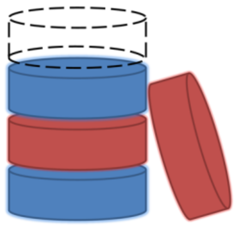


Rafinarea structurii bazelor de date

(Dependențe funcționale)

MovieList

Title	Director	Cinema	Phone	Time
The Hobbit	Jackson	Florin Piersic	441111	11:30
The Lord of the Rings 3	Jackson	Florin Piersic	441111	14:30
Adventures of Tintin	Spielberg	Victoria	442222	11:30
The Lord of the Rings 3	Jackson	Victoria	442222	14:00
War Horse	Spielberg	Victoria	442222	16:30

Screens

Cinema	Time	Title
Florin Piersic	11:30	The Hobbit
Florin Piersic	14:30	The Lord of the Rings 3
Victoria	11:30	Adventures of Tintin
Victoria	14:00	The Lord of the Rings 3
Victoria	16:30	War Horse

Movies

Title	Director
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

Cinema

Cinema	Phone
Florin Piersic	441111
Victoria	442222

Proprietățile descompunerii relațiilor

1. Descompunerea trebuie să păstreze informațiile
   - Datele din relația originală ≡ Data din relațiile descompunerii
   - Crucial for păstrarea consistenței datelor!

2. Descompunerea trebuie să respecte toate DF
   - Dependențele funcționale din relația originală ≡ reuniunea dependențelor funcționale din relațiile descompunerii
   - Facilitează verificarea violărilor DF

Proiecția dependențelor funcționale

- Proiecția mulțimii F pe α (notată prin F_α) este mulțimea acelor dependențe din F^+ care implică doar atribute din α, adică:

$$F_\alpha = \{ \beta \to \gamma \in F^+ \mid \beta\gamma \subseteq \alpha \}$$

- Algoritm pentru determinare proiecției DF:

```
Input: α, F
Output: Fα
result = ∅;
for each  β ⊆ α do
   T =  β⁺ (w.r.t. F)
   result = result ∪ {β→T ∩ α}
return result
```

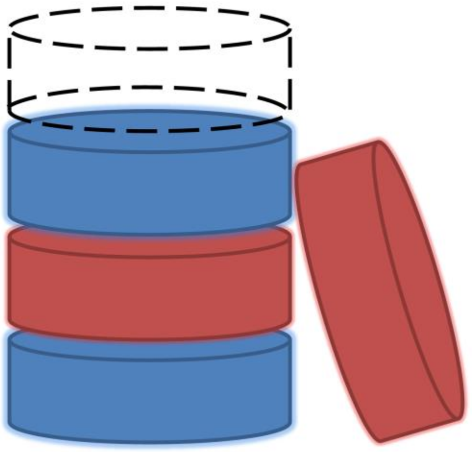
Complexitatea e exponențială

Descompunere cu păstrarea dependențelor

Descompunerea $\{R_1, R_2, \ldots, R_n\}$ a relației R e cu păstrarea dependențelor dacă $(F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})$ şi F sunt echivalente, adică:

$$(F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn}) \Rightarrow F \text{ şi}$$
$$F \Rightarrow (F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})$$

Forme normale



Redundanța

Redundanța este cauza principală a majorității problemelor legate de structura bazelor de date relaționale:

- ***spațiu utilizat,***
- ***anomalii de inserare / stergere / actualizare***

.

Redundanța

- *Dependențele funcționale* pot fi utilizate pentru identificarea problemelor de proiectare şi sugerează posibile îmbunătățiri

- Fie relația R cu 3 atribute, ABC.
  - Nici o DF: nu avem redundanțe.
  - Pentru A→B: Mai multe înregistrări pot avea aceeaşi valoare pentru A, caz în care avem valori identice pentru B!

Tehnica de rafinare a structurii: *descompunerea*

Descompunerea trebuie folosită cu "măsură":

- Este necesară o rafinare? Există motive de decompunere a relației?

- Ce probleme pot să apară prin descompunere?

