

BAZE DE DATE

CURS 10

Normalizare (partea 3). Denormalizare.

Forma normală 5 (FN5)

- FN5 își propune eliminarea redundanțelor care apar în relații $m:n$ dependente.
 - În general, aceste relații nu pot fi descompuse.
 - S-a arătat că o relație de tip 3 este diferită de trei relații de tip 2. Există totuși o excepție, și anume, dacă relația este ciclică
- **Intuitiv**, o relație R este în forma normală 5 dacă și numai dacă:
 - relația este în FN4;
 - nu conține dependențe ciclice.

Forma normală 5 (FN5)

- Dependența funcțională și multidependența permit descompunerea prin proiecție, fără pierdere de informație, a unei relații în **două relații**.
- Regulile de descompunere (FN1 – FN4) nu dau toate descompunerile posibile prin proiecție ale unei relații.
- Există relații care nu pot fi descompuse în două relații dar pot fi descompuse în **trei, patru sau mai multe relații fără a pierde informații**.
- Pentru a obține descompuneri *L-join* în trei sau mai multe relații, s-a introdus conceptul de **join-dependență sau dependență la compunere (JD)**.

Forma normală 5 (FN5)

- Fie $\{R_1, R_2, \dots, R_p\}$ o mulțime de scheme relaționale care nu sunt disjuncte și a căror reuniune este R .
- R satisface **join-dependența** $*\{R_1, R_2, \dots, R_p\}$ dacă la fiecare moment are loc egalitatea:

$$R = \text{JOIN}(\Pi_{\alpha_1}(R), \Pi_{\alpha_2}(R), \dots, \Pi_{\alpha_p}(R))$$

unde α_k reprezintă mulțimea atributelor corespunzătoare lui R_k ($1 \leq k \leq p$).

- **Join-dependența** $*\{R_1, R_2, \dots, R_p\}$ are loc în R , dacă R_1, R_2, \dots, R_p este o **descompunere L-join** a lui R .
- Pentru $p = 2$ se regăsește **multidependența**.
- O **join-dependență** $*\{R_1, R_2, \dots, R_p\}$ în care una dintre R_i este chiar R , definește o **join-dependență trivială**.

Forma normală 5 (FN5)

- *Join*-dependența **generalizează multidependența**.
- Într-adevăr, multidependența $X \twoheadrightarrow Y$ în relația $R(X, Y, Z)$ (deci și $X \twoheadrightarrow Z$), corespunde *join*-dependenței $\{X \cup Y, X \cup Z\}$.
- Invers, *join*-dependența $\{R_1, R_2\}$ corespunde multidependenței $R_1 \cap R_2 \twoheadrightarrow R_1 - (R_1 \cap R_2)$.
- Formal, o relație R este în FN5 dacă și numai dacă orice *join* dependență $\{R_1, R_2, \dots, R_p\}$ care are loc în R fie este trivială, fie conține o supercheie a lui R (adică, o anumită componentă R_i este o supercheie a lui R).
- Cu alte cuvinte, o relație R este în FN5 dacă **orice *join*-dependență definită pe R este implicată de cheile candidat ale lui R .**

Forma normală 5 (FN5)

- Între mulțimile de attribute X , Y și Z din cadrul relației R există o *join* dependență dacă există multidependențe între fiecare dintre perechile de mulțimi (X, Y) , (Y, Z) și (X, Z) .
- Aducerea în FN5 prin eliminarea join dependențelor!

Forma normală 5 (FN5)

- Între mulțimile de attribute X , Y și Z din cadrul relației R există o *join* dependență dacă există multidependențe între fiecare dintre perechile de mulțimi (X, Y) , (Y, Z) și (X, Z) .
- Aducerea în FN5 prin eliminarea join dependențelor!
- Exemple – suport curs.

Concluzii NORMALIZARE

1. **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.
2. **FN2 \rightarrow FN3** elimină redundanțele datorate dependenței tranzitive. Se suprimă dependențele funcționale tranzitive.
3. **FN3 \rightarrow BCNF** elimină redundanțele datorate dependenței funcționale. Se suprimă dependențele în care partea stângă nu este o supercheie.
4. **BCNF \rightarrow FN4** elimină redundanțele datorate multidependenței. Se suprimă toate multidependențele (non-cheie) care nu sunt și dependențe funcționale.
5. **FN4 \rightarrow FN5** elimină redundanțele datorate dependenței ciclice. Se suprimă toate *join*-dependențele care nu sunt implicate de o cheie.
6. **BCNF, FN4 și FN5** corespund la regula că orice determinant este o cheie, dar de fiecare dată dependența cu care se definește determinantul este alta și anume dependența funcțională, multidependența sau *join*-dependența).
7. Descompunerea unei relații FN2 în FN3 conservă datele și dependențele, pe când descompunerea unei relații FN3 în BCNF și, respectiv, a unei relații BCNF în FN4 conservă doar datele.

