

## אלגוריתם Dec3

Dec3(S,F)

```
B - min basis
for( $\forall (f = X \rightarrow Y) \in B$ ) sch(X,Y)
if(none of the schemes from for-loop is a s. key for R)
  add sch=key for R;
```

## דוגמה

$$S = (A, B, C, D, E)$$

$$F = AB \rightarrow C, C \rightarrow B, A \rightarrow D$$

הרעיון הוא לבנות סכימה לכל תלות פונקציונאלית מבסיס מינימלי:

$$S1 = A, B, C \quad S2 = C, B \quad S3 = A, D$$

מתקיים  $S2 \subset S1$ , לכן  $S2$  מיותרת:

$$S1 = A, B, C \quad \del{S2 = C, B} \quad S3 = A, D$$

לכן  $S3$  הוא החדש שלנו:

$$S1 = A, B, C \quad S2 = A, D$$

הלולאה היא בעצם בדיקה אחת - לא באמת נכנסים ללולאה, פשוט בודקים את כל הסכימות של התלויות. לכן עכשיו אפשר לבנות את המפתחות:

$$K1 = A, B, E \quad K2 = A, C, E$$

## תלויות רב ערכיות Multi Valued Dependencies - MUD

name	phone	bank	branch
Avi	03-11	Disc	014
Avi	03-11	Miz	144
Avi	054-2	Dis	014
Avi	054-2	Miz	144
Ben	03-11	Dis	120
Ben	03-11	Leumi	133
Ben	054-3	Dis	120
Ben	054-3	Leumi	133

אין כאן תלויות פונקציונאליות, אבל ברור שיש כאן כפילות!  
לכל אחד מהשמות, יש שתי קבוצות:

$$\begin{aligned} \text{Avi} &\Rightarrow \{03 - 11, 054 - 2\} \\ &\quad \{(Disc, 014), (Miz, 144)\} \\ \text{Ben} &\Rightarrow \{03 - 11, 054 - 3\} \\ &\quad \{(Disc, 120), (Leumi, 133)\} \end{aligned}$$

## הגדרה

יהיו  $S$  סכימה,  $X, Y \subset S$ .  
מסמנים  $X \twoheadrightarrow Y$  אם לכל ערך של  $X$  קיימת קבוצת ערכים ב  $Y$  מוגדרת היטב -  
כלומר קבוצה שלא תלויה ב  $S - X - Y$ .

$$[X] \Rightarrow [Y] \times [S - X - Y]$$

לכל ערך של  $X$  מתאימה מכפלה קרטזית של ערכים ב  $Y$  עם ערכים ב  $S - X - Y$ .

## כללים

1. טריוויאלית: אם  $Y \subset X$  אז  $X \twoheadrightarrow Y$
2. טרנזיטיביות: אם  $X \twoheadrightarrow Y, Y \twoheadrightarrow Z$  אז  $X \twoheadrightarrow Z$  כאשר לוקחים  $Z = Z - X$ .
3. Splitting - לא עושים כאן!
4. FD promotion: אם  $X \rightarrow Y$  אז  $X \twoheadrightarrow Y$ .
5. Complementation: אם  $X \twoheadrightarrow Y$  אז  $X \twoheadrightarrow S - X - Y$ . תלויות רב ערכיות תמיד באות בזוגות.
6. עוד טריוויאליות: אם  $S = (X, Y)$  אז  $X \twoheadrightarrow Y$ .

# 4NF

## הגדרה

יחס הוא ב 4NF אם לכל  $X \twoheadrightarrow Y$  לא טריוויאלית,  $X$  הוא superkey.

## בדוגמה הקודמת

הדוגמה שלנו היא לא ב 4NF, שכן  $\{phone, bank, branch\} \twoheadrightarrow name$  אבל (name) אינו superkey.