Forme Normale

- Dacă o relație se află într-o *formă normală* particulară avem certitudinea că anumite categorii de probleme sunt eliminate/minimizate $\rightarrow$ ne ajută să decidem daca descompunerea unei relații este necesară sau nu.

- Formele normale bazate pe DF sunt:
  - *prima formă normală (1NF),*
  - *a doua formă normală (2NF),*
  - *a treia formă normală (3NF),*
  - *forma normală Boyce-Codd (BCNF).*

$$\{BCNF \subseteq 3NF,\ 3NF \subseteq 2NF,\ 2NF \subseteq 1NF\}$$

1NF

Definiție. O relație se află în *Prima Formă Normală* (1NF) dacă fiecare atribut al relației poate avea doar valori atomice (deci listele şi mulțimile sunt excluse)

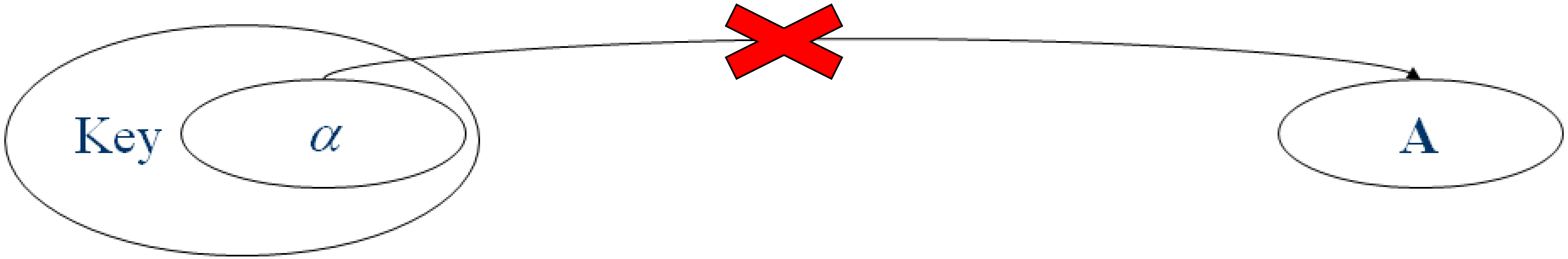
(această condiție este implicită conform definiției modelului relațional)

2NF

Spunem că avem o *dependență funcțională parțială* într-o relație atunci când un atribut *ne-cheie* este dependent funcțional de o parte a unei chei candidat a relației (dar nu de întreaga cheie).

Definiție. O relație se află în *A Doua Formă Normală* (2NF) dacă este 1NF şi nu are dependențe parțiale.

2NF



Partial dependencies (A not in a KEY)

BCNF

Definiție. O relație R ce satisface dependențele funcționale F se află în *Forma Normală Boyce-Codd* (BCNF) dacă se află în 2NF și pentru toate $\alpha \rightarrow A$ din F^+:

- $A \in \alpha$ (DF *trivială*), sau
- α conține o cheie a lui R.

R este în BCNF dacă singurele dependențe funcționale satisfăcute de R sunt cele corespunzătoare constrângerilor de cheie.

BCNF

A not in a KEY

3NF

Definitie. O relație R ce satisface dependențele funcționale F se află în *A Treia Formă Normală* (3NF) dacă se află în 2NF și pentru toate $\alpha \rightarrow A$ din F^+

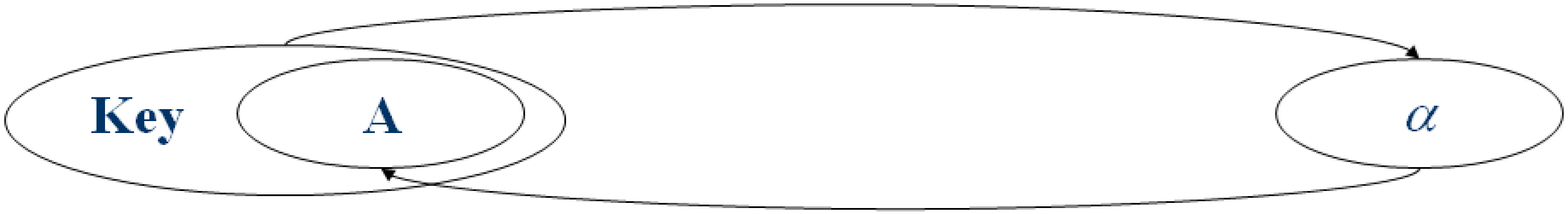
- $A \in \alpha$ (DF *trivială*), sau
- α conține o cheie de-a lui R, sau
- A este un atribut prim.