DENORMALIZARE

- Procedura de normalizare elimină redundanțele prin efectuarea unor proiecții, DAR NU toate redundanțele pot fi eliminate în acest mod.

=> Uneori este necesară **denormalizarea** care presupune:

- Mărirea redundanței;
- Reducerea numărului de join-uri care trebuie efectuate => micșorare timp de execuție!

DENORMALIZARE

- Ideile normalizării sunt utile în proiectarea BD, dar **nu sunt obligatorii!**
- Dependența și normalizarea sunt de natură semantică (cu alte cuvinte, se referă la ceea ce înseamnă datele).
- În schimb, algebra relațională și calculul relațional (limbajele SQL) se referă doar la valorile efective ale datelor și în multe cazuri nu necesită mai mult decât FN1.

DENORMALIZARE

A) Obiectivul denormalizării constă în **reducerea numărului de join-uri** care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans, ca făcând parte din proiectarea bazei de date.

DENORMALIZARE

B) Conceptul de denormalizare suferă de un număr de probleme binecunoscute.

- Odată începută denormalizarea, nu este clar unde trebuie să se oprească.
- Nu se mai lucrează cu relații normalizate și astfel pot apărea anomaliile pe care normalizarea le corectează.
- Un design fizic poate fi bun pentru anumite aplicații, dar prost pentru altele (denormalizarea la nivelul fișierelor stocate).

DENORMALIZARE

Exemplu:

- Se poate presupune că fiecare tabel corespunde unui anumit fișier stocat și că fiecare fișier stocat este format dintr-o mulțime contiguă fizic de înregistrări stocate, câte una pentru fiecare tuplu din tabel.
- Se presupune că informațiile (join-ul) despre creatori, vestimentații și accesorii se reprezintă printr-un tabel și, prin urmare, un fișier stocat. În această structură fizică interogarea « Obținerea informațiilor despre creatorii care oferă vestimentații cu accesorii din marjele » se rezolvă cu ușurință.

DENORMALIZARE

- Interogarea « Obținerea informațiilor despre creatorii din Sibiu » va prezenta performanțe mai reduse în această structură fizică, decât dacă am fi menținut trei tabele de bază pentru cele 3 entități și, prin urmare, 3 fișiere stocate fizic separat.
- În acest ultim design, toate înregistrările despre creatori vor fi fizic contigue, în timp ce în primul design ele sunt dispersate fizic într-o zonă largă și, prin urmare, vor necesita **mai multe operații I/O**.

DENORMALIZARE

Când este utilă denormalizarea?

- Ca o regulă empirică, se poate afirma că, dacă performanțele nu sunt satisfăcătoare și relația are o rată de reactualizare scăzută, dar o rată a interogărilor foarte ridicată, denormalizarea poate constitui o opțiune viabilă.
 - BD de tip OLTP – preponderent normalizare
 - BD de tip OLAP – preponderent denormalizare
- Nu există reguli fixe pentru stabilirea situațiilor în care este indicată denormalizarea relațiilor.
- În general, denormalizarea înseamnă considerarea dublării atributelor și grupării relațiilor
 - Dublarea (duplicarea) atributelor sau gruparea relațiilor are ca scop reducerea numărului de join-uri necesare pentru efectuarea unei interogări

Exemplu de denormalizare

- FILIALA (cod_filiala#, strada, zona, oras, cod_postal, telefon)
- Relația nu este în FN3 deoarece cod_postal → {zona, oras}
- Aplicăm FN3 și se obține:
 - FILIALA1(cod_filiala#, strada, cod-postal, telefon)
 - COD_P(cod_postal#, zona, oras)
- Nu este convenabil, deoarece rareori vom accesa adresa filialei, fără informații referitoare la zonă și oraș. Prin urmare, în acest caz putem prefera varianta denormalizată (cea aflată în FN2)

Cazuri de denormalizare

- Considerarea datelor derivate
- Combinarea relațiilor de tip 1:1
- Duplicarea atributelor care nu sunt chei în relații 1:M
- Tabele de căutare (*Lookup tables*)
- Duplicarea atributelor cheii externe într-o relație 1:M pentru simplificarea join-urilor
- Duplicarea atributelor în relațiile de tip M:N, pentru reducerea join-urilor

Cazuri de denormalizare

Duplicarea atributelor care nu sunt chei în relații 1:M

```
SELECT p.*, pp.nume
FROM   proprietate_de_inchiriat p, proprietar pp
WHERE  p.cod_proprietar = pp.cod AND cod_filiala = 'S3';
```

- Dacă se va duplica atributul nume în relația *proprietate_de_inchiriat*, interogarea devine:

```
SELECT p.*
FROM   proprietate_de_inchiriat p
WHERE  cod_filiala = 'S3';
```

- Avantaje sau dezavantaje? Depinde de problemele care pot apărea.

