3

Proiectarea bazelor de date
Forme normale

Structura bazei de date

=

Structura relațiilor

+

Constrângeri

Exemplu: relația *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

Constrângeri:

- Fiecare film are un regizor
- Fiecare cinematograf are un număr de telefon
- Fiecare cinematograf începe proiecția unui singur film al un moment dat

Proiectare defectuoasă!

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

Anomalie de inserare

Anomalie de ştergere

Anomalie de actualizare

Rafinarea unei structuri defectuoase prin *descompunerea* în mai multe structuri “bune”

Movies

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

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

Cinema

Cinema	*Phone*
Florin Piersic	441111
Victoria	442222

- ✓ Anomalie de inserare
- ✓ Anomalie de ştergere
- ✓ Anomalie de actualizare

Cum determinăm

dacă o structură este

“*bună*” sau ”*defectuoasă*”?

Cum transformăm

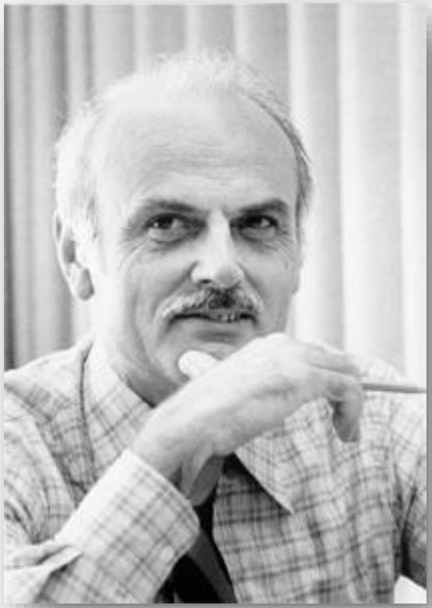
o structură *defectuoasă*

într-una *bună*?

Teoria *dependențelor funcționale* furnizează o abordare sistematică a celor două întrebări

Introdusă de

Edgar Frank Codd în:



"A relational model for large shared data banks", Com. of the ACM, 13(6), 1970, pp.377-387.

Dependențe funcționale

$$\alpha \rightarrow \beta$$

α, β sunt submulțimi de atribute ale R

"**α** determină funcțional **β**"

sau

"**β** depinde functional de **α**"

Definiție dependențe funcționale

Dependența funcțională $\alpha \rightarrow \beta$ este satisfăcută de R dacă şi numai dacă

pentru *orice* instanță a lui R,

oricare două tupluri t_1 şi t_2 pentru care valorile lui α sunt identice

vor avea de asemenea valori identice pentru β.

O dependență funcțională

$$\alpha \rightarrow \beta$$

este **trivială** dacă

$$a \supseteq \beta.$$

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

Dependențe funcționale pentru relația *MovieList*:

1. Title → Director
2. Cinema → Phone
3. Cinema, Time → Title

Fie r instanța unei relații R

- Spunem că r **satisface DF** $\alpha \to \beta$ dacă
  pentru orice pereche de tupluri
  t_1 şi t_2 din r astfel încât $\pi_\alpha(t_1) = \pi_\alpha(t_2)$,
  este de asemenea adevărat că $\pi_\beta(t_1) = \pi_\beta(t_2)$.

sau

$$\forall\ t_1, t_2 \in r$$

$$\pi_\alpha(t_1) = \pi_\alpha(t_2) \Rightarrow \pi_\beta(t_1) = \pi_\beta(t_2)\ *$$

* $\pi_\alpha(t)$ este proiecția atributelor α pentru tuplul t

Fie *r* instanța unei relații *R*

- o **DF** *f* **este satisfăcută pe** *R* dacă şi numai dacă orice instanță *r* a lui *R* satisface *f*

- *r* **nu respectă** o DF *f* dacă *r* nu satisface *f*.

- *r* este o **instanță legală a lui R** dacă *r* satisface toate dependențele funcționale definite pentru *R*.

