

Curs 10 : Normalizarea relatiilor

1. **Anomalii in bazele de date**
2. Dependente functionale
3. Forme normale
4. Denormalizarea

1. Anomalii in bazele de date

3. Dependente functionale

3. Forme normale

4. Denormalizarea



Reamintim:

Tipuri de reguli integritate:

- **a entităților:**
- **a relațiilor**
- **restricții contextuale**

Pentru a prezenta procesul de normalizare, este necesar să definim următoarele două concepte:

1. anomalie,
2. dependență funcțională.



Clasificare

- I. Redundanță logică
- II. Anomaliile la actualizare
 - II.a. anomalie la inserție
 - II.b. anomalie la ștergere
 - II.c. anomalie la modificare
- III. Problema reconexiunii (corelata, in general, cu operatiile de compunere).

Curs 10 : Normalizarea relatiilor

1. Anomalii in bazele de date
2. **Dependente functionale**
3. Forme normale
4. Denormalizarea

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Dependență funcțională =

= o restricție care apare între atributele unei entități la nivelul semanticii (semnificației) acestora și al valorilor lor:

fie a_1 și a_2 atributele unei entități E ; spunem că atributul a_2 este **dependent funcțional** de atributul a_1 (sau: atributul a_1 **determina funcțional** atributul a_2)

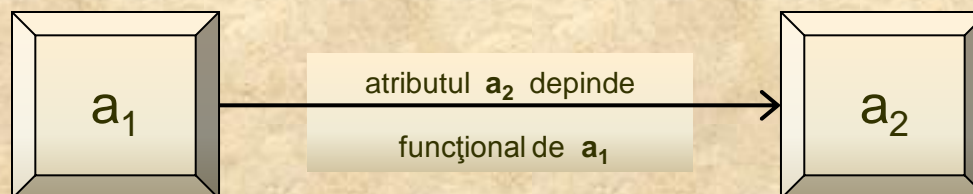
⇔ pentru fiecare valoare a atributului a_1 exista cel mult o valoare a atributului a_2 ;

⇔ atunci cand mai multe tupluri ale entitatii E iau aceleasi valoare pentru atributul a_1 ele iau valoare și pentru atributul a_2

Notatie

$$a_1 \rightarrow a_2$$

Reprezentare grafica



1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Observatii

- Unei valori a atributului a_2 îi pot corespunde mai multe valori ale atributului a_1
(putem spune că a_1 este argumentul iar a_2 este imaginea unei funcții dar NU TOCMAI în sensul matematic al cuvântului:
nu putem calcula valoarea atributului-imagine pe baza unei relatii in care apare atributul-argument ci doar prin examinarea tuturor valorilor atributului-argument si atributului-imagine din acea entitate)
- Vom ignora dependențele triviale, adică dependențele în care a_2 consta dintr-un subset al a_1 ;
- Se pot afla în dependență funcțională nu numai (i) attribute individuale ci și (ii) grupuri de attribute:

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Determinantul unei dependențe funcționale =

= atributul care, prin valorile sale, determină valorile celui alt atribut

(i.e.: atributul aflat, în ambele reprezentări, în stânga săgeții);



Atributul a_1 se numeste **determinant** si atributul a_2 se numeste **determinat** al unei dependente functionale

⇔ pentru fiecare valoare a atributului a_1 există cel mult o valoare a atributului a_2

- Examinarea dependențelor funcționale dintre attributele unei relatii

→ determinarea cheilor candidat precum și a cheii candidat care trebuie să fie aleasă drept cheie primară:

este aleasă cheia candidat care apare ca determinant în toate dependențele funcționale identificate la nivelul entității respective.

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Un algoritm de calcul al inchiderii unui set de attribute aflate in dependenta functionala



Definitie

Fie U o relatie [universala],

$X = \{A_1, A_2, \dots, A_n\}$ o multime de attribute ale U ,

S = o multime de dependente functionale (considerate de proiectantul BD);

se numeste **inchidere a multimii de attribute X determinata de multimea de dependente functionale S** , acea multime de attribute Y cu proprietatea ca orice relatie care satisface toate dependentele din S satisface si dependenta $A_1A_2\dots A_n \rightarrow Y$ (i.e. dependentele din S implica “automat” dependenta $A_1A_2\dots A_n \rightarrow Y$);



Notatie

X^+

$\{A_1, A_2, \dots, A_n\}^+.$

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Un algoritm de calcul al inchiderii unui set de attribute (cont.)



Algoritm de calcul pentru inchiderea X^+ a unui set de attribute X in raport cu un set de dependente functionale S

1. Se initializeaza X^+ cu multimea de attribute $X = \{A_1, A_2, \dots, A_n\}$ implicate in dependentele functionale din multimea S
2. Se cauta o noua dependenta functionala $B_1B_2\dots B_k \rightarrow C$ cu proprietatea ca: $\forall 1 \leq i \leq k, k \leq n: B_i \in A$ dar $C \notin A$
3. $X^+ = X \cup \{C\}$
4. Se reiau Pasul 2 si Pasul 3 pana cand nu se mai pot adauga noi attribute la X
5. Multimea X^+ astfel obtinuta constituie inchiderea multimii de attribute X in raport cu S ;



Observatie

Algoritmul produce, corect, inchiderea X^+ a multimii X pentru ca:

- (i) multimea de attribute ale oricarei relatii [universale] este finita,
- (ii) se adauga – eventual – attribute care fac parte numai din respectiva relatie,
- (iii) la niciun pas din algoritm, dimensiunea multimii X^+ nu se micsoreaza.

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Un algoritm de calcul al inchiderii unui set de atribute (cont.)



