

Rafinarea structurii bazelor de date

(Dependențe funcționale)

MovieList

<i>Title</i>	<i>Director</i>	<i>Cinema</i>	<i>Phone</i>	<i>Time</i>
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

<i>Cinema</i>	<i>Time</i>	<i>Title</i>
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

<i>Title</i>	<i>Director</i>
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

Cinema

<i>Cinema</i>	<i>Phone</i>
Florin Piersic	441111
Victoria	442222

$$\alpha \rightarrow \beta$$

Descompunerea relațiilor

Descompunerea unei relații R

este o mulțime de (sub)relații

$$\{R_1, R_2, \dots, R_n\}$$

astfel încât fiecare $R_i \subseteq R$ și $R = \cup R_i$

Dacă r este o instanță din R ,
atunci r se descompune în

$$\{r_1, r_2, \dots, r_n\},$$

unde fiecare $r_i = \pi_{R_i}(r)$

Proprietățile descompunerii relațiilor

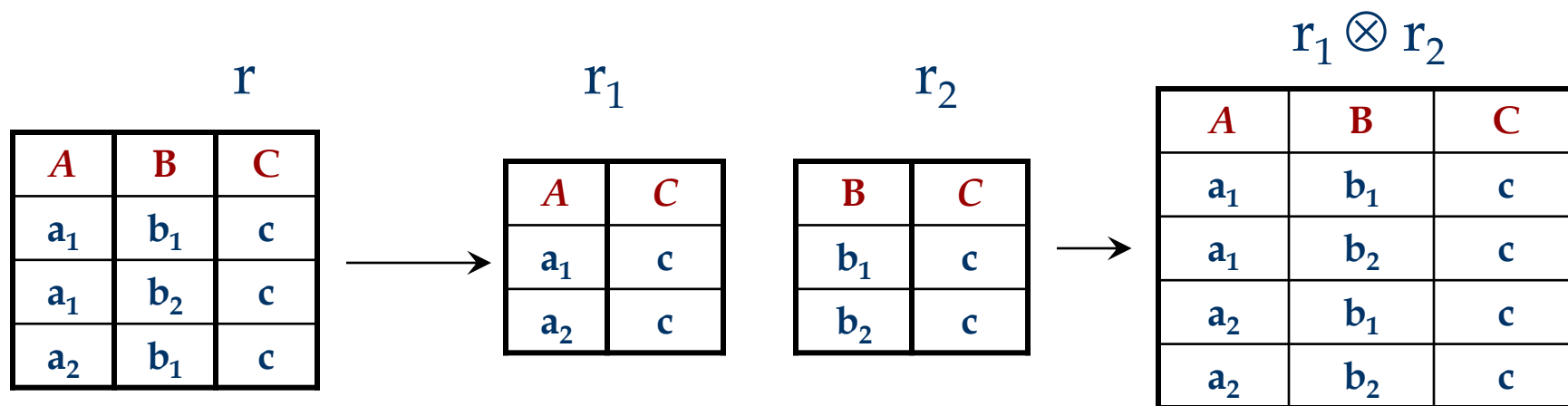
1. Descompunerea trebuie să **păstreze informațiile** (să fie cu joncțiuni fără pierderi)

- Datele din relația originală \equiv Datele din relațiile descompunerii
- Crucial pentru păstrarea consistenței datelor!

2. Descompunerea trebuie să **respecte toate DF**

- Dependențele funcționale din relația originală \equiv reuniunea dependențelor funcționale din relațiile descompunerii
- Facilitează verificarea violărilor DF

Fie descompunere lui $R(A,B,C)$
in $\{R_1(AC), R_2(BC)\}$



- Deoarece $r \subset r_1 \otimes r_2$, descompunerea **nu**
este cu joncțiuni fără pierderi
(lossy decomposition)

Întrebarea 1

*Cum determinăm dacă $\{R_1, R_2\}$
este o descompunere cu joncțiuni fără
pierderi a lui R ?*

Întrebarea 2

*Cum descompunem R în $\{R_1, R_2\}$ astfel
încât aceasta e
cu joncțiuni fără pierderi?*