Exemplu: ***Movie***(*Title, Director, Composer*)

Title	*Director*	*Composer*
Schindler's List	Spielberg	Williams
Saving Private Ryan	Spielberg	Williams
North by Northwest	Hitchcock	Herrmann
Angela's Ashes	Parker	Williams
Vertigo	Hitchcock	Herrmann

- DF *composer* → *director* nu este respectată de relația ***Movie***
- *r* satisface DF *director* → *composer*

Acest lucru nu înseamnă că *director*→*composer* e respectat de ***Movie***!

Problema implicației

Putem deduce că o DF f e respectată de R pe baza unei mulțimi de DF F ?

Exemplu: în ***MovieList***, avem

$$F = \{ \text{Title} \rightarrow \text{Director}$$
$$\text{Cinema} \rightarrow \text{Phone}$$
$$\text{Cinema, Time} \rightarrow \text{Title} \}$$

- *Time* → *Director* este respectată?
- Dar *Cinema, Time* → *Director*?

F **implică logic** pe f

notat prin

$$\mathbf{F} \Rightarrow \mathbf{f}$$

daca fiecare instanță r a relației R

ce satisface F

satisfice şi f

F & G : mulțimi de dependențe funcționale

f : dependeța funcțională

F implică logic G

notat prin

$$\boldsymbol{F \Rightarrow G}$$

dacă $F \Rightarrow g$ pentru fiecare $g \in G$

Închiderea lui **F**

(notată prin F^+)

este mulţimea tuturor DF implicate de F

$$F^+ = \{\, f \mid F \Rightarrow f \,\}$$

F şi G sunt **echivalente**

(notat prin $F \equiv G$)

dacă

$$F^+ = G^+$$

(adică $F \Rightarrow G$ şi $G \Rightarrow F$)

Axiomele lui Armstrong

Fie $\alpha, \beta, \gamma \subseteq R$

Reflexivitate: Dacă $\beta \subseteq \alpha$, atunci $\alpha \rightarrow \beta$

Augmentare: Dacă $\alpha \rightarrow \beta$, atunci $\alpha\gamma \rightarrow \beta\gamma$

Tranzitivitate: Dacă $\alpha \rightarrow \beta$ şi $\beta \rightarrow \gamma$, atunci $\alpha \rightarrow \gamma$

Sistemul axiomelor lui Armstrong este

Corect

(Orice FD derivată este implicată de F)

&

Complet

(Toate DF din F^+ pot fi derivate)

Exemplu: Fie R(A, B, C, D, E) cu mulțimea

$$F = \{A \rightarrow C; B \rightarrow C; CD \rightarrow E\}.$$

Arătați că $F \Rightarrow AD \rightarrow E$

Soluție:

1. $A \rightarrow C$ (dat)
2. $AD \rightarrow CD$ (augumentare cu (1))
3. $CD \rightarrow E$ (dat)
4. $AD \rightarrow E$ (tranzitivitate cu (2) si (3))

Reguli de inferență adiționale

Reuniunea:

Dacă $\alpha \rightarrow \beta$ şi $\alpha \rightarrow \gamma$, atunci $\alpha \rightarrow \beta\gamma$

Descompunerea:

Dacă $\alpha \rightarrow \beta$, atunci $\alpha \rightarrow \beta'$ pentru orice $\beta' \subseteq \beta$

Exemplu: Aratati ca $\{A \to BCD\} \equiv \{A \to B; A \to C; A \to D\}$

Fie $F = \{A \to BCD\}$

Fie $G = \{A \to B; A \to C; A \to D\}$

Prin regula de descompunere avem

$$F \Rightarrow A \to B,$$

$$F \Rightarrow A \to C, \text{ si}$$

$$F \Rightarrow A \to D$$

Prin urmare $F \Rightarrow G$

Din regula reuniunii avem

$$\{A \to B; A \to C\} \Rightarrow A \to BC \text{ si}$$

$$\{A \to BC; A \to D\} \Rightarrow A \to BCD$$

Prin urmare $G \Rightarrow F$, deci $F \equiv G$

Superchei, chei & atribute prime

- O mulțime de atribute α reprezintă o supercheie a relației R (având mulțimea de DF F) dacă

$$F \Rightarrow \alpha \rightarrow R.$$

- O mulțime de atribute α e o cheie a relației R dacă

    (1) α este o supercheie, şi

    (2) nici o submulțime a lui α nu e supercheie

    (adică, pentru fiecare $\beta \subset \alpha$, $\beta \rightarrow R \notin F^+$)

- Un atribut $A \in R$ se numeşte atribut prim dacă A face parte dintr-o cheie a lui R; în caz contrar, A se numeşte atribut neprim.

