Dan Simovici,
Mihaela E. Breabăn,
Henri Luchian

TEORIA ŞI PRACTICA BAZELOR DE DATE

POLIROM
2025

Cuprins

Principii de proiectare a bazelor de date relaționale

SQL și extensii

Procesarea eficientă a datelor

Baze de date NoSQL

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, $F, G \subseteq \mathsf{FD}(H)$.

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, $F, G \subseteq \mathsf{FD}(H)$. Următoarele trei afirmații sunt echivalente:*

(i) $F \subseteq G^+$;
(ii) $F^+ \subseteq G^+$;
(iii) $\mathtt{cl}_F(X) \subseteq \mathtt{cl}_G(X)$ *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 $X \longrightarrow \mathtt{cl}_F(X) \in F^+$ avem $X \longrightarrow \mathtt{cl}_F(X) \in G^+$ astfel încât $\mathtt{cl}_F(X) \subseteq \mathtt{cl}_G(X)$ prin proprietatea de maximalitate a lui $\mathtt{cl}_G(X)$.
(iii) implică *(i)*. Dacă (iii) este adevărată și $X \longrightarrow Y \in F$, din $Y \subseteq \mathtt{cl}_F(X) \subseteq \mathtt{cl}_G(X)$ atunci $X \longrightarrow Y \in G^+$. 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, $F, G \subseteq \mathsf{FD}(H)$. Următoarele trei afirmații sunt echivalente:*

1. $F \subseteq G^+$ *și* $G \subseteq F^+$;
2. *F, G sunt mulțimi echivalente de dependențe funcționale;*
3. $\mathtt{cl}_F(X) = \mathtt{cl}_G(X)$ *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 $\mathsf{FD}(H)$ sunt, desigur, de forma $X \longrightarrow A$, 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 $F \subseteq \mathsf{FD}(H)$, există o mulțime echivalentă $G \subseteq \mathsf{FD}(H)$ astfel încât toate dependențele lui G sunt dependențe funcționale unitare.*

---

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

Demonstraţie. Definim G ca

$$G = \{X \longrightarrow A | X \longrightarrow Y \in F \text{ și } A \in Y\}.$$

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

Definiţia 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 $X \longrightarrow Y \in F$, $(F - \{X \longrightarrow Y\})^+ \subset F^+$. 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:

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

este redundantă. Totuși, $F_1 = \{A \longrightarrow B, B \longrightarrow A, B \longrightarrow C, C \longrightarrow B\}$, $F_2 = \{B \longrightarrow C, C \longrightarrow B, A \longrightarrow C, C \longrightarrow A\}$ și $F_3 = \{A \longrightarrow B, B \longrightarrow A, A \longrightarrow C, C \longrightarrow A\}$ 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, $F \subseteq \mathsf{FD}(H)$.

Ieșire: O acoperire neredundantă F' a lui F.

Metoda: Fie $\phi_1, \ldots, \phi_n$ o secvență care constă din toate dependențele funcționale ale lui F, fără repetări.

Construim o secvență de mulțimi de dependențe funcționale $F_0, F_1, \ldots, F_n$ 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 $F' = F_n$.

Demonstraţia 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 redundantă. De exemplu, pornind de la mulțime neredundantă $F = \{A \longrightarrow BC, C \longrightarrow B\}$ mulțimea construită $G = \{A \longrightarrow B, A \longrightarrow C, C \longrightarrow B\}$ este redundantă, deoarece $G \equiv \{A \longrightarrow C, C \longrightarrow B\}$.

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 reducerea resurselor necesare stocării. Astfel, căutăm să minimizăm dimensiunea lui X în orice dependență funcțională $X \longrightarrow Y$. Următoarea definiție formalizează această cerință.

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

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

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

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

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

Pentru a arăta că $F^+ \subseteq (F')^+$ este suficient să arătăm că $\mathtt{cl}_F(U) \subseteq \mathtt{cl}_{F'}(U)$ pentru fiecare mulțime $U \subseteq H$. Fie $\mathrm{CS}_F(U) = (U_0, U_1, \ldots, U_n)$ și fie $\mathrm{CS}_{F'}(U) = (U'_0, U'_1, \ldots, U'_m)$. Pentru a arăta că $\mathrm{CS}_F(U) \sqsubseteq \mathrm{CS}_{F'}(U)$, considerăm o mulțime U_i din $\mathrm{CS}_F(U)$. 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

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

prin prisma incluziunii 5.3.