Întrebarea 1

*Cum determinăm dacă $\{R_1, R_2\}$
este o descompunere cu joncțiuni fără
pierderi a lui R ?*

Teorema: Descompunerea lui R (cu mulțimea F de DF) în $\{R_1, R_2\}$ este cu joncțiuni fără pierderi cu respectarea mulțimii F dacă și numai dacă :

$$F \Rightarrow R_1 \cap R_2 \rightarrow R_1$$

sau

$$F \Rightarrow R_1 \cap R_2 \rightarrow R_2$$

Întrebarea 2

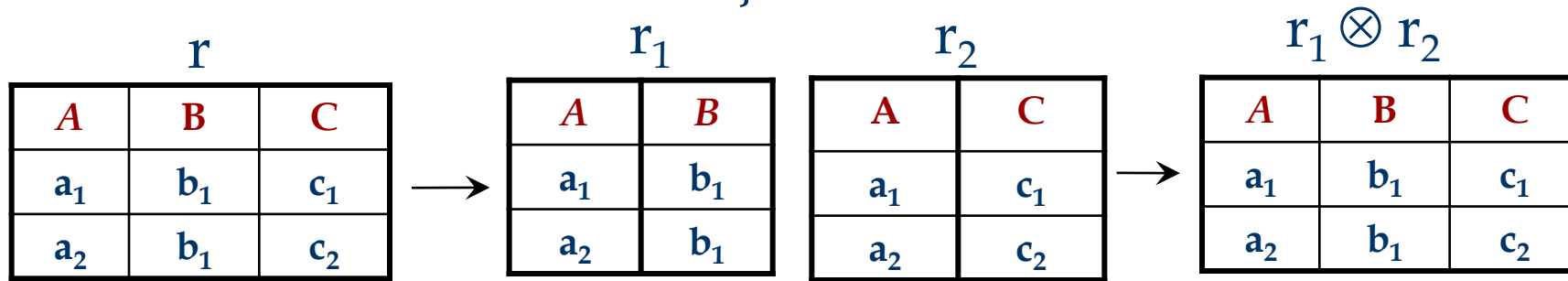
Cum descompunem R în $\{R_1, R_2\}$ astfel încât aceasta e cu joncțiuni fără pierderi?

Corolar: Dacă $\alpha \rightarrow \beta$ este satisfăcută pe R și $\alpha \cap \beta = \emptyset$, atunci descompunerea lui R în $\{R - \beta, \alpha\beta\}$ este o descompunere cu joncțiuni fără pierderi.

