BASI DI DATI 2

ALGORITMI PER LA PROGETTAZIONE DI DB RELAZIONALI E ULTERIORI DIPENDENZE

a.a 2017/2018

Prof.ssa G. Tortora

Top-Down o Bottom-Up?

- Per la progettazione di un db relazionale, esistono due differenti filosofie:
  - L'approccio *Top-Down*
    - Si parte dal diagramma ER (o EER), si mappa in uno schema relazionale e si applicano le procedure di normalizzazione.
  - L'approccio *Bottom-Up*
    - È più formale, poiché considera lo schema del db solo in termini di dipendenze funzionali.
      Dopo aver definito tutte le FD, si applica un algoritmo di normalizzazione per sintetizzare gli schemi di relazione in 3NF o in BCNF.
- In questo capitolo studiamo delle tecniche per il secondo tipo di approccio.

Algoritmi per la progettazione di schemi di db relazionali

Decomposizione delle relazioni

- Gli algoritmi che presentiamo partono da un singolo schema di relazione universale $R = \{A_1, A_2, \ldots, A_n\}$ che include tutti gli attributi del database.

- I progettisti specificano l'insieme F delle dipendenze funzionali che devono valere sugli attributi di R.

- Usando le dipendenze funzionali, gli algoritmi decompongono lo schema di relazione R in un insieme di schemi $D = \{R_1, R_2, \ldots, R_n\}$
  - D è detto decomposizione di R.

Conservazione degli attributi

- Ciascun attributo di R deve apparire in almeno una relazione in D:

$$\bigcup_{i=1}^{n} R_i = R$$

- Questa condizione è detta di conservazione degli attributi.

Decomposizione delle relazioni (2)

- Altro obiettivo è che ogni relazione individuale R_i in D deve essere in BCNF (o in 3NF).
  - Da notare che qualsiasi schema di relazione con soli due attributi è automaticamente in BCNF.

- Questa condizione non è sufficiente per garantire di avere un buon db, in quanto il fenomeno delle tuple spurie può comunque presentarsi anche con relazioni in BCNF.

Conservazione delle dipendenze

- Sarebbe utile che le dipendenze funzionali

  $X \rightarrow Y$ specificate in F, apparissero direttamente in una delle R_i della decomposizione D o possano essere inferite da dipendenze in altre R_i.
  - Questa è (informalmente) la condizione di conservazione delle dipendenze.

- È fondamentale non perdere delle dipendenze, poiché queste rappresentano dei vincoli sul database.

Conservazione delle dipendenze (2)

- Non è necessario che esattamente le stesse dipendenze specificate in F appaiono nelle relazioni della decomposizione D:
  - è sufficiente che l'unione delle dipendenze che valgono nelle singole relazioni di D sia equivalente ad F.

- Per formalizzare tali concetti abbiamo bisogno di alcune definizioni …

Proiezione

- **Definizione**:

  Dato un insieme di dipendenze F in R, la proiezione di F su R_i , denotata da $\pi_{Ri}(F)$ (dove R_i è un sottoinsieme di R), è l'insieme di dipendenze $X \rightarrow Y$ in F^+ tale che gli attributi in $X \cup Y$ sono tutti contenuti in R_i .

