Capitolul 5

PROIECTAREA BAZELOR DE DATE

Prin proiectarea bazei de date, aici se subînțelege proiectarea unei scheme logice care ar înlătura apariția unor anomalii în lucrul cu baza de date, asigurând totodată facilități şi performanțe sporite la exploatarea ei.

Anomaliile care apar în lucrul cu baza de date sunt cunoscute sub anomalii de actualizare a datelor. Ele sunt puse în legătură cu dependențele care se manifestă între atribute. O asemenea abordare a anomaliilor de actualizare permite caracterizarea riguroasă a gradului de perfecțiune a schemei bazei de date şi face posibilă definirea unor tehnici formale de proiectare a unor astfel de scheme.

Prelucrarea datelor o perioadă de timp, cum se întâmplă în bazele de date, poate provoca o serie de probleme personalului responsabil de menținerea integrității datelor. Anomaliile în date cum ar fi datele duplicate sau pierderile de informații pot apărea, dacă datele nu sunt organizate într-un mod rezonabil. În ce constă o organizare rezonabilă a datelor? Cercetările la zi şi experiența acumulată în domeniul proiectării bazelor de date au arătat că unele aranjări de date lucrează mai bine decât altele. S-au elaborat tehnici de analiză a datelor şi organizare a lor într-o structură flexibilă şi stabilă.

Procesul de normalizare constă în aplicarea unui set de reguli predefinite asupra unei aranjări a datelor cu scopul reducerii structurii complexe şi transformării lor în structuri mai mici si stabile ce vor facilita manipularea şi menținerea datelor.

La fiecare pas o regulă este aplicată, datele pot fi restructurate şi când regula este satisfăcută se spune că datele sunt într-o formă normală.

Deci normalizarea este o abordare formală de analiză şi grupare a datelor în structuri mai eficiente ce se pot acomoda viitoarelor actualizări. În afară de aceasta normalizarea minimizează impactul ce poate avea loc asupra aplicațiilor în procesul actualizării bazei de date.

Pentru a produce o bază de date bine proiectată de obicei se porneşte de la relații nenormalizate şi printr-o serie de paşi se descompun structurile de date pentru a obține schema finală a bazei de date.

5.1. Prezumția schemei universale

Materia expusă până acum (şi mai departe) presupune că toată mulțimea de atribute formează schema unei relații "mari" şi toate constrângerile asupra atributelor sunt constrângeri ale acestei scheme.

Unul din scopurile, pe care şi le propune să le atingă modelul relațional este eliberarea utilizatorului de a specifica căile de acces la date. Această problemă e cunoscută sub denumirea de problema navigației logice. Însă, dacă baza de date constă din mai multe relații independența navigației logice nu este asigurată.

De exemplu, fie baza de date are două relații *angajați*(FUNCȚIONAR DEPARTAMENT) şi *departamente*(DEPARTAMENT MANAGER). Pentru a obține

asocierile FUNCȚIONAR MANAGER se joncționează relațiile *angajați* şi *departamente* şi apoi relația obținută se proiectează pe atributele FUNCȚIONAR şi MANAGER. Dar indicarea operațiilor şi este specificarea căilor de acces. Dacă baza de date se restructurează, reprezentându-se printr-o singură relație, atunci trebuie să se modifice corespunzător şi programele ce specifică joncțiunea.

Formulând o interpelare (cerere) la baza de date ce se referă la mai multe relații din baza de date e comod a interpreta lumea reală ca o singură relație, schema căreia include toate atributele din schemele relațiilor bazei. Această relație se numeşte relație universală, iar schema ei – schema relației universale sau schema universală.

Modelul relației universale realizează complet independența navigației logice, excluzând astfel definirea unor căi de acces neoptimale din partea unor utilizatori neinițiați. Deci acest model facilitează interacțiunea sistem-utilizator, cerând de la ultimul doar cunoaşterea atributelor şi semanticii.

Cea mai strictă formă de realizare a relației universale constă în construirea propriu-zisă a bazei de date dintr-o singură relație pe mulțimea de atribute universală U. Dar aici apar multe dezavantaje. În primul rând, nu toate tuplurile vor avea valori definite. În al doilea rând, păstrarea tuturor datelor într-o singură relație inevitabil va genera o serie de anomalii de actualizare.

De aceea, realmente, baza de date constă dintr-o mulțime de relații normalizate definite pe submulțimi de atribute ale schemei relației imaginare universale. Însă în acest caz baza de date trebuie să satisfacă unele condiții.

Una din condiții presupune că baza de date trebuie să posede proprietatea joncțiunii fără pierderi. Această problemă a fost abordată în capitolul precedent.

A doua – că descompunerea relației universale imaginare conservă dependențele. Această cerință va fi considerată în acest capitol.

A treia condiție presupune ca atributele în schema universală joacă un singur "rol". Astfel ambiguitățile sunt excluse. Dacă numele exprimă diverse noțiuni problema se soluționează prin renumirea atributelor sau divizarea unui atribut în mai multe .

5.2. Descompunerea relațiilor cu conservarea dependențelor

S-a constatat că e binevenit ca descompunerea unei relații (inclusiv celei universale) să posede proprietatea joncțiunii fără pierderi. Aceasta este garanția că relația poate fi refăcută din proiecțiile sale.

O altă importantă proprietate a descompunerii unei relații r(R) pe mulțimea de scheme $R_1,...,R_m$, unde $R=R_1 \ldots R_m$, constă că mulțimea de dependențe valide în r(R) să se deducă din dependențele valide în proiecțiile $\pi_{R1}(r)$, ..., $\pi_{Rm}(r)$.

Definiția 5.1. Fie F o mulțime de dependențe funcționale asupra schemei R. Proiecția mulțimii F asupra unei mulțimi de atribute Z, notată $\pi_Z(F)$, este mulțimea de dependențe $X \rightarrow Y$ din F^+ şi $XY \subseteq Z$.

Să observăm că dependența $X \rightarrow Y$ nu neapărat trebuie să aparțină mulțimii F. E de ajuns ca ea să aparțină închiderii F^+, adică $F \models X \rightarrow Y$.

Definiția 5.2. Fie relația r(R) este descompusă pe mulțimea de scheme R_1,..., R_m, unde $R=R_1 \ldots R_m$, şi fie o mulțime F de dependențe asupra R. Vom spune că descompunerea relației r(R) asupra R_1,..., R_m conservă dependențele F, dacă $\{\pi_{R1}(F) \cup ... \cup \pi_{Rm}(F)\}| = F$.

Tendința de conservare a dependențelor e firească. Dependențele sunt constrângeri de integritate asupra relațiilor cu schema dată. Dacă din dependențele proiectate nu ar urma mulțimea F, atunci s-ar găsi o descompunere $\pi_{R1}(r)$,..., $\pi_{Rm}(r)$ a relației r(R) ce nu satisface mulțimea de dependențe F, dar posedă proprietatea joncțiunii fără pierderi.

Exemplul 5.1. Fie relația r(ORAŞ ADRESĂ COD), unde ADRESĂ este denumire de stradă, număr de casă, iar COD este codul oficiului poştal ce deserveşte o anumită adresă. În această relație avem valide următoarele dependențe funcționale

$$F = \{\text{ORAŞ ADRESĂ} \rightarrow \text{COD}, \text{COD} \rightarrow \text{ORAŞ}\}.$$

Adică adresa completă determină funcțional codul poştal, dar codul poştal e de ajuns pentru a determina oraşul. Descompunerea relațiilor r asupra schemelor ORAŞ COD şi ADRESĂ COD posedă proprietatea joncțiunii fără pierderi. Într-adevăr, întrucât COD→ORAŞ, atunci COD→→ORAŞ. Dar conform teoremei 4.3 relația r se descompune fără pierderi în două relații definite pe schemele de mai sus.

Proiecția mulțimii de dependențe F asupra schemei ADRESĂ COD produce numai dependențe triviale ce urmează din axioma reflexivității, în timp ce proiecția mulțimii F pe schema COD ORAŞ produce dependența COD→ORAŞ şi dependențele triviale. Deci

$$\{\pi_{\text{ADRESĂ COD}}(F) \cup \pi_{\text{COD ORAŞ}}(F)\}| \neq F.$$

Din altă parte descompunerea unei relații poate conserva dependențele, dar să nu posede proprietatea joncțiunii fără pierderi.

Exemplul 5.2. Fie relația r(ABCD) şi fie mulțimea F={A→B, C→D} de dependențe funcționale valide în r. Descompunerea relației r în $\pi_{AB}(r)$ şi $\pi_{CD}(r)$ conservă dependențele din F, însă ea nu posedă proprietatea joncțiunii fără pierderi. Într-adevăr, tabloul ce defineşte descompunerea arată ca în fig.5.1. Asupra acestui tablou nu pot fi aplicate F-regulile şi întrucât el nu conține un tuplu-scop, descompunerea nu posedă proprietatea joncțiunii fără pierderi.

tab	A	B	C	D
	a_1	a_2	b_{13}	b_{14}
	b_{21}	b_{22}	a_3	a_4

Fig.5.1.