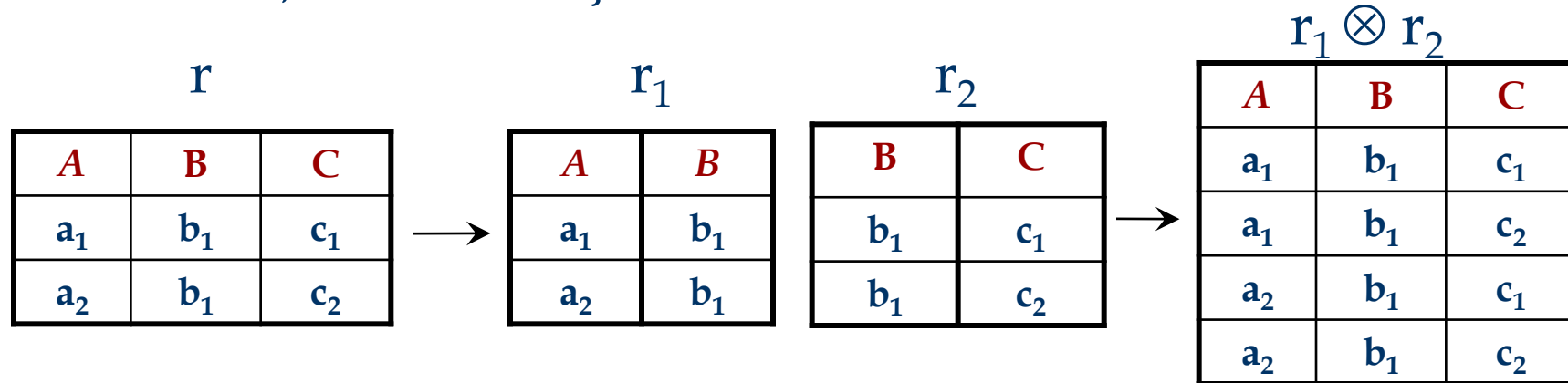
Exemplu

- Fie $R(A,B,C)$ cu mulțimea de dependențe funcționale $F = \{ A \rightarrow B \}$
- Descompunerea $\{AB, AC\}$ e cu joncțiuni fără pierderi deoarece

$$AB \cap AC = A \quad \text{și} \quad A \rightarrow AB$$



- Descompunerea $\{AB, BC\}$ nu este cu joncțiuni fără pierderi în raport cu F deoarece $AB \cap BC = B$, iar $B \rightarrow AB$ și $B \rightarrow BC$ nu sunt satisfăcute de R



Teoremă

Dacă

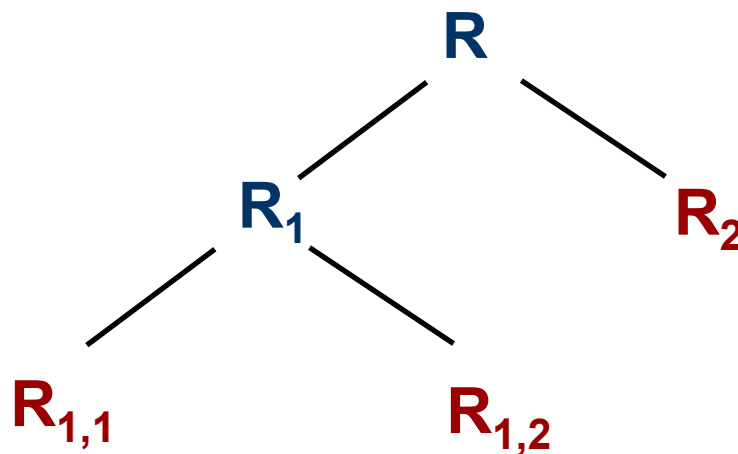
$\{\mathbf{R}_1, \mathbf{R}_2\}$ este o descompunere cu joncțiuni fără pierderi a lui \mathbf{R} ,

și dacă

$\{\mathbf{R}_{1,1}, \mathbf{R}_{1,2}\}$ e o descompunere cu joncțiuni fără pierderi a lui \mathbf{R}_1 ,

atunci

$\{\mathbf{R}_{1,1}, \mathbf{R}_{1,2}, \mathbf{R}_2\}$ e o descompunere cu joncțiuni fără pierderi a \mathbf{R} :



MovieList

<i>Title</i>	<i>Director</i>	<i>Cinema</i>	<i>Phone</i>	<i>Time</i>
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
War Horse	Spielberg	Victoria	442222	14:00
The Lord of the Rings 3	Jackson	Victoria	442222	16:30

Cinema-Screens

<i>Cinema</i>	<i>Phone</i>	<i>Time</i>	<i>Title</i>
F. Piersic	441111	11:30	The Hobbit
F. Piersic	441111	14:30	The Lord of the Rings 3
Victoria	442222	11:30	Adventures of Tintin
Victoria	442222	14:00	War Horse
Victoria	442222	16:30	The Lord of the Rings 3

Movie

<i>Title</i>	<i>Director</i>
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

Movie

<i>Title</i>	<i>Director</i>
The Hobbit	Jackson
The Lord of the Rings 3	Jackson
Adventures of Tintin	Spielberg
War Horse	Spielberg

Cinema-Screens

<i>Cinema</i>	<i>Phone</i>	<i>Time</i>	<i>Title</i>
F. Piersic	441111	11:30	The Hobbit
F. Piersic	441111	14:30	The Lord of the Rings 3
Victoria	442222	11:30	Adventures of Tintin
Victoria	442222	14:00	War Horse
Victoria	442222	16:30	The Lord of the Rings 3

