

**Dan Simovici,
Mihaela E. Breabăn,
Henri Luchian**

**TEORIA
ȘI PRACTICA
BAZELOR
DE DATE**

POLIROM
2025

Cuprins

<i>Prefață</i>	9
--------------------------	---

Concepte fundamentale

1	Concepte de bază din domeniul bazelor de date	13
1.1	Baze de date și sisteme de gestiune a bazelor de date	13
1.1.1	Ce este o bază de date?	13
1.1.2	Sisteme de gestiune a bazelor de date	14
1.2	Arhitectura unui sistem de gestiune de baze de date	17
1.3	O perspectivă istorică a SGBD-urilor	19
1.4	Recomandări bibliografice	20
2	Modelul relațional	21
2.1	Introducere	21
2.2	Tabelul – principala structură de date în modelul relațional	21
2.2.1	Proiecții	26
2.3	Metadata	27
2.4	Exerciții	28
2.5	Recomandări bibliografice	29
3	Regăsirea datelor în bazele de date relaționale	31
3.1	Introducere	31
3.2	Operații cu tabele	32
3.2.1	Redenumirea tabelelor și atributelor	32
3.2.2	Operații din teoria mulțimilor	33
3.2.3	Selecția	36
3.2.4	Proiecția	39
3.2.5	Operația „join”	40
3.2.6	Împărțirea	43
3.3	Operațiile de bază ale algebrei relaționale	44
3.4	Alte operații din algebra relațională	48

3.5	Scheme de tabel și scheme de baze de date.	51
3.6	Exerciții	52
3.7	Recomandări bibliografice.	54

Principii de proiectare a bazelor de date relaționale

4	Proiectare conceptuală: modelul Entitate-Asociere	57
4.1	Introducere	57
4.2	Concepte de bază în modelul E/A	57
4.3	Atribute	59
4.4	Chei	62
4.5	Constrângeri de participare.	67
4.6	Entități slabe.	69
4.7	Asocierile is-a	71
4.8	Capcane de conectare.	74
4.9	Modelarea E/A în limbajul UML	75
4.10	Transformarea unei diagrame E/A în schemă relațională.	82
	4.10.1 Interpretarea mulțimilor de entități	83
	4.10.2 Interpretarea mulțimilor de asocieri	85
4.11	Integritatea referențială	90
4.12	Rafinarea schemei relaționale	91
	4.12.1 Rolul constrângerilor de participare.	91
	4.12.2 Asocierile de tip specializare.	92
4.13	Concluzii.	93
4.14	Exerciții	93
4.15	Recomandări bibliografice.	95
5	Proiectare prin normalizare.	97
5.1	Introducere	97
5.2	Dependențe funcționale	98
	5.2.1 Demonstrații și dependențe funcționale.	100
	5.2.2 Reguli de inferență derivate	103
	5.2.3 Închiderea unei mulțimi de atribute	104
	5.2.4 Corectitudine și completitudine	106
	5.2.5 Calculul închiderii	109
5.3	Funcții proiecție-join	111
5.4	Chei și dependențe funcționale	115
5.5	Acoperiri.	117
5.6	Tablouri	122
	5.6.1 Mapări proiecție-join și tablouri	125
	5.6.2 Tablouri și dependențe funcționale	127
5.7	Dependențe multivaluate	134
5.8	Forme normale.	140
	5.8.1 Valori atomice și prima formă normală	141
	5.8.2 Forme normale și dependențe funcționale	141
5.9	Normalizare prin descompunere sau sinteză	147
5.10	Exerciții	153
5.11	Recomandări bibliografice.	157