În capitolul 1 s-a definit noțiunea de cheie a unei relații sau a unei scheme şi sau discutat problemele legate de această noțiune. E evident că noțiunea de cheie este în strânsă corelație cu noțiunea de dependență funcțională. Prin urmare, aici ne vom opri asupra repetării noțiunii de cheie în termenii dependențelor funcționale.

Vom presupune mai departe, când vom avea nevoie, că schema unei relații constă din două componente S=(R, F), unde R este propriu-zis schema, iar F mulțimea de dependențe definite pe mulțimea R.

Definiția 5.3. Fie S=(R, F) o schemă relațională. O submulțime de atribute K a schemei R se numeşte supercheie pentru schema S, dacă K→R este în F^+. Submulțimea de atribute K din R se numeşte cheie, dacă K e supercheie şi pentru orice submulțime proprie K^1 a supercheii K dependența K^1→R nu este în F^+. Dependențele de forma K→R, unde K este cheie sau supercheie a schemei S, le vom numi *dependențe cheie*.

Exemplul 5.3. Fie schema relațională S=(ABCDEFG, {A→BCF, C→D, BD→E, EF→G}). Să găsim cheile şi supercheile acestei scheme.

Calculăm A^+=ABCDEFG, C^+=CD, $(BD)^+$=BDE şi $(EF)^+$=EFG. Întrucât atributul A determină funcțional toate atributele schemei, el este cheie. Uniunea atributului A cu orice submulțime din BCDEFG formează supercheile schemei S.

5.3. Anomalii şi redundanțe

De ce o schemă a bazei de date poate fi "rea"? Anomaliile, care apar în lucrul cu baza de date, se produc datorită dependențelor "nedorite" care se manifestă între atributele din cadrul schemelor relațiilor din baza de date. Aceste dependențe determină creşterea redundanței datelor şi reducerea flexibilității structurii bazei de date, făcând extrem de dificil lucrul cu ea.

Deci, în primul rând, o schemă poate fi ineficientă fiindcă conține o mulțime de *date redundante*.

În al doilea rând, ca o consecință a primei cauze, actualizarea unei baze redundante poate duce la situația când ea va conține fapte logic contradictorii. O parte de date pot rămâne nemodificate. Deci o bază de date "rea" duce la apariția unor *inconsistențe la modificarea datelor*.

În al treilea rând, o bază de date "rea" *poate limita posibilitatea de inserare a datelor*. Într-o relație nu pot fi introduse date despre o entitate până nu se cunosc alte date conform restricțiilor de integritate ale entității.

În al patrulea rând, pot apărea *pierderi de date la ştergere*. În mod normal, prin operația de ştergere trebuie să se poată elimina din baza de date numai datele pe care dorim să le ştergem. Atunci când, concomitent cu aceste date sunt şterse şi altele, care nu mai pot fi reconstruite din baza de date, spunem că la operația de ştergere se produc pierderi de date.

5.4. Forma normală unu

Precum s-a menționat în capitolul 1, domeniile atributelor sunt simple, adică ele sunt atomice (nu pot fi descompuse din punctul de vedere al sistemului de gestiune al bazei de date). Cu alte cuvinte, valorile ce le pot primi atributele nu sunt liste, mulțimi sau alte structuri complexe.

Definiția 5.4. Schema relațională R se găseşte în *forma normală unu*, dacă pentru orice atribut A din R valorile din dom(A) sunt atomice. Schema unei baze de date se

găseşte în forma normală unu, dacă orice schemă relaţională din ea este în forma normală unu.

Forma normală unu este forma de bază a relaţiilor, care figurează ca cerinţă minimală la majoritatea SGBD-urilor. Toate exemplele de relaţii considerate până aici au fost în forma normală unu.

Definirea noţiunii de valoare atomică e destul de dificilă. Valoarea atomică dintr-o aplicaţie în altă aplicaţie poate fi considerată nonatomică. De aceea ne vom conduce de următoarea regulă: atributul nu este atomic, dacă în aplicaţii el se utilizează pe părţi.

În general se cunosc două tipuri de atribute nonatomice. Unul din ele sunt listele sau mulţimile de valori.

cămin	NUME_STUDENT	CAMERĂ
	Ionescu, Vasilachi	301
	Popovici	302
	Gârlea, Efim	303

Fig.5.2(a)

cămin	NUME_STUDENT	CAMERA
	Ionescu	301
	Vasilachi	301
	Popovici	302
	Gârlea	303
	Efim	303

Fig.5.2(b)

Exemplul 5.4. Relaţia *cămin* din fig.5.2(a) nu se află în forma normală unu, fiindcă atributul NUME_STUDENT nu e atomic.

Aducerea relaţiei *cămin* în forma normală unu presupune eliminarea listelor de valori. Pentru orice valoare din listă pe care o poate primi atributul NUME_STUDENT se formează un tuplu aparte, conţinând numele studentului şi camera unde locuieşte. Relaţia *cămin* adusă în forma normală unu arată ca în fig.5.2(b).

Alt tip de atribute nonatomice sunt atributele compuse.

data_naştere	NUME_STUDENT	DATA_NAŞTERE
	Ionescu	9 ianuarie 1979
	Vasilachi	21 februarie 1978
	Popovici	15 decembrie 1977
	Gârlea	6 iunie 1979
	Efim	9 ianuarie 1978

Fig.5.3(a)

Exemplul 5.5. Relaţia *data_naştere* din fig.5.3(a) nu este în forma normală unu, dacă dorim să avem accesul la unele componente ale atributului DATA_NAŞTERE.

Pentru a aduce relaţia *data_naştere* în forma normală unu atributul compus DATA_NAŞTERE se divizează în trei atribute ZI, LUNĂ, AN. Noua relaţie *data_naştere* din fig.5.3(b) se găseşte în forma normală unu.

data_naştere	NUME_STUDENT	ZI	LUNĂ	AN
	Ionescu	9	ianuarie	1979
	Vasilachi	21	februarie	1978
	Popovici	15	decembrie	1977
	Gârlea	6	iunie	1979
	Efim	9	ianuarie	1978

Fig.5.3(b)

Utilitatea formei normale unu este destul de evidentă. Listele de valori distrug structura naturală dreptunghiulară a unei relaţii. Este extrem de greu să te referi la un element din grupul de valori, fiindcă trebuie specificată cumva poziţia valorii căutate. Şi, bineînţeles, că operaţia de actualizare nu poate fi efectuată. Cu atât mai mult, că cheia NUME_STUDENT a relaţiei *cămin* nu poate fi specificată în cazul unei liste de valori.

În afară de aceasta, diverse părţi ale unui atribut partiţionat pot să se comporte în mod diferit din punctul de vedere al dependenţelor. Presupunem că în prima relaţie *data_naştere* din fig.5.3(a) s-a adăugat atributul SEMN valorile căruia sunt semnele zodiacului. Tot ce se poate de făcut în această relaţie este să stabilim dependenţa funcţională DATA_NAŞTERE→SEMN. Dar această constrângere de integritate permite ca doi indivizi născuţi în aceeaşi zi şi aceeaşi lună, dar ani diferiţi, să aibă semne diferite ale zodiacului.

Relaţia a doua *dată_naştere* din fig.5.3(b) este lipsită de acest dezavantaj, fiindcă aici se poate defini dependenţa funcţională ZI, LUNĂ→SEMN, ce corespunde semanticii semnului zodiacului, care nicidecum nu depinde de anul în care este născută persoana dată, ci numai de ziua şi luna naşterii. Deci unul din avantajele formei normale unu constă în aceea că ea poate exprima dependenţele la aşa grad de detaliere, de care avem nevoie.

5.5. Forma normală doi

Apariţia formei normale doi a fost motivată de reducerea redundanţei şi eliminarea unor anomalii ce apar la actualizarea schemelor în forma normală unu.

Considerăm relaţia r din fig. 5.4.

r	DISCIPLINĂ	PROFESOR	GRUPĂ	ŞEF_GR
	Baze de date	Popescu	CIB-941	Vasilachi
	Programarea logică	Petrache	CIB-942	Gârlea
	Structuri de date	Ciobanu	CIB-942	Gârlea
	Cerc. operaţionale	Cazacu	CIB-942	Gârlea

Fig.5.4. Relația r în forma normală unu

Relația r(DISCIPLINĂ PROFESOR GRUPĂ ŞEF_GR) este constrânsă de două dependențe funcționale: GRUPĂ→ŞEF_GR, semnificând că grupa de studenți are un singur şef şi GRUPĂ DISCIPLINĂ→PROFESOR ce presupune că o disciplină într-o grupă de studiu este predată de un singur profesor. Este evident că singura cheie a acestei relații este mulțimea {GRUPĂ DISCIPLINĂ}.

Relația dată se găseşte în forma normală unu. Dar să observăm dezavantajele ce le posedă o relație cu astfel de schemă.