Cazuri de denormalizare

Tabele de căutare (de referință)

- Acestea conțin, de obicei, un cod și o descriere (și / sau denumire)
- Le întâlnim în forma normalizată a bazei de date
- De exemplu, se poate defini un tabel de căutare pentru tipul de proprietate și modifica tabelul *proprietate_de inchiriat* astfel:

TIP_PROPRIETATE(cod_tip#, descriere)

PROPRIETATE_DE_INCHIRIAT(cod#, strada, zona, oras, cod_postal, cod_tip, nr_camere, chirie, cod_proprietar, cod_filiala, nr_personal)

Cazuri de denormalizare

Tabele de căutare (de referință) - continuare

- Avantaje:
 - Dacă este modificată descrierea, atunci se va modifica o singură dată, în tabelul de căutare
 - Se reduce dimensiunea relației de la capătul M al relației (PROPRIETATE_DE_INCHIRIAT)
- Varianta de pe slide-ul anterior este în FN3
- Dacă se consideră că accesarea descrierii tipului de proprietate are loc frecvent odată cu accesarea informațiilor despre proprietate, atunci putem denormaliza astfel:

~~TIP_PROPRIETATE(cod_tip#, descriere)~~

PROPRIETATE_DE_INCHIRIAT(cod#, strada, zona, oras, cod_postal, **descriere_tip**, nr_camere, chirie, cod_proprietar, cod_filiala, nr_personal)

Cazuri de denormalizare

Duplicarea atributelor cheii externe într-o relație de tip 1:M

- Obiectiv: simplificarea join-urilor
- Relațiile pot să nu fie învecinate în diagramă
- **Exemplu:** Să se enumere proprietarii de proprietăți de închiriat dintr-o filială.

```
SELECT pp.nume
FROM   proprietate_de_inchiriat p, proprietar pp
WHERE  p.cod_proprietar = pp.cod AND cod_filiala = 'S3';
```

- Dacă se duplică cheia externă *cod_filiala* în relația PROPRIETAR, adică se introduce o relație directă între FILIALA și PERSONAL, atunci cererea devine:

```
SELECT pp.nume
FROM   proprietar pp
WHERE  cod_filiala = 'S3';
```

Cazuri de denormalizare

Duplicarea atributelor cheii externe într-o relație de tip 1:M (continuare)

- **Atenție!** Sunt necesare constrângeri suplimentare asupra cheilor externe. De exemplu, dacă un proprietar ar închiria prin mai multe filiale atunci modificările nu mai sunt valabile.
 - Prezența unui atribut multiplu ar însemna non-FN1.
- **Observație:** Singurul motiv pentru care relația `PROPRIETATE_DE_INCHIRIAT` conține atributul *cod_filiala* constă în faptul că este posibil ca o proprietate să nu aibă alocat un membru de personal, mai ales la început, atunci când este preluată de către agenție.

Cazuri de denormalizare

Duplicarea atributelor în relațiile de tip M:N

- Presupunem că relația M:N dintre CHIRIAS și PROPRIETATE_DE_INCHIRIAT a fost descompusă prin introducerea relației intermediare VIZITARE.
- **Exemplu:** Care sunt chiriașii care au vizitat proprietăți, dar mai au de făcut comentarii asupra unora dintre ele? Personalul de la agenție are nevoie de atributul *strada* atunci când discută cu chiriașii.

```
SELECT p.strada, c.*, v.data
FROM   chirias c, vizitare v, proprietate_de_inchiriat p
WHERE  v.cod_proprietate = p.cod AND c.cod = v.cod_chirias AND
comentarii IS NULL;
```

Cazuri de denormalizare

Duplicarea atributelor în relațiile de tip M:N (continuare)

- Dacă se introduce atributul strada în relația VIZITARE, atunci cererea devine:

```
SELECT v.strada, c.*, v.data
FROM   chirias c, vizitare v
WHERE  c.cod = v.cod_chirias AND comentarii IS NULL;
```