Curs 3. Dependenţe funcţionale

Proiectare bună / defectuoasă

Datele reprezentate prin diverse structuri au în general constrângeri relativ la valorile pe care le pot primi atributele. Aceste constrângeri pot fi utilizate pentur a determina dacă o structură este corect definită/proiectată sau nu.

Exemplu: Să considerăm relaţia *MovieList* având următoarea instanţă:

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

Figura 3.1 Instanţa relaţiei *MovieList*

Datele stocate în această relaţie trebuie să respecte următoarele constrângeri:

- Fiecare film are un regizor (*director*)
- FIecare cinematograf are un singur număr de telefon
- Fiecare cinematograf începe proiecţia unui singur film al un moment dat

Problemele comune care apar în cazul unei proiectări defectuoase sunt:

- **Anomalie de inserare**: Nu putem insera informaţii despre un nou film dacă nu cunoaştem cel puţin un cinematograf în care acesta este proiectat

- **Anomalie de ştergere**: Dacă eliminăm toate filmele regizate de Peter Jackson, vom pierde informaţiile despre cinematograful *Cinema City*

- **Anomalie de modificare**: Dacă numărul de telefon al unui cinematograf se schimbă, trebuie să propagăm această modificare în toate înregistrările unde apare acest cinematograf, în caz contrar rămânând informaţii inconsistente în baza de date

În general, putem rafina o structură defectuoassă prin *decompunerea* sa 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

Cinema

Cinema	*Phone*
Cinema City	441111
Odeon	442222

Screens

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

Figura 3.2 Descompunerea relaţiei *MovieList*

Rezultatul rafinării structurii prezentat în figura 3.2 permite:

- Inserarea de noi filme (în *Movies*) fără a cunoaşte detalii legate de proiectarea acestora
- Eliminarea filmelor fără a pierde informaţia despre cinematografe
- Modificarea unei singure înregistrări pentru a actualiza numărul de telefon al unui cinematograf.

E important să putem răspunde la următoarele două întrebări:

- Cum determinăm dacă o structură este "*bună*" sau "*defectuoasă*"?
- Cum transformăm o structură *defectuoasă* într-una *bună*?

Teoria bazată pe *dependenţe funcţionale* furnizează o abordare sistematică a întrebărilor de mai sus. Această teorie a fost introdusă de E.F. Codd în lucrarea: "*A relational model for large shared data banks*", Com. of the ACM, 13(6), 1970, pp.377-387.

Dependenţe funcţionale

Dependenţele funcţionale (DF) sunt constrângeri ale structurilor (tabelelor) unei baze de date care specifică faptul că valoarea unei anumite mulţimi de atribute determină în mod unic valoarea unei alte mulţimi de atribute.

Fie α şi β două submuţimi de atribute alte relaţiei R. Vom nota prin:

$$\alpha \rightarrow \beta$$

faptul că α determină în mod funcţional β (sau β depinde funcţional de α)

În exemplul precedent (relaţia *MovieList*) putem identifica următoarele dependenţe funcţionale:

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

Definiţie. Dependenţa funcţională $\alpha \rightarrow \beta$ este satisfăcută de R dacă şi numai dacă pentru *orice* instanţă a lui R, oricare două tuple t_1 şi t_2 pentru care valorile lui α sunt identice vor avea de asemenea valori identice pentru β.

Acest lucru mai poate fi scris astfel:

$$\pi_\alpha(t_1) = \pi_\alpha(t_2) \quad \Rightarrow \quad \pi_\beta(t_1) = \pi_\beta(t_2),$$

Notă: $\pi_\alpha(t)$ reprezintă proiecţia pe atributele α aplicată tuplului t (v. curs 6)

Fie *r* instanţa unei relaţii *R*

Spunem că *r* **satisface DF** $\alpha \rightarrow \beta$ dacă pentru orice pereche de tuple t_1 şi t_2 în *r* astfel încât $\pi_\alpha(t_1) = \pi_\alpha(t_2)$, este de asemenea adevărat că $\pi_\beta(t_1) = \pi_\beta(t_2)$. Deci, o **DF** *f* **este satisfăcută pe** *R* dacă şi numai dacă orice instanţă *r* a lui *R* satisface *f*

r **nu respectă** o DF *f* dacă *r* nu satisface *f*. Spunem că *r* este o **instanţă legală a lui R** dacă *r* satisface toate dependenţele funcţionale definite pentru *R*.