Screens

<i>Cinema</i>	<i>Time</i>	<i>Title</i>
F. Piersic	11:30	The Hobbit
F. Piersic	14:30	Saving Private Ryan
Victoria	11:30	Adventures of Tintin
Victoria	14:00	War Horse
Victoria	16:30	Saving Private Ryan

Cinema

<i>Cinema</i>	<i>Phone</i>
F. Piersic	441111
Victoria	442222

Proprietățile descompunerii relațiilor

1. Descompunerea trebuie să **păstreze informațiile**

- Datele din relația originală \equiv Data din relațiile descompunerii
- Crucial for păstrarea consistenței datelor!

2. Descompunerea trebuie să **respecte toate DF**

- Dependențele funcționale din relația originală \equiv 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 attribute din α , adică:

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

- Algoritm pentru determinare proiecției DF:

Input: α, F

Output: F_α

result = \emptyset ;

for each $\beta \subseteq \alpha$ do

$T = \beta^+$ (w.r.t. F)

 result = result $\cup \{ \beta \rightarrow T \cap \alpha \}$

return result

Complexitatea
e exponențială

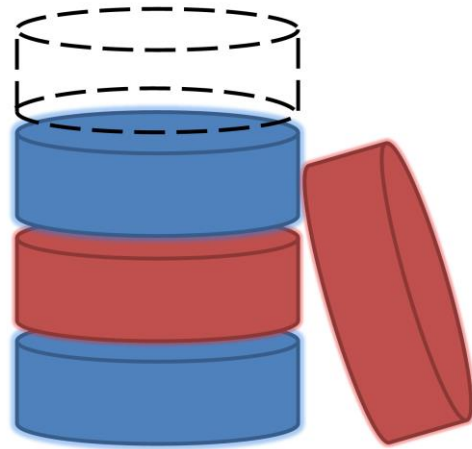
Descompunere cu păstrarea dependențelor

Descompunerea $\{R_1, R_2, \dots, R_n\}$ a relației R e cu **păstrarea dependențelor** dacă $(F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n})$ și F sunt echivalente, adică:

$$(F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n}) \Rightarrow F \text{ și}$$

$$F \Rightarrow (F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n})$$

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 și sugerează posibile îmbunătățiri
- Fie relația R cu 3 attribute, ABC.
 - **Nici o DF:** nu avem redundanțe.
 - **Pentru $A \rightarrow B$:** Mai multe înregistrări pot avea aceeași 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 descompunere 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 → ne ajută să decidem dacă 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 și mulțimile sunt excluse)

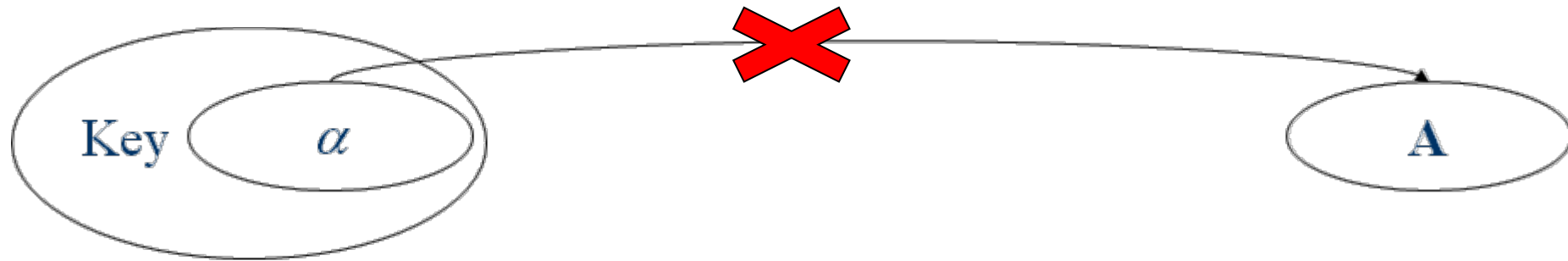
(această condiție este implicită conform definiției modelului relațional)