Decomposizione dependency-preserving

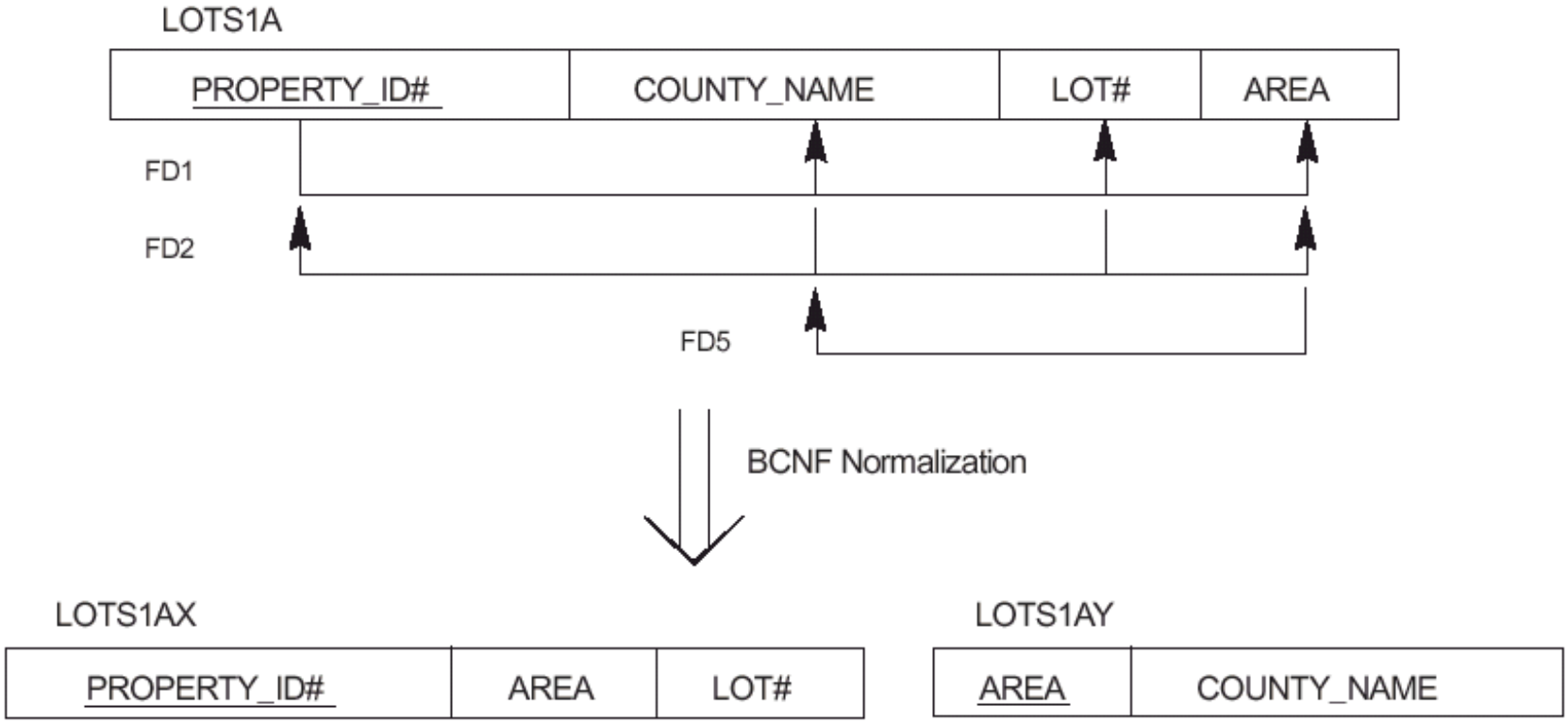
- Diciamo che una decomposizione $D = \{R_1, R_2, \ldots, R_m\}$ di R è dependency-preserving rispetto a F se l'unione delle proiezioni di F su ciascun R_i in D è equivalente a F, cioè:

$$((\pi_{R1}(F)) \cup \ldots \cup (\pi_{Rm}(F)))^+ = F^+$$

- Se una decomposizione non è dependency-preserving, si ha la perdita di qualche dipendenza nella decomposizione.