Observatie

Fie R o relatie, X multimea tuturor atributelor sale și $A \subset X \Rightarrow$

$A^+ = X \leftrightarrow A$ este o supercheie pentru R

(i.e. A determina functional toate atributele lui R atunci și numai atunci cand A^+ coincide cu multimea tuturor atributelor din R)

\Rightarrow putem cauta / verifica daca o multime de atribute A este o supercheie pentru relatia R astfel:

- (i) aplicam multimii A algoritmul de mai sus și calculam inchiderea A^+ ;
- (ii) testam daca $A^+ = X$;
- (iii) daca da, testam pentru $\forall B \subset A$ daca $B^+ = X$;
- (iv) daca nu exista nicio multime $B \subset A$ a.i. $B^+ = X$, atunci algoritmul se incheie cu $A =$ supercheie pentru R ;
- (v) daca $\exists B' \subset A$ a.i. $B'^+ = X$, atunci inlocuim pe A cu B' și reluam Pasul (i).

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Definitii (tipuri de dependente)

(1) O dependență funcțională $X \rightarrow Y$ se numește **dependență funcțională totală (FT)**

\Leftrightarrow nu există nicio submulțime proprie $X' \subset X$ a. î. $X' \rightarrow Y$

(2) O dependență funcțională $X \rightarrow Y$ se numește **dependență funcțională parțială**

\Leftrightarrow dacă există o submulțime proprie $X' \subset X$ a. î. $X' \rightarrow Y$

(3) Fie D mulțimea dependențelor unei relații și

$p_1, p_2, \dots, p_r, r \geq 1$, proprietăți formale ale acestor depend.

\Rightarrow orice mulțime D' cu proprietatea ca orice dependență a mulțimii D este derivabilă din D' prin aplicarea proprietăților p_1, p_2, \dots, p_r se numeste **acoperire** a lui D in raport cu proprietățile p_1, p_2, \dots, p_r

(4) mulțimea D' se numeste **acoperire minimală** pentru D

\Leftrightarrow nu există nicio submulțime proprie, nevidă a lui D' care să fie o acoperire pentru D

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. Denormalizarea



Definitii (inchiderea unei relatii)

Fie U mulțimea dependențelor considerate de proiectantul bazei pentru o schemă relațională sau pentru o relație universală; **inchiderea relatii [universale]** U , notata cu U^* = mulțimea maximală de dependențe asociate multimei U , obtinuta:

- prin metode euristice sau
- pe baza mulțimii de proprietăți formale ale dependențelor, proprietăți considerate drept reguli de deducție (axiome);

⇒ utilizarea unei multimi de dependente de baza si a axiomelor accelereaza obtinerea inchiderii lui U .

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Definitii (multime de axiome)

Fie D o multime de dependente functionale si D^* inchiderea sa:

(1) O **mulțime de axiome** se numeste **completă**

\Leftrightarrow este suficienta pentru a obtine, plecand de la D , toate dependențele din D^*

(2) O **mulțime de axiome** se numeste **închisă**

\Leftrightarrow orice dependenta dedusa pe baza ei, plecand de la D , face parte D^* .

1. Anomalii in bazele de date
2. **Dependente functionale**
3. Forme normale
4. Denormalizarea



Axiomele lui Armstrong

Fie X, Y, Z, W mulțimi de attribute ale unei entitati:

Ax. 1 – reflexivitate:

$$X \rightarrow X$$

Ax. 2 – creșterea determinantului:

dacă $X \rightarrow Y$ și $X \subseteq Z$, atunci $Z \rightarrow Y$

dacă $X \rightarrow Y$ și $W \subseteq Z$, atunci $X \cup Z \rightarrow Y \cup W$

dacă $X \rightarrow Y$ atunci $X \cup Z \rightarrow Y \cup Z$

Ax. 3 – tranzitivitate:

dacă $X \rightarrow Y$ și $Y \rightarrow Z$, atunci $X \rightarrow Z$.

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. Denormalizarea



Axiomele lui Armstrong (cont.)

Teorema (J.D. Ullman)

Axiomele Ax.1 – Ax.3 reprezintă o mulțime închisă și completă de axiome

Corolar

Închiderea unei multimi de dependente D consta din mulțimea dependențelor deduse din D prin aplicarea axiomelor lui Armstrong.



Observatie

(1) Fie D o dependența funcțională totală

⇒ axiomele lui Armstrong se reduc la o axiomă unică, numită **pseudo-tranzitivitatea**:

dacă $X \rightarrow Y$ și $W \cup Y \rightarrow Z$, atunci $W \cup X \rightarrow Z$

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Reguli de inferenta pentru dependente funct.

Fie X, Y, Z, W mulțimi de attribute ale unei entitati:

Reg. 1 – descompunere:

dacă $X \rightarrow Y$ si $Z \subseteq Y$, atunci $X \rightarrow Z$

Reg. 2 – reuniune:

dacă $X \rightarrow Y$ si $X \rightarrow Z$, atunci $X \rightarrow YZ$

Reg. 3 – semitransitivitatea:

dacă $X \rightarrow Y$ si $YZ \rightarrow W$, atunci $XZ \rightarrow W$

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Definitii (dependente echivalente)

(1) Fie F o mulțime de dependențe funcționale totale

⇒ **închiderea sa pseudo-tranzitivă** $F^+ =$

reuniunea mulțimilor dependențelor funcționale totale care pot fi obținute din F folosind axioma de pseudo-tranzitivitate (sau, echivalent, axiomele lui Armstrong si cele 3 reguli);

(2) Fie F_1 si F_2 două mulțimi de dependențe funcționale totale

⇒ $F_1 \Leftrightarrow F_2$ ddacă $F_1^+ \equiv F_2^+$

(F_1 si F_2 sunt **echivalente** \leftrightarrow inchiderile lor pseudotranzitive coincid);

(3) Fie F o mulțime de dependențe funcționale totale asociată unei mulțimi de attribute A :

F definește o **acoperire minimală** dacă satisface următoarele proprietăți:

- nicio dependență funcțională din F nu este redundantă,
- toate dependențele funcționale totale între submulțimi ale lui A se afla în închiderea pseudo-tranzitivă a lui F ;

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea



Definitii

Fie F o multime de dependente funcționale ale unei entitati:

⇒ F este in **forma canonica** =

- i. orice dependenta functionala din F are in membrul stang un singur atribut,
- ii. F este minimala (nu contine dependente redundante);

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Reprezentarea grafica a dependentelor functionale: digraful [bipartit]

Fie $A = \{A_1, A_2, \dots, A_n\}$ o mulțime de attribute și

$\{X_i \rightarrow A_j\}$ o mulțime de dependențe functionale unde X_i este o submulțime a lui A :

dacă m.stg al dependenței functionale consta dintr-un singur atribut \rightarrow reprezentarea grafica: digraf in care:

- fiecarui atribut ii corespunde unui nod
- dependența $A_i \rightarrow A_j$ este reprezentată printr-un arc de la A_i la A_j ;

dacă m. stg al dependenței functionale consta din mai multe attribute \rightarrow reprezentarea grafica: \approx digraf bipartit

1. Anomalii in bazele de date

2. Dependente functionale

3. Forme normale

4. Denormalizarea

Reprezentarea grafica a dependentelor functionale: digraful [bipartit] (cont.)

