Forme normali e normalizzazione

Paolo Atzeni

02/04/2007
(aggiornato 17/4/2007 con piccole correzioni)

Contenuti

- Riepilogo del capitolo 10 (“La normalizzazione”) del testo di Atzeni, Ceri, Paraboschi, Torlone
  Basi di dati McGraw-Hill, 1996-2006
- Approfondimenti e giustificazioni formali

Forme normali

- Una forma normale è una proprietà di una base di dati relazionale che ne garantisce la “qualità”, cioè l'assenza di determinati difetti
- Quando una relazione non è normalizzata:
  - presenta ridondanze,
  - si presta a comportamenti poco desiderabili durante gli aggiornamenti
- Le forme normali sono di solito definite sul modello relazionale, ma hanno senso in altri contesti, ad esempio il modello E-R

Normalizzazione

- Procedura che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale

- La normalizzazione va utilizzata come tecnica di verifica dei risultati della progettazione di una base di dati
- Non costituisce una metodologia di progettazione

Una relazione con anomalie

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

Anomalie

- Lo stipendio di ciascun impiegato è ripetuto in tutte le ennuple relative
  - ridondanza
- Se lo stipendio di un impiegato varia, è necessario andarne a modificare il valore in diverse ennuple
  - anomalia di aggiornamento
- Se un impiegato interrompe la partecipazione a tutti i progetti, dobbiamo cancellarlo
  - anomalia di cancellazione
- Un nuovo impiegato senza progetto non può essere inserito
  - anomalia di inserimento

Perché questi fenomeni indesiderabili?

- abbiamo usato un'unica relazione per rappresentare informazioni eterogenee
  - gli impiegati con i relativi stipendi
  - i progetti con i relativi bilanci
  - le partecipazioni degli impiegati ai progetti con le relative funzioni

Per studiare in maniera sistematica questi aspetti, è necessario introdurre un vincolo di integrità:
la dipendenza funzionale

Proprietà

- Ogni impiegato ha un solo stipendio (anche se partecipa a più progetti)
- Ogni progetto ha un bilancio
- Ogni impiegato in ciascun progetto ha una sola funzione (anche se può avere funzioni diverse in progetti diversi)

Dipendenza funzionale

- relazione r su R(X)
- due sottoinsiemi non vuoti Y e Z di X
- esiste in r una dipendenza funzionale (FD) da Y a Z se, per ogni coppia di ennuple t_1 e t_2 di r con gli stessi valori su Y, risulta che t_1 e t_2 hanno gli stessi valori anche su Z

Notazione

$Y \rightarrow Z$

- Esempi:

$$\text{Impiegato} \rightarrow \text{Stipendio}$$
$$\text{Progetto} \rightarrow \text{Bilancio}$$
$$\text{Impiegato Progetto} \rightarrow \text{Funzione}$$

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

Impiegato → Stipendio

Progetto → Bilancio

Impiegato Progetto → Funzione

Un inciso: altre FD, particolari

- Impiegato Progetto $\rightarrow$ Progetto

- Si tratta però di una FD "banale" (sempre soddisfatta)

- $Y \rightarrow A$ è non banale se A non appartiene a Y
- $Y \rightarrow Z$ è non banale se nessun attributo in Z appartiene a Y

Le anomalie sono legate ad alcune FD

- gli impiegati hanno un unico stipendio

  $$\text{Impiegato} \rightarrow \text{Stipendio}$$

- i progetti hanno un unico bilancio

  $$\text{Progetto} \rightarrow \text{Bilancio}$$

Non tutte le FD causano anomalie

- In ciascun progetto, un impiegato svolge una sola funzione

$$\text{Impiegato Progetto} \rightarrow \text{Funzione}$$

- Il soddisfacimento è più “semplice”, perché Impiegato Progetto è chiave

FD e anomalie

- La terza FD corrisponde ad una chiave e non causa anomalie
- Le prime due FD non corrispondono a chiavi e causano anomalie

- La relazione contiene alcune informazioni legate alla chiave e altre ad attributi che non formano una chiave
- Le anomalie sono causate dalla presenza di concetti eterogenei:
  - proprietà degli impiegati (lo stipendio)
  - proprietà di progetti (il bilancio)
  - proprietà della chiave Impiegato Progetto

Forma normale di Boyce e Codd (BCNF)

- Una relazione r è in forma normale di Boyce e Codd se, per ogni dipendenza funzionale (non banale) $X \rightarrow Y$ definita su di essa, X contiene una chiave K di r

- La forma normale richiede che i concetti in una relazione siano omogenei (solo proprietà direttamente associate alla chiave)

Che facciamo se una relazione non soddisfa la BCNF?

- La rimpiazziamo con altre relazioni che soddisfano la BCNF

Come?

- Decomponendo sulla base delle dipendenze funzionali, al fine di separare i concetti

Impiegato	Stipendio
Rossi	20
Verdi	35
Neri	55
Mori	48
Bianchi	48

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Verdi	Giove	progettista
Verdi	Venere	progettista
Neri	Venere	direttore
Neri	Giove	consulente
Neri	Marte	consulente
Mori	Marte	direttore
Mori	Venere	progettista
Bianchi	Venere	progettista
Bianchi	Giove	direttore

Progetto	Bilancio
Marte	2
Giove	15
Venere	15

Procedura intuitiva di normalizzazione

- Non valida in generale, ma solo nei "casi semplici"
  - Per ogni dipendenza $X \rightarrow Y$ che viola la BCNF, definire una relazione su XY ed eliminare Y dalla relazione originaria

Non sempre così facile

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato → Sede
Progetto → Sede

Decomponiamo sulla base delle dipendenze

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Proviamo a ricostruire

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

Diversa dalla relazione di partenza!

Decomposizione senza perdita

- Una relazione r si decompone senza perdita su X_1 e X_2 se il join delle proiezioni di r su X_1 e X_2 è uguale a r stessa (cioè non contiene ennuple spurie)
- La decomposizione senza perdita è garantita se gli attributi comuni contengono una chiave per almeno una delle relazioni decomposte
- Ma, attenzione:
  - Se le relazioni sono tre o più, come si generalizza la proprietà?

