

Baze de date relaționale

Normalizarea bazelor de date

Nicolae-Cosmin Vârlan

November 1, 2018

Modelul relațional - chei (recapitulare de la primul curs)

- ▶ **Supercheie** - un atribut sau o mulțime de attribute care identifică unic un tuplu într-o relație
- ▶ **Cheie candidat** - o supercheie cu proprietatea că nici o submulțime proprie a sa nu este supercheie
- ▶ **Cheie primară** - o cheie candidat selectată pentru a identifica în mod unic tuplele într-o relație
- ▶ **Cheie alternativă** - Chei candidat care nu au fost selectate pentru a juca rolul de cheie primară
- ▶ **Cheie străină** - un atribut sau o submulțime de attribute dintr-o relație care face referință la o cheie candidat a altei relații

Găsirea cheilor candidat utilizând dependențele funcționale

Intuitiv: Cheia candidat, este de fapt formată dintr-o combinație de attribute care pot determina unic linia. Dacă pot determina unic linia (deci oricare dintre valorile celorlalte attribute), atunci putem considera că avem o dependența funcțională de la $X \subseteq U$ către toate attributele din U atunci X este cheie în orice relație r construită peste $R[U]$.

Formal: Fie $R[U]$ o schemă de relație și Σ o mulțime de dependențe funcționale satisfăcute de $R[U]$. $X \subseteq U$ este cheie candidat d.dacă $X^+ = U$ și $\forall X' \subseteq X, X'^+ \neq U$

Un atribut se numește **prim** dacă face parte dintr-o cheie candidat.

Un atribut este **neprim** dacă nu este parte din nicio cheie candidat.

...reminder (din cursul 2)

Fie $X \subseteq U$ și \mathcal{R}_A regulile de inferență ale lui Armstrong. Notăm cu

$$X_{\mathcal{R}_A}^+ = \{A \mid \Sigma \vdash_{\mathcal{R}_A} X \rightarrow A\}$$

Regulile de inferență ale lui Armstrong:

$$A1: \frac{}{A_1 \dots A_n \rightarrow A_i}, i = \overline{1, n}$$

$$A2: \frac{A_1, \dots, A_m \rightarrow B_1, \dots, B_r}{A_1 \dots A_m \rightarrow B_j}, j = \overline{1, r}$$

$$\frac{A_1, \dots, A_m \rightarrow B_j, j = \overline{1, r}}{A_1 \dots A_m \rightarrow B_1, \dots, B_r}$$

$$A3: \frac{A_1, \dots, A_m \rightarrow B_1, \dots, B_r, B_1, \dots, B_r \rightarrow C_1, \dots, C_p}{A_1 \dots A_m \rightarrow C_1, \dots, C_p}$$

Revenim - Exemplul 1:

Să considerăm $U = \{A, B, C\}$ și $\Sigma = \{A \rightarrow B, B \rightarrow C\}$. Vom construi mulțimea $X_{\mathcal{R}_A}^+$ pentru fiecare dintre atribute:

$$A_{\mathcal{R}_A}^+ = \{A, B, C\} \quad (A \text{ din } A_1, B \text{ din } A \rightarrow B, \\ C \text{ din } A \rightarrow B, B \rightarrow C \text{ și folosind } A_3)$$

$$B_{\mathcal{R}_A}^+ = \{B, C\} \quad (B \text{ din } A_1, C \text{ din } B \rightarrow C)$$

$$C_{\mathcal{R}_A}^+ = \{C\} \quad (C \text{ din } A_1)$$

Se observă că A este cheie candidat pentru că de el depind (funcțional) celelalte atribute.

Atribute prime: $\{A\}$

Atribute neprime: $\{B, C\}$

Să considerăm $U = \{A, B, C\}$ și $\Sigma = \{A \rightarrow B, B \rightarrow C\}$. Vom construi mulțimea $X_{\mathcal{R}_A}^+$ pentru fiecare dintre atribute:

Putem organiza atributele ținând cont de locul unde apar ele în cadrul dependențelor din Σ :

- ▶ Stânga: Apar numai în partea stângă a dependențelor din Σ .
- ▶ Mijloc: Apar și în stânga și în dreapta dependențelor din Σ .
- ▶ Dreapta: Apar numai în partea dreaptă în dependențele din Σ .