SQL și extensii

6	SQL – Limbajul relațional	161
6.1	Introducere	161
6.2	SQL în sistemul MySQL	162
6.3	Comenzi MySQL	163
6.4	Sortarea rezultatelor și redenumirea coloanelor	169
6.5	Completitudinea relațională a sistemului MySQL	171
6.6	Funcții încorporate în MySQL	178
6.7	Funcții care acționează asupra datelor calendaristice	179
6.8	Grupuri în SQL	195
6.9	Funcții și proceduri definite de utilizatori	197
6.10	Directive de modificare a datelor	200
6.11	Controlul accesului	202
6.12	Tabele virtuale	203
6.13	Dialectul SQL al sistemului ORACLE	206
6.14	Exerciții	210
6.15	Recomandări bibliografice	211
7	PL/SQL – o extensie procedurală a limbajului SQL	213
7.1	Introducere	213
7.2	Sintaxa limbajului PL/SQL	214
7.3	Tipuri de date în PL/SQL	215
7.4	Structuri de control în PL/SQL	217
7.5	Interogări în PL/SQL	219
7.6	Cursoare în PL/SQL	222
7.7	Proceduri în PL/SQL	225
7.8	Funcții în PL/SQL	227
7.9	Colecții în PL/SQL	230
	7.9.1 Tabele asociative	230
	7.9.2 Tabele imbricate	232
	7.9.3 Tabele variabile	233
7.10	Întregistrări în PL/SQL	236
7.11	PL/SQL ca limbaj orientat pe obiecte	240
	7.11.1 Ereditatea tipurilor	243
	7.11.2 Tipuri abstracte în PL/SQL	245
7.12	Declanșatoare în PL/SQL	246
7.13	Exerciții	248
7.14	Recomandări bibliografice	248

Procesarea eficientă a datelor

8	Indexarea datelor	251
8.1	Introducere	251
8.2	Organizarea fizică a datelor	253
8.3	Fișiere index	256
	8.3.1 Indecși secvențiali	256
	8.3.2 Indecși B-arbore	261

8.3.3	Indecși hash	272
8.3.4	Indecși bitmap	276
8.3.5	Indecși multi-atribut	279
8.4	Comenzi SQL pentru indexare	281
8.5	Considerente practice privind indexarea	282
8.6	Exerciții	284
8.7	Recomandări bibliografice	284
9	Procesarea interogărilor	285
9.1	Introducere	285
9.2	Operații cu tabele în planurile de execuție	286
9.3	Procesarea selecțiilor	290
9.4	Procesarea joinurilor	294
9.5	Procesarea frazelor sub-select	298
9.6	Decizii optime de procesare	299
9.7	Exerciții	302
9.8	Recomandări bibliografice	302
10	Tranzacții și acces concurent în bazele de date	303
10.1	Introducere	303
10.2	Tranzacții	304
10.3	Serializabilitate	310
10.4	Algoritmi de control al tranzacțiilor	318
10.5	Ierarhii de memorie	323
10.6	Recuperare	324
10.7	Puncte de control	328
10.8	Exerciții	332
10.9	Recomandări bibliografice	333

Baze de date NoSQL

11	Baze de date NoSQL	337
11.1	De ce NoSQL?	337
11.2	Clase de sisteme NoSQL	338
11.3	Sisteme de fișiere distribuite	339
11.4	Procesare distribuită Map-Reduce	342
11.5	Depozite de documente în MongoDB	345
	11.5.1 Modelarea datelor	346
	11.5.2 Operații asupra datelor	349
11.6	Exerciții	354
11.7	Recomandări bibliografice	355
	<i>Bibliografie</i>	357
	<i>Index</i>	361

5.5 Acoperiri

Restricționând și standardizând dependențele funcționale, devine mult mai ușoară manipularea și compararea lor.

Definiția 5.5.1 Fie F, G două mulțimi de dependențe funcționale, $F, G \subseteq \text{FD}(H)$.

F și G sunt *echivalente* dacă $F^+ = G^+$. În acest caz, numim F o *acoperire* pentru G și G o acoperire pentru F .¹

Dacă F, G sunt mulțimi echivalente de dependențe funcționale, scriem $F \equiv G$. □

Teorema 5.5.2 Fie F și G două mulțimi de dependențe funcționale, $F, G \subseteq \text{FD}(H)$. Următoarele trei afirmații sunt echivalente:

- (i) $F \subseteq G^+$;
- (ii) $F^+ \subseteq G^+$;
- (iii) $\text{cl}_F(X) \subseteq \text{cl}_G(X)$ pentru fiecare submulțime X a lui H .

Demonstrație. (i) implică (ii). Presupunem că $F \subseteq G^+$. Prima parte a teoremei 5.2.22 implică $F^+ \subseteq (G^+)^+$. A doua parte a acestei teoreme dă $(G^+)^+ = G^+$, de unde $F^+ \subseteq G^+$.