La decomposizione senza perdita non basta

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

Impiegato $\rightarrow$ Sede

Progetto $\rightarrow$ Sede

Un altro problema

- Supponiamo di voler inserire una nuova ennupla che specifica la partecipazione dell'impiegato Neri, che opera a Milano, al progetto Marte

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

$$\text{Impiegato} \rightarrow \text{Sede}$$
$$\text{Progetto} \rightarrow \text{Sede}$$

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Neri	Marte	Milano

Conservazione delle dipendenze

- Una decomposizione conserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti

- Progetto $\rightarrow$ Sede non è conservata

Qualità delle decomposizioni

- Una decomposizione dovrebbe sempre soddisfare:
  - la decomposizione senza perdita, che garantisce la ricostruzione delle informazioni originarie
  - la conservazione delle dipendenze, che garantisce il mantenimento dei vincoli di integrità originari

Possibile obiettivo

- Data una relazione
  - Verificare se è in BCNF
  - In caso negativo, ottenere una decomposizione
    - senza perdita
    - che conserva le dipendenze
    - in cui ciascuna delle relazioni è in BCNF

- In effetti, l'obiettivo BCNF non è sempre raggiungibile

Una relazione non normalizzata

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

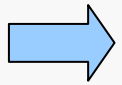
Progetto Sede → Dirigente
Dirigente → Sede

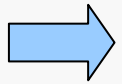
La decomposizione è problematica

- Progetto Sede $\rightarrow$ Dirigente coinvolge tutti gli attributi e quindi nessuna decomposizione può preservare tale dipendenza
- quindi in alcuni casi la BCNF “non è raggiungibile”
- Per questo si studia anche la terza forma normale (3NF)

I punti lasciati aperti

- Quali sono le dipendenze definite su una relazione?
- Decomposizione senza perdita, come si verifica in generale?





Forma normale di Boyce e Codd (BCNF)

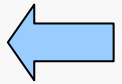
- Una relazione r è in forma normale di Boyce e Codd se, per ogni dipendenza funzionale (non banale) $X \rightarrow Y$ definita su di essa, X contiene una chiave K di r

Conservazione delle dipendenze

- Una decomposizione conserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti
- Progetto $\rightarrow$ Sede non è conservata

Dipendenze “implicate”

- R(ABC)
- $A \rightarrow B$, $B \rightarrow C$, $A \rightarrow C$
  - $A \rightarrow C$ è derivabile (più precisamente implicata) da $A \rightarrow B$, $B \rightarrow C$; la debbo considerare
    - nel verificare la BCNF?
    - nel valutare la conservazione delle dipendenze?
  - In generale, come si studia l’implicazione?



Decomposizione senza perdita

- Una relazione r si decompone senza perdita su X_1 e X_2 se il join delle proiezioni di r su X_1 e X_2 è uguale a r stessa (cioè non contiene ennuple spurie)
- La decomposizione senza perdita è garantita se gli attributi comuni contengono una chiave per almeno una delle relazioni decomposte
- Ma, attenzione:
  - Se le relazioni sono tre o più, come si generalizza la proprietà?

Per approfondire

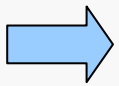
- Teoria delle dipendenze funzionali
  - Implicazione
  - Caratterizzazione della decomposizione senza perdita
- Algoritmi per la normalizzazione

Vincoli di integrità, terminologia

- Dato uno schema, un vincolo di integrità è un predicato definito sulle istanze dello schema
- Un vincolo può essere soddisfatto o violato da una specifica istanza
- Ad uno schema $\mathbf{R}$, associamo di solito un insieme di vincoli Γ
- Un'istanza $\mathbf{r}$ di $\mathbf{R}$ è consistente (o legale o valida) se soddisfa tutti i vincoli in Γ
- Esistono classi di vincoli, ognuna con la sua sintassi e semantica
  - Consideriamo per ora le dipendenze funzionali (definite su una relazione), di cui conosciamo sintassi e semantica

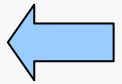
Implicazione di vincoli

- Un insieme di vincoli Γ implica un vincolo γ se ogni istanza che soddisfa Γ soddisfa anche γ
  - Esempio: dato lo schema $R(ABC)$, l'insieme di dipendenze funzionali $\{A \rightarrow B , B \rightarrow C\}$ implica la dipendenza $A \rightarrow C$
  - Teorema 1 (semplice)
    - Γ implica γ se e solo se non esiste un'istanza che soddisfa Γ e non soddisfa γ
    - Γ non implica γ se e solo se esiste un'istanza che soddisfa Γ e non soddisfa γ
- Un vincolo è banale (in inglese "trivial") se è soddisfatto da ogni istanza
  - Esempio: la dipendenza $AB \rightarrow B$ è banale
  - Teorema 2:
    - un vincolo è banale se e solo se è implicato dall'insieme vuoto di vincoli



Nota

- Per tutti i teoremi “semplici” è importante sviluppare, per esercizio, la dimostrazione.



Il problema dell'implicazione

- "Dato un insieme di vincoli Γ e un vincolo γ, stabilire se Γ implica γ"
- In generale, è un problema di logica matematica, ma noi lo affrontiamo in casi specifici, in cui ci sono soluzioni semplici ed efficienti
- Supponiamo che Γ e γ siano dipendenze funzionali (FD)

Notazione

- Le prime lettere dell'alfabeto A, B, C indicano attributi
- Le ultime lettere X, Y, Z indicano insiemi attributi
- Nell'indicare insiemi di attributi omettiamo parentesi graffe e unioni
  - Invece di scrivere $X \cup \{A, B\}$ scriviamo XAB