Stânga	Mijloc	Dreapta
A	B	C

Regulă: întotdeauna atributele din Stânga sunt attribute prime, cele din Dreapta sunt neprime. Cele din Mijloc pot fi în oricare dintre categorii.

Exemplul 2:

Fie $U = \{A, B, C, D, E, F\}$ și

$\Sigma = \{A \rightarrow BD, B \rightarrow C, DE \rightarrow F\}$. Care sunt cheile candidat ?

Stânga	Mijloc	Dreapta
A, E	B, D	C, F

$A_{\mathcal{R}_A}^+ = \{A, B, C, D\}$ - nu e cheie (nu conține F), dar cu siguranță apare în orice cheie.

De fapt, am stabilit că fiecare cheie candidat va conține toate atributele din "Stânga" - în cazul nostru pe A și pe E :

$AE_{\mathcal{R}_A}^+ = \{A, B, C, D, E, F\}$

AE = **cheie multivaluată** (este compusă din mai multe atribute).

Exemplul 3:

Fie $U = \{A, B\}$ și $\Sigma = \{A \rightarrow B, B \rightarrow A\}$. Care sunt cheile candidat ?

Stânga	Mijloc	Dreapta
	A, B	

$$A_{\mathcal{R}_A}^+ = \{A, B\}$$

$$B_{\mathcal{R}_A}^+ = \{A, B\}$$

Ambele sunt chei candidat.

Atribute prime: $\{A, B\}$

Atribute neprime: \emptyset

Dependențe pline

Fie $R[U]$ o schema de relație peste o mulțime de atribute U și Σ o mulțime de dependențe funcționale ce au loc în $R[U]$. O dependența $X \rightarrow A \in \Sigma^+$ se numește **plină** dacă $\nexists X' \subset X$ astfel încât $X' \rightarrow A \in \Sigma^+$.

Exemplu:

$R[A, B, C, D]$

$\Sigma = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$

Toate dependențele din Σ sunt pline (e.g. nu avem în Σ^+ nici una dintre dependențele: $A \rightarrow C, B \rightarrow C, B \rightarrow A, C \rightarrow A$).

$AB \rightarrow D$ nu este plină pentru că $B \subset AB$ și avem $B \rightarrow D \in \Sigma^+$

Atribut tranzitiv dependent

Fie $R[U]$ o schemă de relație peste o mulțime de atribute U și Σ o mulțime de dependențe funcționale ce au loc în $R[U]$. Un atribut $A \in U$ se numește **tranzitiv dependent de X** ($X \subset U, A \notin X$) dacă $\exists Y \subset U$ astfel încât:

- ▶ $A \in U - Y$
- ▶ $X \rightarrow Y \in \Sigma^+$
- ▶ $Y \rightarrow A \in \Sigma^+$
- ▶ $Y \rightarrow X \notin \Sigma^+$

Exemplu:

$R[A, B, C, D, E]$

$\Sigma = \{AB \rightarrow C, AB \rightarrow D, CD \rightarrow E\}$

E este tranzitiv dependent de AB ($X = AB, Y = CD$).

Dependențe triviale

O dependență funcțională $X \rightarrow Y$ este trivială dacă și numai dacă $Y \subseteq X$

O dependență multivaluată $X \twoheadrightarrow Y$ este trivială dacă $Y \subseteq X$ sau dacă $Z = \emptyset$ ($X \cup Y = U$)

Normalizarea schemelor bazelor de date

Normalizarea este necesară din două motive:

- ▶ Pentru eliminarea redundanțelor.
- ▶ Pentru a păstra datele într-o manieră consistentă.

Normalizarea trebuie făcută încă din faza de proiectare a bazei de date și din acest motiv este firesc să vorbim despre **NORMALIZAREA SCHEMEI** bazei de date și nu despre normalizarea anumitor relații.