În primul rând, sunt limitate posibilitățile de inserare a datelor. În relația r nu pot fi introduse date despre o grupă, adică şeful grupei, decât atunci când se cunoaşte măcar o disciplină ce va fi predată în această grupă. Atributul DISCIPLINĂ face parte din cheie şi nu poate avea valoare nedeterminată.

În al doilea rând, pot fi pierderi de date la ştergere. De exemplu, în situația când disciplina Baze de date nu se mai predă grupei 941 tuplul dat trebuie şters. Însă ştergerea acestui tuplu din relația considerată determină pierderea datelor despre şeful grupei 941, întrucât acestei grupe nu se mai predau de acum nici o disciplină (fie din cauza că pentru această grupă s-a terminat semestrul de studiu mai devreme în legătură cu plecarea la practică).

În al treilea rând, persistă o redundanță de date. De exemplu, faptul că Gârlea este şeful grupei CIB-942 se repetă de trei ori.

Această redundanță implică al patrulea dezavantaj: apariția unor inconsistențe la modificarea datelor. Presupunem că s-a schimbat şeful grupei 942. Modificarea tuplurilor poate duce la apariția inconsistențelor, dacă numele şefului de grupă nu este actualizat în toate tuplurile. În mod normal, numele şefului grupei trebuie de scris o singură dată, lucru posibil de realizat numai dacă datele despre grupă s-ar păstra într-o relație separată.

Din punctul de vedere al actualizării fără anomalii şi îndepărtării redundanței, păstrarea datelor în două relații r_1şi r_2 (vezi fig. 5.5(a), 5.5(b)) e binevenită.

r_1	DISCIPLINĂ	PROFESOR	GRUPĂ
	Baze de date	Popescu	CIB-941
	Programare logică	Petrache	CIB-942
	Structuri de date	Ciobanu	CIB-942
	Cerc. operaționale	Cazacu	CIB-942

Fig.5.5(a) Relația r_1

r_2	GRUPĂ	ŞEF
	CIB-941	Vasilachi
	CIB-942	Gârlea

Fig.5.5(b) Relația r_2

Este evident că relația r se restabileşte din r_1şi r_2, adică $r = r_1 \,|\times|\, r_2$. În afară de aceasta, au dispărut anomaliile de actualizare şi ne-am eliberat de careva redundanță.

Să trecem la expunerea strictă a formei normale doi.

Definiția 5.4. Fie relația r(R) şi A∈R. Atributul A se numeşte *primar*, dacă el aparține unei chei a schemei R şi *nonprimar* în caz contrar.

Definiția 5.5. Fie X→A o dependență funcțională netrivială. Atributul A este *parțial dependent* de X, dacă există o submulțime proprie Y a mulțimii X şi Y→A. Dacă nu există o astfel de submulțime proprie, se spune că A *depinde complet* de X.

Exemplul 5.6. Fie F={DISCIPLINĂ GRUPĂ → PROFESOR, GRUPĂ→ŞEF} asupra schemei R=DISCIPLINĂ PROFESOR GRUPĂ ŞEF. Aici atributul ŞEF depinde parțial de DISCIPLINĂ GRUPĂ, iar atributul PROFESOR depinde complet de DISCIPLINĂ GRUPĂ. Întrucât mulțimea de atribute {DISCIPLINĂ, GRUPĂ} este singura cheie a schemei R, atributele DISCIPLINĂ şi GRUPĂ sunt primare, iar PROFESOR şi ŞEF sunt nonprimare.

Definiția 5.6. Schema unei relații R se găseşte în *forma normală doi* în raport cu mulțimea de dependențe funcționale F, dacă ea se găseşte în forma normală unu şi orice atribut nonprimar nu depinde parțial de careva cheie a schemei R. Schema bazei de date se găseşte în forma normală doi, dacă orice schemă relațională a ei se găseşte în forma normală doi.

Exemplul 5.7. Fie R şi F din exemplul 5.6. Schema bazei de date Db={R} nu se găseşte în forma normală doi, fiindcă atributul ŞEF depinde parțial de cheia {DISCIPLINĂ, GRUPĂ}. Dar schema Db={DISCIPLINĂ PROFESOR GRUPĂ, GRUPĂ ŞEF} bazei de date din fig.5.5(a) şi 5.5(b) se găseşte în forma normală doi.

Într-adevăr, în schema relațională R_1 = DISCIPLINĂ PROFESOR GRUPĂ, atributul PROFESOR este nonprimar şi el depinde complet de cheia DISCIPLINĂ GRUPĂ. Cheia schemei R_2 = GRUPĂ ŞEF este atributul GRUPĂ. Deci singurul atribut nonprimar e ŞEF care depinde complet de cheie.

Problema determinării, dacă un atribut e primar e legată de problema găsirii cheilor unei relații. Prin urmare, determinarea dacă o schemă se găseşte în forma normală doi e o problemă NP-completă.

5.6. Forma normală trei

Considerăm relația r(STUDENT DISCIPLINĂ PROFESOR COD_PROF) din fig.5.6.

r	STUDENT	DISCIPLINĂ	PROFESOR	COD_PROF
	Vasilachi	Baze de date	Popescu	P.021
	Marin	Baze de date	Popescu	P.021
	Guțu	Baze de date	Popescu	P.021
	Vasilachi	Programarea logică	Petrache	P.024

Fig.5.6. Relația r(STUDENT DISCIPLINĂ PROFESOR COD_PROF) în forma normală doi

Relaţia r este constrânsă de următoarele dependenţe funcţionale:
STUDENT DISCIPLINĂ→COD_PROF,
COD_PROF→PROFESOR,
PROFESOR→COD_PROF.

Cheia acestei relaţii e formată din două atribute STUDENT şi DISCIPLINĂ. Deci ele sunt primare. Atributele PROFESOR şi COD_PROF sunt nonprimare. Întrucât ele depind complet de cheie, relaţia r se găseşte în forma normală doi.

Dar să observăm că şi această relaţie nu e lipsită de unele anomalii.

În relaţia r nu poate fi inserat numele unui profesor şi codul lui, decât atunci când acest profesor predă măcar o disciplină în momentul dat. Deci în r se manifestă anomalia de inserare a datelor.

În situaţia, când profesorul Petrache nu mai predă disciplina Programarea logică, operaţia de ştergere a acestui tuplu determină pierderea codului acestui profesor. Deci schema dată nu e liberă nici de anomaliile de ştergere a datelor.

În afară de aceasta, perechea de date <Popescu P.021> se repetă de atâtea ori câţi studenţi ascultă prelegerile profesorului Popescu. Prin urmare, persistă o redundanţă, care poate duce la apariţia anomaliilor de modificare şi inconsistenţă a datelor. De exemplu, dacă codul profesorului Popescu se va schimba cu P.022, atunci neapărat trebuie modificate toate tuplurile (dar nu se ştie numărul lor) şi de făcut substituirea corespunzătoare. O singură greşeală poate duce la violarea dependenţelor PROFESOR→COD_PROF şi COD_PROF→PROFESOR.

Aceste anomalii pot fi eliminate, dacă relaţia r se descompune în două relaţii r_1 şi r_2 din fig.5.7(a) şi 5.7(b). În afară de aceasta, relaţia r poate fi restabilită prin joncţiunea proiecţiilor sale r_1 şi r_2. Existenţa joncţiunii fără pierderi este evidentă.

r_1	STUDENT	DISCIPLINĂ	COD_PROF
	Vasilachi	Baze de date	P.021
	Marin	Baze de date	P.021
	Guţu	Baze de date	P.021
	Vasilachi	Programarea logică	P.024

Fig.5.7(a). Relaţia r_2(STUDENT DISCIPLINĂ COD_PROF)

r_2	PROFESOR	COD_PROF
	Popescu	P.021
	Petrache	P.024

Fig.5.7(b). Relaţia r_2(COD_PROF PROFESOR)

Definiţia 5.7. Fie relaţia r(R), $X,Y \subseteq R$ şi $A \in R$. Vom spune că atributul A *depinde tranzitiv* de X prin Y, dacă sunt satisfăcute condiţiile:

(1) $X \rightarrow Y$,
(2) $Y \not\rightarrow X$ (adică X nu depinde funcţional de Y),
(3) $Y \rightarrow A$
(4) $A \notin XY$.

În această definiție, condițiile (1) şi (3) implică $X \rightarrow A$ conform regulii tranzitivității. Condiția (2) este esențială, de altfel $X \leftrightarrow Y$. Condiția (4), de asemenea, e esențială, în caz contrar $X \rightarrow A$ poate fi dedusă cu ajutorul reflexivității (dacă $A \in X$) sau cu ajutorul regulii proiectivității (dacă $A \in Y$).