Una proprietà interessante

- Siano:
  - $R(U)$ uno schema di relazione
  - X e Y sottoinsiemi qualunque di U
  - F un insieme di FD su attributi in U
- Teorema 3
  - Un'istanza r di R soddisfa $X \rightarrow Y$ se e solo se, per ogni $A \in Y$, essa soddisfa $X \rightarrow A$
  - L'insieme F implica $X \rightarrow Y$ se e solo se, per ogni $A \in Y$, esso implica $X \rightarrow A$
- Quindi
  - "Senza perdita di generalità, possiamo limitarci a considerare FD il cui secondo membro è un singolo attributo"

La chiusura di un insieme di attributi

- $R(U)$ uno schema di relazione
- X sottoinsieme qualunque di U
- F un insieme di FD su attributi in U

- X^+_F (chiusura di X rispetto a F):
  - $\{ A \mid A \in U$ e F implica $X \rightarrow A \}$
- Teorema 4
  - $X \subseteq X^+_F$
- Teorema 5
  - F implica $X \rightarrow Y$ se e solo se $Y \subseteq X^+_F$
- Quindi:
  - Per verificare se F implica $X \rightarrow Y$ è sufficiente calcolare X^+_F e verificare se $Y \subseteq X^+_F$

Algoritmo per il calcolo della chiusura

- **R(U)** uno schema di relazione (**U** insieme finito di attributi …)
- **X** sottoinsieme qualunque di **U**
- **F** un insieme di FD su attributi in **U**

- X^+_F (chiusura di **X** rispetto a **F**):
  - $\{ A \mid A \in U$ e **F** implica $X \rightarrow A \}$

insiemeDiAttributi calcolaChiusura(X,F)
{ insiemeDiAttributi chiusura = X;
repeat
if c'è una $Z \rightarrow Y \in F$ tale che $Z \subseteq$ **chiusura**
then chiusura = chiusura $\cup$ **Y**
until chiusura non è cambiato nell'ultima esecuzione del ciclo**;**
return chiusura
}

Algoritmo per la chiusura: correttezza

- Teorema
  - Per ogni **X** e **F**, **CalcolaChiusura(X,F)** termina e restituisce X^+_F
- Dimostrazione
  1. Terminazione:
     - Si rimane nel ciclo solo se nell'iterazione appena conclusa si è aggiunto almeno un attributo all'insieme **chiusura**
     - Poiché il numero di attributi è finito, il ciclo può essere ripetuto solo un numero finito di volte
  2. Correttezza: se **fin** è valore finale di **chiusura** (quindi il risultato)
     - $\mathbf{fin} \subseteq X^+_F$
     - $X^+_F \subseteq \mathbf{fin}$

Algoritmo per la chiusura: correttezza, 2

- $\mathbf{fin} \subseteq \mathbf{X}^+_F$
  - Per induzione completa sul numero di modifiche a **chiusura**:
    - siano $\mathbf{X_0, X_1, X_2, \ldots, X_k}$ i valori che **chiusura** assume durante l'esecuzione (e quindi $\mathbf{X = X_0}$ e $\mathbf{X_k = fin}$)
    - dimostriamo che, per ogni **i**, si ha $\mathbf{X_i \subseteq X^+_F}$
  - Passo base, $\mathbf{i = 0}$: per il teorema 4, sappiamo che $\mathbf{X \subseteq X^+_F}$ e quindi, poiché $\mathbf{X = X_0}$,abbiamo $\mathbf{X_i \subseteq X^+_F}$

Algoritmo per la chiusura: correttezza, 3

- Induzione: supposto $X_{i-1} \subseteq X^+_F$, dimostriamo $X_i \subseteq X^+_F$ per $i > 0$; la FD utilizzata è $Z \rightarrow Y$ e quindi $Z \subseteq X_{i-1}$ e $X_i - X_{i-1} \subseteq Y$
  - È sufficiente dimostrare che $Y \subseteq X^+_F$ cioè che
  - per ogni $A \in Y$, si ha che F implica $X \rightarrow A$ cioè che
  - per ogni $A \in Y$, tutte le relazioni che soddisfano F soddisfano anche $X \rightarrow A$
  - Consideriamo una qualunque relazione r su $R(U)$ che soddisfi F e mostriamo che soddisfa $X \rightarrow A$: ogni coppia di ennuple con uguali valori su X deve avere uguali valori su A (il resto è ok); consideriamo $t_1, t_2 \in r$ con $t_1[X] = t_2[X]$; per l'ipotesi induttiva, $Z \subseteq X_{i-1} \subseteq X^+_F$ e quindi $t_1[Z] = t_2[Z]$ (altrimenti, $X \rightarrow Z$ sarebbe violata); ma allora, poiché r soddisfa F e $Z \rightarrow Y \in F$, si ha che r soddisfa $Z \rightarrow Y$ e quindi $t_1[Y] = t_2[Y]$; dato che $A \in Y$, abbiamo che r soddisfa $X \rightarrow A$

Algoritmo per la chiusura: correttezza, 4

- $X^+_F \subseteq$ **fin**; quindi dobbiamo dimostrare che,
  - se $A \in X^+_F$ allora $A \in$ **fin** o, in modo equivalente
  - se **F** implica $X \rightarrow A$ allora $A \in$ **fin** o, in modo equivalente
  - se $A \notin$ **fin** allora **F** non implica $X \rightarrow A$
- Con un controesempio: una relazione che soddisfa **F** e non $X \rightarrow A$

r	**fin**	**U - fin**
	0 0 … 0	0 0 … 0
	0 0 … 0	1 1 … 1

  - **r** non soddisfa $X \rightarrow A$ (perché $X \subseteq$ **fin** e $A \notin$ **fin**)
  - per ogni $V \rightarrow W \in F$
    - se **V** non è contenuto in **fin**, allora **r** soddisfa $V \rightarrow W$
    - se $V \subseteq$ **fin**, allora l'algoritmo prima o poi considera $V \rightarrow W$ e quindi $W \subseteq$ **fin** e quindi le due ennuple sono uguali su **W** e quindi **r** soddisfa $V \rightarrow W$