Există mai multe forme normale (clasice), fiecare aducând ceva în plus față de forma precedentă: 1NF, 2NF, 3NF, BCNF, 4NF.

Să vedem în ce constau...

1NF (1971)

Fie U o mulțime de atribute și $R[U]$ o schema de relație.
Spunem că schema de relație $R[U]$ este în Forma Normală 1 (1NF) dacă și numai dacă domeniile atributelor sunt indivizibile și fiecare valoare a fiecărui atribut este atomică.

Pentru a avea o relație în 1NF, trebuie efectuate următoarele operații:

- ▶ Eliminarea grupurilor repetitive din fiecare relație.
- ▶ Identificarea tupelilor ce ar putea avea duplicate în coloană printr-o cheie.
- ▶ Crearea unei noi scheme de relație având ca atribute: identificatorul de la punctul precedent și valoarea repetată ca atribut atomic.

1NF

Exemplu:

Avem schema: Studenti[nr_matricol, nume, prenume, an] în care studenții ar putea avea câte două prenume:

	nr_matricol	nume	prenume	an
$r :$	1	Ionescu	Maria Ioana	1
	2	Popescu	Vasile	1
	3	Vasilescu	Vali Cristi	2

Pasul 1: Eliminăm grupul repetitiv:

	nr_matricol	nume	an
$r :$	1	Ionescu	1
	2	Popescu	1
	3	Vasilescu	2

1NF

Pasul 2: Identificam cheia:

	nr_matricol	nume	an
$r :$	1	Ionescu	1
	2	Popescu	1
	3	Vasilescu	2

Pasul 3: Creare relație în care fiecare valoare apare pe un rând nou și e legată de relația principală prin intermediul cheii:

	nr_matricol	prenume
$r' :$	1	Maria
	1	Ioana
	2	Vasile
	3	Vali
	3	Cristi

1NF

Deși ar putea părea că am utilizat efectiv relația care este construită peste $R[U]$, în fapt nu am făcut decât să modificăm schema de relație $R[U]$ și să construim o nouă schemă de relație în schema bazei noastre de date denumită $R'[U]$.

Pasul 1: de fapt a eliminat atributul “prenume” din mulțimea U .

Pasul 2: de fapt a identificat o cheie candidat (ce se poate face direct din schemă - vezi cum am găsit o cheie candidat din dependențele funcționale).

Pasul 3: de fapt a construit o nouă schemă de relație $R'[U]$

S-ar putea ca în unele locuri să întâlnești ideea că relația este într-o anumită formă normală. Aceasta idee este incorectă datorită faptului că normalizarea se realizează înainte de a crea baza de date, în stadiul de proiectare a acesteia !

2NF (1971)

O schema de relație $R[U]$ care este în 1NF, împreună cu o mulțime de dependențe funcționale Σ este în a doua formă normală (2NF) dacă orice atribut neprim din $R[U]$ este dependent plin de orice cheie a lui $R[U]$.

Să reluăm exemplul de la dependențe pline:

$R[A, B, C, D]$

$\Sigma = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$

1. Gasiți cheile
2. Gasiți attributele neprime
3. Attributele neprime sunt dependențe plin de cheile găsite ?

2NF

Sa reluam exemplul de la dependențe pline:

$R[A, B, C, D]$

$\Sigma = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$

Posibilele chei: AB, BC

Atribute prime: A, B, C

Atribute neprime: D

$B \rightarrow D \in \Sigma^+$. D nu este dependent plin de AB pentru că, deși avem $AB \rightarrow D \in \Sigma^+$, avem și $B \rightarrow D \in \Sigma^+$ rezultă că $R[U]$ împreună cu Σ nu este în 2NF.

Observație: dacă nu avem chei multivaluate, cu siguranță $R[U]$ este în 2NF (pentru că avem numai dependențe pline de la chei - nu putem găsi submulțimi de atribute atunci când cheia este formată dintr-un singur atribut).