- Considerăm din nou relația

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

cu DF

(1) Cinema, Time $\rightarrow$ Title

(2) Cinema $\rightarrow$ Phone

(3) Title $\rightarrow$ Director

- {*Cinema, Time*} este singura cheie a relației ***MovieList***.
- *Cinema* şi *Time* sunt singurele atribute prime din ***MovieList***.
- Orice mulțime ce include {*Cinema; Time*} e supercheie a ***MovieList***.

Închiderea atributelor

Fie $\alpha \subseteq R$ şi F o mulţime de DF satisfăcute pe R

- Închiderea lui α (cu respectarea mulțimii F de DF), notată cu α^+, este mulțimea de atribute ce sunt determinate funcțional din α pe baza dependențelor funcționale din F; adică

$$\alpha^+ = \{A \in R \mid F \Rightarrow \alpha \rightarrow A\}$$

- Se observă că $F \Rightarrow \alpha \rightarrow \beta$ dacă şi numai dacă $\beta \subseteq \alpha^+$(cu respectarea DF din F)

Algoritm pt deteminarea închiderii atributelor

Input: α, F

Output: α^+ (w.r.t. F)

Compute a sequence of sets of attrs α_0, α_1,... α_k, α_{k+1} as follows:

$$\alpha_0 = \alpha$$

$$\alpha_{i+1} = \alpha_i \cup \gamma \text{ such that there is some FD } \beta \rightarrow \gamma \in F \text{ and } \beta \subseteq \alpha_i$$

Terminate the computation once

$$\alpha_{k+1} = \alpha_k \text{ for some } k$$

Return α_k

```
Input: α, F
Output: α⁺ (w.r.t. F)
Compute a sequence of sets of attrs α_0, α_1,... α_k, α_{k+1}
as follows:
      α_0 = α
      α_{i+1} = α_i ∪ γ such that there is some FD
            β→γ ∈ F and β ⊆ α_i
Terminate the computation once α_{k+1} = α_k for some k
Return α_k
```

Exemple: Fie $F = \{A \to C; B \to C; CD \to E\}$, aratati ca $F \Rightarrow AD \to E$

i	α_i	*FD folosit*
0	AD	dat
1	ACD	$A \to C$
2	ACDE	$CD \to E$
3	ACDE	-

Deci $AD^+ = ACDE$. Deoarece $E \in AD^+$, rezulta ca $F \Rightarrow AD \to E$

Descompunerea relațiilor

Descompunerea unei relații R

este o mulțime de (sub)relații

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

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

Dacă *r* este o instanță din R,

atunci *r* se descompune în

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

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

Descompunerea relațiilor

$$\{ M_1 = (\text{Cinema, Time})$$
$$M_2 = (\text{Time, Title}),$$
$$M_3 = (\text{Title, Director}),$$
$$M_4 = (\text{Cinema, Phone})\}$$

e o descompunere a:

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

Proprietățile descompunerii relațiilor

1. Descompunerea trebuie să păstreze informațiile
   - Datele din relația originală ≡ Datele din relațiile descompunerii
   - Crucial pentru 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

1. Descompunerea trebuie să păstreze informațiile

Cu alte cuvinte:

putem reconstrui *r*

prin jonctiunea proiectiilor sale

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

Observatie: daca $\{R_1, R_2, \dots, R_n\}$ e o descompunere a R, atunci pentru orice instanta *r* din R, avem

$$r \subseteq \pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \dots \otimes \pi_{Rn}(r)$$

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

M1

Cinema	*Time*
Florin Piersic	11:30
Florin Piersic	14:30
Victoria	11:30
Victoria	14:00
Victoria	16:30

M2

Time	*Title*
11:30	The Hobbit
14:30	The Lord of the Rings 3
11:30	Adventures of Tintin
14:00	The Lord of the Rings 3
16:30	War Horse

M3

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

M4

Cinema	*Phone*
Florin Piersic	441111
Victoria	442222

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

Descompunere cu joncțiune fără pierderi
(Lossless - Join Decomposition)

O descompunere a R (având DF *F*) în

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

este o

descompunere cu joncțiuni fără pierderi
cu respectarea mulțimii *F*

dacă

$$\pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r) = r$$

pentru orice instanță r din R ce satisface F.