⇒ **Graful dependentelor functionale:** un digraf bipartit, definit astfel:

- ✓ pentru fiecare atribut A_j există un singur nod având eticheta A_j , reprezentat printr-un punct;
- ✓ pentru fiecare dependență funcțională de forma $A_i \rightarrow A_j$, există un arc de la A_i la A_j ;
- ✓ pentru fiecare dependență funcțională de forma $X_i \rightarrow A_j$, unde mulțimea X_i este definită de $X_i = \{A_{i_1}, \dots, A_{i_p}\}$ cu $p > 1$, există:
 - un nod auxiliar etichetat prin X_i , reprezentat printr-un dreptunghi și
 - o mulțime de arce plecând de la A_{i_1} la X_i , ..., de la A_{i_p} la X_i ;
 - pentru fiecare dependență de forma $X_i \rightarrow A_j$, există un arc de la X_i la A_j ;

Curs 10 : Normalizarea relatiilor

1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea

1. Anomalii in bazele de date
 2. Dependente functionale
 3. **Forme normale**
 4. Denormalizarea
-

Istoric

- Această tehnică a fost inițiată și fundamentată matematic tot de E.F. Codd (CODD, E.F.: "A Relational Model of Data for Large Shared Databanks", *Comm. ACM*, vol. 13 (1970), no. 6, p. 377-387)
- Codd a propus inițial trei seturi de reguli pe care o relație trebuie să le satisfacă pentru a fi coerentă și pe care le-a denumit prima (**FN1**), a doua (**FN2**), respectiv a treia (**FN3**) formă normală (dând astfel și numele tehnicii însăși)
- Ulterior (1974), Raymond Boyce a introdus, împreună cu E.F. Codd, o definiție mai tare a FN3 – denumită **forma normală Boyce-Codd (FNBC)**.

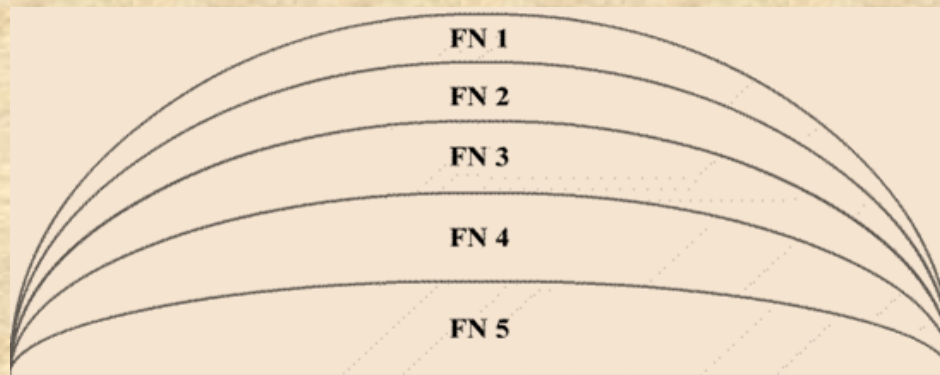
1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea



Observatie

- Cele 5 forme normale au un caracter progresiv:
ex.: o relație aflată în FN3 este automat în FN2 și deci și în FN1;
- Din punctul de vedere al modelului relațional, singura formă normală obligatorie pentru toate relațiile din BD este FN1; dacă însă dorim să evităm toate anomaliile de actualizare (analizate mai sus) este necesar să continuăm procesul de normalizare cel puțin până la FN3;
- Din punct de vedere al performanțelor în exploatare, este preferabil ca BD să fie lăsată într-o formă normală inferioară (se execută procesul invers normalizării: **denormalizarea** BD).



1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea



Observatie

Este justificată întrebarea: câte forme normale mai așteaptă să fie descoperite?

Ronald FAGIN: "A Normal Form for Relational Databases That is Based on Domains and Keys", în *ACM Transactions on Database Systems* 6, 3 (Sept. **1981**), pp.387-415)

În acest articol:

- este introdusă o formă normală care se bazează pe noțiunile de domeniu de valori și cheie primară (**FN/DK**) și
- se demonstrează că o relație este în FN/DK \Leftrightarrow nu prezintă anomalii la modificarea datelor

Această teoremă arată că nu mai este nevoie de nicio altă formă normală (cel puțin din punctul de vedere al eliminării anomaliilor la modificarea datelor).

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Normalizarea BD

- un proces de ameliorare progresivă a schemei conceptuale, prin care un set de relații care încalcă anumite principii de proiectare este înlocuit cu un alt set de relații adecvat, coerent și bine structurat;
- acest proces trebuie să satisfacă următoarele cerințe:
 - ✓ să garanteze **conservarea datelor**
 - ✓ să garanteze **conservarea dependențelor** dintre date,
 - ✓ să reprezinte o **descompunere minimală** a relațiilor inițiale

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Normalizarea BD (cont.)