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 și 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^+

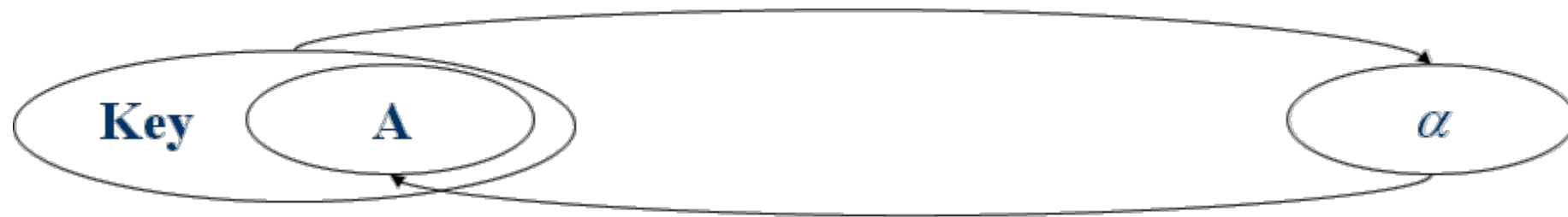
- $A \in \alpha$ (DF *trivială*), sau
- α conține o cheie de-a lui R, sau
- A este un atribut prim.

■ Dacă R este în BCNF, evident este și 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.

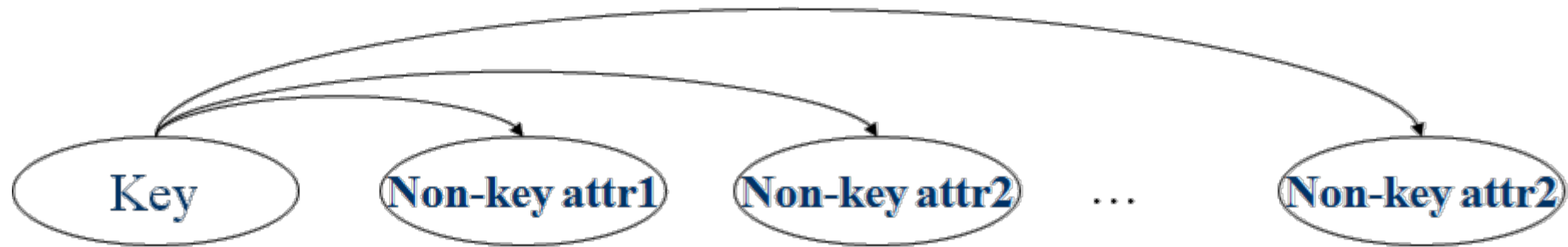
■ Descompunerea *cu joncțiune fără pierderi & cu păstrarea dependențelor* a relației R într-o mulțime de relații 3NF este întotdeauna posibilă.

3NF



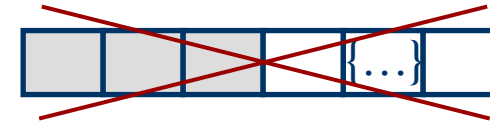
A is in KEY

BCNF & 3NF

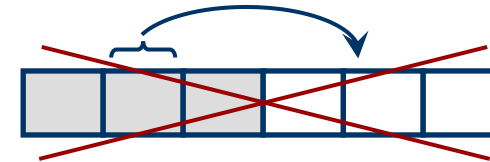
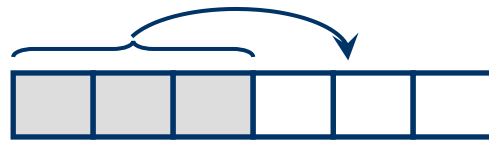


Forme Normale bazate pe DF

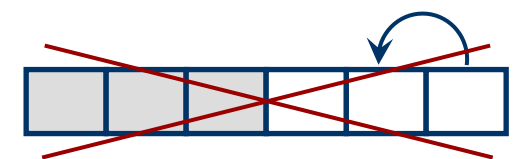
1NF - toate valorile atributelor sunt atomice