■ Dacă R este în BCNF, evident este şi în 3NF.

■ Dacă R este în 3NF, este posibil ca să apară anumite redundanțe. Este un compromis, utilizat atunci când BCNF nu se poate atinge.

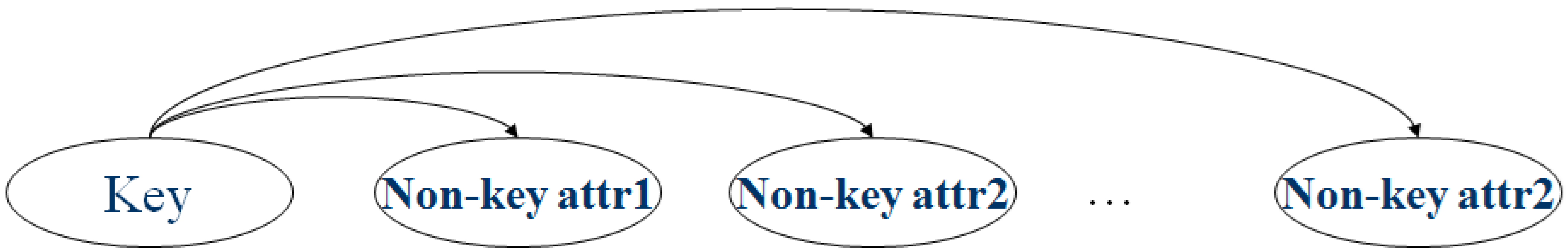
■ *Descompunerea cu joncțiune fără pierderi & cu păstrarea dependențelor a relației R într-o mulțime de relații 3NF este întotdeauna posibilă.*

3NF



A is in KEY

BCNF & 3NF

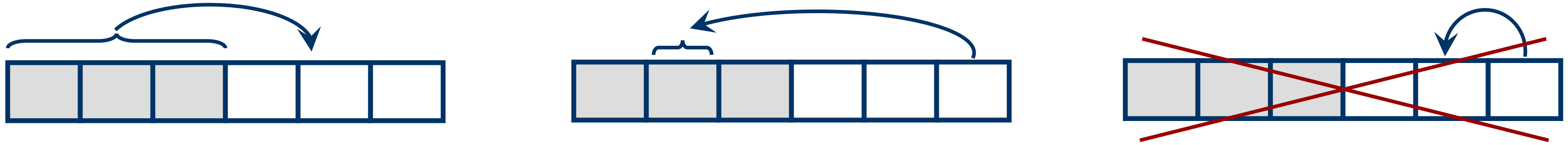


Forme Normale bazate pe DF

1NF - toate valorile atributelor sunt atomice

2NF - toate atributele non-cheie depind de **întreaga** cheie (nu sunt dependențe parțiale)

3NF – tabele în 2NF și toate atributele non-prime depind **doar** de cheie (nu sunt depedențe tranzitive)



BCNF - Toate dependențele sunt date de chei

Normalizarea pe scurt

Fiecare atribut depinde:

de cheie, → definiție cheie

de întreaga cheie, → 2NF

şi de nimic altceva

decât de cheie → BCNF

Normalizarea pe scurt

Fiecare atribut $\overbrace{\text{depinde}}^{\text{neprim}}$:

de cheie, → definiție cheie

de întreaga cheie, → 2NF

şi de nimic altceva

decât de cheie → 3NF

Exemple de nerespectare a FN

2NF - toate atributele neprime trebuie să depindă de **întreaga** cheie

Exam (***Student***, ***Course***, Teacher, Grade)