(ii) implică (iii). Presupunem că (ii) este adevărată. Deoarece $X \rightarrow \text{cl}_F(X) \in F^+$ avem $X \rightarrow \text{cl}_F(X) \in G^+$ astfel încât $\text{cl}_F(X) \subseteq \text{cl}_G(X)$ prin proprietatea de maximalitate a lui $\text{cl}_G(X)$.

(iii) implică (i). Dacă (iii) este adevărată și $X \rightarrow Y \in F$, din $Y \subseteq \text{cl}_F(X) \subseteq \text{cl}_G(X)$ atunci $X \rightarrow Y \in G^+$. Prin urmare, (i) este adevărată. ■

Următorul corolar constituie un instrument util pentru demonstrarea echivalenței dependențelor funcționale.

Corolar 5.5.3 Fie F, G două mulțimi de dependențe funcționale, $F, G \subseteq \text{FD}(H)$. Următoarele trei afirmații sunt echivalente:

1. $F \subseteq G^+$ și $G \subseteq F^+$;
2. F, G sunt mulțimi echivalente de dependențe funcționale;
3. $\text{cl}_F(X) = \text{cl}_G(X)$ pentru orice submulțime X a lui H .

Demonstrație. Corolarul este o consecință imediată a Teoremei 5.5.2. ■

Definiția 5.5.4 O *dependență funcțională unitară* este o dependență funcțională al cărui membru drept constă dintr-un singur atribut. □

Dependențele funcționale unitare din $\text{FD}(H)$ sunt, desigur, de forma $X \rightarrow A$, unde X este o submulțime a lui H și A este un membru al lui H .

Teorema 5.5.5 Pentru orice mulțime F de dependențe funcționale $F \subseteq \text{FD}(H)$, există o mulțime echivalentă $G \subseteq \text{FD}(H)$ astfel încât toate dependențele lui G sunt dependențe funcționale unitare.

¹Utilizarea termenului *acoperire* implică în general asimetrie. E de notat însă că în terminologia bazelor de date acest lucru nu mai este valabil.

Demonstrație. Definim G ca

$$G = \{X \rightarrow A \mid X \rightarrow Y \in F \text{ și } A \in Y\}.$$

Regula de proiectivitate implică $X \rightarrow A \in F^+$ pentru fiecare $X \rightarrow A \in G$. Pe de altă parte, dacă $X \rightarrow Y \in F$ și $Y = A_1 \dots A_m$, atunci $X \rightarrow A_1, \dots, X \rightarrow A_m \in G$ și regula de aditivitate implică $X \rightarrow Y \in G^+$. Prin urmare, Corolarul 5.5.3 implică echivalența lui F și G . ■

Definiția 5.5.6 O mulțime F de dependențe funcționale este *neredundantă* dacă nu există nicio submulțime proprie G a lui F astfel încât $G \equiv F$. Altfel, F este o mulțime de dependențe funcționale *redundantă*. □

Evident, o mulțime F este neredundantă dacă pentru orice $X \rightarrow Y \in F$, $(F - \{X \rightarrow Y\})^+ \subset F^+$. De asemenea, orice submulțime a unei mulțimi de dependențe funcționale neredundante este la rândul ei neredundantă. Dată o mulțime F de dependențe funcționale, este posibil să existe mai mult de o acoperire neredundantă a lui F . De exemplu, mulțimea de dependențe funcționale unitare:

$$F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B, A \rightarrow C, C \rightarrow A\}$$

este redundanță. Totuși, $F_1 = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}$, $F_2 = \{B \rightarrow C, C \rightarrow B, A \rightarrow C, C \rightarrow A\}$ și $F_3 = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A\}$ sunt fiecare în parte neredundante și echivalente cu F .

Algoritm 5.5.7 (Calculul unei acoperiri neredundante)

Intrare: O mulțime finită H de atribute și o mulțime F de dependențe funcționale, $F \subseteq \text{FD}(H)$.

Ieșire: O acoperire neredundantă F' a lui F .

Metoda: Fie ϕ_1, \dots, ϕ_n o secvență care constă din toate dependențele funcționale ale lui F , fără repetări.

Construim o secvență de mulțimi de dependențe funcționale F_0, F_1, \dots, F_n unde $F_0 = F$ și

$$F_{i+1} = \begin{cases} F_i - \{\phi_{i+1}\} & \text{dacă } F_i - \{\phi_{i+1}\} \equiv F_i \\ F_i & \text{altfel} \end{cases}$$

pentru $0 \leq i < n$. Returnează mulțimea $F' = F_n$.