Algoritmo per la chiusura, complessità

- Con una implementazione immediata:
  - $O(|F| \times |U|)$
- Con una implementazione più sofisticata
  - $O(||F||)$

  dove $||F||$ è il numero di simboli in F:

  $$||F|| = \Sigma_{i=1,\ldots,k} (|V_i| + |W_i|)$$

  se $F = \{V_1 \rightarrow W_1 , V_2 \rightarrow W_2 , \ldots , V_k \rightarrow W_k \}$
- "Il problema dell'implicazione di FD è risolubile in tempo lineare"

Forma normale di Boyce e Codd (BCNF)

- Una relazione r è in forma normale di Boyce e Codd se, per ogni dipendenza funzionale (non banale) $X \rightarrow Y$ definita su di essa, X contiene una chiave K di r

Chiavi e dipendenze funzionali

- $R(U)$ uno schema di relazione
- K sottoinsieme di U
- F un insieme di FD su attributi in U

- K è superchiave per una relazione r su $R(U)$ se e solo se
  - r soddisfa $K \rightarrow U$ o, equivalentemente
  - r soddisfa $K \rightarrow U - K$
- Dato F, possiamo dire che K è superchiave a livello di schema (cioè che è superchiave per tutte le relazioni che soddisfano F), se e solo se:
  - F implica $K \rightarrow U$, cioè se e solo se
  - $K^+_F = U$

Chiavi, problemi interessanti

- Verifica:
  - dati $R(U)$, F e $X \subseteq U$
    - X è chiave (cioè superchiave minimale)?
    - Basta verificare se
      - $X^+_F = U$
      - $(X\text{-}A)^+_F \neq U$ per ogni $A \in X$
    - Si esegue $|X|+1$ volte l'algoritmo di chiusura

Chiavi, problemi interessanti, 2

- Ricerca;
  - diversi sottoproblemi
    - ricerca di una chiave
    - ricerca di tutte le chiavi
    - verifica dell’unicità della chiave
    - ricerca della chiave minima (con meno attributi)

Ricerca di una chiave

- **R(U)**, **F** (con **U=A_1,A_2,…,A_n**)
- Osservazioni
  - **U** è superchiave
  - Ogni superchiave o è chiave oppure contiene propriamente (almeno) una chiave

```
insieme TrovaChiave(U,F)
{   insieme super = U;
    for(int i =1; i<=n; i++) {
            if ( (super – A_i )+_F = U ) super = super – A_i ;
    }
    return super
}
```

Ricerca di una chiave, esempio

- **R(ABCDEGH)**
- **F = {GD → B , BG → D , BD → G, G → H, A → C}**
- **A B C D E G H**

- **A D E G** è una chiave
- Se ci sono più chiavi, l'algoritmo produce risultati diversi, a seconda dell'ordine in cui considera gli attributi
- Altre chiavi:
  - **A B D E**
  - **A B E G**

Quante chiavi può avere una relazione?

- Anche un numero esponenziale!
- Esempio
  - $R(A_1A_2\ldots A_n\, B_1B_2\ldots B_n)$
  - $F = \{A_i \rightarrow B_i\,,\, B_i \rightarrow A_i \mid i = 1, 2, \ldots, n\}$
  - Chiavi:
    - $A_1A_2\ldots A_n$, $B_1A_2\ldots A_n$, $A_1B_2\ldots A_n$, …
    - ogni chiave ha n attributi, k da $A_1A_2\ldots A_n$ e $n\text{-}k$ da $B_1B_2\ldots B_n$
    - ci sono tante chiavi quanti sono i sottoinsiemi di $\{1, 2, \ldots, n\}$ e cioè 2^n

Ricerca di tutte le chiavi

- In generale il problema ha complessità esponenziale!
- Però i casi pratici sono quasi sempre più semplici:
  - $F = \{X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_n \rightarrow A_n\}$ (senza perdita di generalità abbiamo dipendenze con …)
    - Gli attributi che non compaiono a secondo membro ($U - A_1A_2\ldots A_n$) appartengono a tutte le chiavi
    - Gli attributi in $(U - A_1A_2\ldots A_n)^+_F - (U - A_1A_2\ldots A_n)$ non appartengono a nessuna chiave
    - La relazione ha una sola chiave se e solo se:
      - $(U - A_1A_2\ldots A_n)^+_F = U$
    - Gli attributi che appaiono solo a secondo membro ($A_1A_2\ldots A_n - X_1\ X_2\ \ldots\ X_n$) non appartengono a nessuna chiave

Ricerca di tutte le chiavi, esempio

- **R(ABCDEGHL)**
- **{ABC → BCD, AGE → EH , ABD → ACD, H → B, EA → B, DC → L}**
- **{ABC → D, AGE → H , ABD → C , H → B, EA → B, DC → L}**
- Attributi che non compaiono a secondo membro:
  - **AEG** in tutte le chiavi
- **$(AEG)^{+}_{F} = ABEGH$**
  - **BH** in nessuna chiave
- Attributi che compaiono solo a secondo membro:
  - **L** in nessuna chiave
- Analizzando i casi possibili (sono rimasti **CD**) troviamo tutte le chiavi:
  - **ADEG**
  - **ACEG**

Ricerca della chiave minima

- Le chiavi sono minimali per definizione:
  - data una chiave $\mathbf{K}$, non esiste un'altra chiave $\mathbf{K'} \subset \mathbf{K}$
- Potrebbe interessare cercare chiavi minime:
  - chiavi $\mathbf{K}$, tali che non esista un'altra chiave $\mathbf{K'}$ con meno attributi: $|\mathbf{K'}| < |\mathbf{K}|$
- Purtroppo:
  - Il problema di trovare chiavi con cardinalità minima è NP-completo (riduzione ad un problema di vertex cover)
- Quindi:
  - In generale non esiste un algoritmo efficiente per trovare la chiave minima