Decomposizione dependency-preserving: *Esempio*

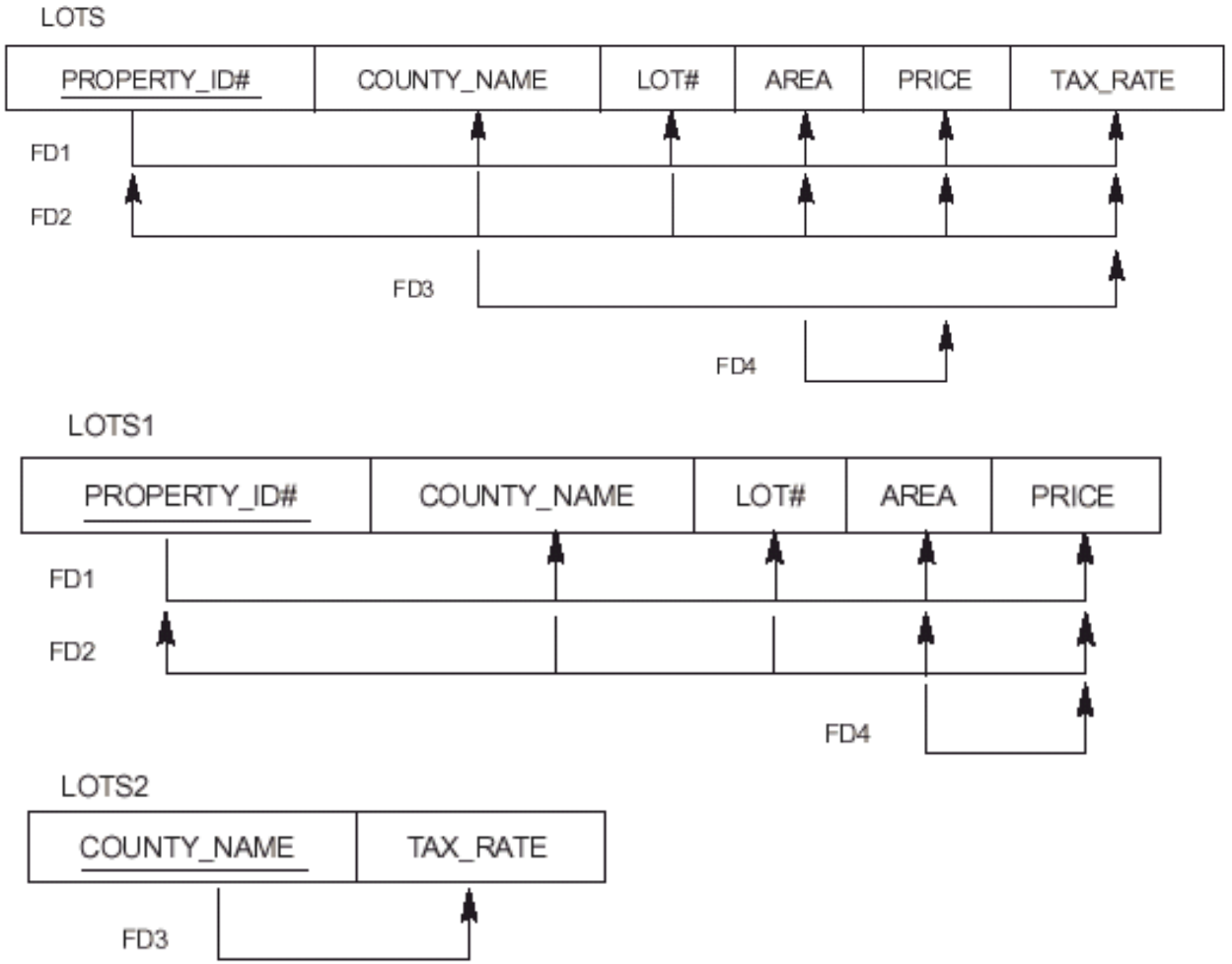
- Decomposizione non dependency-preserving:



La FD2 viene persa

Decomposizione dependency-preserving: *Esempio 2*

- Decomposizione dependency-preserving:



Proposizione 1

- È sempre possibile trovare una decomposizione dependency-preserving D rispetto a F tale che ogni R_i in D è in 3NF.