Demonstrația Corectitudinii: Se poate arăta imediat că mulțimea F_n este neredundantă și echivalentă cu F . ■

Mulțimea neredundantă de dependențe funcționale obținută în Algoritmul 5.5.7 depinde de ordinea în care considerăm dependențele funcționale. Acest lucru nu este surprinzător ținând seama de remarca ce precede algoritmul.

Observăm că, atunci când F este o mulțime neredundantă de dependențe funcționale, mulțimea G de dependențe funcționale unitare construită în Teorema 5.5.5 poate fi redundanță. De exemplu, pornind de la mulțime neredundantă $F = \{A \rightarrow BC, C \rightarrow B\}$ mulțimea construită $G = \{A \rightarrow B, A \rightarrow C, C \rightarrow B\}$ este redundanță, deoarece $G \equiv \{A \rightarrow C, C \rightarrow B\}$.

Din motive enunțate în Secțiunea 5.8, este de dorit să lucrăm cu scheme de tabele ce conțin dependențe funcționale cu proprietatea că cea mai mică mulțime de atribute determină cel mai mare număr posibil dintre atributele rămase. Printre alte beneficii, acest lucru permite reducererea resurselor necesare stocării. Astfel, căutăm să minimizăm dimensiunea lui X în orice dependență funcțională $X \rightarrow Y$. Următoarea definiție formalizează această cerință.

Definiția 5.5.8 Fie F o mulțime de dependențe funcționale și fie $X \rightarrow Y$ o dependență funcțională din F . $X \rightarrow Y$ este *F-redușă* dacă nu există nicio submulțime proprie X' a lui X astfel încât $(F - \{X \rightarrow Y\}) \cup \{X' \rightarrow Y\} \equiv F$. Mulțimea F este *redușă* dacă aceasta constă doar din dependențe funcționale F -redușe. \square

Lema 5.5.9 Fie F o mulțime de dependențe funcționale, $F \subseteq \text{FD}(H)$ și fie $X \rightarrow Y \in F$. Dacă $X' \subset X$, atunci $F^+ \subseteq ((F - \{X \rightarrow Y\}) \cup \{X' \rightarrow Y\})^+$.

Demonstrație. Fie $F' = (F - \{X \rightarrow Y\}) \cup \{X' \rightarrow Y\}$.

Observăm că definiția lui F' implică proprietatea că pentru fiecare mulțime de atribute W avem

$$\bigcup \{V \mid U \rightarrow V \in F, U \subseteq W\} \subseteq \bigcup \{V' \mid U' \rightarrow V' \in F', U' \subseteq W\}. \quad (5.3)$$

Pentru a arăta că $F^+ \subseteq (F')^+$ este suficient să arătăm că $\text{cl}_F(U) \subseteq \text{cl}_{F'}(U)$ pentru fiecare mulțime $U \subseteq H$. Fie $\text{CS}_F(U) = (U_0, U_1, \dots, U_n)$ și fie $\text{CS}_{F'}(U) = (U'_0, U'_1, \dots, U'_m)$. Pentru a arăta că $\text{CS}_F(U) \sqsubseteq \text{CS}_{F'}(U)$, considerăm o mulțime U_i din $\text{CS}_F(U)$. Arătăm prin inducție după i , că $U_i \subseteq U'_m$. Pentru $i = 0$ această afirmație este imediată, deoarece $U_0 = U'_0 \subseteq U'_m$. Prin urmare, presupunem că $U_i \subseteq U'_m$.

Avem

$$\begin{aligned} U_{i+1} &= U_i \cup \bigcup \{V \mid U \rightarrow V \in F \text{ și } U \subseteq U_i\} \\ &\subseteq U'_m \cup \bigcup \{V' \mid U' \rightarrow V' \in F' \text{ și } U' \subseteq U_i\} \\ &= U'_{m+1} = U'_m, \end{aligned}$$

prin prisma incluziunii 5.3.

Deoarece $\text{CS}_F(U) \sqsubseteq \text{CS}_{F'}(U)$, înseamnă că $\text{cl}_F(U) \subseteq \text{cl}_{F'}(U)$. \blacksquare

Teorema 5.5.10 Pentru fiecare mulțime F de dependențe funcționale există o mulțime F' de dependențe funcționale echivalentă și redușă.