Exemplul 5.8. Considerăm relația din fig.5.6. Atributul nonprimar PROFESOR este tranzitiv dependent de cheia {STUDENT, DISCIPLINĂ} prin COD_PROF fiindcă

(1) STUDENT DISCIPLINĂ$\rightarrow$ COD_PROF (întrucât {STUDENT, DISCIPLINĂ} e cheie şi deci determină toate atribuitele din schema relației r),
(2) COD_PROF$\not\rightarrow$STUDENT DISCIPLINĂ,
(3) COD_PROF$\rightarrow$PROFESOR,
(4) PROFESOR$\notin$STUDENT DISCIPLINĂ.

Definiția 5.8. Schema R se găseşte în *forma normală trei* în raport cu o mulțime de dependențe funcționale F, dacă R se găseşte în forma normală unu şi orice atribut nonprimar nu depinde tranzitiv de careva cheie a schemei R. Schema bazei de date $Db=\{R_1, ...,R_m\}$ se găseşte în forma normală trei, dacă orice schemă relațională $R_i \in Db$, $1 \leq i \leq m$, se găseşte în forma normală trei.

Exemplul 5.9. Schema relației din fig.5.6 nu se găseşte în forma normală trei, fiindcă, cum s-a văzut în exemplul 5.8, atributul nonprimar PROFESOR depinde tranzitiv de cheia {STUDENT, DISCIPLINĂ}. În schimb, schema bazei de date din fig.5.7(a) şi 5.7(b) Db = {STUDENT DISCIPLINĂ COD_PROF, COD_PROF PROFESOR} se găseşte în forma normală trei.

Într-adevăr, să examinăm pe rând schemele relaționale din Db. Cheia relației r_1 este {STUDENT, DISCIPLINĂ}. Atributele antrenate în această cheie sunt primare. Singurul atribut nonprimar este COD_PROF. El nu depinde tranzitiv de {STUDENT, DISCIPLINĂ}. Deci relația r_1 (sau schema ei) se găseşte în forma normală trei.

Cât priveşte relația r_2, în ea sunt valide dependențele funcționale COD_PROF$\rightarrow$PROFESOR şi PROFESOR$\rightarrow$ COD_PROF. Deci r_2 are două chei COD_PROF şi PROFESOR. Întrucât schema relației r_2 nu conține atribute nonprimare ea este în forma normală trei.

E firească întrebarea, care este corelația dintre forma normală trei şi forma normală doi. Răspunsul îl dă următoarea teoremă.

Teorema 5.1. Schema unei relații ce se găseşte în forma normală trei se găseşte şi în forma normală doi.

Demonstrație. Teorema poate fi reformulată în felul următor: dacă schema unei relații nu se găseşte în forma normală doi, atunci ea nu se găseşte nici în forma normală trei. Deci trebuie să arătăm că dependența parțială implică dependența tranzitivității.

Fie schema relațională S=(R,F) şi presupunem că atributul nonprimar A e parțial dependent de o cheie, fie K. Adică $K \rightarrow A \in F^+$ şi $K^1 \rightarrow A \in F^+$, unde $K^1 \subset K$. Conform regulii reflexivității, $K^1 \subset K$ implică $K \rightarrow K^1 \in F^+$ şi atunci condiția (1) a definiției 5.7 e satisfăcută. Condiția (2) tot e satisfăcută, adică $K^1 \not\subset K$, fiindcă în caz contrar K nu este cheie. Condiția (3) urmează din ipoteză, iar condiția (4) e satisfăcută din presupunerea

că A este atribut nonprimar, adică nu aparţine cheii K şi deci nici lui K^1. Prin urmare, atributul nonprimar A depinde tranzitiv de cheia K.

Exemplul 5.10. Schema bazei de date Db={DISCIPLINĂ PROFESOR GRUPĂ, GRUPĂ ŞEF} din fig.5.5(a) şi 5.5(b) se găseşte în forma normală trei, deci se găseşte şi în forma normală doi.

Într-adevăr, schema R_1 = DISCIPLINĂ PROFESOR GRUPĂ se găseşte în forma normală trei, fiindcă cheia e {DISCIPLINĂ, GRUPĂ}, iar singurul atribut nonprimar este PROFESOR şi el nu depinde tranzitiv de cheie. Cheia schemei R_2 = GRUPĂ ŞEF este atributul GRUPĂ. Atributul nonprimar ŞEF nu depinde tranzitiv de GRUPĂ.

E uşor de observat, din cele expuse mai sus, că definiţia formei normale trei poate fi formulată şi altfel.

Definiţia 5.9. Schema unei relaţii se găseşte în forma normală trei, dacă orice atribut ce depinde tranzitiv de cheie este primar.

Atunci putem formula următoarea teoremă.

Teorema 5.2. Schema R se găseşte în forma normală trei în raport cu mulţimea de dependenţe funcţionale F, dacă pentru orice dependenţă netrivială $X \rightarrow A \in F^+$

(1) X este supercheie pentru R

sau

(2) A este atribut primar.

Demonstraţie. Fie $X \rightarrow A$ o dependenţă netrivială şi fie K o cheie a schemei R. Din definiţia 5.9 reiese că nu e necesară examinarea cazului, când A este atribut primar. Condiţia (2) e evidentă. Să arătăm că pentru orice atribut nonprimar A, dependenţa $K \rightarrow A$ nu este tranzitivă. Dacă presupunem că condiţia (1) nu e satisfăcută, atunci $K \rightarrow X \in F^+$ (fiindcă K e cheie) şi $X \not\rightarrow K$. Dar aceasta este dependenţa tranzitivă a atributului nonprimar de cheie.

Exemplul 5.11. Fie schema bazei de date Db = {(AC, {C→A}), (ABE, {AE→B}), (BCDEF, {BF→C, CD→EF, EF→CD})}.

Este uşor de constatat că schema Db se găseşte în forma normală trei. Într-adevăr, schemele relaţionale R_1 = (AC, {C→A}) şi R_2 = (ABE, {AE→B}) se găsesc în forma normală trei fiindcă nu au dependenţe tranzitive. Să examinăm schema R_3 = (BCDEFE, {BF→C, CD→EF, EF→CD}). Schema R_3 are trei chei BCD, BDF şi BEF. Dependenţele tranzitive BCD→EF, BDF→C, BEF→CD includ numai atribute primare (ele toate fac parte dintr-o cheie). Prin urmare şi schema R_3 este în forma normală trei.

5.7. Forma normală Boyce-Codd

Forma normală trei nu interzice dependenţa tranzitivă a atributelor primare de cheie. Însă şi unele relaţii în forma normală trei nu sunt lipsite de anomaliile de actualizare a datelor.

r	ORAŞ	ADRESĂ	COD

o_1	a_1	c_1
o_1	a_2	c_1
o_1	a_3	c_1
o_1	a_4	c_2

Fig.5.8. Relaţia r(ORAŞ ADRESĂ COD)
în forma normală trei

Exemplul 5.12. Considerăm relaţia r(ORAŞ ADRESĂ COD) din fig.5.8 examinată şi în exemplul 5.1. Relaţia r satisface dependenţele funcţionale ORAŞ ADRESĂ→COD şi COD→ORAŞ. Mulţimea de atribute {ORAŞ, ADRESĂ} formează cheia relaţiei. Atributul ORAŞ e primar. Nici un atribut nonprimar nu depinde tranzitiv de această cheie. Deci relaţia r se află în forma normală trei.

Însă, în ea nu putem introduce un tuplu ce conţine date despre un oraş şi codul lui, până nu cunoaştem adresa asociată de acest cod.

În afară de aceasta, în r se repetă perechea ORAŞ COD ce poate duce la apariţia inconsistenţelor la modificarea acestor date.

Dacă relaţia r e descompusă în două relaţii r_1 şi r_2 (precum sunt prezentate în fig.5.9(a) şi 5.9(b)), atunci aceste dezavantaje lipsesc.

r_1	COD	ADRESĂ
	c_1	a_1
	c_1	a_2
	c_1	a_3
	c_2	a_4

Fig.5.9(a). Relaţia r_1(COD ADRESĂ)

r_2	ORAŞ	COD
	o_1	c_1
	o_1	c_2

Fig.5.9(b). Relaţia r_2(ORAŞ COD)

Definiţia 5.10. Schema R se găseşte în forma normală Boyce-Codd în raport cu o mulţime de dependenţe funcţionale F, dacă pentru orice dependenţă $X \rightarrow Y \in F^+$, determinantul X este o supercheie a schemei R. Schema bazei de date se găseşte în forma normală Boyce-Codd, dacă orice schemă relaţională din ea se găseşte în forma normală Boyce-Codd.

Exemplul 5.13. Considerând relaţia din fig.5.8 ce satisface dependenţa funcţională COD→ORAŞ, conchidem că COD nu este supercheie. Deci schema acestei relaţii nu se găseşte în forma normală Boyce-Codd.

Examinăm relaţiile r_1 şi r_2 din fig.5.9(a) şi 5.9(b).

Cheia relaţiei r_1 constă din toată mulţimea de atribute {COD, ADRESĂ}. În r_1 nu e validă nici o dependenţă funcţională, în afara celor triviale. Deci schema R_1 = COD ADRESĂ se găseşte în forma normală Boyce-Codd.