Copertura minimale

- Un insieme di dipendenze funzionali F è minimale se:
  - Ogni dipendenza in F ha un singolo attributo sul lato destro.
  - Non è possibile rimpiazzare una dipendenza $X \rightarrow A$ con una dipendenza $Y \rightarrow A$ dove Y è un sottoinsieme proprio di X ed avere ancora un insieme di dipendenze che è equivalente ad F.
  - Non possiamo rimuovere una dipendenza da F ed ottenere un insieme di dipendenze equivalente ad F.
- **Definizione**: Una copertura minimale di un insieme E di dipendenze funzionali è un insieme minimale F di dipendenze che è equivalente ad E.

Algoritmo di sintesi relazionale

- Algoritmo 1
  Decomposizione dependency-preserving in schemi di relazione 3NF:
  1. trovare una copertura minimale G di F.
  2. per ogni parte sinistra X di una dipendenza funzionale che appare in G, creare uno schema di relazione $\{X \cup A_1 \cup A_2 \cup \ldots \cup A_m\}$ in D dove
     $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_m$ sono le sole dipendenze in G aventi X come parte sinistra.
  3. mettere in uno schema di relazione singolo tutti gli attributi rimanenti, per garantire la proprietà di attribute-preservation.

Algoritmo per trovare una copertura minimale G per F

- Il passo 1 dell'algoritmo richiede di trovare una copertura minimale.

  Ecco l'algoritmo per fare ciò:

- Algoritmo 1.a

  Consta di quattro passi.

  G è l'insieme minimale di dipendenze funzionali equivalente ad F.

1. *porre* G:= F;
2. rimpiazzare ogni dipendenza funzionale $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ in G, con **n** dipendenze funzionali $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$;

…

Algoritmo per trovare una copertura minimale G per F (2)

3. *per ogni* dipendenza funzionale $X \rightarrow A$ in G
   per ogni attributo B che è un elemento di X
   {
   se { {G - {$X \rightarrow A$} } $\cup$ { $(X - \{B\}) \rightarrow A$} } è equivalente a G
   allora sostituire $X \rightarrow A$ con
   $(X - \{B\}) \rightarrow A$ in G;
   }
4. *per ogni* dipendenza funzionale rimanente $X \rightarrow A$ in G
   {
   se { G – {$X \rightarrow A$} } è equivalente a G
   allora rimuovere $X \rightarrow A$ da G;
   }

Decomposizione e Join Lossless (o Non-additive)

- La lossless (o non-additive) join (LJ) è un'altra proprietà di cui dovrebbe godere una decomposizione.

- Una decomposizione con LJ garantisce che non vengono generate tuple spurie applicando un'operazione di NATURAL-JOIN alle relazioni nella decomposizione.

Lossless Join

- **Definizione**:
  Una decomposizione D = {R_1, R_2, …, R_m} di R ha la proprietà di lossless join rispetto all'insieme di dipendenze di F su R, se per ogni stato di relazione r di R che soddisfa F, vale:

  $$*(\pi_{R1}(r), \ldots, \pi_{Rm}(r)) = r$$

  dove con * indichiamo la natural-join.

Test della proprietà di lossless join

- Algoritmo 2
  Verifica in cinque passi se una decomposizione D ha la proprietà di lossless join rispetto ad un insieme F di dipendenze funzionali:

1. Creare una matrice S con una riga i per ogni relazione R_i nella decomposizione D, e una colonna j per ogni attributo A_j in R;
2. *porre* S(i,j) = b_{ij} per tutte le entrate della matrice;
   / ogni b_{ij} è un simbolo distinto associato agli indici (i,j) */*

…

Test della proprietà di lossless join (2)

3. *per ogni* riga i che rappresenta lo schema di relazione R_i
   per ogni colonna j che rappresenta l'attributo A_j
   se R_i include l'attributo A_j
   allora porre $S(i,j) = a_j$;
4. *ripetere* quanto segue *finché* l'esecuzione del ciclo non modifica più S
   per ogni dipendenza funzionale $X \rightarrow Y$ in F
   per tutte le righe in S, che hanno gli stessi simboli nelle colonne corrispondenti agli attributi in X:

…

Test della proprietà di lossless join (3)

rendere uguali i simboli in ogni colonna che corrispondono ad un attributo in Y, come segue:

se ogni riga ha un simbolo ‘a’ nella colonna, porre le altre righe allo stesso simbolo ‘a’ nella colonna. Se non esiste in nessuna riga un simbolo ‘a’ per l’attributo, scegliere uno dei simboli ‘b’ che appaiono in una delle righe e porre le altre righe a quel simbolo ‘b’ nella colonna;

5. *se* una riga contiene solo simboli ‘a’,

   allora la decomposizione ha la proprietà lossless join,

   altrimenti no.

Descrizione dell'algoritmo 2

- Data una relazione R, decomposta in R_1, R_2, …, R_m, l'algoritmo crea uno stato di relazione r nella matrice S. La riga i-esima rappresenta la relazione R_i, con un simbolo 'a' in corrispondenza degli attributi di R_i, ed un simbolo 'b' nelle colonne rimanenti.

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} D={R_1, R_2}
R_1=EMP_LOCS={ENAME, PLOCATION}
R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

Descrizione dell'algoritmo 2 (2)

- L'algoritmo trasforma le righe della matrice (*nel ciclo del passo 4*) in modo che rappresentino tuple soddisfacenti tutte le FD in F.

- Alla fine del loop, per ogni coppia di righe in S per i cui attributi vale la parte sinistra X di una FD $X \rightarrow Y$, deve valere anche la parte destra Y di tale dipendenza.

Descrizione dell'algoritmo 2 (3)

- Può essere dimostrato che dopo il ciclo del passo 4, la decomposizione D ha la proprietà di lossless join se e solo se una riga ha tutti simboli 'a'.

- Se non si hanno tutti simboli 'a', lo stato di relazione r (*e quindi la decomposizione D*) soddisfa le FD di F, ma non gode della proprietà di lossless join.

Algoritmo 2: *Esempio*

EMP	
SSN	ENAME

PROJECT		
PNUMBER	PNAME	PLOCATION

WORKS_ON		
SSN	PNUMBER	HOURS

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS}
R_1=EMP={SSN, ENAME}
R_2=PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

D={R_1, R_2, R_3}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

Trovare una decomposizione

- L'algoritmo proposto consente di verificare se una decomposizione ha la proprietà di lossless join.

- Vogliamo trovare una decomposizione di uno schema di relazione R che sia in BCNF e che presenti la proprietà di lossless join rispetto ad un determinato insieme di dipendenze funzionali.

Proprietà delle decomposizioni lossless join

- Proprietà LJ1:
  Una decomposizione D = $\{R_1, R_2\}$ di R ha la proprietà lossless join rispetto a un insieme di dipendenze funzionali F su R se e solo se:
  - la FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ è in F^+,

  oppure
  - la FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ è in F^+

- Questo è un test molto più immediato dell'algoritmo 2 ma funziona solo per decomposizioni in due schemi.

Proprietà delle decomposizioni lossless join (2)

- Proprietà LJ2:
  Se una decomposizione $D = \{R_1, R_2, \ldots, R_m\}$ di R ha la proprietà LJ rispetto a un insieme di FD F su R, e se una decomposizione

  $D_1 = \{Q_1, Q_2, \ldots, Q_k\}$ di R_i ha la proprietà LJ rispetto alla proiezione di F su R_i allora la decomposizione

  $D_2 = \{R_1, R_2, \ldots, R_{i-1}, Q_1, Q_2, \ldots, Q_{k,} R_{i+1}, \ldots R_m\}$ di R ha la proprietà LJ rispetto a F.

- Usando queste proprietà possiamo sviluppare un algoritmo per creare decomposizioni LJ con schemi di relazione in BCNF.