Vincoli: chiusura, equivalenza e copertura

- **chiusura** Γ_C^+ di Γ rispetto ad una classe C :
  - $\{ \gamma \mid \gamma$ è un vincolo della classe C e Γ implica $\gamma \}$
- Il riferimento alla classe è fondamentale, ma viene spesso omesso perché implicito (ad esempio, noi parleremo quasi solo di FD e quindi la classe sarà quella delle FD)
- Γ_1 e Γ_2 sono **equivalenti** (e ciascuno è una **copertura** dell'altro) se:
  - $\Gamma_1^+ = \Gamma_2^+$
- Teorema
  - $\Gamma_1^+ = \Gamma_2^+$ se e solo se
    - Γ_1 implica ciascun vincolo in Γ_2 e
    - Γ_2 implica ciascun vincolo in Γ_1

Forma normale di Boyce e Codd (BCNF)

- Una relazione r è in forma normale di Boyce e Codd se, per ogni dipendenza funzionale (non banale) $X \rightarrow Y$ definita su di essa, X contiene una chiave K di r

- $R(U)$, F
- Quali dipendenze dobbiamo considerare?
  - quelle in F o quelle in F^+ ?
- Può succedere che
  - X contiene una chiave per ogni dipendenza (non banale) $X \rightarrow Y$ in F e ciò non è vero per una dipendenza in F^+ ?

Indipendenza della BCNF dalla copertura

- $R(U)$, F
- Teorema:
  - Se per ogni $Y \rightarrow B$ non banale in F abbiamo che Y contiene una chiave di R , allora lo stesso vale per ogni FD in F^+
- Dimostrazione: supponiamo che Y è superchiave di R per ogni $Y \rightarrow B$ non banale in F. Prendiamo $X \rightarrow A$ (non banale) in F^+ (e cerchiamo di dimostrare che $X^+_F = U$). Poiché $X \rightarrow A$ è in F^+ se calcoliamo X^+_F con l'algoritmo **calcolaChiusura**, ad un certo punto una qualche FD $Z \rightarrow A$ in F ci fa aggiungere A; ma questo può accadere solo se $Z \subseteq X^+_F$; e quindi $X \rightarrow Z \in F^+$ ma allora, poiché $Z \rightarrow A \in F$ abbiamo $Z^+_F = U$ e quindi $Z \rightarrow U \in F^+$ e quindi, per transitività, $X \rightarrow U \in F^+$

BCNF, definizione formale

- Uno schema di relazione $R(U)$ con le dipendenze F è in BCNF se per ogni FD (non banale) $X \rightarrow Y$ in F, il primo membro X contiene una chiave K di r

BCNF verifica e altro

- Sappiamo quindi verificare se una relazione soddisfa la BCNF
- Ma se non la soddisfa, come decomponiamo?
  - Vogliamo (abbiamo visto informalmente a suo tempo)
    - Decomposizione senza perdita
    - Conservazione delle dipendenze
  - Perché?

Relative information capacity

- A notion used to compare database schemas
  - The **information capacity** of a schema is the ability of the schema to hold information
- The information capacity of two schemas can be
  - "the same" (**information capacity equivalence**)
  - comparable (**information capacity dominance**)
  - incomparable
- The approach is based on mappings between allowed database instances

Dominance and equivalence

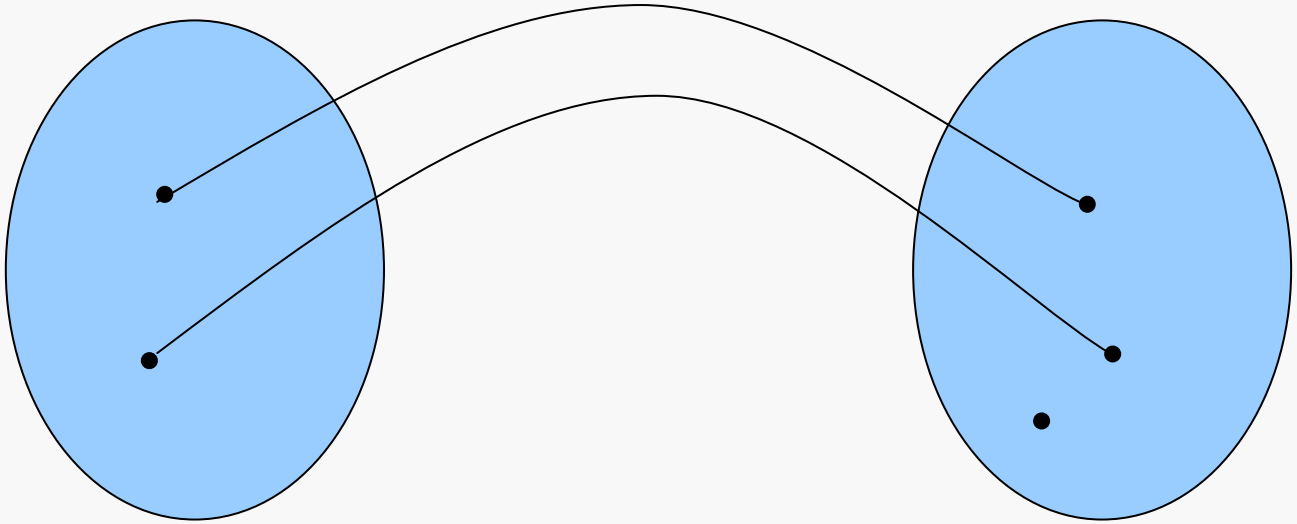
- Schemas S1, S2
- S2 **dominates** S1
  - Whatever can be represented by S1 can also be represented by S2
- S1 and S2 are **equivalent**
  - S1 dominates S2 and S2 dominates S1
  - Whatever can be represented by S1 can be represented by S2 and viceversa

Information capacity dominance

- Schemas S1, S2
- The basic idea
  - Whatever can be represented by S1 can also be represented by S2
- More in detail:
  - For every instance i1 of S1, there is an instance i2 of S2, **with the same information**

