rightarrow BD$, $ACD\rightarrow B$, $CE\rightarrow AG$. Fie X=D. Avem:

1) $X^{(0)}=D$; $Y_1=D$, $Z_1=EG$; $X^{(1)}=D\cup\{E\}\cup\{G\}=\{D, E, G\}=DEG$.
2) $X^{(1)}=DEG$; $Y_1=D$; $Z_1=EG$; rezultă imediat că $X^{(1)}=DEG=X^{(0)}$ deci STOP.

Aşadar $(D)^+=DEG$.

Similar se poate arăta că $(CG)^+=ABCDEG$, $(BE)^+=ABCDEG$, $(BD)^+=ABCDEG$, etc.

Acoperiri de mulțimi de dependențe

Fie F, G – mulțimi de dependențe funcționale. Dacă $F^+=G^+$, spunem că F şi G sunt *echivalente*.

Dacă F şi G sunt echivalente, spunem că F acoperă G (şi G acoperă F).

Fie F, G. Pentru a stabili că F şi G sunt echivalente (sau F=G), pentru fiecare dependență $Y\rightarrow Z$ din F se verifică dacă $Y\rightarrow Z$ este în G^+, folosind algoritmul anterior pentru a calcula Y^+ şi pentru a verifica apoi dacă $Z\subseteq Y^+$ (potrivit axiomelor, dacă $Z\subseteq Y^+$, atunci $Y\rightarrow Z$). Dacă o dependență $Y\rightarrow Z$ din F nu este în G^+, atunci sigur $F^+\neq G^+$.

Dacă fiecare dependență din F este în G^+, atunci fiecare dependeță $V\rightarrow W$ din F^+ este în G^+; pentru a arăta că $V\rightarrow W$ este în G^+, se demonstrează că fiecare $Y\rightarrow Z$ din F este în G^+, apoi că $V\rightarrow W$ este în F^+.

Pentru a arăta că fiecare dependență din G este, de asemenea, şi în F^+, se procedează în mod analog că *F şi G vor fi echivalente dacă fiecare dependență fin F este şi în G^+, iar fiecare dependență din G este şi în F^+*.

Lemă. Fiecare mulțime de dependențe funcționale F este acoperită de o mulțime de dependențe G, în care nicio parte dreaptă nu conține mai mult de un atribut.

Demonstrație. Fie G o mulțime de dependențe $X\rightarrow A$, astfel încât pentru o dependență $X\rightarrow Y$ în F, A este în Y. Atunci $X\rightarrow A$ decurge din $X\rightarrow Y$ prin regula de descompunere. Aşadar, $G\subseteq F^+$. Dar $F\subseteq G^+$, deoarece, dacă $Y=A_1A_2...A_n$, atunci $X\rightarrow Y$ rezultă din $X\rightarrow A_1$, $X\rightarrow A_2$, ..., $X\rightarrow A_n$, prin regula de reuniune.

Mulțime minimală de dependențe

O mulțime de dependențe este minimală dacă:
1) partea dreaptă a fiecărei dependențe din F conține un singur atribut;
2) pentru nicio dependență $X\rightarrow A$ din F, mulțimea $F-\{X\rightarrow A\}$ nu este echivalentă cu F;
3) pentru nicio dependență $X\rightarrow A$ din F şi pentru nicio submulțime $Z\subseteq X$, mulțimea $F-\{X\rightarrow A\}\cup\{Z\rightarrow A\}$ nu este echivalentă cu F.

De fapt, condiția 2 garantează că nicio dependență din F nu este redundantă, iar 3 că niciun atribut în nicio parte stângă nu este redundant. Desigur, niciun atribut din dreapta nu este redundant, deoarece fiecare parte dreaptă conține un singur atribut (condiția 1).

Teoremă. Fiecare mulțime de dependențe F este echivalentă cu o mulțime F`, care este minimală.

Exemplu. Fie: AB→C, D→EG, C→A, BE→C, BC→D, CG→BD, ACD→B, CE→AG.

În ultima lemă a fost indicat – la demonstrație – „un fel" de algoritm pentru secționarea părților drepte ale dependențelor funcționale. Aplicându-l obținem:

AB→C, C→A, BC→D, ACD→B, D→E, D→G, BE→C, CG→B, CG→D, CE→A, CE→G.

CG→B şi CE→A sunt redundante.

Să arătăm, de exemplu, că CG→B este redundantă cu ACD→B.

Se vede că ACD→B este echivalentă cu CD→B, deoarece C→A.

Aşadar, să arătăm că CG→B este redundantă, față de CD→B.

Pornind de la CD→B, avem CG→D, deci înlocuim pe D în CD→B şi obținem C(CG) →B, sau CG→B.

Procedând astfel, se obține sistemul de dependențe minimale din (a). Dacă din F eliminăm dependențele CE→A, CG→D şi ACD→B, obținem acoperirea minimală din (b). Observăm că cele două acoperiri minimale conțin numere diferite de dependențe.

(a) AB→C, C→A, BC→D, CD→B, D→E, D→G, BE→C, CG→D, CE→G.

(b) AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CE→G.