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

r

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c

→

r_1

A	*C*
a_1	c
a_2	c

r_2

B	*C*
b_1	c
b_2	c

→

$r_1 \otimes r_2$

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c
a_2	b_2	c

- 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 joncţiuni fără pierderi a lui R?

Întrebarea 2

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

Întrebarea 1

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

Teorema: Descompunerea lui R (cu mulţimea F de DF) în $\{R_1, R_2\}$ este cu joncţiuni fără pierderi cu respectarea mulţimii F dacă şi 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 joncţiuni fără pierderi?

Corolar: Dacă $\alpha \rightarrow \beta$ este satisfăcută pe R şi $\alpha \cap \beta = \emptyset$, atunci descompunerea lui R în $\{R- \beta , \alpha\beta\}$ este o descompunere cu joncţiuni 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

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

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

→

r_1

A	*B*
a_1	b_1
a_2	b_1

r_2

A	C
a_1	c_1
a_2	c_2

→

$r_1 \otimes r_2$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

- Descompunerea{AB, BC} **nu** e cu joncțiuni fără pierderi F deoarece $AB \cap BC = B$ şi nici una dintre dependențele $B \rightarrow AB$ şi $B \rightarrow BC$ nu sunt respectate pe R

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

→

r_1

A	*B*
a_1	b_1
a_2	b_1

r_2

B	C
b_1	c_1
b_1	c_2

→

$r_1 \otimes r_2$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

Teorema

Dacă

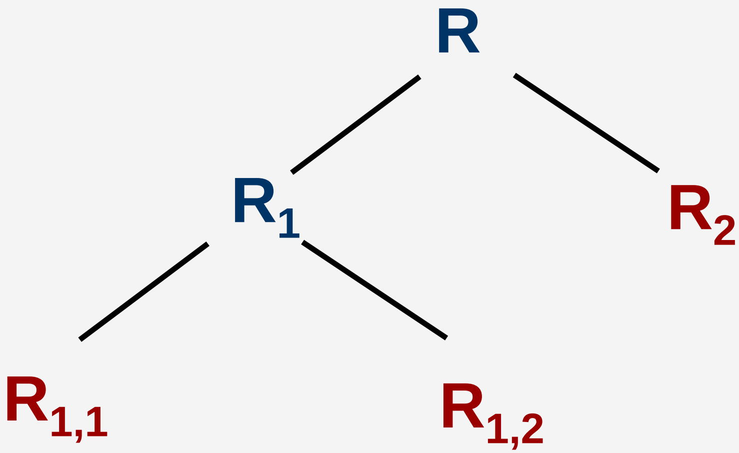
$\{R_1, R_2\}$ este o descompunere cu joncțiuni fără pierderi a lui R,

și dacă

$\{R_{1,1}, R_{1,2}\}$ e o descompunere cu joncțiuni fără pierderi a lui R_1,

atunci

$\{R_{1,1}, R_{1,2}, R_2\}$ e o descompunere cu joncțiuni fără pierderi a R:



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
War Horse	Spielberg	Victoria	442222	14:00
The Lord of the Rings 3	Jackson	Victoria	442222	16:30

Movie

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

Cinema-Screens

Cinema	*Phone*	*Time*	*Title*
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

Cinema-Screens

Cinema	*Phone*	*Time*	*Title*
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

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

Cinema

Cinema	*Phone*
F. Piersic	441111
Victoria	442222

Screens

Cinema	*Time*	*Title*
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

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: 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 apărea 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 dor valori atomice (deci listele şi mulțimile sunt excluse)