Decomposizione LJ in schemi in BCNF

- Algoritmo 3
  Consente di ottenere in due passi una decomposizione LJ con schemi di relazione in BCNF:

1. *porre* D := R;
2. *finché* ci sta uno schema di relazione Q in D che non è in BCNF
   {
   si scelga uno schema di relazione Q in D che non sia in BCNF;
   si trovi una dipendenza funzionale $X \rightarrow Y$ che violi BCNF;
   si sostituisca Q in D con due schemi di relazione (Q-Y) e ($X \cup Y$).
   }

Decomposizione LJ e dependency-preserving in schemi di relazione in 3NF

- Algoritmo 4
  Consente di ottenere in tre passi una decomposizione LJ e dependency-preserving con schemi di relazione in 3NF:
  1. Trovare una copertura minimale G per F;
  2. *per ogni* parte sinistra X di una FD in G, creare uno schema di relazione in D con attributi

  $\{X \cup A_1 \cup A_2 \cup \ldots \cup A_m\}$ dove $X \rightarrow A_1$, $X \rightarrow A_2$, …, $X \rightarrow A_m$ sono le sole dipendenze in G aventi X come parte sinistra.
  3. *Se* nessuno degli schemi di relazione in D contiene una chiave di R, creare un altro schema di relazione che contiene attributi che formano una chiave per R.

Trovare una chiave K per uno schema di relazione R

- L'algoritmo che segue consente di identificare una chiave K di R, come richiesto nel passo 3.

  La chiave restituita dipende dall'ordine in cui gli attributi sono stati rimossi al passo 2.

- Algoritmo 4a:

1. *porre* K := R;
2. *per ogni* attributo A in K
   {
   calcolare $(K - A)^+$ rispetto all' insieme di FD;
   se $(K - A)^+$ contiene tutti gli attributi in R,
   allora K = K - {A};
   }

Nota

- Non è sempre possibile trovare una decomposizione LJ che sia dependency-preserving e i cui schemi di relazione siano in BCNF.

Decomposizioni e valori null

- La teoria delle decomposizioni LJ si basa sulla seguente assunzione:
  *nessun valore **null** è ammesso per gli attributi di join.*

- Non esiste una teoria di progettazione di database relazionali con i valori **null** soddisfacente.

Decomposizioni e valori null: *Esempio*

- Due impiegati non sono ancora assegnati a un dipartimento …

EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUM
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX	**null**
Benitez, Carlos M.	888664444	1963-01-09	7654 Beech, Houston, TX	**null**

DEPARTMENT

DNAME	DNUM	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Decomposizioni e valori null: *Esempio*

- Siano interessati ad una lista (ENAME, DNAME) per tutti gli impiegati.

ENAME	SSN	BDATE	ADDRESS	DNUM	DNAME	DMGRSSN
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

- Risultato di un Natural-Join: mancano gli impiegati non assegnati.

Decomposizioni e valori null: *Esempio*

ENAME	SSN	BDATE	ADDRESS	DNUM	DNAME	DMGRSSN
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX	**null**	**null**	**null**
Benitez, Carlos M.	888664444	1963-01-09	7654 Beech, Houston, TX	**null**	**null**	**null**

- Risultato di un Outer-Join: abbiamo tutti gli impiegati, ma anche dei valori **null** per i dipartimenti.

Decomposizioni e valori null (2)

- Potenziali problemi con valori **null** possono sorgere nell'uso di:
  - Join, con perdita di informazioni se non si utilizza un Outer-Join.
  - Funzioni di aggregazione, quali SUM o AVERAGE, con comportamenti non sempre definiti su **null**.