Demonstrație. Justificarea este constructivă. Pentru fiecare dependență funcțională $X \rightarrow Y$ a lui F și fiecare atribut $A \in X$, determinăm dacă $Y \subseteq \text{cl}_F(X - A)$; dacă suntem în acest caz, înlocuim $X \rightarrow Y$ în F cu $(X - A) \rightarrow Y$. Afirmăm că F este echivalent cu $F - \{X \rightarrow Y\} \cup \{(X - A) \rightarrow Y\}$. Observăm că $F \subseteq (F - \{X \rightarrow Y\}) \cup \{(X - A) \rightarrow Y\}^+$. Pe de altă parte, $F - \{X \rightarrow Y\} \cup \{(X - A) \rightarrow Y\} \subseteq F^+$ pentru că $Y \subseteq \text{cl}_F(X - A)$, astfel că F și $F - \{X \rightarrow Y\} \cup \{(X - A) \rightarrow Y\}$ sunt echivalente conform Corolarului 5.5.3.

Deoarece F este finită, procedura poate fi aplicată de un număr finit de ori. La final, mulțimea de dependențe funcționale obținută constă din dependențe funcționale F -reduse. ■

Exemplul 5.5.11 Fie $H = ABC$ și fie $F = \{AB \rightarrow C, A \rightarrow B\} \subseteq \text{FD}(H)$. Este ușor de verificat următoarele egalități:

$$\text{cl}_F(A) = ABC, \text{cl}_F(B) = B, \text{cl}_F(C) = C,$$

$$\text{cl}_F(AB) = \text{cl}_F(AC) = ABC, \text{cl}_F(BC) = BC.$$

Dacă renunțăm la A din $AB \rightarrow C$ observăm că nu putem obține prin inferență $B \rightarrow C$ din F fiindcă $\text{cl}_F(B) = B$. Pe de altă parte, dacă renunțăm la B din $AB \rightarrow C$, observăm că putem obține în continuare $A \rightarrow C$ din F , deoarece $\text{cl}_F(A) = ABC$. Prin urmare, $\{A \rightarrow C, A \rightarrow B\}$ este o mulțime de dependențe funcționale echivalentă, redusă. □

Lema 5.5.12 Fie F o mulțime de dependențe funcționale redusă, $F \subseteq \text{FD}(H)$ și F' o mulțime neredundantă obținută din F aplicând Algoritmul 5.5.7. Atunci F' este o mulțime redusă de dependențe funcționale.

Demonstrație. Justificarea este imediată și e lăsată cititorului. ■

Definiția 5.5.13 Fie F o mulțime de dependențe funcționale, $F \subseteq \text{FD}(H)$. O formă canonică a lui F este o mulțime G neredundantă și redusă de dependențe funcționale unitare care este echivalentă cu F . □

Teorema 5.5.14 Pentru orice mulțime finită F de dependențe funcționale, există o formă canonică a lui F .

Demonstrație. Pornind de la F , construim o mulțime echivalentă F_1 de dependențe funcționale de forma $X \rightarrow A$ ca în Teorema 5.5.5. Apoi, din F_1 construim o mulțime echivalentă F_2 care este redusă și constă din dependențe funcționale unitare. În final, din F_2 construim o mulțime echivalentă neredundantă F_3 aplicând Algoritmul 5.5.7.

Din Lema 5.5.12 urmează că F_3 este redusă. ■

Exemplul 5.5.15 Fie $H = ABCDE$ o mulțime de atribute și fie F o mulțime de dependențe funcționale dată de

$$F = \{A \rightarrow BCD, AB \rightarrow DE, BE \rightarrow AC\}.$$

Mulțimea F_1 este $F_1 = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, AB \rightarrow D, A \rightarrow E, BE \rightarrow A, BE \rightarrow C\}$. Pentru a construi mulțimea redusă F_2 trebuie să examinăm dependențele funcționale din F_1 care au mai mult de un atribut în membrii lor stângi: $AB \rightarrow D, AB \rightarrow E, BE \rightarrow A, BE \rightarrow C$. Observăm că $\text{cl}_{F_1}(A) = ABCDE$. Prin urmare, putem elimina B din membrul stâng al $AB \rightarrow D$. Dependența funcțională rezultată este deja în F_1 . Deoarece $\text{cl}_{F_1}(B) = B$, observăm că A nu poate fi eliminată din $AB \rightarrow D$. Începând de la $AB \rightarrow E$ obținem $A \rightarrow E$. Deoarece $\text{cl}_{F_1}(E) = E$ nicio altă dependență funcțională nu poate fi obținută. Astfel, $F_2 = \{A \rightarrow B, A \rightarrow C, A \rightarrow D,$