2NF

Având o schemă de relație $R[U]$ care nu este în 2NF, putem să o ducem în 2NF urmând următorii pași:

Pasul 1: Identificăm cheile candidat.

Pasul 2: Găsim attributele neprime.

Pasul 3: Pentru fiecare din attributele neprime A identificăm care sunt attributele dintr-o cheie de care depinde A .

Pasul 4: Creăm o nouă relație R' peste acele attribute identificate la pasul anterior împreună cu atributul neprim pentru care s-a găsit dependența.

Urmați cei patru pași pentru exemplul anterior care nu era în 2NF.

2NF

Exemplu: Dacă avem schema de relație
OS[nume, versiune, an, companie]

nume	versiune	an	companie
Windows	XP	2001	Microsoft
MacOS	Sierra	2017	Apple
Ubuntu	Bionic Beaver	2018	Ubuntu
Windows	7	2009	Microsoft
MacOS	Mojave	2018	Apple

De obicei, când ne exprimăm, spunem că Windows XP este făcut de Microsoft. Avem dependența: **nume,versiune** → **companie**. În același timp, știm că Windows este făcut numai de Microsoft și MacOS numai de Apple. Deci avem și **nume** → **companie**. Fiecare versiune este făcută într-un anumit an. Avem: **versiune** → **an**.

2NF

dependențe:

- ▶ $\text{nume,versiune} \rightarrow \text{companie}$
- ▶ $\text{nume} \rightarrow \text{companie}$
- ▶ $\text{versiune} \rightarrow \text{an}$

Cheie: (nume, versiune) (apar în stânga dependențelor)

Atribute neprime: companie

companie nu este dependent plin pentru că, deși avem $\text{nume,versiune} \rightarrow \text{companie}$, totodată avem și $\text{nume} \rightarrow \text{companie}$.
Vom elimina din schema OS atributul companie și vom adăuga o schemă de relație $R'[\text{nume, companie}]$.

Vom avea așadar...

2NF

nume	versiune	an
Windows	XP	2001
MacOS	Sierra	2017
Ubuntu	Bionic Beaver	2018
Windows	7	2009
MacOS	Mojave	2018

și

nume	companie
Windows	Microsoft
MacOS	Apple
Ubuntu	Ubuntu

Este în 2NF ? Mai avem de verificat dacă an este dependent plin sau nu... să vedem cum faceți asta voi :D

3NF (1971)

Schema de relație $R[U]$ împreună cu mulțimea de dependențe funcționale Σ este în forma a treia normală (**3NF**) dacă este în 2NF și orice atribut neprim din R **NU** este tranzitiv dependent de nici o cheie a lui R .

(Adică e în 2NF și orice atribut neprim depinde de chei și nu de un alt atribut neprim sau grupare de attribute neprime)

Exemplu:

Considerăm schema de relație $R[A, B, C]$ și
 $\Sigma = \{AB \rightarrow C, C \rightarrow A\}$

Chei: AB, BC

Atribute neprime: \emptyset - deci este în 2NF și în 3NF.

3NF - glumița de pe wikipedia. . . .

An approximation of Codd's definition of 3NF, paralleling the traditional pledge to give true evidence in a court of law, was given by Bill Kent: *[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key.*¹

A common variation supplements this definition with the oath: *so help me Codd*².

¹Kent, William - A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM 26 (2), Feb. 1983, pp. 120 – 125.

²Diehr, George. Database Management (Scott, Foresman, 1989), p. 331.

3NF

Exemplu de normalizare 3NF

Fie schema: *Concursuri*[*materie*, *an*, *castigator*, *IQ*], prin *IQ* referindu-ne la *IQ*-ul câștigătorului.

Se observă că avem dependențele funcționale:

$$\Sigma = \{(materie, an) \rightarrow castigator, castigator \rightarrow IQ\}$$

Cheie primara: (*materie*, *an*)

Atribute neprime: *IQ*

Se observă că $(materie, an) \rightarrow IQ \in \Sigma^+$, și în același timp $materie \rightarrow IQ \notin \Sigma^+$ respectiv $an \rightarrow IQ \notin \Sigma^+$. Deci schema de relație se află în 2NF (atributele neprime fiind dependente plin de chei).