Algoritmi di normalizzazione

Algoritmo	Input	Output	Proprietà/Scopo	Note
Algoritmo 1	Insieme F di dipendenze funzionali	Un insieme di relazioni in 3NF	Conservazione delle dipendenze	Nessuna garanzia sulla verifica della proprietà di join senza perdita.
Algoritmo 2	Una decomposizione D di R e un insieme F di dipendenze funzionale	Risultato booleano; si o no per la proprietà di join non-additivo	Test per la decomposizione non-additiva	Esiste un test più semplice per l
Algoritmo 3	Insieme F di dipendenze funzionali	Un insieme di relazioni in BCNF	Decomposizione non-additiva	Nessuna garanzia della conservazione delle dipendenze.
Algoritmo 4	Insieme F di dipendenze funzionali	Un insieme di relazioni in 3NF	Decomposizione non-additiva e con conservazione delle dipendenze	Può capitare di non avere una BCNF; ma si ottiene la 3NF e tutte le proprietà auspicabili.
Algoritmo 4a	Schema di relazione R con un insieme F di dipendenze funzionali	Chiave K di R	Per trovare una chiave K (cioè un sottoinsieme di R)	L'intera relazione R è sempre una superchiave di default.

Dipendenze Multivalore

Dipendenze Multivalore (MVD)

- Sono conseguenza della 1NF, che non consente ad un attributo di assumere un insieme di valori.

- In presenza di due o più attributi indipendenti *multivalued* nello stesso schema, siamo costretti a ripetere ogni valore di un attributo con ogni valore dell'altro.

Dipendenze Multivalore: *Esempio*

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

- Questo vincolo è specificato come una dipendenza multivalued sulla relazione EMP:
  - Informalmente, si può avere una MVD ogni volta che si fondono due relazioni indipendenti, A:B e A:C con cardinalità 1:N, nella stessa relazione.

Definizione formale di MVD: X multidetermina Y

- **Definizione**:

  Una dipendenza multivalore $X \twoheadrightarrow Y$, specificata su uno schema di relazione R, dove X e Y sono sottoinsiemi di R, specifica il seguente vincolo su qualche relazione r di R:

  - Se esistono in r due tuple t_1 e t_2, con $t_1[X] = t_2[X]$, allora dovrebbero esistere anche t_3 e t_4 con le seguenti proprietà:
    - $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
    - $t_3[Y] = t_1[Y]$ e $t_4[Y] = t_2[Y]$
    - Dato $Z = R - (X \cup Y)$, allora $t_3[Z] = t_2[Z]$ e $t_4[Z] = t_1[Z]$

Definizione formale di MVD (2)

- Data la simmetria della definizione,

  posto $Z = R - (X \cup Y)$,

  si ha che se $X \twoheadrightarrow Y$ allora $X \twoheadrightarrow Z$:

  - $X \twoheadrightarrow Y \Rightarrow X \twoheadrightarrow Z$ e perciò talvolta lo si scrive come $X \twoheadrightarrow Y \mid Z$

MVD: *Esempio*

- Dato un valore di X, l'insieme di valori di Y determinati da X è completamente determinato solo da X e non dipendono dai valori dei rimanenti attributi Z di R:

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

  - Nella relazione EMP valgono:
    ENAME ↠ PNAME e
    ENAME ↠ DNAME
    (o ENAME ↠ PNAME | DNAME)

MVD banali

- Una MVD X ↠ Y in R è detta MVD banale (*trivial*) se:
  - Y è un sottoinsieme di X, oppure
  - X ∪ Y = R

- *Esempio*:
  In EMP-PROJECTS,
  ENAME ↠ PNAME è una MVD banale.

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

EMP_PROJECTS

ENAME	PNAME
Smith	X
Smith	Y

EMP_DEPENDENTS

ENAME	DNAME
Smith	John
Smith	Anna

Regole di inferenza per FD e MVD

- Come già visto per le FD, anche per le MVD esiste un insieme di regole di inferenza:
- Presi $R = \{A_1, A_2, \ldots, A_n\}$ e $X, Y, Z, W \subseteq R$:
  - (IR1) (Regola riflessiva per FD): se $X \supseteq Y$ allora $X \rightarrow Y$
  - (IR2) (Augmentation rule per FD): $\{ X \rightarrow Y \} \models XZ \rightarrow YZ$
  - (IR3) (Transitive rule per FD) : $\{ X \rightarrow Y, Y \rightarrow Z \} \models X \rightarrow Z$