3NF - toate atributele neprime trebuie să depindă **doar** de cheie

Dissertation(***Student***, Title, Teacher, Department)

BCNF - toate DF sunt implicate de cheile candidat

Schedule (***Day***, ***Route***, ***Bus***, Driver)

"Strategia" de normalizare

BCNF prin descompunere cu joncțiune fără pierderi şi păstrarea dependențelor
(prima alegere)

3NF prin descompunere cu joncțiune fără pierderi şi păstrarea dependențelor
(a doua alegere)

deoarece uneori dependențele nu pot fi păstrate pt a obține BCNF

Descompunerea în BCNF

Fie relația R cu dependențele funcționale F. Dacă $\alpha \rightarrow A$ nu respectă BCNF, descompunem R în

R - A și αA.

Aplicarea repetată a acestei idei va conduce la o colecție de relații care

- sunt în BCNF;
- conduc la joncțiune fără pierderi;
- garantează terminarea.

Descompunerea în BCNF

Exemplu:

R(<u>C</u>, S, J, D, P, Q, V), **C** cheie,

{JP→ C, SD→ P, J→ S}

Alegem SD→ P, decompunând în

(<u>**S**</u>, <u>**D**</u>, P), (<u>C</u>, S, J, D, Q, V).

Apoi alegem J→ S, decompunând (<u>C</u>, S, J, D, Q, V) în

(<u>J</u>, S) şi (<u>C</u>, J, D, Q, V)

În general, mai multe dependențe pot cauza nerespectarea BCNF. Ordinea în care le ``abordăm'' poate conduce la decompuneri de relații complet diferite!

În general, descompunerea în BCNF nu păstrează dependențele.

Exemplu. $\boldsymbol{R}$(C,S,Z), $\{CS \rightarrow Z,\ Z \rightarrow C\}$

Exemplu. $\boldsymbol{R}$(C, S, J, P, D, Q, V) descompus în

(S, D, P), (J, S) şi (C, J, D, Q, V)

nu păstrează dependențele inițiale $\{JP \rightarrow C,\ SD \rightarrow P,\ J \rightarrow S\}$).

! adăugând JPC la mulțimea de relații obținem *descompunere cu păstrarea dependențelor*.

⇨ BCNF & redundanță

Exemplu

1. Fie $\alpha \rightarrow$ A o DF din F ce nu respectă BCNF
2. Descompunem R în $R_1 = \alpha A$ şi $R_2 = R - A$.
3. Dacă R_1 sau R_2 nu sunt în BCNF, descompunerea continuă

(Title, Director, **Cinema**, Phone, **Time**)

(Title, **Cinema**, Phone, **Time**) (**Title**, Director)

(**Cinema**, Phone) (Title, **Cinema, Time**)

Title → Director

Cinema → Phone

Cinema, Time → Title

Descompunerea în 3NF

Evident, procedeul descompunerii din BCNF poate fi utilizat şi pentru descompunerea 3NF.

- Cum asigurăn păstrarea dependențelor?
  - Dacă X→Y nu se păstrează, adăugăm XY.
  - Problema este că XY e posibil să nu respecte 3NF! (pp. că adăugăm CJP pt \`păstrarea' JP→C. Dacă însă are loc şi J→C atunci nu e corect.)

- *Rafinare*: În loc de a utiliza mulțimea inițială F, folosim o *acoperire minimală a lui F*.

Redundanța in DF

- Un atribut $A \in \alpha$ e redundant în DF $\alpha \rightarrow B$ dacă

$$(F - \{\alpha \rightarrow B\}) \cup \{\alpha\text{-}A \rightarrow B\} \equiv F$$

- Pentru a verifica dacă $A \in \alpha$ e redundant în $\alpha \rightarrow B$, calculăm $(\alpha\text{-}A)^+$. Apoi $A \in \alpha$ e redundant în $\alpha \rightarrow B$ dacă $B \in (\alpha\text{-}A)^+$

- *Exerciţiu*: Care sunt atributele redundante în $AB \rightarrow C$ având:

$$\{AB \rightarrow C,\ A \rightarrow B,\ B \rightarrow A\}?$$

Redundanța in DF

- O DF $f \in F$ e redundantă dacă F -{f} e echivalent cu F

- Verificăm că $\alpha \rightarrow A$ e redundantă in F, calculând α^+ pe baza F-{$\alpha \rightarrow A$}. Atunci $\alpha \rightarrow A$ e redundantă în F dacă $A \in \alpha^+$

- *Exercițiu*: Care sunt dependențele funcționale redundante în:

  $\{A \rightarrow C, A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow A\}$?