Relația r_2 satisface dependența funcțională COD→ORAŞ. Atributul COD este cheie, deci şi supercheie. Prin urmare, r_2 se găseşte în forma normală Boyce-Codd.

Relația r poate fi restabilită din proiecțiile sale, r_1 şi r_2. Deci descompunerea dată posedă proprietatea joncțiunii fără pierderi. Însă descompunerea dată nu conservă dependențele funcționale. Dependența ORAŞ ADRESĂ→COD validă în relația r nu se deduce din dependențele valide în relațiile r_1 şi r_2.

Din exemplul 5.13 putem face concluzia că forma normală trei nu implică forma normală Boyce-Codd.

Următoarea afirmație stabileşte legătura dintre forma normală Boyce-Codd şi forma normală trei.

Teoremă 5.3. Dacă schema R se găseşte în forma normală Boyce-Codd, atunci R se găseşte şi în forma normală trei.

Demonstrație. Validitatea acestei afirmații urmează direct din definiția formei normale Boyce-Codd şi teorema 5.2.

5.8. Normalizarea prin descompunere

5.8.1. Aducerea schemelor în forma normală trei

Normalizarea este procesul de aducere a schemei într-o formă normală dată. Algoritmul FN3 aduce schema R în forma normală trei prin descompunere.

Algoritmul FN3 (R, F, Db)
Intrare: Schema R; F – o mulțime de dependențe funcționale definite pe schema R.
Ieşire: Db – schema bazei de date în forma normală trei.

```
begin
1     k:=1;
2     Rk:=R;
3     for i=1 to k do
4           keys (Ri,F,K);
5           AttrNP:=Ri \ ∪Kj, ∀Kj∈K;
6           while ∃ X→Y∈F+ & X⟶/R_i & X∩Y=∅ & XY⊆Ri & Y⊆AttrNP
            do
            begin
7                 k:=k+1;
8                 Rk:=XY;
9                 Ri:= Ri \ Y;
            end
10    Db:={R1,..., Rk};
      return (Db);
end.
```

La început vom porni de la ideea că orice schemă ce nu se găseşte în forma normală trei poate fi descompusă într-o serie de scheme ce se găsesc în forma normală trei. Descompunerea presupune divizarea unei scheme R în două scheme R_1 şi R_2 astfel că orice relație r(R) ce satisface mulțimea dată de dependențe F se proiectează fără pierderi asupra schemelor R_1 şi R_2, adică $r = \pi_{R1}(r) |\times| \pi_{R2}(r)$. Acest proces se repetă

asupra schemelor R_1 şi R_2, bineînțeles, până subschemele formate sunt aduse în forma normală trei.

În linia 3 variabila i denotă schema curentă, iar k numărul de scheme deja create.

Linia 4 determină mulțimea K de chei a schemei curente R_i.

Linia 5 construieşte mulțimea AttrNP de atribute nonprimare din R_i.

Linia de bază a algoritmului este 6. Ea analizează dacă schema curentă se găseşte în forma normală trei. Pentru fiecare dependență validă în schema curentă cu determinatul format numai din atribute nonprimare se verifică, dacă determinantul ei este supercheie. Dacă nu, atunci conform teoremei 5.2 schema R_i nu se găseşte în forma normală trei. În acest caz (liniile 7-9) se produce descompunerea schemei R_i: se formează o nouă schemă din atributele implicate în dependență, R_i=XY, iar schema R_i e substituită de $R_i \setminus Y$. Conform teoremei 4.3, această descompunere posedă proprietatea joncțiunii fără pierderi. Apoi continuă analiza schemelor deja formate. Dacă în cadrul lor nu se mai manifestă dependențe ce satisfac condițiile din linia 6, atunci schema obținută se găseşte în forma normală trei.

Exemplul 5.14. Considerăm schema relațională R = DPOCSN, unde D e disciplină, P – profesor, O – ora, C – clasa, S – student şi N – nota. Presupunem că pe schema dată sunt definite următoarele dependențe funcționale:

$D \rightarrow P$ – orice disciplină e predată de un singur profesor;

$OC \rightarrow D$ – într-o clasă în acelaşi timp se predă o singură disciplină;

$OP \rightarrow C$ – profesorul într-un anumit timp se găseşte într-o singură clasă;

$DS \rightarrow N$ – orice student are o singură notă finală la o disciplină;

$OS \rightarrow C$ – studentul se găseşte la ora dată într-o singură clasă.

Să se aducă schema R în forma normală trei.

De la începutul algoritmului se formează schema R_1 = R. În linia 4 a fost găsită o singură cheie pentru R_1 şi anume OS. Deci {D,P,C,N} formează mulțimea de atribute nonprimare. Pentru a aduce schema R_1 la forma normală trei considerăm dependența funcțională $D \rightarrow P$ care satisface toate condițiile din linia 6. Formăm o nouă schemă (linia 8) R_2 = DP, iar R_1 este substituită de R_1 = DOCSN. Schema R_2 se găseşte în forma normală trei, fiindcă nu există vre-o dependență definită pe această schemă şi să satisfacă condițiile liniei 6. Schema R_1 nu este în forma normală trei, fiindcă există o dependență, de exemplu, $OC \rightarrow D$ ce satisface condițiile liniei 6. Se formează a treia schemă R_3 = OCD, dar R_1 devine de acum egală cu OSCN. Este evident că schema R_3 se găseşte în forma normală trei. Schema R_1 de asemenea se găseşte în forma normală trei, fiindcă singura dependență, $OS \rightarrow C$, definită pe atributele schemei R_1 nu satisface condițiile liniei 6.

Deci schema R s-a descompus fără pierderi în R_1, R_2 şi R_3. Schema bazei de date Db={R_1,R_2,R_3} se găseşte în forma normală trei.

Să menționăm că schema Db={R_1,R_2,R_3} se găseşte şi în forma normală Boyce-Codd.

5.8.2. Aducerea schemei la forma normală Boyce-Codd

Adresându-ne la definiția formei normale Boyce-Codd, un algoritm de aducere a schemelor în această formă poate fi prezentat astfel.

Algoritmul FNBC (R, F, Db)

Intrare: Schema R;

F- o mulţime de dependenţe funcţionale asupra schemei R.
Ieşire: Db – schema bazei de date în forma normală Boyce-Codd.
begin
k:=1;
R_k:=R;
for i=1 to k do
while $\exists X \rightarrow Y \in F^+ \& X \not\rightarrow R_i \& X \cap Y = \varnothing \& XY \subseteq R_i$ do
begin
k:=k+1;
R_k:=XY;
R_I:=$R_i \setminus Y$;
end
Db:={R_1,..., R_k};
return (Db);
end

Exemplul 5.15. Fie R = (ABCDEF, {AB→E, AC→F, AD→B, B→C, C→D}). Să se aducă schema R în forma normală Boyce-Codd.

R_1 = ABCDEF. Considerăm pe rând dependenţele funcţionale valide în R_1. Dependenţa AB→E nu participă la descompunere, fiindcă $(AB)^+$=ABCDEF. Considerăm dependenţă AC→F. $(AC)^+$ = ABCDEF, deci AC→F, de asemenea, nu poate fi aplicată la descompunerea schemei R_1. Acelaşi lucru putem spune şi despre dependenţa AD→B, fiindcă $(AD)^+$ = ABCDEF.

Determinantul dependenţei B→C nu este supercheie, fiindcă B^+ = BC. Deci R_1 se descompune în două scheme: R_2 = BC şi R_1 = ABDEF. Schema R_2 evident se găseşte în forma normală Boyce-Codd, fiindcă constă numai din două atribute. Altă dependenţă, ce poate fi aplicată de acum asupra schemei R_1 modificate, este $B \rightarrow D \in F^+$, unde $BD \subseteq R_1$. Construim schemele R_3 = BD şi R_1 = ABEF. Cheia schemei R_3 este B, iar a schemei R_1 - AB. Schema bazei de date în forma normală Boyce-Codd este Db={(ABEF, {AB}), (BC, {B}), (BD, {B})}.

5.8.3. Dezavantajele normalizării prin descompunere

Dat fiind faptul că algoritmul FN3 necesită calcularea cheilor schemei şi determinarea atributelor nonprimare, iar algoritmii FN3 şi FNBC necesită examinarea dependenţelor din F^+ valide în schema curentă, complexitatea procesului de normalizare nu e polinomială. Acesta e primul dezavantaj.

Într-al doilea rând, nu întotdeauna putem obţine un număr minimal de scheme relaţionale normalizate dintr-o schemă dată.

Exemplul 5.16. Fie schema R = ABCDE şi F = {AB→CDE, AC→BDE, B→C, C→B, C→D, B→E}. Să se aducă R în forma normală trei.

Cheile schemei R sunt K = {AB, AC}. Aplicăm pentru descompunere dependenţa C→D. Atunci

R_2 = CD, K_2 = {C};
R_1 = ABCE, K_1 = {AB, AC}.