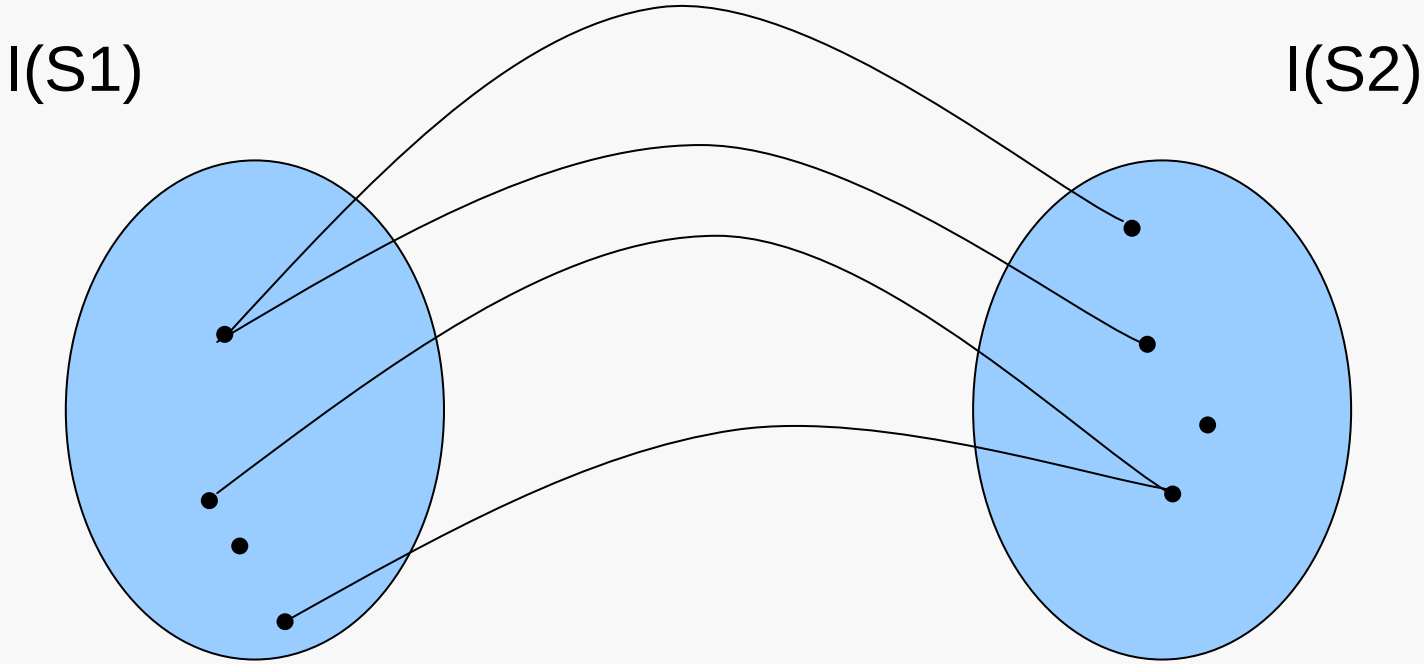
I(S1)

I(S2)



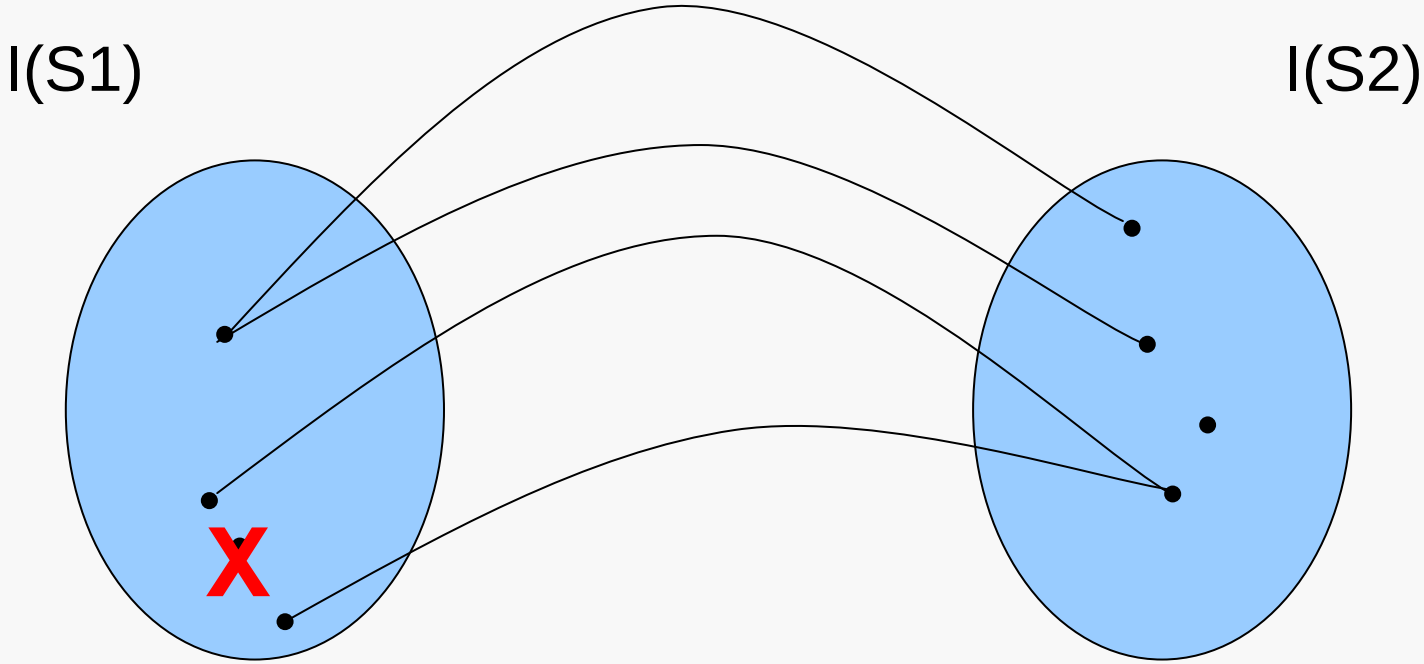
Comparing information capacity

- Schemas S1 and S2, with their legal instances I(S1) and I(S2)
- A mapping f (binary relation) from I(S1) and I(S2)