## דוגמה

name	phone	cr_card	valid
Avi	03-11	V1111	11/11
Avi	03-11	M2222	11/11
Avi	054-2	M3333	10/12
Avi	054-2	V1111	11/11
Ben	03-11	V4444	09/13
Ben	03-11	A5555	10/13
Ben	054-3	V4444	09/13
Ben	054-3	A5555	10/13
Avi	03-11	M3333	10/12
Avi	054-2	M2222	11/11

$S = \text{name, phone, cr\_card, valid}$

$$F = \left\{ \begin{array}{l} \text{name} \rightarrow\rightarrow \text{phone} \\ \text{name} \rightarrow\rightarrow \text{cr\_card, valid} \\ \text{cr\_card} \rightarrow \text{valid} \\ \text{cr\_card} \rightarrow \text{name} \end{array} \right\}$$

## Dec4 אלגוריתם

$\text{dec4}(S,F)$

```
if(S in 4NF) return S
else
  if( $X \rightarrow\rightarrow Y \in F$  violates 4NF) // X not a s.k.
  {
    compute F1; compute F2;
    return ( $\text{dec4}(S1,F1) \cup \text{dec4}(S2,F2)$ )
  }
```

## בדוגמה

$(\text{name} \rightarrow\rightarrow \text{phone}) \in F$

$S1 = (\text{name, phone}) \quad S2 (\text{name, cr\_card, valid})$

$F1 = \emptyset \quad S2 = \{\text{cr\_card} \rightarrow \text{name}, \text{cr\_card} \rightarrow \text{valid}\}$

נשים לב שבdec4, בניגוד לdec3, צריך להמשיך עד שמגיעים ליחס ב4NF

$$\text{dec4}(S1, F1) = \text{dec}(S2, F2)$$

התוצאה הסופית היא:

name	phone	name	cr_card	valid
Avi	03-11	Avi	V1111	11/11
Avi	054-2	Avi	M2222	11/11
Ben	03-11	Avi	M3333	10/12
Ben	054-3	Ben	V4444	09/13
		Ben	A5555	10/13

נשים לב! פעם ראשונה שבשני היחסים פחות שורות מאשר במקור!  
בכל הפירוקים הקודמים, תמיד באחד היחסים היה מפתח, ולכן אותו מספר שורות כמו ביחס המקומי.

## ניהול טרנזקציות - Transactions Management

### מושגים

- $T$  - טרנזקציה (Transaction)
- $E$  - אלמנט (Element)
- מצב מסד הנתונים (State of DB) - ערכים של כל הנתונים

### תכונות

- אטומיות (atomicity) - או שהטרנזקציה מתבצעת או שהיא לא מתבצעת. אסור שהיא תתבצע רק חצי.
- נכונות (Correctness principle) - אם הטרנזקציה מתבצעת לבד, ללא שגיאות של המערכת, וכשהיא מתחילה המערכת הייתה במצב תקין (DB consistent state) - המערכת תימצא במצב תקין לאחר הטרנזקציה.

מצב תקין - שומר על אילוצים מפורשים ומרומזים

## פעולות בסיסיות - Primitive operations

יש 4 פעולות פרימיטיביות:

- input( $X$ ) •
- output( $X$ ) •
- read( $X$ ) •
- write( $X$ ) •

לכל טרנזיציה יש מקום בזיכרון. אם כמה טרנזקציות קוראות ערך של  $X$ , כל אחד קוראת אותו לזיכרון שלה. יש גם בפרים בין הטרנזקציות לבסיס הנתונים.  
 קוראים לבסיס הנתונים בלוק שלם ממסד הנתונים(פעולת input), וקוראים את הערך  $X$  לזיכרון הפרטי של הטרנזיציה עם(פעולת read). פעולות output וwrite עושות אותו דבר רק הפוך.

## תזמון - Serializability of schedules

Schedule זה סדרה של צעדים, פעולות, של טרנזקציות. לדוגמה:

$S1=T1:r(A,t);T1:t=t+100;T1:w(A,t);T1:r(B,t);T1:t=t+100;T1:w(B,t)$   
 $S2=T2:r(A,s);T2:s=s*2;T2:w(A,s);T2:r(B,s);T2:s=s*2;T2:w(B,s)$   
 $S3=T1,T2$   
 $S4=T2,T1$

אם יש אילוק האילוק:

constraint:  $A=B$

$T1$  מוסיף 100 ל  $A$ , ואז מוסיף 100 ל  $B$ . אם מתחילים ממצב תקין, מגיעים למצב תקין.  
 $T2$  מכפיל את  $A$  פי 2, ואז מכפיל את  $B$  פי 2. אם מתחילים ממצב תקין, מגיעים למצב תקין.

$S3$  ו  $S4$  מריצים את  $S1, S2$  אחת אחרי השנייה, ולכן גם הן תקינות.  
 מתי יכולה להיות בעיה? אם מריצים אותם אחד ביחד עם השני, בו זמנית.