Vom normaliza schema pentru

Concursuri[*materie*, *an*, *castigator*, *IQ*] prin construirea a două scheme diferite: *Concursuri*[*materie*, *an*, *castigator*] și *Inteligenta*[*castigator*, *IQ*].

BCNF (Boyce-Codd Normal Form) \equiv 3.5NF (1975)

O schemă de relație $R[U]$ împreună cu o mulțime de dependențe Σ este în BCNF dacă este în 1NF și pentru orice dependența funcțională netrivială $X \rightarrow A \in \Sigma^+$, X este cheie în $R[U]$.

Observație: O schemă de relație ce este în BCNF este în 3NF.

Obs: O schemă ce este în BCNF este și în 3NF.

Consideram (R, Σ) în BCNF - în 1NF este sigur din definiție.

a) **PP (RA) că nu e în 2NF**, adică există un atribut neprim A și o cheie K și A nu este dep. plin de K . Adică există $X \subset K$ a.i. $X \rightarrow A \in \Sigma^+$. Deoarece nu considerăm dep. triviale avem că $A \notin K$. Atunci X este cheie (pentru că (R, Σ) este în BCNF). Ceea ce înseamnă că K nu este cheie (pentru că trebuia să fie minimală).

b) Deci (R, Σ) ce este în BCNF trebuie să fie în 2NF. **PP (RA) că nu ar fi în 3NF**: adică există un atribut A tranzitiv dependent de o cheie X . Adică există Y a.i. $A \notin X$, $A \notin Y$ și avem că: $X \rightarrow Y \in \Sigma^+$, $Y \rightarrow A \in \Sigma^+$ și $Y \rightarrow X \notin \Sigma^+$. Y nu conține nicio cheie pentru că altfel am avea $Y \rightarrow \forall \in \Sigma^+$ deci și $Y \rightarrow X \in \Sigma^+$. Deoarece Y nu este cheie și totuși am $Y \rightarrow A \in \Sigma^+$ avem că (R, Σ) NU este în BCNF - fals. Deci este și 3NF

Descompunerea de tip join fără pierdere

Considerăm o schemă de relație $R[A_1, A_2, \dots, A_n]$. Spunem despre o mulțime $\rho = \{R_1[A_{i_1}^1, A_{i_1}^1, \dots, A_{i_{k_1}}^1], R_2[A_{i_1}^2, A_{i_2}^2, \dots, A_{i_{k_2}}^2] \dots R_t[A_{i_1}^t, A_{i_2}^t, \dots, A_{i_{k_t}}^t]\}$ că este o **descompunere de tip join** a lui $R[A_1, A_2, \dots, A_n]$ dacă:

$$\bigcup_{p=1}^t \bigcup_{r=1}^{k_p} A_{i_r^p}^p = \{A_1 \dots A_n\}$$

ρ este o **descompunere de tip join fără pierdere** dacă $\forall r$ peste R ce satisface mulțimea de dependențe funcționale Σ , avem că $r[A_{i_1}^1, A_{i_1}^1, \dots, A_{i_{k_1}}^1] \bowtie r[A_{i_1}^2, A_{i_2}^2, \dots, A_{i_{k_2}}^2] \bowtie \dots \bowtie r[A_{i_1}^t, A_{i_2}^t, \dots, A_{i_{k_t}}^t]$ satisface mulțimea de dependențe funcționale Σ .

Descompunerea de tip join fără pierdere

Teoremă: Dacă $\rho = \{R_1, R_2\}$ este o descompunere a lui R și Σ o mulțime de dependențe funcționale, atunci ρ este o descompunere de tip join fără pierdere cu privire la Σ dacă și numai dacă $R_1 \cap R_2 \rightarrow R_1 - R_2 \in \Sigma^+$ sau $R_1 \cap R_2 \rightarrow R_2 - R_1 \in \Sigma^+$ (operațiile sunt de fapt pe atributele peste care sunt construite schemele)

Exemplu: Considerăm $R[A, B, C]$ și $\Sigma = \{A \rightarrow B\}$.