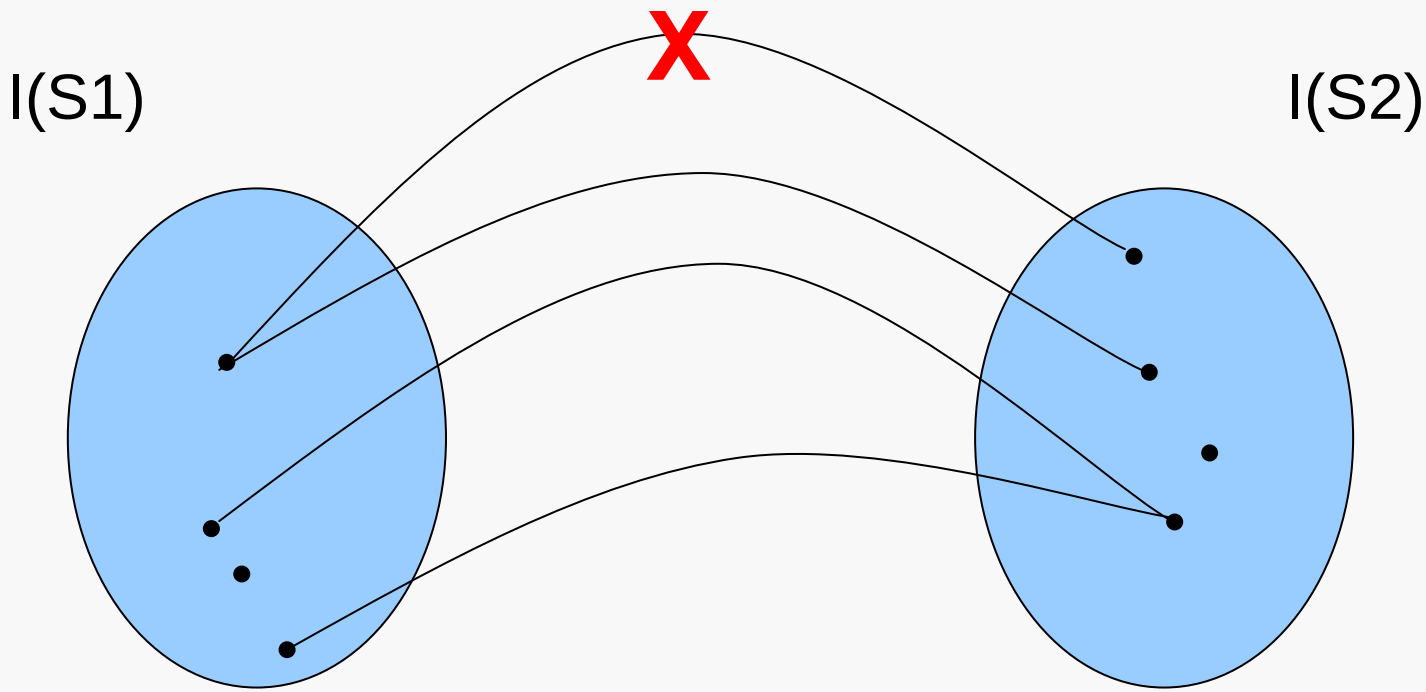
Properties of the mapping, 1

- f is **total** if it is defined on every element of I(S1)



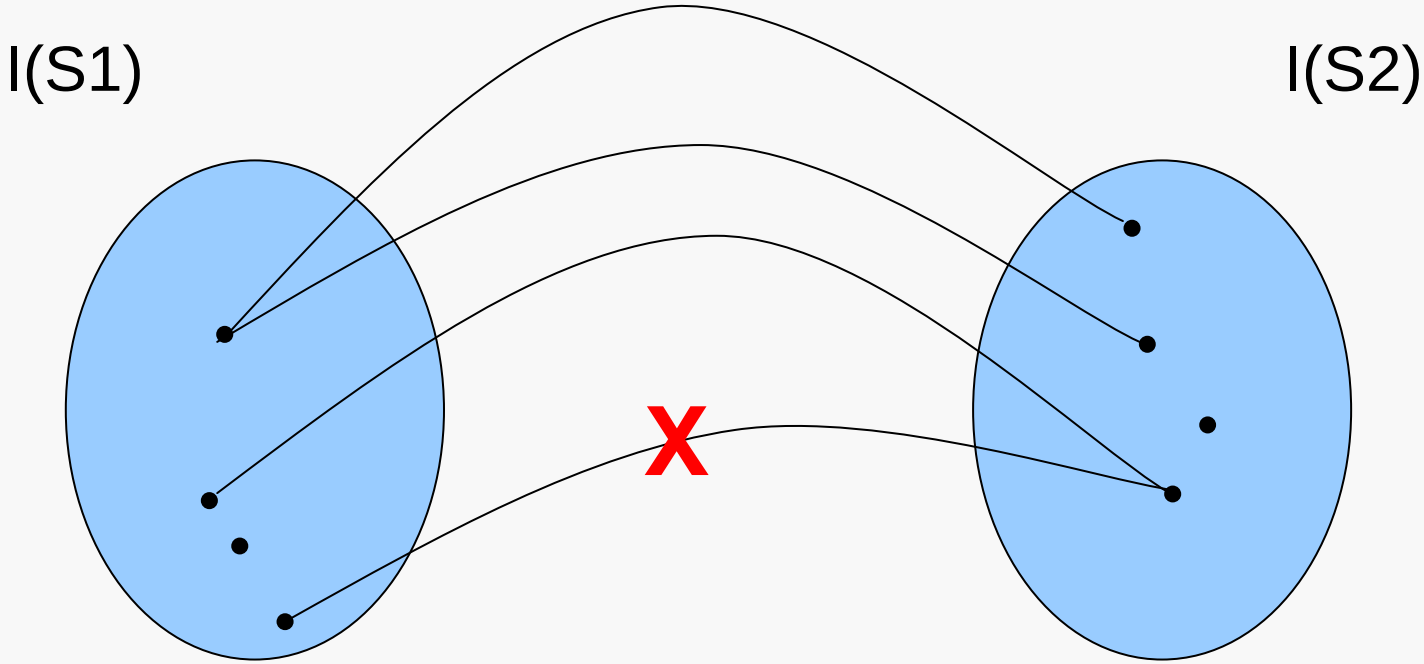
Properties of the mapping, 2

- f is **functional** if for every element of I(S1) there is at most one associated element in I(S2)