Mai departe pentru descompunerea schemei R_1 utilizăm dependența $B \rightarrow E$:
$R_3 = BE$, $K_3 = \{B\}$;
$R_1 = ABC$, $K_1 = \{AB, AC\}$.
Schema finală în forma normală trei este
Db = {(ABC, {AB, AC}),(CD, {C}), (BE, {B})}.
Există, în schimb, altă descompunere cu mai puține scheme relaționale. Dacă utilizăm dependența funcțională $B \rightarrow DE \in F^+$, atunci obținem schemele
$R_2 = BDE$, $K_1 = \{B\}$;
$R_1 = ABC$, $K_1 = \{AB, AC\}$.
Deci, Db = {(ABC, {AB, AC}), (BDE, {B})}.

A treia problemă constă în apariția dependențelor parțiale în procesul descompunerii schemelor. Aceste dependențe generează scheme cu mai multe scheme relaționale decât e nevoie.

Exemplul 5.17. Fie schema R = ABCD şi F = {$A \rightarrow BCD$, $C \rightarrow D$}. Să se aducă schema R în forma normală trei.
Singura cheie a schemei R este A. Utilizăm dependența $BC \rightarrow D$ pentru descompunerea schemei R. Atunci
$R_2 = BCD$, $K_2 = \{BC\}$;
$R_1 = ABC$, $K_1 = \{A\}$.
În R_2 atributul D depinde parțial de cheia BC, deci poate fi aplicată dependența $C \rightarrow D$ pentru descompunerea schemei R_2:
$R_3 = CD$, $K_3 = \{C\}$;
$R_2 = BC$, $K_2 = \{BC\}$.
Deci, schema bazei de date în forma trei este Db={(ABC, {A}), (BC, {BC}), (CD, {C})}.
Însă, dacă pentru descompunere asupra schemei R e aplică deodată dependența $C \rightarrow D$, atunci obținem
$R_2 = CD$, $K_2 = \{C\}$;
$R_1 = ABC$, $K_1 = \{A\}$.
În acest caz Db={(ABC, {A}), (CD, {C})}. Ultima schemă a bazei de date este o submulțime a primei scheme.
Acest dezavantaj poate fi evitat, dacă dependențele utilizate în descompunere sunt reduse în stânga.

A patra problemă este că normalizarea prin descompunere nu întotdeauna conservă dependențele funcționale.

Exemplul 5.18. Schema bazei de date obținută în exemplul 5.14 nu conservă dependențele $OP \rightarrow C$ şi $DS \rightarrow N$. Schema bazei de date obținută în exemplul 5.15 nu conservă dependențele $AC \rightarrow F$ şi $AD \rightarrow B$.

A cincea problemă constă în faptul că normalizarea prin descompunere poate produce scheme, în care dependențele, ce pot fi utilizate mai departe în descompunere, sunt latente.

Exemplul 5.19. Fie pe schema R = ABCD e definită mulţimea de dependenţe funcţionale F = {A→B, B→C}. Cheia relaţiei R este AD. Dependenţa A→B poate fi utilizată în descompunerea schemei R:

R_2 = AB, K_2 = {A};

R_1 = ACD, K_1 = {AD}.

S-ar părea că schema R_1 se găseşte în forma normală trei, însă în R_1 există dependenţa funcţională latentă A→C, ce ar trebui să fie utilizată în descompunerea de mai departe.

5.9. Normalizarea prin sinteză

În această secţiune va fi prezentată o altă metodă de aducere a schemelor la forma normală trei, ce nu generează problemele descrise în secţiunea precedentă.

Metoda propusă este o procedură de sinteză, fiindcă pleacă de la mulţimea de dependenţe funcţionale F cu determinaţii dintr-un singur atribut şi produce schema bazei de date Db={R_1,..., R_m} asupra R= R_1...R_m. Schema bazei de date trebuie să satisfacă următoarele patru condiţii:

(1) Mulţimea de dependenţe formate de cheile fiecărei scheme R_i trebuie să fie o acoperire a mulţimii iniţiale F de dependenţe funcţionale, adică F ≡ {K→R_i | R_i∈Db, 1≤ i ≤ m, K - cheie}.

(2) Orice schemă relaţională R_i din Db se află în forma normală trei.

(3) Nu există o schemă a bazei de date ce satisface condiţiile (1) şi (2) cu mai puţine scheme relaţionale.

(4) Orice relaţie r(R) ce satisface F se descompune fără pierderi asupra schemei Db, adică r=π_{R1}(r)|x|...|x|π_{Rm}(r).

Să considerăm aceste condiţii.

Condiţia (1) garantează că descompunerea relaţiei r asupra schemei Db conservă dependenţele funcţionale F. În afară de aceasta, condiţia (1) ne asigură că unicele dependenţe valide în R_i, 1 ≤ i ≤ m, au în calitate de determinanţi cheile schemei R_i. Satisfacerea condiţiei (1) este soluţia problemei patru din secţiunea precedentă.

Condiţia (2) este scopul principal al normalizării şi necesitatea ei a fost detaliat studiată.

Condiţia (3) ne ocroteşte de scheme redundante, deci ea soluţionează problemele doi şi trei din secţiunea precedentă.

Condiţia (4) de asemenea a fost considerată.

Problema cinci din secţiunea precedentă nu apare graţie îndeplinirii concomitente a condiţiilor (1) şi (3). Iar problema unu nu se mai pune, fiindcă complexitatea algoritmului de sinteză descris mai jos este polinomială.

Algoritmul SYNT (F, Db)

Intrare: F – o mulţimea de dependenţe funcţionale

Ieşire: Db schema bazei de date în forma normală trei.

1. Se găseşte o acoperire nonredundantă F_n a mulţimii F.
2. Se construieşte o acoperire redusă în stânga F_r a mulţimii F_n.
3. Mulţimea F_r se partiţionează în clase de echivalenţă.
4. Se construieşte o mulţime J în felul următor. Fie J=∅. Pentru orice două dependenţe funcţionale din F_r cu determinaţii X şi Y, unde $X \leftrightarrow Y$, se modifică J, $J:=J \cup \{X \rightarrow Y, Y \rightarrow X\}$. Pentru orice $A \in Y$, dacă $X \rightarrow A$ se găseşte în F_r, atunci $F_r := F_r \setminus \{X \rightarrow A\}$. Acelaşi lucru e valabil şi pentru orice $B \in X$. Dacă $Y \rightarrow B \in F_r$, atunci $F_r := F_r \setminus \{Y \rightarrow B\}$.
5. Se elimină dependenţele tranzitive. Se găseşte o mulţime $F_r^1 \subseteq F_r$, ce satisface $(F_r^1 \cup J)^+ \equiv (F_r \cup J)^+$ şi nici o submulţime proprie a mulţimii F_r^1 nu satisface condiţia dată. Apoi se includ dependenţele din J în clasele de echivalenţă a mulţimii F_r^1 şi fie că obţinem mulţimea G de dependenţe funcţionale.
6. Se construiesc schemele $R_1,...,R_m$. Fiecare schemă R_i include atributele dependenţelor funcţionale din clasa de echivalenţă i şi în final obţinem schema bazei de date $Db:=\{R_1, ..., R_m\}$.

Să ne oprim acum asupra corectitudinii algoritmului. Algoritmul expus formează un număr minimal de scheme relaţionale în Db.

Teorema 5.4. Dacă Db este schema bazei de date sintetizate din mulţimea F, atunci Db conţine cel puţin $|\bar{E}_{Fn}|$ scheme relaţionale.

Demonstraţie. Dependenţele ce sunt incluse într-o schemă relaţională R_i, $1 \le i \le m$, au determinanţi echivalenţi. Deci Db conţine atâtea scheme în câte clase de echivalenţă este partiţionată mulţimea G (vezi algoritmul SYNT). Din lema 3.3 urmează $|\bar{E}_G|=|\bar{E}_{Fn}|$. Dar e cunoscut faptul că $|\bar{E}_F| \ge |\bar{E}_{Fn}|$, adică mulţimea nonredundantă constă dintr-un număr minimal de clase de echivalenţă.

Această teoremă ne garantează că algoritmul de sinteză satisface condiţia (3).

Condiţia (1) este asigurată, fiindcă $F \equiv F_n \equiv F_r \equiv G$.

Condiţia (2) e satisfăcută de pasul 5 al algoritmului ce elimină dependenţele tranzitive.

Acum să vedem dacă e satisfăcută şi condiţia (4). Cu toate că algoritmul de sinteză soluţionează toate cele cinci probleme din secţiunea 5.8.3, nu întotdeauna schema bazei de date posedă proprietatea joncţiunii fără pierderi. Adică nu întotdeauna e satisfăcută condiţia (4). Acest lucru nu se petrecea în cazul normalizării prin descompunere.

Exemplul 5.20. Fie $F=\{A \rightarrow C, B \rightarrow C\}$. Algoritmul de sinteză generează schema $Db=\{R_1, R_2\}$, unde

$R_1 = AC$, $K_1=\{A\}$;

$R_2 = BC$, $K_2=\{B\}$.