- exista 2 metode de modelare a BD fara anomalii si fara pierdere de informatii:
 - (1) **top-down**
 - (2) **bottom-up**

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Normalizarea BD (cont.):



Definitie formala (metoda *top-down*)

- = procesul prin care relatia universală care modelează o situație reală și **respectă restricțiile contextuale** încalcând astfel regulile de integritate este înlocuită cu un set de reguli din ce în ce mai adecvate, coerente și bine structurate;
- se realizează plecând de la o relație universală ce conține toate atributele sistemului de modelat;
 - se desfășoară în mai mulți pași;
 - fiecare pas (cu excepția aducerii BD la FN1) presupune:
 - identificarea dependențelor funcționale,
 - verificarea îndeplinirii unor anumite proprietăți denumite generic **forme normale**.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Normalizarea BD (cont.):

- Orice formă normală se obține aplicând o **schemă de descompunere**
- Există două tipuri de descompuneri:

I. Descompuneri care conservă dependențele

relatia R este descompusa într-o mulțime de proiecții R_1, R_2, \dots, R_k
 a.î. dependențele relatiei initiale R sunt echivalente (au închideri pseudo-tranzitive identice) cu reuniunea dependențelor noilor relatii R_1, R_2, \dots, R_k

II. Descompuneri fără pierderi de informație (*L-join*)

relatia R este descompusa într-o mulțime de proiecții R_1, R_2, \dots, R_k
 a.î. pentru orice realizare a lui R este adevărată relația:

$$R = \text{JOIN}(\Pi_{B_1}(R), \Pi_{B_2}(R), \dots, \Pi_{B_j}(R))$$

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Normalizarea BD (cont.):

Regula Casey-Delobel

- descrie conditia ca o descompunere utilizată în procesul normalizării sa se efectueze fără pierdere de informație:
"dacă este satisfăcută o anumită dependență funcțională, atunci există o descompunere fără pierderi";
- Fie: $R(A)$ o schemă relațională,
 α, β, γ o partiție a multimii de attribute A a.i.
 α determină funcțional pe β iar γ contine restul atributelor in A ;

atunci:

$$R(A) = \text{JOIN}(\Pi_{\alpha \cup \beta}(R), \Pi_{\alpha \cup \gamma}(R))$$

unde $\alpha \cup \beta$ reprezintă mulțimea atributelor care intervin în dependențele funcționale,

$\alpha \cup \gamma$ reprezintă reuniunea determinantului (atributul comun al compunerii) cu restul atributelor lui A .

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 1 (FN1)



O relație *R* este în FN1 dacă fiecărui atribut care o compune îi corespunde o valoare indivizibilă (atomică).

În plus, o relație nu trebuie să conțină attribute sau grupuri de attribute repetitive



Această formă figurează ca cerință minimală în majoritatea sistemelor relaționale;



Algoritm AFN1

(aducerea unei relații în FN1 prin eliminarea atributelor compuse și a celor repetitive)

1. se introduc în relație, în locul atributelor compuse, componentele acestora,
2. se plasează grupurile de attribute repetitive, fiecare în câte o nouă relație,
3. se introduce în schema fiecărei noi relații de la pasul 2 cheia primară a relației din care a fost extras atributul repetitiv,
4. se stabilește cheia primară a fiecărei noi relații create la pasul 2, aceasta este compusă din cheia introdusă la pasul 3, precum și din attribute proprii ale acestor noi relații.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 2 (FN2)

 O relație R este în **FN2** ddacă:

- relația R este în **FN1**
- fiecare atribut care nu participă la cheia primară este dependent de întreaga cheie primară;

 Se poate aplica regula Casey-Delobel:

fie relația $R(K1, K2, X, Y)$;

unde $K1$ și $K2$ definesc cheia primară

X și Y sunt mulțimi de attribute astfel încât $K1 \rightarrow X$;

observăm ca R nu este în FN2;

dar R poate fi înlocuită (fără pierdere de informație) cu două proiecții:

$R1(K1, K2, Y)$ și
 $R2(K1, X)$.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 2 (FN2)



Algoritm AFN2

(aducerea unei relații în *FN2* prin eliminarea dependențelor funcționale parțiale din cadrul unor relații aflate în *FN1*)

1. pentru fiecare dependență funcțională parțială se creează o nouă relație având schema formată din determinantul și determinatul acestei dependențe
2. se elimină din cadrul relației inițiale attributele care formează determinatul dependenței parțiale
3. dacă în relația inițială există mai multe dependențe parțiale cu același determinant, pentru acestea se creează o singură relație cu schema formată din determinant (luat o singură dată) și din determinatii dependențelor considerate
4. se determină cheia primară a fiecărei noi relații create; aceasta va conține attributele din determinantul dependenței funcționale parțiale care au stat la baza constituirii relației
5. dacă noile relații create conțin dependențe parțiale, atunci se face reia de la pasul 1, altfel STOP.

1. Anomalii in bazele de date
2. Dependente functionale

3. Forme normale
4. Denormalizarea

Forma normala 3 (FN3)



O relație R este în FN3 ddacă:

- relația R este în FN2
- fiecare atribut care nu participă la o cheia candidat este dependent direct de cheia primară;



A doua condiție interzice utilizarea dependențelor funcționale tranzitive în cadrul relației R

⇒ o relație este în FN3 dacă și numai dacă fiecare atribut care nu este cheie depinde de cheie, de întreaga cheie și numai de cheie.



Se poate aplica regula Casey-Delobel:

fie relația $R(K, X_1, X_2, X_3)$,

unde K este cheia primară a lui R și

atributul X_2 depinde tranzitiv de K ,

presupunem că $K \rightarrow X_1 \rightarrow X_2$.

dependența funcțională $X_1 \rightarrow X_2$ care arată că R nu este în FN3

⇒ se înlocuiește R (fără pierdere de informație) prin două proiecții $R1(K, X_1, X_3)$ și $R2(X_1, X_2)$.