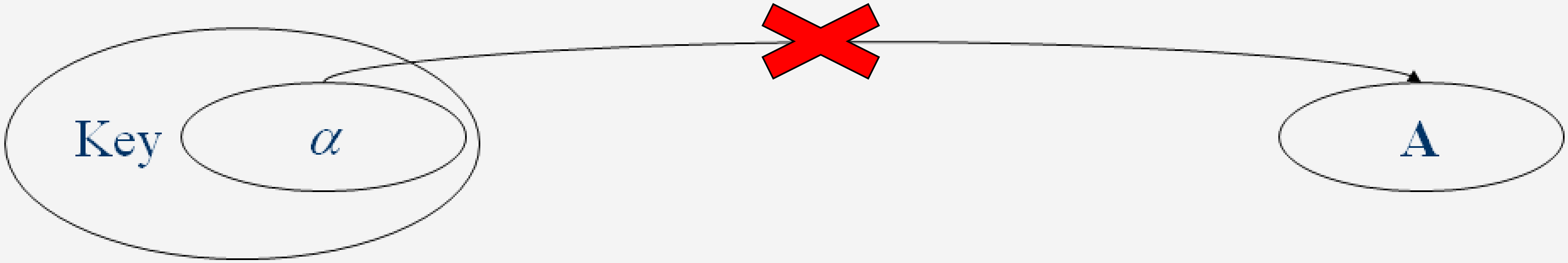
(această condiție este implicită conform definiției modelului relațional)

2NF

Spunem că avem o *dependență funcțională parțială* într-o relație atunci când un atribut *ne-cheie* este dependent funcțional de o parte a cheii primare 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)

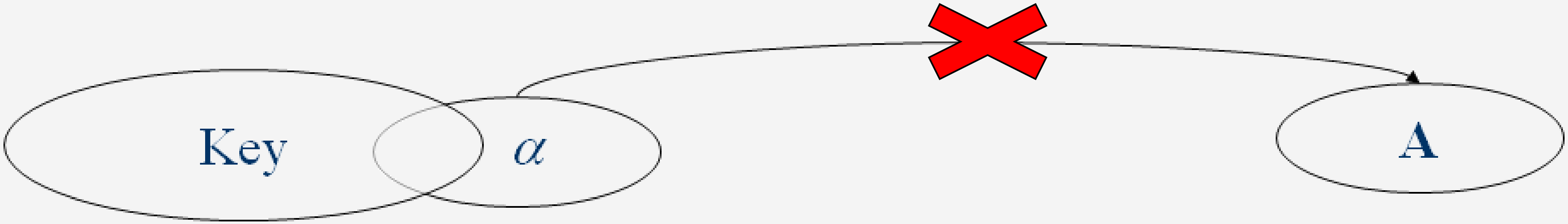
BCNF

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

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

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, pentru toate $\alpha \to A$ din F^+

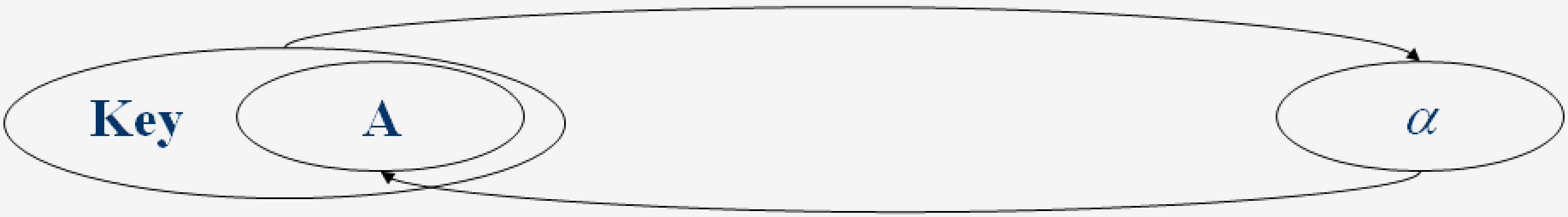
- $A \in \alpha$ (DF *trivială*), sau
- α este o supercheie pentru R, sau
- A este un atribut prim.