Însă e uşor de văzut că relaţia r(ABC) din fig.5.10 nu se descompune fără pierderi asupra R_1 şi R_2.

r	A	B	C

a_1	b_1	c_1
a_2	b_2	c_1

Fig.5.10.

Un alt dezavantaj al algoritmului de sinteză e legat de atributele ce nu sunt antrenate de mulțimea de dependențe funcționale F. Aceste două dezavantaje pot fi eliminate, introducând aşa-numita cheie universală.

Definiția 5.11. Fie Db = $\{R_1,...,R_m\}$ o schemă a bazei de date asupra atributelor R = $R_1...R_m$ şi F o mulțime de dependențe funcționale. Mulțimea $X \subseteq R$ se numeşte *cheie universală*, dacă $F \models X \rightarrow R$ şi nu există X^1, unde $X^1 \subset X$, ce ar satisface $F \models X^1 \rightarrow R$.

Deci, ca schema bazei de date să posede proprietatea joncțiunii fără pierderi, ea trebuie să conțină o schemă relațională în care o cheie a ei e universală.

Vom modifica algoritmul de sinteză pentru a elimina cele două dezavantaje menționate mai sus.

La mulțimea inițială de dependențe funcționale F se adaugă o dependență funcțională $R \rightarrow C$, unde R= $R_1...R_m$, iar $C \notin R$.

Este clar că la primul pas al algoritmului dependența $R \rightarrow C$ nu va fi eliminată, fiindcă ea nu e redundantă în F.

La al doilea pas ea va fi redusă în stânga, fie $R^1 \rightarrow C$.

La etapa de partiție, dacă ea va intra într-o clasă de echivalență cu alte dependențe, atunci ea se elimină din F_r şi algoritmul continuă mai departe. Dacă ea singură formează o clasă de echivalență, atunci $R^1 \rightarrow C$ generează schema $R_m = R^1C$ cu cheia R^1.

La sfârşitul algoritmului se elimină atributul C din schema R_m, deci $R_m = R^1$.

Exemplul 5.21. Fie F ca în exemplul 5.20, adică $F=\{A \rightarrow C, B \rightarrow C\}$. Adăugăm la F dependența $ABC \rightarrow D$.

Algoritmul de sinteză va genera schema Db={(AC, {A}), (BC, {B}), (ABD, {AB})}. Apoi eliminând din ultima schemă relațională atributul D, obținem schema bazei de date în forma normală trei ce posedă proprietatea joncțiunii fără pierderi Db={(AC, {A}), (BC, {B}), (AB, {AB})}.

Cu părere de rău, trebuie să recunoaştem că algoritmul modificat violează condiția (3) de minimalitate a schemei.

5.10. Forma normală patru

Definiția 5.12. Schema R se găseşte în *forma normală patru* în raport cu mulțimea de dependențe multivaloare şi funcționale M, dacă ea se găseşte în forma normală unu şi orice dependență $X \rightarrow\rightarrow Y$ din M^+ satisface.

(1) $X \rightarrow\rightarrow Y$ este trivială (adică $Y \subseteq X$ sau XY=R)

sau

(2) X este supercheie pentru schema R.

Schema bazei de date se găseşte în forma normală patru, dacă orice schemă a ei se găseşte în forma normală patru.

Procedura de aducere în forma normală patru se bazează pe teorema 4.3, care ne spune că o dependenţă $X \rightarrow\rightarrow Y$ e validă într-o relaţie r(R), dacă şi numai dacă r(R) este joncţiunea proiecţiilor $\pi_{XY}(r)$ şi $\pi_{XZ}(r)$, unde $Z = R \setminus XY$.

Procesul de normalizare decurge în felul următor. Se începe cu schema iniţială R. Dacă în această schemă e validă dependenţa multivaloare netrivială $X \rightarrow\rightarrow Y$ şi X nu e supercheie, atunci schema R se descompune în două scheme $R_1=XY$ şi $R_2=XZ$, unde $Z = R \setminus XY$. La rândul lor, schemele R_1 şi R_2 sunt considerate în privinţa satisfacerii condiţiilor de a fi în forma normală patru. Dacă careva schemă nu e în forma normală patru, procesul de descompunere continuă până toate schemele sunt normalizate. Evident că procesul este finit.

Exemplul 5.22. Fie mulţimea de dependenţe $M = \{C \rightarrow\rightarrow DE, A \rightarrow BC\}$ definită pe schema R=ABCDE. Schema R nu se află în forma normală patru, fiindcă $C \rightarrow\rightarrow DE$ nu e trivială şi C nu este cheie. Schema bazei de date ce constă din două scheme $R_1=CDE$ şi $R_2=ABC$ se găseşte în forma normală patru. Într-adevăr, cu toate că $A \rightarrow B \in M^+$, deci $A \rightarrow\rightarrow B \in M^+$ şi $A \rightarrow\rightarrow B$ nu este trivială, însă A este supercheie pentru R_2.

Exemplul 5.23. Să se normalizeze schema R=ABCDEI, dacă pe ea e definită mulţimea de dependenţe $M=\{A \rightarrow\rightarrow BCD, B \rightarrow AC, C \rightarrow D\}$. Fiindcă $A \rightarrow\rightarrow BCD$ nu este trivială şi A nu e supercheie schema R se descompune fără pierderi în schemele $R_1=ABCD$ şi $R_2=AEI$. Schema R_2 se găseşte în forma normală patru: pe ea e definită o singură dependenţă multivaloare trivială $A \rightarrow\rightarrow EI$. Cu toate că în M^+ este $B \rightarrow\rightarrow AC$, din $B \rightarrow AC$ şi $C \rightarrow D$ urmează $B \rightarrow ACD$. Deci, B este cheia schemei R_1 şi $B \rightarrow\rightarrow AC$ nu participă la descompunerea de mai departe a schemei R_1. Însă, din $C \rightarrow D$ urmează dependenţa netrivială $C \rightarrow\rightarrow D$ ce descompune schema R_1 în două: $R_3=CD$ şi $R_1=ABC$. Schema bazei de date în forma normală patru este $DB=\{ABC, AEI, CD\}$.

Teorema 5.5. Dacă schema R se găseşte în forma normală patru, atunci R se găseşte în forma normală Boyce-Codd.

Demonstraţie. Vom demonstra o afirmaţie echivalentă: dacă R nu se găseşte în forma normală Boyce-Codd, atunci R nu se găseşte nici în forma normală patru.

Fie R nu se găseşte în forma normală Boyce-Codd. Adică trebuie să existe o dependenţă funcţională netrivială $X \rightarrow A$ şi X nu este supercheie a schemei R. Atunci există un atribut B în R, încât $B \notin AX$ şi $X \not\rightarrow B$. Prin urmare, $B \in R \setminus AX$ şi atunci dependenţa $X \rightarrow\rightarrow A$ care este alter ego al dependenţei $X \rightarrow A$ nu e trivială. Din condiţiile că dependenţa $X \rightarrow\rightarrow A$ nu e trivială şi X nu e supercheie, urmează că R nu se găseşte în forma normală patru.

5.11. Forma normală proiecţie-joncţiune

Definiţia 5.13. Dependenţa joncţiune $|\times|(R_1, ..., R_m)$ este aplicabilă schemei R, dacă $R=R_1...R_m$.

Definiția 5.14. Schema R se găseşte în *forma normală proiecție-joncțiune* în raport cu o mulțime de dependențe joncțiune (dependențele multivaloare sunt aceleaşi dependențe joncțiune) şi funcționale J, dacă ea se găseşte în forma normală unu şi orice dependență joncțiune $|x|(R_1, ..., R_m)$ aplicabilă din J^+ este trivială sau orice R_i este supercheie pentru R.

Exemplul 5.24. Fie mulțimea de dependențe J = {|x|(ABCD, CDE, BDF), |x|(AB, BCD, AD), A→BCDE, BC→A} definită pe schema R = ABCDEF.

Schema R nu se găseşte în forma normală proiecție-joncțiune din cauza dependenței aplicabile |x|(ABCD, CDE, BDF).

Schema bazei de date Db = {ABCD, CDE, BDF} se găseşte în forma normală proiecție-joncțiune în raport cu J. Cu toate că dependența joncțiune |x|(AB, BCD, AD) e aplicabilă schemei relaționale ABCD, Db se găseşte în forma normală proiecție-joncțiune datorită faptului că orice subschemă AB, BCD şi AD este supercheie pentru schema ABCD.

Exemplul 5.25. Fie R = ABCDEF şi J = {|x|(ABC, ADEF), A→BCDE, BC→AF}.

Schema R se găseşte în forma normală proiecție-joncțiune, fiindcă cu toate că dependența |x|(ABC, ADEF) este aplicabilă schemei R, subschemele ei sunt superchei pentru R.

Teorema 5.6. Dacă schema R se găseşte în forma normală proiecție-joncțiune în raport cu mulțimea de dependențe J, atunci R se află în forma normală patru.