O DF $\alpha \rightarrow \beta$ se numeşte **trivială** dacă $\alpha \supseteq \beta$

Exemplu. Fie relaţia ***Movie***(Title, Director, Composer) şi *r* o instanţă legală a lui ***Movie*** prezentată mai jos:

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

Figura 3.3. Instanţă a relaţiei ***Movie***

Dependenţa funcţională *composer* → *director* nu este respectată de relaţia ***Movie***. În acelaşi timp, *r* satisface DF *director* → *composer*, dar de aici nu putem trage concluzia că *director*→*composer* este satisfăcută pe întreaga relaţie ***Movie***!

Concluzie: pe baza unor simple instanţe legale ale lui R putem determina care sunt DF ce nu sunt satisfăcute de R, dar nu putem deduce care sunt DF netriviale care sunt respectate de toate instanţele lui R.

Problema implicaţiei: Având dată o mulţime de dependenţe funcţionale *F* (care sunt respectate de *R*) şi o dependenţa funcţională oarecare *f*, putem deduce dacă *f* este de asemenea respectată de *R*? Spunem că *F* **implică logic (**sau doar **implică)** *f*, şi notăm prin $F \Rightarrow f$, dacă fiecare instanţă *r* a relaţiei *R* ce satisface DF din *F* satisface de asemenea şi pe *f*.

Examplu: În ***MovieList***, avem următoarea mulţime predefinită de dependenţe funcţionale:

F = { Title → Director

Cinema → Phone

Cinema, Time → Title }

Dependenţele *Cinema, Time → Director* sau *Time → Director* sunt de asemenea respectate?

Fie *F* & *G* două mulţimi de dependenţe funcţionale. Vom nota prin $F \Rightarrow G$ (F implică G) dacă $F \Rightarrow g$ pentru oricare $g \in G$.

Inchiderea lui F (notată prin F^+) este mulţimea tuturor dependenţelor funcţionale implicate de F şi anume,

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

Spunem că două mulţimi de dependenţe funcţionale, *F* şi *G*, sunt **echivalente** (notat prin $F \equiv G$) dacă $F^+ = G^+$ (adică, $F \Rightarrow G$ şi $G \Rightarrow F$)

Axiomele dependenţelor funcţionale

= o colecţie de reguli formale de derivare a dependenţelor funcţionale dintr-un set iniţial de dependenţe funcţionale.

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 $\alpha \rightarrow \gamma$

Axiomele lui Armstrong sunt *necesare* şi *suficiente* pentru a determina DF implicate de F

Necesare: Orice FD derivată este implicată de F

Complete: Toate DF din F^+ pot fi derivate

Problemă: Considerăm R(A, B, C, D, E) având 3 dependenţe funcţionale:

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

Arătaţi că $F \Rightarrow AD \rightarrow E$

Soluţie:

1. $A \rightarrow C$ (dată)
2. $AD \rightarrow CD$ (augmentare cu (1))
3. $CD \rightarrow E$ (dată)
4. $AD \rightarrow E$ (tranzitivitate cu (2) şi (3))

Reguli de inferenţă adiţionale

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

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

Superchei, chei & atribute prime

O mulţime de atribute α reprezintă o supercheie a relaţiei R (având mulţimea de DF F) dacă

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

O mulţime de atribute α o cheie a relaţiei R dacă

(1) α este o supercheie, şi

(2) nici o submulţime a lui α nu este 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.

Exemplu: Considerăm din nou relaţia ***MovieList*** (Title, Director, Cinema, Phone, Time) cu mulţimea de dependenţe funcţionale:

(1) Cinema, Time $\rightarrow$ Title

(2) Cinema $\rightarrow$ Phone

(3) Title $\rightarrow$ Director

{*Cinema, Time*} este singura cheie a relaţiei ***MovieList***.

Cinema şi *Time* sunt singurele atribute prime din ***MovieList***.

Orice mulţime ce include {*Cinema; Time*} este o supercheie a lui ***MovieList***.

Închiderea atributelor

Identificarea tuturor dependenţelor funcţionale din F^+ pornind de la o mulţime F de DF nu este eficientă deoarece dimensiunea lui F^+ creşte exponenţial!

Este, astfel, mult mai eficient să se calculeze <u>închiderea unei mulţimi de atribute</u>

Fie $\alpha \subseteq R$ şi F o mulţime de DF satisfăcute pe R. Închiderea lui α (cu respectarea mulţimii F de DF), notată cu α^+, este mulţimea de atribute ce sunt determinate funcţional din α pe baza dependenţelor funcţionale 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 de determinarea a închiderii unui atribut/unei mulţîmi de atribute:

Input: α, F

Output: α^+ (w.r.t. F)

Se calculează secvenţa de mulţimi de atribute α_0, α_1,... α_k, α_{k+1} astfel:

$\alpha_0 = \alpha$

$\alpha_{i+1} = \alpha_i \cup \gamma$ astfel incat exista o FD

$\beta \rightarrow \gamma \in F$ si $\beta \subseteq \alpha_i$

Algoritmul se termina atunci cand $\alpha_{k+1} = \alpha_k$ pentru un anume *k*

Return α_k

Problemă: Fiind dată F = {A → C;B → C;CD → E}, arătaţi că F ⇒ AD→E.