Regole di inferenza per FD e MVD (2)

- (IR4) (Complementation rule per MVD):
  $\{ X \twoheadrightarrow Y \} \models \{ X \twoheadrightarrow (R - (X \cup Y)) \}$
- (IR5) (Augmentation rule per MVD):
  se $X \twoheadrightarrow Y$ e $W \supseteq Z$ allora $WX \twoheadrightarrow YZ$
- (IR6) (Transitive rule per MVD):
  $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z \} \models X \twoheadrightarrow (Z - Y)$
- (IR7) (Replication rule da FD a MVD):
  $\{X \rightarrow Y\} \models X \twoheadrightarrow Y$

Regole di inferenza per FD e MVD (3)

- (IR8) (Regola di unione per FD e MVD):
  se X -» Y e ∃W con le proprietà che
  1. W ∩ Y è vuoto
  2. W → Z
  3. Y ⊇ Z

  allora X → Z

- L'insieme delle regole di inferenza è corretto e completo:
  - Dato un insieme F di FD e MVD specificate su R = {A_1, A_2, …,A_n}, possiamo usare IR1 – IR8 per inferire l'insieme completo di tutte le dipendenze funzionali e multivalued.

Quarta Forma Normale (4NF)

Quarta Forma Normale

- **Definizione**:

  Uno schema di relazione R è detto essere in 4NF rispetto a un insieme di dipendenza F se, per ogni dipendenza multivalued non-trivial $X \twoheadrightarrow Y$ in F^+, X è una superchiave per R.

4NF: *Esempio*

- Nella relazione EMP risulta:
  - ENAME ↠ PNAME,
  - ENAME ↠ DNAME,

  ma ENAME non è una superchiave.

- Ciò implica che EMP non è in 4NF.

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

4NF: *Esempio* (2)

EMP

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John
Brown	W	Jim
Brown	X	Jim
Brown	Y	Jim
Brown	Z	Jim
Brown	W	Joan
Brown	X	Joan
Brown	Y	Joan
Brown	Z	Joan
Brown	W	Bob
Brown	X	Bob
Brown	Y	Bob
Brown	Z	Bob

EMP_PROJECTS

ENAME	PNAME
Smith	X
Smith	Y
Brown	W
Brown	X
Brown	Y
Brown	Z

EMP_DEPENDENTS

ENAME	DNAME
Smith	Anna
Smith	John
Brown	Jim
Brown	Joan
Brown	Bob

- EMP viene decomposta in EMP_PROJECTS e EMP_DEPENDENTS, che sono in 4NF:
  - ENAME ↠ PNAME è una MVD in EMP_PROJECTS
  - ENAME ↠ DNAME è una MVD banale in EMP_DEPENDENTS.

Nota sulle MVD non-trivial

- Relazioni contenenti MVD non-trivial tendono ad essere relazioni *“tutta chiave”*, nel senso che la chiave è formata da tutti gli attributi.

Decomposizione LJ in Relazioni 4NF

- Decomponendo uno schema R in $R_1 = (X \cup Y)$ e $R_2 = (R - Y)$ in base a una MVD $X \twoheadrightarrow Y$, la decomposizione gode della proprietà LJ (*condizione necessaria e sufficiente*).
- Proprietà LJ1': Gli schemi di relazione R_1 e R_2 formano una decomposizione LJ di R se e solo se:
  - $(R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$
    oppure, per simmetria se e solo se
  - $(R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$

Decomposizione LJ in relazioni 4NF

- Algoritmo 5
  Decomposizione di un schema in relazioni 4NF con la proprietà LJ:
  1. *porre* D = {R};
  2. *finché* esiste uno schema di relazione Q in D che non è in 4NF
     {
     scegliere uno schema di relazione Q in D che non è in 4NF;
     trovare una MVD non banale X ↠ Y in Q che viola 4NF;
     rimpiazzare Q in D con due schemi (Q – Y) e (X ∪ Y);
     }