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

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

Demonstrație. Justificarea este constructivă. Pentru fiecare dependență funcțională $X \longrightarrow Y$ a lui F și fiecare atribut $A \in X$, determinăm dacă $Y \subseteq \mathtt{cl}_F(X - A)$; dacă suntem în acest caz, înlocuim $X \longrightarrow Y$ în F cu $(X - A) \longrightarrow Y$. Afirmăm că F este echivalent cu $F - \{X \longrightarrow Y\} \cup \{(X - A) \longrightarrow Y\}$. Observăm că $F \subseteq (F - \{X \longrightarrow Y\} \cup \{(X - A) \longrightarrow Y\})^+$. Pe de altă parte, $F - \{X \longrightarrow Y\} \cup \{(X - A) \longrightarrow Y\} \subseteq F^+$ pentru că $Y \subseteq \mathtt{cl}_F(X - A)$, astfel că F și $F - \{X \longrightarrow Y\} \cup \{(X - A) \longrightarrow Y\}$ 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 $F = \{AB \longrightarrow C, A \longrightarrow B\} \subseteq \mathsf{FD}(H)$. Este ușor de verificat următoarele egalități:

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

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

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

Lema 5.5.12 *Fie F o mulțime de dependențe funcționale redusă, $F \subseteq \mathsf{FD}(H)$ 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, $F \subseteq \mathsf{FD}(H)$. 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.*

Demonstraţie. Pornind de la F, construim o mulțime echivalentă F_1 de dependențe funcționale de forma $X \longrightarrow A$ 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

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

Mulțimea F_1 este $F_1 = \{A \longrightarrow B, A \longrightarrow C, A \longrightarrow D, AB \longrightarrow D, A \longrightarrow E, BE \longrightarrow A, BE \longrightarrow C\}$. 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: $AB \longrightarrow D, AB \longrightarrow E, BE \longrightarrow A, BE \longrightarrow C$. Observăm că $\mathtt{cl}_{F_1}(A) = ABCDE$. Prin urmare, putem elimina B din membrul stâng al $AB \longrightarrow D$. Dependența funcțională rezultată este deja în F_1. Deoarece $\mathtt{cl}_{F_1}(B) = B$, observăm că A nu poate fi eliminată din $AB \longrightarrow D$. Începând de la $AB \longrightarrow E$ obținem $A \longrightarrow E$. Deoarece $\mathtt{cl}_{F_1}(E) = E$ nicio altă dependență funcțională nu poate fi obținută. Astfel, $F_2 = \{A \longrightarrow B, A \longrightarrow C, A \longrightarrow D,$

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

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

care reprezintă forma canonică a lui F. □

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

1. *Nicio mulțime X_iA_i nu este inclusă în K;*
2. *$K \cup \bigcup\{A_i \mid 1 \leq i \leq n\} = H$;*
3. *$\mathbf{H} = (K, X_1A_1, \ldots, X_nA_n)$ este o descompunere fără pierderi a fiecărui tabel din $\mathcal{S}$.*

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

Pentru fiecare $A \in H$, definim numărul p_A prin $p_A = \min\{\ell \mid 0 \leq \ell \leq m$ și $A \in K_\ell\}$. Observăm că p_A există, deoarece $\mathtt{cl}_G(K) = H$. 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 $X_i \longrightarrow A_i \in G$ 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 $\mathtt{S}$. Fie $t, t_1, \ldots, t_n$ $n + 1$ tuple care pot participa în join astfel încât $t_i \in \rho[X_iA_i]$ pentru $1 \leq i \leq n$ și $t \in \rho[K]$. Atunci, ρ conține tuplele $s, s_1, \ldots, s_n$ astfel încât $s_i[X_iA_i] = t_i$ pentru $0 \leq i \leq n$ și $s[K] = t$. Presupunem că atributele $A_1, \ldots, A_n$ sunt listate astfel încât $p_{A_i} \leq p_{A_j}$ implică $i \leq j$. Fie $L_0 = K$ și $L_i = KA_1 \ldots A_i$ 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 $(t \bowtie t_1 \bowtie \cdots \bowtie t_i)[L_i] = s[L_i]$.

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

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

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

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

Exemplul 5.5.17 Considerăm schema $\mathtt{S} = (A_1 \ldots A_6, F)$, unde A_1A_2 este o cheie pentru F. Fie G o formă canonică a lui F:

$$G = \{A_1 \longrightarrow A_3, A_2 \longrightarrow A_4, A_1A_4 \longrightarrow A_5, A_2A_3 \longrightarrow A_6\}.$$

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

$$\mathbf{H} = (A_1A_2, A_1A_3, A_2A_4, A_1A_4A_5, A_2A_3A_6).$$

□

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

$$\textit{codcurs sem an} \longrightarrow \textit{codinst}$$
$$\textit{matr codcurs sem an} \longrightarrow \textit{nota}$$

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

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

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 $\mathtt{S}$. □

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 $\mathtt{d}^A$ numit *simbolul distinctiv al atributului* A și o mulțime $V^A = \{\mathtt{n}_0^A, \mathtt{n}_1^A, \ldots\}$ de *simboluri nedistinctive*. Ne vom referi la mulțimea $\widehat{D}_A = \{\mathtt{d}^A\} \cup V^A$ ca *pseudodomeniul atributului* A. Presupunem că dacă $A \neq A'$, atunci $\widehat{D}_A \cap \widehat{D}_{A'} = \emptyset$.

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

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.