Dependența tranzitivă poate fi mai complexă:

fie K_1 o parte a cheii K ,


tranzitivitatea poate fi de forma $K \rightarrow Y \rightarrow X_2$ unde $Y = \{K_1, X_1\}$

in acest caz, R poate fi descompusă în $R1(K, X_1, X_3)$ și $R2(K_1, X_1, X_2)$.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 3 (FN3)

 Observăm că atributul care asigură tranzitivitatea, X_1 , nu este nici cheie primară în R nici măcar parte a cheii primare. Tocmai din acest motiv, dependența $X_1 \rightarrow X_2$ nu este dezirabilă la nivelul R

Algoritm AFN3

(aducerea unei relații $FN2$ în $FN3$ prin eliminarea dependențelor funcționale tranzitive)

1. pentru fiecare dependență funcțională tranzitivă se transferă attributele implicate în dependența tranzitivă într-o nouă relație
2. se determină cheia primară a fiecărei noi relații create la pasul 1
3. se introduc în relația inițială, în locul atributelor transferate, cheile primare determinate la pasul 2
4. se reanalizează relația inițială; dacă în cadrul ei există noi dependențe tranzitive, atunci se reia de la pasul 1, altfel STOP.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 3 (FN3)

🔔 FN3 poate fi obținută și cu ajutorul unei **scheme de sinteză**.

Informal:

algoritmul de sinteză construiește o acoperire minimală F a dependențelor funcționale totale:

- se elimină attributele și dependențele funcționale redundante.
 - mulțimea F este partiționată în grupuri F_i , astfel încât:
 - în fiecare grup F_i se află dependențe funcționale care au același membru stâng și
 - nu există două grupuri având același membru stâng
 - fiecare grup F_i produce o schemă FN3
- 🔔 Algoritmul realizează o descompunere ce conservă dependențele.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 3 (FN3)



Algoritm SNF3

(aducerea unei relații în FN3 prin utilizarea unei scheme de sinteză)

1. se determină F , o acoperire minimală a lui D (mulțimea dependențelor funcționale)
2. se descompune mulțimea F în grupuri notate F_i , astfel încât în cadrul fiecărui grup să existe dependențe funcționale având aceeași parte stângă
3. se determină perechile de chei echivalente (X, Y) în raport cu F
4. pentru fiecare pereche de chei echivalente:
 - se identifică grupurile F_i și F_j care conțin dependențele funcționale cu partea stângă X și respectiv Y
 - se formează un nou grup de dependențe F_{ij} , care va conține dependențele funcționale având membrul stâng (X, Y)
 - se elimină grupurile F_i și F_j , iar locul lor va fi luat de grupul F_{ij}
5. se determină o acoperire minimală a lui F , care va include toate dependențele $X \rightarrow Y$, unde X și Y sunt chei echivalente (celelalte dependențe sunt redundante)
6. se construiesc relații FN3 (câte o relație pentru fiecare grup de dependențe funcționale).

1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea

Forma normala 3 (FN3)

Algoritm EAR

(elimină attributele redundante din determinantul dependențelor funcționale)

Pentru fiecare dependență funcțională din E și pentru fiecare atribut din partea stângă a unei dependențe funcționale:

1. se elimină atributul considerat
2. se calculează închiderea părții stângi reduse
3. dacă închiderea conține toate attributele din determinantul dependenței, atunci atributul eliminat la pasul 1 este redundant și rămâne eliminat;

altfel: atributul nu este redundant și se reintroduce în partea stângă a dependenței funcționale.

1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea

Forma normala 3 (FN3)

Algoritm EDF

(elimină dependențele funcționale redundante din E)

Pentru fiecare dependență funcțională $X \rightarrow Y$ din E:

1. se elimină dependența din E
2. se calculează închiderea X , în raport cu mulțimea redusă de dependențe
3. dacă Y este inclus în X , atunci dependența $X \rightarrow Y$ este redundantă și rămâne eliminată. În caz contrar, se reintroduce în E;

Algoritm AIDF

(determină închiderea lui A)

1. se caută dacă există în E dependențe $X \rightarrow Y$ pentru care determinantul X este o submulțime a lui A, iar determinatul Y nu este inclus în A
2. pentru fiecare astfel de dependență funcțională se adaugă mulțimii A atributele care constituie determinatul dependenței
3. dacă nu mai există dependențe funcționale de tipul de la pasul 1, atunci $A^+ = A$.

1. Anomalii in bazele de date
2. Dependente functionale




3. Forme normale
4. Denormalizarea

	<i>Condiție de verificat</i>	<i>Soluție (normalizare)</i>
FN1	<ul style="list-style-type: none"> Toate attributele relației trebuie să fie atomice 	<ul style="list-style-type: none"> Fiecare atribut neatomic se transformă într-o nouă relație Se stabilesc relațiile necesare între noile relații și relația inițială modificată
FN2	<ul style="list-style-type: none"> Relația este în FN1; Cheia sa primară constă din mai multe attribute; Toate attributele care nu fac parte din cheia primară sunt complet dependente funcțional de cheia primară 	<ul style="list-style-type: none"> Fiecare parte a cheii primare, împreună cu attributele care depind funcțional complet de ea formează o nouă relație; Se stabilesc relațiile necesare între noile relații care au înlocuit-o pe cea inițială
FN3	<ul style="list-style-type: none"> Relația este în FN2; Nici un atribut care nu face parte dintr-o cheie candidat nu este funcțional dependent de un alt atribut care nu face nici el parte dintr-o cheie candidat (nici un atribut care nu face parte dintr-o cheie candidat nu este funcțional dependent de cheia primară prin tranzitivitate) 	<ul style="list-style-type: none"> Se păstrează în relația inițială numai cheia primară și attributele care depind funcțional de ea direct (inclusiv atributul "incriminat"); Se creează câte o nouă relație din fiecare atribut care nu face parte din cheia primară împreună cu toate attributele (care nu fac nici ele parte din cheia primară a relației inițiale) care sunt dependente funcțional de acesta; Se stabilesc relațiile necesare între noile relații și relația inițială modificată



1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea

Forma normala Boyce-Codd (BCNF)

-  BNCF elimina toate anomalile generate de dependentele functionale.
-  Intuitiv, o relație ***R*** este în forma normală Boyce-Codd \Leftrightarrow fiecare determinant este o cheie candidat
-  Formal, o relație ***R*** este în forma normală Boyce-Codd \Leftrightarrow pentru orice dependență funcțională totală $X \rightarrow A$, X este o cheie a lui R .