$\rho_1 = \{R_1[A, B], R_2[A, C]\}$ este fără pierdere deoarece:
 $AB \cap AC = A, AB - AC = B$ și $A \rightarrow B \in \Sigma^+$

$\rho_2 = \{R_1[A, B], R_2[B, C]\}$ este cu pierdere deoarece:
 $AB \cap BC = B, AB - BC = A$ și $B \rightarrow A \notin \Sigma^+$
 $AB \cap BC = B, BC - AB = C$ și $B \rightarrow C \notin \Sigma^+$

Descompunerea de tip join fără pierdere

Putem calcula Σ_i pentru $R_i[U_i]$ și continua procesul de descompunere până când ajungem la scheme de relație ce sunt în BCNF. $\Sigma_i = \{X \rightarrow Y | X, Y \in U_i\}$ - adică, pentru $R_i[U_i]$ ce este un element al descompunerii ρ luăm acele dependențe din Σ care sunt peste attributele ce sunt în U_i .

Pentru exemplul precedent:

Considerăm $R[A, B, C]$ și $\Sigma = \{A \rightarrow B\}$.

$\rho_1 = \{R_1[A, B], R_2[A, C]\}$ este fără pierdere deoarece:
 $AB \cap AC = A, AB - AC = B$ și $A \rightarrow B \in \Sigma^+$

avem că $\Sigma_1 = \{A \rightarrow B\}$ și $\Sigma_2 = \emptyset$

Alg. pt. descompunerea în join fără pierdere de tip BCNF

Intrare: (R, Σ)

Ieșire: $\rho = \{(R_1, \Sigma_1), \dots (R_t, \Sigma_t)\} =$ descompunere fără pierdere a lui R cu privire la Σ . Unde (R_i, Σ_i) în BCNF, $\forall i \in \{1..t\}$

Pas 1: $\rho = R = R_1$; se caculează Σ^+ și cheile din R .

Pas 2: Cât timp există în ρ un cuplu (R_i, Σ_i) ce nu e în BCNF:

Pas 2.1: Alege $X \rightarrow A$ din Σ_i a.i. $A \notin X$ și X nu conține cheie

Pas 2.2: $S_1 = X \cup \{A\}$; $S_2 = R_i - A$;

Pas 2.3: $\rho = \rho - R_i$; $\rho = \rho \cup S_1 \cup S_2$;

Pas 2.4: Se caculează $\Sigma_{S_1}^+, \Sigma_{S_2}^+$ și cheile pentru S_1 și S_2 .

Exemplu

Schema de relație:

Absolvent(CNP, aNume, adresa, ICod, INume, IOras, medie, prioritate)

$\Sigma = \{ \text{CNP} \rightarrow \text{aNume, adresa, medie} \quad \text{medie} \rightarrow \text{prioritate}$
 $\text{ICod} \rightarrow \text{INume, IOras} \}$

Se descompune în:

Exemplu

Schema de relație:

Absolvent(CNP, aNume, adresa, ICod, INume, IOras, medie, prioritate)

$\Sigma = \{ \text{CNP} \rightarrow \text{aNume}, \text{adresa}, \text{medie} \quad \text{medie} \rightarrow \text{prioritate}$
 $\text{ICod} \rightarrow \text{INume}, \text{IOras} \}$

Se descompune în:

$\rho = \{ R1[\text{ICod}, \text{INume}, \text{IOras}], R2[\text{medie}, \text{prioritate}],$
 $R3[\text{CNP}, \text{aNume}, \text{adresa}, \text{medie}], R4[\text{CNP}, \text{ICod}] \}$

Bibliografie

- ▶ Further Normalization of the Data Base Relational Model. - *Frank Edgar Codd*; IBM Research Report RJ909 (August, 1971)
- ▶ Baze de date relaționale. Dependențe - *Victor Felea*; Univ. Al. I. Cuza, 1996