Demonstrația acestei teoreme urmează direct din condiția că dependența multivaloare netrivială X→Y definită asupra schemei R este dependența de joncțiune |x|(XY, XZ), unde Z = R \ XY.

5.12. Concluzii

Etapele de proiectare a bazei de date pot fi cele de mai jos. Fiecare din aceste etape produce o bază de date mai "bună" decât precedenta. Corelația dintre diverse forme normale este reprezentată în fig.5.11.

(1) Inițial datele sunt nenormalizate.

(2) Se elimină atributele ce formează mulțimi de valori sau sunt complexe. În consecință se obține schema relației universale. Se spune că schema acestei relații se găseşte în forma normală unu (FN1).

(3) Pentru a ajunge la forma normală doi (FN2) se elimină dependențele parțiale de chei ale atributelor nonprimare.

(4) Forma normală trei (FN3) cere eliminarea dependențelor tranzitive ale atributelor nonprimare de chei. De obicei mulți profesioniști în proiectarea bazelor de date se limitează la această formă normală.

(5) După înlăturarea tuturor dependențelor tranzitive, se obține forma normală Boyce-Codd (FNBC).

(6) Forma normală patru (FN4) soluționează problemele cauzate de dependențele multivaloare netriviale.

(7) Forma normală proiecție-joncțiune (FNPJ) se referă la soluționarea problemei descompunerii fără pierderi a relațiilor.

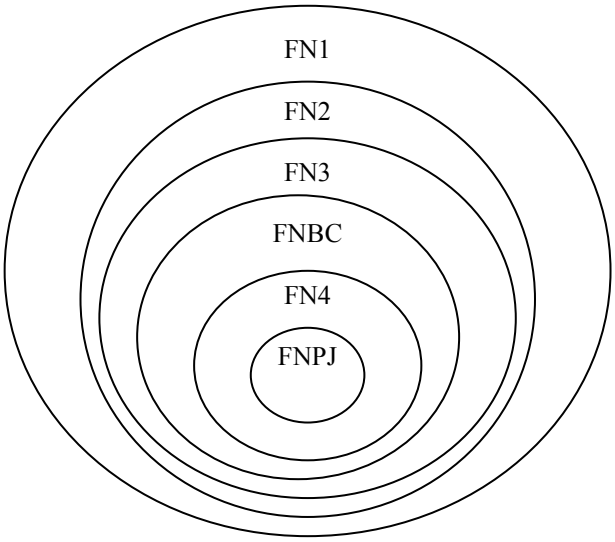

Fig.5.11. Corelația dintre formele normale

În afară de facilitățile pe care ni le oferă o schemă a bazei de date, construită conform rigorilor ştiinţifice, normalizarea creează şi două probleme: micşorarea eficienței căutării datelor şi apariția unor duplicate de date.

Un efect adițional al normalizării este creşterea numărului de structuri de date în baza de date. Aceasta însă afectează eficiența de căutare a datelor în sistemul informatic. Deşi normalizarea reduce spațiul total necesar de păstrare a datelor, însă creşte timpul în care poate fi căutată informația. Pentru procesarea interpelărilor şi extragerii răspunsurilor apare necesitatea rejoncționării relațiilor. Prin aceasta şi se explică faptul că primele baze de date relaționale au apărut pe calculatoare performante, iar mai târziu au apărut sisteme instalate pe microcomputere.

În ceea ce priveşte duplicatele de date trebuie de menționat că ele nu pot fi comparate cu redundanța de date ce este redusă în procesul de normalizare. Redundanța generează anomalii de actualizare a datelor. Pe când duplicatele apărute după normalizare, nu generează asemenea anomalii. Aici e vorba de atributele ce formează determinantul dependenței care participă la descompunere. Atributele determinantului apar în ambele scheme produse din schema precedentă.

5.13. Exerciții

5.1. Fie mulțimea de dependențe funcționale G = {AB→EF, A→C, D→B, C→F, F→B}. Să se determine cheile schemelor de mai jos.

(a) R_1 = ABCDEF;

(b) R_2 = ABDF;

(c) R_3 = ACE;

(d) R_4 = BCD;

(e) R_5 = DEF.

5.2. Fie mulțimea de dependențe funcționale G = {A→D, AB→E, BF→E, CD→F, E→C} definită pe schema R = ABCDEF.

(a) Să se determine închiderile oricărei combinații de atribute din schema R.

(b) Să se identifice atributele primare.

5.3. Să se aducă un exemplu de schemă (alta decât cele descrise în secțiunea curentă), în care se manifestă anomalii de inserare, ştergere şi modificare a datelor.

5.4. Să se aducă schema din exercițiul 5.3 la forma normală necesară, încât anomaliile să fie eliminate.

5.5. Fie pe schema R = ABCDE e definită mulțimea de dependențe funcționale F = {A→C, B→C , C→D, DE→C, CE→A}. Să determine dacă descompunerea R_1=AD, R_2=AB, R_3=BE, R_4=CDE, R_5=AE a schemei R, posedă proprietatea joncțiunii fără pierderi.

5.6. Fie F = {AC→BE, BC→AD, C→DE, A→D, D→B}. Să se determine dacă descompunerea R_1=ABC, R_2=AB, R_3=BDE a schemei R=ABCDE, se bucură de proprietatea joncțiunii fără pierderi.

5.7. Să se aducă schema relațională R = ABCDEF în forma normală doi, dacă pe ea e definită mulțimea de dependențe funcționale G = {AB→CE, BC→A, C→A, ACE→B, E→DF, BD→C, CF→BE, CD→AF, E→F}.

5.8. Să se descompună schema R = ABCDEF în forma normală trei, dacă pe ea e definită mulțimea de dependențe funcționale G = {AB→C, C→A, BC→D, ACD→B, CD→B, BE→C, CF→BD, CE→AF}.

5.9. Să se aducă un exemplu de schemă în forma normală trei cu un atribut primar ce depinde tranzitiv de cheie.

5.10. Fie F = {C→T, HR→C, CS→G, HS→R, HRS→T}. Să se sintetizeze schema bazei de date în forma normală trei.

5.11. Să se construiască schema bazei de date în forma normală trei din mulțimea de dependențe F = {A→CF, B→ED, E→F, F→BC}.

5.12. Să se aducă un exemplu de relație ce se descompune fără pierderi în trei relații, dar nu se descompune în două. Bineînțeles că toate schemele trebuie să fie diferite.

5.13. Fie mulțimea de dependențe funcționale F = {AB→C, A→D, BD→C} valide în relația r(ABCD).

(a) Să se aducă trei exemple de anomalii ce apar în actualizarea relației r.

(b) Să se descompună schema acestei relații în două scheme astfel ca schemele obținute să se găsească în forma normală trei şi descompunerea să conserve dependențele funcționale.

5.14. Fie că relația r(ABCD) satisface mulțimea de dependențe funcționale F = {AC→B, AB→D}.

(a) Să se găsească cheile schemei relației r.

(b) Să se arate că schema R = ABCD se găseşte în forma normală doi şi nu se găseşte în forma normală trei.

(c) Să se aducă exemple de anomalii ce pot apărea în procesul de actualizare a relației r.

5.15. Considerăm schema R = ABCD, mulțimea de dependențe funcționale F = {A→B, C→B, D→ABC, AC→D} definită pe ea şi descompunerea lui R în două scheme R_1 = AB şi R_2= BCD.

(a) Descompunerea dată posedă proprietatea joncțiunii fără pierderi? Dacă nu, atunci să se găsească o descompunere fără pierderi.

(b) Descompunerea dată conservă mulțimea de dependențe F?

5.16. Considerăm schema R = ABCD şi mulțimea de dependențe funcționale F = {AC→B, AB→D}.

(a) Care sunt cheile schemei R?

(b) În ce formă normală cea mai înaltă se găseşte schema R?

(c) Care este cea mai înaltă formă la care poate fi adusă schema R, dar să posede proprietatea joncțiunii fără pierderi şi să conserve dependențele?

5.17. Utilizând algoritmul de normalizare prin descompunere să se aducă schema R = ABCDEF în forma normală Boyce-Codd, dacă pe ea e definită mulțimea de dependențe funcționale G = {A→B, CD→A, CB→D, AE→F}.

5.18. Fie schema R = ABCDEGHIKLMN şi mulțimea de dependențe funcționale F = {AC→ED, A→BGHL, G→HIK, L→MN, N→L}. Să se aducă schema R în forma normală Boyce-Codd prin descompunere, astfel ca

descompunerea să posede proprietatea joncțiunii fără pierderi. Schema normală obținută conservă dependențele?

5.19. De ce o schemă relaţională cu cel mult două atribute se găseşte în forma normală Boyce-Codd?

5.20. Să se arate că, dacă pentru orice pereche distinctă de atribute A şi B din schema R, atributul A nu se conține în închiderea mulțimii R \ AB, atunci R se găseşte în forma normală Boyce-Codd.

5.21. Să se aducă un exemplu de schemă în forma normală Boyce-Codd, dar care nu se găseşte în forma normală patru.