Soluţie

i	α_i	*FD used*
0	AD	given input
1	ACD	A→C
2	ACDE	CD → E
3	ACDE	none

Deci AD^+ = ACDE. Deoarece E ∈AD^+, atunci rezultă că F ⇒ AD→E

Descompunerea relaţiilor

Descompunerea unei relaţii R este un set de (sub)relaţii $\{R_1, R_2, ..., R_n\}$ astfel încât fiecare $R_i \subseteq R$ şi $R = \cup R_i$. Dacă *r* este o relaţie din R, atunci *r* se descompune în $\{r_1, r_2, ..., r_n\}$, unde fiecare $r_i = \pi_{Ri}(r)$

Exemplu:

{ (Cinema, Time, Title),

(Title, Director),

(Cinema, Phone)}

Este o descompunere a relaţiei ***MovieList***(Title, Director, Cinema, Phone, Time)

Proprietăţile descompunerii relaţiilor:

1. Descompunerea trebuie să păstreze informaţiile
   - Datele din relaţia originală ≡ Data din relaţiile descompunerii
   - Crucial for păstrarea consistenţei datelor!
2. Descompunerea trebuie să respecte toate DF
   - Dependenţele funcţionale din relaţia originală ≡ reuniunea dependenţelor funcţîonale din relaţiile descompunerii
   - Facilitează verificarea violărilor DF

Descompunere cu joncţiune fără pierderi

Este important ca descompunerea unei relaţii să păstreze informaţiile; ceea ce înseamnă că putem reconstrui o instanţă *r* prin joncţiunea proiecţiilor sale $\{r_1, r_2, ... , r_n\}$. De notat că dacă $\{R_1, R_2, ... , R_n\}$ este o descompunere a lui R, atunci pentru orice instanţă *r* a lui R, este întotdeauna adevărat că

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

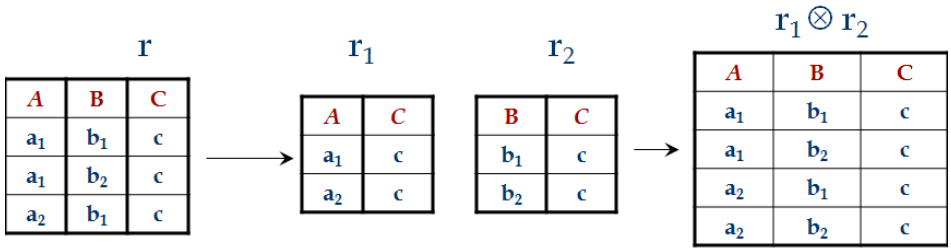
Notă: ⊗ reprezintă operatorul algebric join natural (v cursul 6)

O descompunere a lui R (cu mulţimea de dependenţe funcţionale *F*) în $\{R_1,R_2,...,R_n\}$ este o descompunere cu joncţiuni fără pierderi cu respectarea mulţimii *F* dacă

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

Pentru fiecare instanţă r a lui R ce satisface *F*.

Exemplu. Se consideră descompunerea relaţiei R(A,B,C) în $\{R_1(AC), R_2(BC)\}$



Deoarece $r \subset r_1 \otimes r_2$, descompunerea de mai sus nu este cu joncţiuni fără pierderi.

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

Teoremă: 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:

$$F \Rightarrow R_1 \cap R_2 \rightarrow R_1 \quad \text{sau} \quad F \Rightarrow R_1 \cap R_2 \rightarrow R_2$$

Cum descompunem R în {R_1, R_2} astfel încât ?

Corollar: Dacă $\alpha \rightarrow \beta$ este satisfăcută pe R şi $\alpha \cap \beta = \varnothing$, atunci descompunerea lui R în {R- β , $\alpha\beta$} este o descompunere cu joncţiuni fără pierderi.

Example. Se consideră R(A,B,C) cu mulţimea de DF F = { $A \rightarrow B$}

Descompunerea {AB, AC} este cu joncţiuni fără pierderi deoarece $AB \cap AC = A$ şi $A \rightarrow AB$

Descompunerea {AB, BC} nu este cu joncţiuni fără pierderi deoarece $AB \cap BC = B$ şi nici $B \rightarrow AB$ sau $B \rightarrow BC$ nu sunt satisfăcute de R.