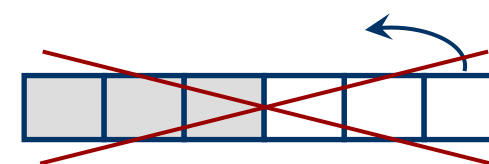
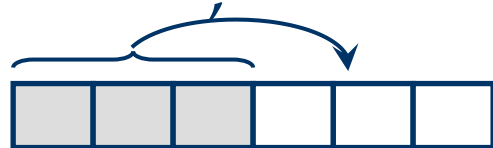
2NF - toate attributele non-cheie depind de **întreaga** cheie (nu sunt dependențe parțiale)



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



BCNF - Toate dependențele sunt date de chei



Normalizarea pe scurt

Fiecare atribut depinde:

de cheie,► definiție cheie

de întreaga cheie,► 2NF

și de nimic altceva

decât de cheie► BCNF

Normalizarea pe scurt

Fiecare atribut ^{neprim} depinde:

- de cheie,► definiție cheie
- de întreaga cheie,► 2NF
- și de nimic altceva
decât de cheie► 3NF

Exemple de nerespectare a FN

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

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



3NF - toate attributele 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 jonctiune fără pierderi și păstrarea dependențelor
(prima alegere)

3NF prin descompunere cu jonctiune fără pierderi și 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:

$R(\underline{C}, S, J, D, P, Q, V)$, **C** cheie,

$\{JP \rightarrow C, SD \rightarrow P, J \rightarrow S\}$

Alegem $SD \rightarrow P$, decompunând în

$(\underline{S}, \underline{D}, P), (\underline{C}, S, J, D, Q, V)$.

Apoi alegem $J \rightarrow S$, decompunând $(\underline{C}, S, J, D, Q, V)$ în

(\underline{J}, S) și $(\underline{C}, J, D, Q, V)$

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

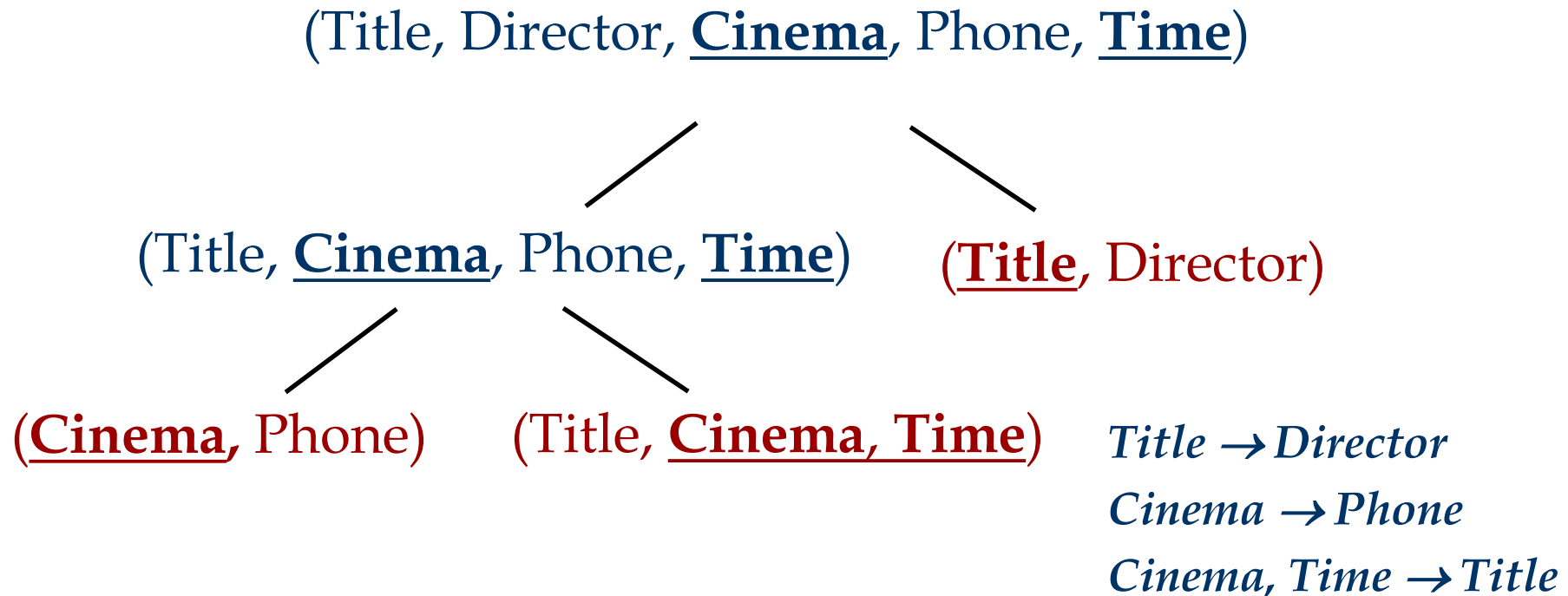
Exemplu. $R(C, S, J, P, D, Q, V)$ descompus în
(S, D, P), (J, S) și (C, J, D, Q, V)
nu păstrează dependențele inițiale $\{JP \rightarrow C, SD \rightarrow P, J \rightarrow S\}$).