$A \rightarrow E, AB \rightarrow D, AB \rightarrow E, BE \rightarrow A, BE \rightarrow C$. Aplicând Algoritmul 5.5.7, obținem mulțimea de dependențe funcționale unitare

$$F_3 = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, BE \rightarrow A\}$$

care reprezintă forma canonică a lui F . \square

Următoarea teoremă joacă un rol esențial în sintetizarea schemelor bazei de date care satisfac anumite forme normale. O vom folosi în Secțiunea 5.9.

Teorema 5.5.16 *Fie $S = (H, F)$ o schemă cu dependențe funcționale și fie K o cheie pentru S . Dacă $G = \{X_i \rightarrow A_i \mid 1 \leq i \leq n\}$ este o formă canonică pentru F , atunci*

1. *Nicio mulțime $X_i A_i$ nu este inclusă în K ;*
2. *$K \cup \bigcup \{A_i \mid 1 \leq i \leq n\} = H$;*
3. *$\mathbf{H} = (K, X_1 A_1, \dots, X_n A_n)$ este o descompunere fără pierderi a fiecărui tabel din S .*

Demonstrație. Pentru a demonstra prima parte a teoremei, observăm că dacă $X_i A_i$ ar fi o submulțime a lui K , atunci $K - A_i$ ar fi de asemenea cheie, contrazicând astfel proprietatea de minimalitate a lui K .

Pentru a doua parte a teoremei, observăm că $\text{cl}_G(K) = \text{cl}_F(K) = H$, deoarece F, G sunt mulțimi echivalente de dependențe funcționale și K este o cheie pentru F . Fie $\text{CS}_G(K) = (K_0, \dots, K_\ell, \dots, K_m)$ o G -secvență de închidere a lui K , unde $K_m = H$.

Pentru fiecare $A \in H$, definim numărul p_A prin $p_A = \min\{\ell \mid 0 \leq \ell \leq m \text{ și } A \in K_\ell\}$. Observăm că p_A există, deoarece $\text{cl}_G(K) = H$. Dacă $p_A = 0$, atunci $A \in K$. Altfel, $A \in K_{p_A} - K_{p_A-1}$ ceea ce înseamnă că există o dependență funcțională $X_i \rightarrow A_i \in G$ astfel încât $X_i \subseteq K_{p_A-1}$ și $A_i = A \in K_{p_A}$. Astfel, în oricare caz, avem $A \in K \cup \bigcup \{A_i \mid 1 \leq i \leq n\}$.

Pentru a demonstra ultima parte a teoremei, considerăm un tabel $\tau = (T, H, \rho)$ al schemei S . Fie t, t_1, \dots, t_n $n+1$ tuple care pot participa în join astfel încât $t_i \in \rho[X_i A_i]$ pentru $1 \leq i \leq n$ și $t \in \rho[K]$. Atunci, ρ conține tuplele s, s_1, \dots, s_n astfel încât $s_i[X_i A_i] = t_i$ pentru $0 \leq i \leq n$ și $s[K] = t$. Presupunem că attributele A_1, \dots, A_n sunt listate astfel încât $p_{A_i} \leq p_{A_j}$ implică $i \leq j$. Fie $L_0 = K$ și $L_i = K A_1 \dots A_i$ pentru $1 \leq i \leq n$, unde $L_n = H$. Avem $X_i \subseteq L_{i-1}$ pentru $1 \leq i \leq n$.

Demonstrăm prin inducție după i , $1 \leq i \leq n$, că $(t \bowtie t_1 \bowtie \dots \bowtie t_i)[L_i] = s[L_i]$.

Pentru $i = 1$, aplicabilitatea joinului asupra lui t și t_1 implică $t[X_1] = t_1[X_1]$, astfel $s[X_1] = s_1[X_1]$, ceea ce dă $s[A_1] = s_1[A_1]$. Astfel, $(t \bowtie t_1)[L_1] = s[L_1]$.