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

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

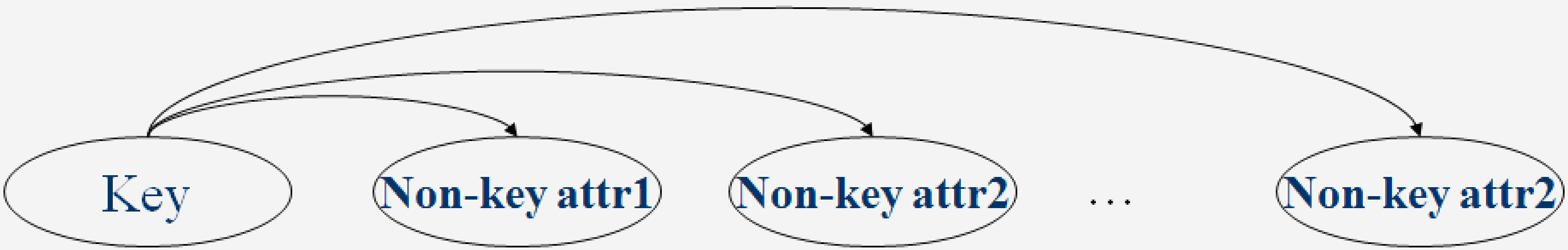
■ *Descompunerea cu joncțiune fără pierderi & cu păstrarea dependențelor a relației R într-o mulțime de relații 3NF este întotdeauna posibilă.*

3NF



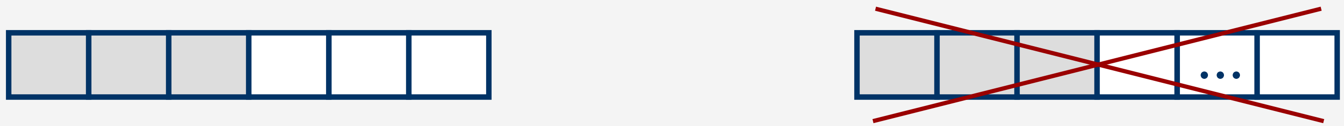
A is in KEY

BCNF & 3NF

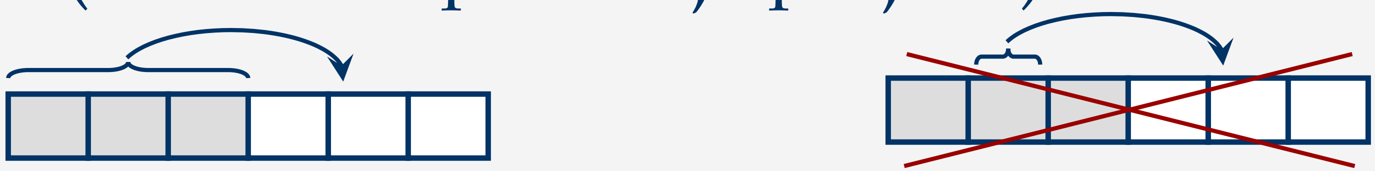


Forme Normale bazate pe DF

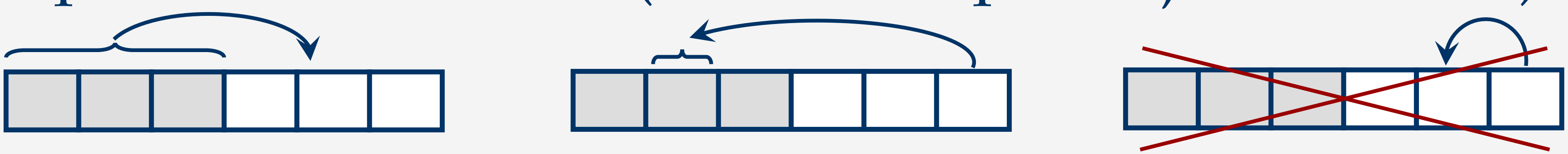
1NF - toate valorile atributelor sunt atomice



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



3NF – tabele în 2NF şi 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

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 <u>joncțiune fără pierderi</u> şi <u>păstrarea dependențelor</u> (prima alegere)

3NF prin descompunere cu <u>joncțiune fără pierderi</u> şi <u>păstrarea dependențelor</u> (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

$R - A$ şi 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).