! 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 $R_1 = \alpha A$ și $R_2 = R - A$.
3. Dacă R_1 sau R_2 nu sunt în BCNF, descompunerea continuă



Descompunerea în 3NF

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

- Cum asigurăm păstrarea dependențelor?
 - Dacă $X \rightarrow Y$ nu se păstrează, adăugăm XY.
 - Problema este că XY e posibil să nu respecte 3NF! (pp. că adăugăm CJP pt `păstrarea' $JP \rightarrow C$. Dacă însă are loc și $J \rightarrow C$ atunci nu e corect.)
- *Rafinare:* În loc de a utiliza mulțimea inițială F, folosim o *acoperire minimală a lui F*.

Redundanța în DF

- Un atribut $A \in \alpha$ e redundant în DF $\alpha \rightarrow B$ dacă
$$(F - \{\alpha \rightarrow B\}) \cup \{\alpha - A \rightarrow B\} \equiv F$$
- Pentru a verifica dacă $A \in \alpha$ e redundant în $\alpha \rightarrow B$, calculăm $(\alpha - A)^+$. Apoi $A \in \alpha$ e redundant în $\alpha \rightarrow B$ dacă $B \in (\alpha - A)^+$
- *Exercițiu:* Care sunt attributele redundante în $AB \rightarrow C$ având:
$$\{AB \rightarrow C, A \rightarrow B, B \rightarrow A\}?$$

Redundanța în DF

- O DF $f \in F$ e **redundantă** dacă $F - \{f\}$ e echivalent cu F
- Verificăm că $\alpha \rightarrow A$ e redundantă în 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 attribute redundante
3. Nu sunt DF redundante in **G**
4. **G** și **F** sunt echivalente

Fiecare multime de DF are cel puțin o acoperire minimală!

Algoritm de calcul al acoperirii minime pt F:

1. Folosim descompunerea pentru a obține DF cu 1 atribut în partea dreaptă.
2. Se elimină attributele 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 $D = \emptyset$

Apply *union rule* to combine FDs in F with same L.H.S. into a single FD

For each FD $\alpha \rightarrow \beta$ in F do

 Insert the relation schema $\alpha\beta$ into D

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

Remove redundant relation schema from D as follows:

 delete R_i from D if $R_i \subseteq R_j$, where $R_j \in D$

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 minime 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:

$R = (\text{Teacher}, \text{Dept}, \text{Phone}, \text{Office})$

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

$R = (\text{Teacher}, \text{Dept}, \text{Phone}, \text{Office})$



$R_1 = (\text{Teacher}, \text{Dept}) \quad R_2 = (\text{Teacher}, \text{Phone}) \quad R_3 = (\text{Teacher}, \text{Office})$

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