Teoremă: Dacă {R_1, R_2} este o descompunere cu joncţiuni fără pierderi a lui R, şi dacă {$R_{1,1}$, $R_{1,2}$} este o descompunere cu joncţiuni fără pierderi a lui R_1, atunci {$R_{1,1}$, $R_{1,2}$, R_2} este o descompunere cu joncţiuni fără pierderi a lui R :

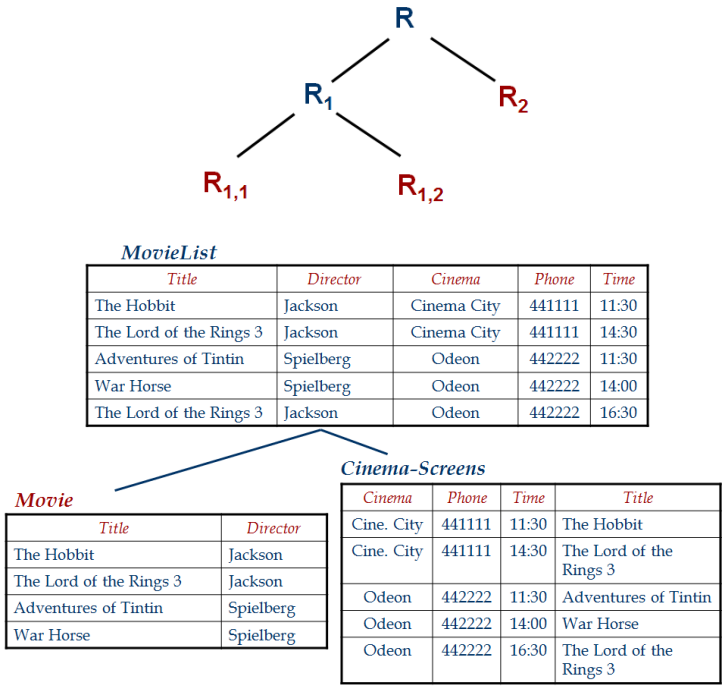


Figura 3.4 Primul pas al descompunerii relaţiei ***MovieList*** pe baza DF

Title → Director

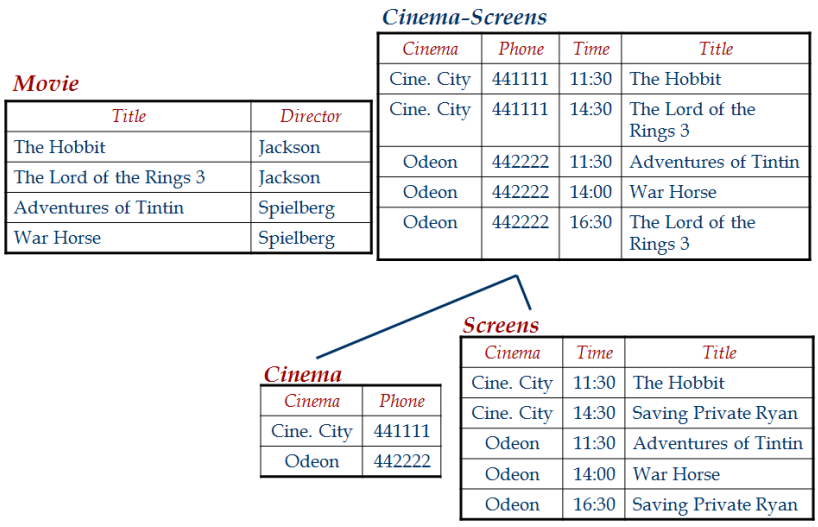


Figura 3.5 Ultimul pas al descompunerii ***MovieList***, pe baza DF

Cinema → *Phone*

Păstrarea dependenţelor funcţionale

Numim proiecţia lui F pe α (notată prin F_α) mulţimea dependenţelor funcţionale din F^+ ce implică doar atributele din α; adică $F_\alpha = \{ \beta \rightarrow \gamma \in F^+ \mid \beta\gamma \subseteq \alpha \}$

Algoritmul de determinare a proiecţiei unei mulţimi de dependenţe funcţionale:

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

Definiţie. Descompunerea $\{R_1, R_2, ..., R_n\}$ relaţiei R păstrează dependenţele dacă

$(F_{R1} \cup F_{R2} \cup ... \cup F_{Rn})$ şi F sunt echivalente, adică:

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