Acoperire minimală

- O acoperire minimală pentru mulțimea F de dependente functionale este o multime G de dependente functionale pentru care:
  1. Fiecare DF din G e de forma $\alpha \rightarrow A$
  2. Pt fiecare DF $\alpha \rightarrow A$ din G, α nu are atribute redundante
  3. Nu sunt DF redundante in G
  4. G şi F sunt echivalente

Fiecare multime de DF are cel putin o acoperire minimala!

Algoritm de calcul al acoperirii minimale pt F:

1. Folosim descompunerea pentru a obține DF cu 1 atribut în partea dreaptă.
2. Se elimină atributele redundante
3. Se elimină dependențele funcționale redundante

Calcul Acoperire Minimală

Fie $F = \{ABCD \rightarrow E, E \rightarrow D, A \rightarrow B, AC \rightarrow D\}$

Atributele BD din $ABCD \rightarrow E$ sunt redundante:

$\Rightarrow$ $F = \{AC \rightarrow E, E \rightarrow D, A \rightarrow B, AC \rightarrow D\}$

$AC \rightarrow D$ este redundanta

$\Rightarrow$ $F = \{AC \rightarrow E, E \rightarrow D, A \rightarrow B\}$

care este o acoperire minimala

Acoperirile minimale nu sunt unice
(depind de ordinea de alegere a DF/atr. redundante)

Decompunere în 3NF

Input: Schema R with *F* which is a minimal cover
Output: A dependency-preserving, lossless-join 3NF decomposition of R

```
Initialize D = ∅
Apply union rule to combine FDs in F with same
L.H.S. into a single FD

     For each FD α →β  in F do
          Insert the relation schema αβ into D

     Insert δ into D, where δ is some key of R

Remove redundant relation schema from D as follows:
     delete Ri from D if Ri ⊆ Rj, where Rj ∈  D

return D
```

Exemplu

Fie R(A,B,C,D,E) cu dependentele functionale:

$$F = \{ABCD \rightarrow E,\ E \rightarrow D,\ A \rightarrow B,\ AC \rightarrow D\}$$

- Acoperirea minimala a F este $\{AC \rightarrow E,\ E \rightarrow D,\ A \rightarrow B\}$
- Unica cheie: AC
- R nu e in 3NF deoarece $A \rightarrow B$ nu respecta 3NF
- descompunerea 3NF a R:
  - Relatii pentru fiecare DF: R_1(A, C, E), R_2(E,D), si R_3(A,B)
  - Relatie pentru cheia lui R: R_4(A, C)
  - Eliminare relatie redundanta: R_4 (deoarece $R_4 \subseteq R_1$)
  - $\Rightarrow$ descompunerea 3NF este $\{R_1(A,C,E), R_2(E,D), R_3(A,B)\}$

- Descompunerea 3NF nu este unică. Depinde de:
  - Alegerea *acoperirii minimale* sau
  - Alegerea *relatiei redundante care va fi eliminata*

Din nou despre... descompunere

- Descompunerea este ultima solutie de rezolvare a problemelor generate de redundanțe & anomalii

- Excesul poate fi nociv!

Exemplu:

$$R = (Teacher, Dept, Phone, Office)$$

cu DF $F = \{Teacher \rightarrow Dept\ Phone\ Office\}$

R = (Teacher,Dept, Phone, Office)

R_1 = (Teacher,Dept) R_2 = (Teacher, Phone) R_3 = (Teacher, Office)

- Uneori, din motive de performanță se practica de-normalizarea