1. Anomalii in bazele de date
2. Dependente functionale
3. **Forme normale**
4. Denormalizarea

Forma normala Boyce-Codd (BCNF)



Algoritm ABCNF

(aducerea unei relații FN3 în BCNF prin eliminarea dependențelor funcționale ai căror determinanți nu sunt chei candidat)


1. dacă relația conține unul sau cel mult două atribute, atunci nu pot exista dependențe noncheie și deci relația este în BCNF
2. dacă relația conține mai mult de două atribute, se verifica dacă ea conține dependențe noncheie;
dacă nu există astfel de dependențe, relația este în BCNF
3. pentru fiecare dependență noncheie $X \rightarrow Y$ se creează două relații

una dintre ele va avea schema formată din attributele $\{X, Y\}$, cealaltă va avea schema formată din toate attributele relației inițiale, mai puțin Y

4. se reiau pașii 1, 2, 3 pentru relațiile obținute la pasul 3.

- | | |
|-------------------------------|-------------------------|
| 1. Anomalii în bazele de date | 3. Forme normale |
| 2. Dependente functionale | 4. Denormalizarea |

Forma normala Boyce-Codd (BCNF)

 Se pot aduce în BCNF și relații aflate în FN1 sau FN2 pt ca dependențele funcționale parțiale și cele tranzitive sunt tot dependențe noncheie, adică dependențe ai căror determinanți nu sunt chei candidat

Fie R este o relație ce conține multimea de attribute A :

Algoritm TFBCNF

(aducerea unei relații R din FN1 în BCNF)

1. dacă relația conține cel mult două attribute, atunci R este în BCNF și algoritmul s-a terminat
2. dacă relația conține mai mult de două attribute se consideră toate perechile (X, Y) de attribute distincte din A
3. se determină A_1^+ , închiderea mulțimii de attribute $A_1 = A - \{X, Y\}$.
4. dacă pentru orice pereche (X, Y) , $X \notin A_1^+$ atunci relația R este în BCNF și algoritmul s-a terminat
5. altfel: (pentru cel puțin o pereche (X, Y) , $X \in A_1^+$), relația R nu este în BCNF
6. se reduce progresiv schema relației și se reia algoritmul, exploatând relația redusă.

Orice relație obținută prin reducerea lui R și care este în BCNF se consideră ca făcând parte din descompunerea lui R în procesul aducerii sale în BCNF.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala Boyce-Codd (BCNF)

 Definițiile din

ZANIOLO, CARLO: A New Normal Form for the Design of Relational Schemata, *ACM Transactions on Database Systems*, vol.7 (1982), nr.3, p. 489-499:

evidențiază diferența dintre **FN3** și **BCNF**:



Fie R o relație,

X o submulțime a atributelor lui R ,

A un atribut singular:

R se află în FN3 \Leftrightarrow

pentru fiecare dependență funcțională $X \rightarrow A$ din R este adevărată cel puțin una dintre următoarele afirmații:

1. X conține pe A (a.î. dependența funcțională este banală);
2. X este o supercheie;
3. A este conținut într-o cheie candidat a lui R .



R se afla in BCNF dacă, in definiția anterioară, se elimina posibilitatea (3)

(\Rightarrow BCNF este mai puternică decât FN3).

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 4 (FN4)

- 🔔 Se bazeaza pe un alt tip de dependenta intre valorile atributelor din relatii: dependenta multivaloare;
- 🔔 Acest tip de dependenta apare – in general – in relatiile in care:
 - mai mult de un atribut prezinta valori multiple si
 - intre acele attribute exista relatii de tip 1:m INDEPENDENTE, impuse de context.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 4 (FN4)



Dependenta multivaloare = (MVD = multi-valued dependency)

= o dependenta intre minimum 3 attribute apartinand aceleiasi relatii R cu proprietatea ca:

- (i) pentru fiecare valoare a lui A exista un set de valori ale lui B si un set de valori ale lui C
- (ii) aceste seturi de valori sunt independente unul de celalalt.



Notatie

$(A \twoheadrightarrow B, A \twoheadrightarrow C)$



Exemplu

$(nrG \twoheadrightarrow tipAutoReparat, nrG \twoheadrightarrow Client).$

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 4 (FN4)



Definitii

O dependenta multivaloare $A \twoheadrightarrow B$ dintr-o relatie R se numeste **triviala** \Leftrightarrow

(a) $B \subset A$ **sau**

(b) $A \cup B = R$;

O dependenta multivaloare $A \twoheadrightarrow B$ dintr-o relatie R se numeste **netriviala** \Leftrightarrow

niciuna dintre cele 2 conditii de mai sus nu are loc;

1. Anomalii in bazele de date
2. Dependente functionale

3. Forme normale
4. Denormalizarea

Forma normala 4 (FN4)

Observatie

Fie W, V, X, Y și Z submulțimi de attribute ale unei scheme relaționale R .

- \forall T multime de multidependențe \Rightarrow
 \exists o mulțime completă de axiome (Ax1–Ax8) care permit obținerea tuturor multidependențelor ce se pot deduce din mulțimea T :

Ax1. Dacă $Y \subset X$, atunci $X \rightarrow Y$

Ax2. Dacă $X \rightarrow Y$, atunci $X \cup Z \rightarrow Y \cup Z$

Ax3. Dacă $X \rightarrow Y$ și $Y \rightarrow Z$, atunci $X \rightarrow Z$

Ax4. Dacă $X \twoheadrightarrow Y$, atunci $X \twoheadrightarrow R - \{X \cup Y\}$

Ax5. Dacă $X \twoheadrightarrow Y$ și $V \subset W$, atunci $W \cup X \twoheadrightarrow V \cup Y$

Ax6. Dacă $X \twoheadrightarrow Y$ și $Y \twoheadrightarrow Z$, atunci $X \twoheadrightarrow (Z - Y)$

Ax7. Dacă $X \rightarrow Y$, atunci $X \twoheadrightarrow Y$

Ax8. Dacă $X \twoheadrightarrow Y, Z \rightarrow W, W \subset Y$ și $Y \cap Z = \emptyset$, atunci $X \rightarrow W$

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 4 (FN4)



O relatie R este in $FN4$ ddaca

- relatia R este in $BCNF$
- nu contine nicio MVD netriviala (i.e.: orice dependență multivaloare este o dependență funcțională);



Observatii

1. $FN4$ este mai tare decat $BCNF$ pt ca elimina complet redundanta in date
2. Descompunerea unei relatii in 2 relatii se bucura de proprietatea de compunere fara pierdere de informatie;



Algoritm AFN4




(aducerea unei relații $BCNF$ la $FN4$ prin eliminarea MVD)

1. Se identifica toate MVD netriviale care apar in relatie,
2. pentru fiecare MVD netriviala: $A \twoheadrightarrow B$ se creaza o relatie separata care sa contina atributul/atributele B și o copie a atributului/atributelor-determinant A .