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

Exemplu. ***R***(C, S, J, P, D, Q, V) 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.

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

- 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 $B \in (\alpha\text{-}A)^+$

- *Exercițiu*: Care sunt atributele redundante in $AB \rightarrow C$ 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*

BCNF vs 3NF

- BCNF: joncțiune fără pierderi (posibil să nu păstreze dependențele)
- 3NF: joncțiune fără pierderi & păstrare dependențe

R(Course, Teacher, Time) cu DF

$\{Course \rightarrow Teacher;\ Teacher, Time \rightarrow Course\}$

- Chei: {Course, Time} şi { Teacher, Time }
- R este în 3NF dar nu în BCNF

- descompunere BCNF $\{R_1$(Course, Teacher), R_2(Course, Time)} este (doar) cu joncțiune fără pierderi

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

Dependențe multivaloare

course	teacher	book
alg101	Green	Alg Basics
alg101	Green	Alg Theory
alg101	Brown	Alg Basics
alg101	Brown	Alg Theory
logic203	Green	Logic B.
logic203	Green	Logic F.
logic203	Green	Logic intro.

relația e în BCNF

Dependențe multivaloare

	X	Y	Z
t_1	a	b_1	c_1
t_2	a	b_2	c_2
t_3	a	b_1	c_2
t_4	a	b_2	c_1

$\forall t_1, t_2 \in r$ şi $\pi_x(t_1) = \pi_x(t_2) \Rightarrow$
$\exists\, t_3 \in r$ astfel încât
$\pi_{XY}(t_1) = \pi_{XY}(t_3)$,
$\pi_z(t_2) = \pi_z(t_3)$

Reguli adiționale:

- Complementare: $X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow R - XY$
- Augumentare: $X \rightarrow\rightarrow Y,\ Z \subseteq W \Rightarrow WX \rightarrow\rightarrow YZ$
- Tranzitivitate: $X \rightarrow\rightarrow Y,\ Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow Z - Y$
- Replicare: $X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$
- Fuzionare: $X \rightarrow\rightarrow Y,\ W \cap Y = \emptyset,\ W \rightarrow Z,\ Z \subseteq Y \Rightarrow X \rightarrow Z$

A patra formă normală (4NF)

Definiție. Fie R o schemă relațională şi F o mulțime de dependențe funcționale şi multivaloare pe R.

Spunem că R este în a patra forma normală NF4 dacă, pentru orice dependență multivaloare $X \rightarrow\rightarrow Y$:

- $Y \subseteq X$ sau
- $XY = R$ sau
- X e super-cheie

A patra formă normală (4NF)

course	teacher	book
alg101	Green	Alg Basics
alg101	Green	Alg Theory
alg101	Brown	Alg Basics
alg101	Brown	Alg Theory
logic203	Green	Logic B.
logic203	Green	Logic F.
logic203	Green	Logic intro.

course→→teacher

Relatia se poate descompune in:
(Course,Teacher)
si (Course,Book)

course	teacher
alg101	Green
alg101	Brown
logic203	Green

course	book
alg101	Alg Basics
alg101	Alg Theory
logic203	Logic B.
logic203	Logic F.
logic203	Logic intro.

Dependența *Join*

- Spunem ca R satisface dependența *join* $\otimes\{R_1,\ldots,R_n\}$ dacă $R_1, R_2,\ldots,R_n$ este o descompunere cu joncțiuni fără pierderi a lui R.

O dependență multivaloare $X \rightarrow\rightarrow Y$ poate fi exprimată ca o dependență join: $\otimes\{XY, X(R-Y)\}$.

A cincea formă normală (5NF)

O relație R este în NF5 dacă şi numai dacă pentru orice dependență *join* a lui R:

- R_i=R pentru un *i* oarecare, sau
- dependența este implicată de o mulțime de dependențe functionale din R in care partea stângă e o cheie pentru R