Presupunem că $(t \bowtie t_1 \bowtie \dots \bowtie t_i)[L_i] = s[L_i]$. Afirmăm că

$$(t \bowtie t_1 \bowtie \dots \bowtie t_i \bowtie t_{i+1})[L_{i+1}] = s[L_{i+1}].$$

Observăm că $X_{i+1} \subseteq L_i$.

Tuplul t_{i+1} poate intra în join cu $(t \bowtie t_1 \bowtie \dots \bowtie t_i)$; această proprietate implică $(t \bowtie t_1 \bowtie \dots \bowtie t_i)[X_{i+1}] = t_{i+1}[X_{i+1}]$, astfel $s_{i+1}[A_{i+1}] = s[A_{i+1}]$. Obținem astfel concluzia dorită.

Pentru $i = n$ obținem $t \bowtie t_1 \bowtie \dots \bowtie t_n = s$, ceea ce demonstrează că \mathbf{H} este o descompunere fără pierdere. ■

Exemplul 5.5.17 Considerăm schema $\mathbf{S} = (A_1 \dots A_6, F)$, unde $A_1 A_2$ este o cheie pentru F . Fie G o formă canonică a lui F :

$$G = \{A_1 \longrightarrow A_3, A_2 \longrightarrow A_4, A_1 A_4 \longrightarrow A_5, A_2 A_3 \longrightarrow A_6\}.$$

Pentru orice tabel al schemei \mathbf{S} avem descompunerea fără pierdere:

$$\mathbf{H} = (A_1 A_2, A_1 A_3, A_2 A_4, A_1 A_4 A_5, A_2 A_3 A_6).$$

□

Exemplul 5.5.18 Fie $\mathbf{S} = (H, F)$ schema de tabel introdusă în Exemplul 5.4.7. Fie $K = \text{matr codcurs sem an}$. Mulțimea F care constă din dependențele funcționale

$$\begin{aligned} \text{codcurs sem an} &\longrightarrow \text{codinst} \\ \text{matr codcurs sem an} &\longrightarrow \text{nota} \end{aligned}$$

este deja în formă canonică. Prin urmare, orice tabel τ al lui \mathbf{S} are descompunerea fără pierdere $\mathbf{H} = (H_1, H_2, H_3)$, unde

$$\begin{aligned} H_1 &= \text{matr codcurs sem an} \\ H_2 &= \text{codcurs sem an codinst} \\ H_3 &= \text{matr codcurs sem an nota} \end{aligned}$$

În continuare, deoarece $H_1 \subseteq H_3$, putem renunța la H_1 din această descompunere. Astfel, $\mathbf{H}' = (H_2, H_3)$ este de asemenea o descompunere fără pierdere a oricărui tabel τ al lui \mathbf{S} . □

Rezultatele prezentate aici sunt independente de orice tabel anume al unei scheme. De-a lungul timpului, tabelele se schimbă, dar proprietățile schemei rămân constante.

5.6 Tablouri

Noțiunea de tablou pe care o introducem în această secțiune ne permite să studiem proprietățile dependențelor funcționale și multivaluate într-o manieră mult mai eficientă.

Fie \mathcal{U} o mulțime de atribute ale unei relații. Pentru fiecare atribut $A \in \mathcal{U}$, considerăm un simbol \mathbf{d}^A numit *simbolul distinctiv al atributului A* și o mulțime $V^A = \{\mathbf{n}_0^A, \mathbf{n}_1^A, \dots\}$ de *simboluri nedistinctive*. Ne vom referi la mulțimea $\hat{D}_A = \{\mathbf{d}^A\} \cup V^A$ ca *pseudodomeniul atributului A* . Presupunem că dacă $A \neq A'$, atunci $\hat{D}_A \cap \hat{D}_{A'} = \emptyset$.

Mulțimea \hat{D}_A este echipată cu o relație de ordine a cărei diagramă este dată în Figura 5.1: $\mathbf{d}^A < \mathbf{n}_0^A < \mathbf{n}_1^A < \dots$.

Noțiunea de tablou este foarte similară noțiunii de tabel. Diferența majoră dintre tabele și tablouri este aceea că valorile care apar în tablouri aparțin pseudodomeniilor atributelor și nu domeniilor lor.