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 5 (FN5)

-  Se bazeaza pe un alt tip de dependenta intre valorile atributelor din relatii: dependenta la descompunerea fara pierdere de informatie (la descompunerea nonaditiva)
-  Acest tip de dependenta apare atunci cand o relatie trebuie descompusa in mai mult de 2 relatii (ca in cazul *FN4*) si este rezolvata prin *FN5*
-  **Dependenta la descompunerea fara pierdere de informatie (dependenta la descompunerea nonaditiva) = (*Lossless-join dependency*)**
- = o proprietate a operatiei de descompunere care impiedica aparitia de linii nelegitime atunci cand are loc o operatie de compunere naturala a mai multor relatii



Notatie

$$*(R_1, R_2, \dots, R_k)$$


1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 5 (FN5)

 **O relatie R este in FN5 ddaca**

relatia R nu prezinta nicio dependenta la compunere

 Observatii

1. $FN5$ se mai numeste si forma normala proiectie-compunere ($PJNF = Project-Join Normal Form$)
2. Între mulțimile de attribute X , Y și Z din cadrul relației R există o dependenta la compunere dacă există multidependențe între fiecare dintre perechile de mulțimi (X, Y) , (Y, Z) și (X, Z) .
 \Rightarrow o dependenta la compunere poate exista numai în cadrul acelor relații $FN4$ care prezintă chei compuse și attribute comune în chei.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Forma normala 5 (FN5)



Algoritm AFN5

(aducerea unei relații *FN4* la *FN5* prin eliminarea dependentelor la compunere)

1. Se cauta dependentele la compunere ținând seama de observația anterioară. Dacă nu există dependente la compunere, atunci relația este în *FN5* si STOP.
2. Daca exista dependente la compunere , atunci se descompune relația în scopul obținerii *FN5*. Considerând că schema relației conține mulțimile de attribute X , Y , Z și că între fiecare pereche (X, Y) , (Y, Z) , (X, Z) există multidependențe, relația se rupe prin proiecție în trei relații: $R1(X, Y)$, $R2(Y, Z)$ și $R3(X, Z)$.

1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Concluzii

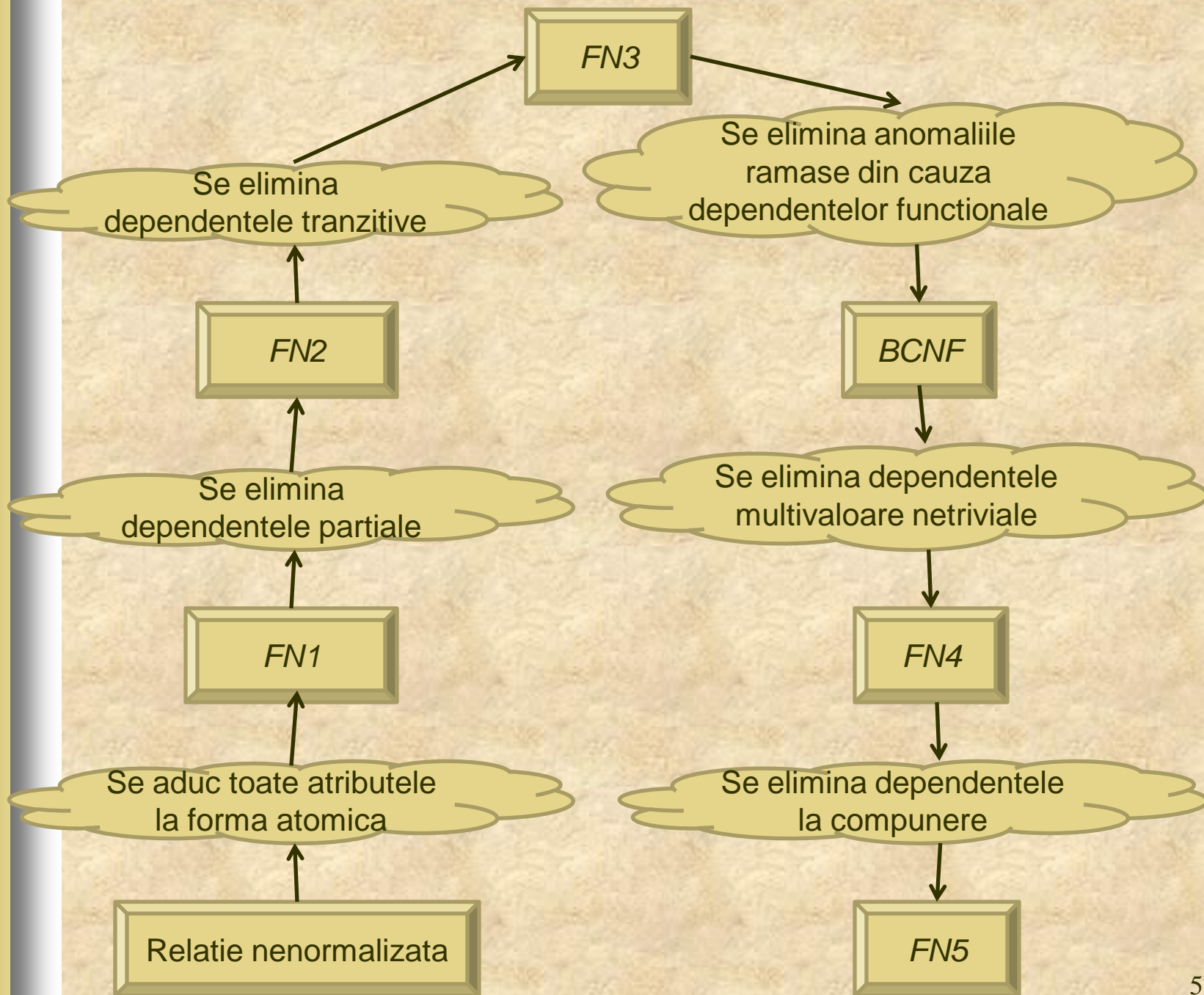
Normalizarea

= o tehnica de obtinere a unei multimi de relatii inzestrate cu anumite proprietati, in conformitate cu constrangerile specifice, impuse de situatia concreta modelata prin baza de date,

= o metoda formală prin care se pot identifica relatiile cu ajutorul campurilor de cheie și se pot descoperi diversele tipuri de dependente care exista intre attributele lor,

= un proces care transforma o relatie trecand-o dintr-o *FN* in alta;

- la fiecare pas, se incearca eliminarea acelor caracteristici ale relatiei care o fac vulnerabila la anomaliiile de actualizare
- trecerea intr-o *FN* superioara face a relatia mai invulnerabila dar și mai restrictiva ca format:



1. Anomalii in bazele de date
2. Dependente functionale

3. **Forme normale**
4. Denormalizarea

Concluzii (cont.):

$FN1 \rightarrow FN2$

- elimină redundanțele datorate dependenței netotale a atributelor care nu participă la o cheie, față de cheile lui R
- se suprimă dependențele funcționale care nu sunt totale;

$FN2 \rightarrow FN3$

- elimină redundanțele datorate dependenței tranzitive
- se suprimă dependențele funcționale tranzitive;
- se conserva si datele si dependentele;

$FN3 \rightarrow BCNF$

- elimină redundanțele datorate dependenței funcționale
- se suprimă dependențele în care partea stângă nu este o supercheie
- se conserva doar datele.

1. Anomalii in bazele de date
2. Dependente functionale

3. Forme normale
4. Denormalizarea

Concluzii (cont.):

$BCNF \rightarrow FN4$

- elimină redundanțele datorate multidependenței
- se suprimă toate multidependențele care nu sunt și dependențe funcționale
- se conserva doar datele;

$FN4 \rightarrow FN5$

- elimină redundanțele datorate dependenței la compunere
- se suprimă toate dependențele la compunere care nu sunt implicate de o cheie;

$BCNF, FN4$ și $FN5$

- toate FN se bazeaza pe regula “orice determinant este o cheie”,
- pentru fiecare FN determinantul se definește in raport cu un alt tip de dependenta:
 - ✓ dependența funcțională, ($BCNF$)
 - ✓ multidependența ($FN4$)
 - ✓ dependența la compunere ($FN5$).

Curs 10 : Normalizarea relatiilor

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. **Denormalizarea**

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. Denormalizarea

Deormalizarea

Fie $R = \{R_1, R_2, \dots, R_p\}$ o mulțime de relații

Denormalizarea R înseamnă înlocuirea R cu

$$R' = \text{JOIN}(R_1, R_2, \dots, R_p),$$

astfel încât $\forall 1 \leq i \leq p$: proiecția lui R după atributele lui R_i va produce din nou relația R_i

Observatii

Denormalizare \equiv rafinarea schemei relaționale a.î. gradul de normalizare a unei relații modificate să fie mai mic decât gradul de normalizare a cel puțin uneia dintre relațiile inițiale;

Obiectivul denormalizarii

mărirea redundanței (relația R' se află la un nivel de normalizare mai scăzut decât relațiile R_1, R_2, \dots, R_p componente)

\equiv reducerea numărului de *join*-uri care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans (ca parte din proiectarea bazei de date).

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. Denormalizarea



Problemele denormalizarii

1. Reaparitia anomaliilor pe care diversele forme normale reusisera sa le elimine;
2. Urmari negative ale denormalizarii asupra fisierelor stocate: un design fizic poate fi bun pentru anumite aplicații, dar prost pentru altele (in tabelul obtinut prin denormalizare - *join* – liniile par adiacente dar in memorie inregistrarile nu sunt! --> interogările care vizeaza inregistrari numai in unul dintre tabelele participante la *join* vor fi mai lente);
3. Lipsa unui criteriu formal pt stabilirea nivelului la care denormalizarea trebuie sa se opreasca;
4. Lipsa unor reguli formale pentru stabilirea situațiilor în care este indicată denormalizarea relațiilor.

1. Anomalii in bazele de date
2. Dependente functionale

3. Forme normale
4. Denormalizarea



Cazuri in care denormalizarea este indicata:

Normalizarea “completa” a BD → scaderea semnificativa a performantelor BD

⇒ 3 exemple de situatii in care trebuie considerata posibilitatea reducerii gradului de normalizare in favoarea cresterii performantelor:

A. Actualizari vs. interogari

Dacă o relație are:

- o rată de reactualizare scăzută
- o rată de interogare foarte ridicată

atunci denormalizarea este o solutie.

1. Anomalii in bazele de date
2. Dependente functionale

3. Forme normale
4. Denormalizarea



Cazuri in care denormalizarea este indicata:

(cont.)

B. Calcularea vs. memorarea datelor derivate

Din perspectiva proiectării fizice a BD se poate opta intre:

- stocarea in BD a atributelor derivate
- calcularea atributelor derivate de fiecare dată când este necesar

Criterii:

- costul memorarii vs costul recalcularii
- costul menținerii concordanței dintre datele calculate si datele operationale din care sunt derivate,

C. Pro sau contra redundantei

Din perspectiva proiectării fizice a BD se poate opta intre:

- eliminarea completa a redundantei (aducerea la FN5)
- dublarea unor attribute (pentru simplificarea *join*-urilor)

dupa evaluarea costurilor /pericolelor implicate de necesitatea actualizarii mai multor copii ale aceleasi informatii.

Curs 10 : Normalizarea relatiilor

1. Anomalii in bazele de date
2. Dependente functionale
3. Forme normale
4. Denormalizarea