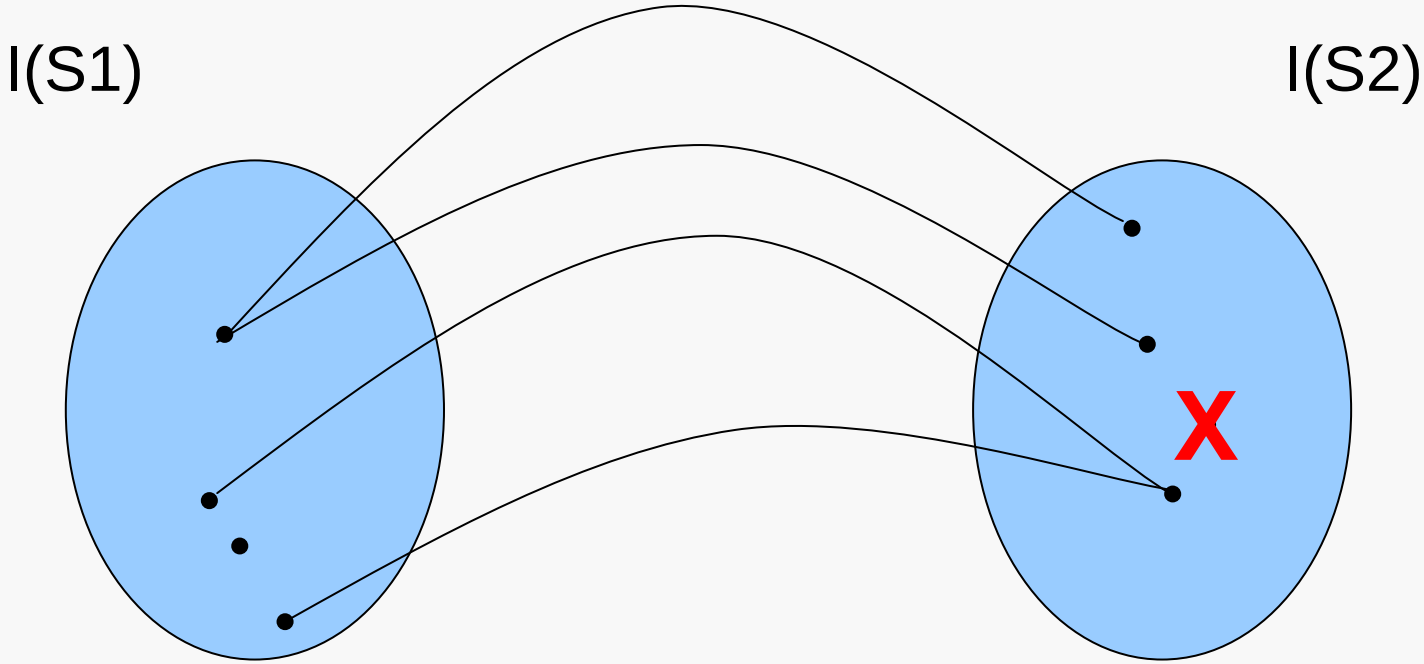
Properties of the mapping, 3

- f is **injective** if no two elements of I(S1) are associated with the same element of I(S2)
  - the inverse is functional



Properties of the mapping, 4

- f is **surjective** if every element of I(S2) is associated with at least one element of I(S1)
  - the inverse of f is total



Information capacity and properties of the mapping

- S2 **dominates** S1
  - Whatever can be represented by S1 can also be represented by S2
- If the mapping is functional, total and injective, then
  - **S2 dominates S1**
- Note:
  - This implies that it is a bijection between

    I(S1) and a subset of I(S2)

Information capacity and properties of the mapping, 2

- S1 and S2 are **equivalent**
  - S1 dominates S2 and S2 dominates S1
  - Whatever can be represented by S1 can be represented by S2 and viceversa
- If the mapping is functional, total, injective, and surjective then
  - **S2 and S1 are equivalent**
- Note:
  - This implies that it is a bijection between

    I(S1) and I(S2)

Il mapping, nella decomposizione

- In generale, nulla è detto sul mapping, ma nel nostro caso possiamo sceglierne uno specifico
  - Proiezioni (da S1 a S2)
  - Join delle proiezioni (da S2 a S1)

Impiegato	Stipendio
Rossi	20
Verdi	35
Neri	55
Mori	48
Bianchi	48

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Verdi	Giove	progettista
Verdi	Venere	progettista
Neri	Venere	direttore
Neri	Giove	consulente
Neri	Marte	consulente
Mori	Marte	direttore
Mori	Venere	progettista
Bianchi	Venere	progettista
Bianchi	Giove	direttore

Progetto	Bilancio
Marte	2
Giove	15
Venere	15

Proprietà del mapping, nella decomposizione

- $R(X)$ con insieme di FD F
- $R_1(X_1), \ldots, R_n(X_n)$ (con $X_1X_2\ldots X_n=X$) ognuno con il relativo insieme di FD, $F_1, \ldots, F_n$
- Mapping
  - Proiezioni su $X_1, X_2, \ldots, X_n$
  - Join naturale delle n relazioni
- Proprietà
  - Totale?
  - Funzionale?
  - Iniettivo?
  - Suriettivo?

Suriettività del mapping

- Decomponiamo perché vogliamo istanze che altrimenti non sarebbero ammissibili

Impiegato	Stipendio
Rossi	54
Verdi	62
Bruni	42

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Rossi	Venere	direttore
Verdi	Venere	tecnico
Verdi	Giove	consulente

Progetto	Bilancio
Marte	200
Venere	450
Giove	150
Plutone	250

- Quindi non possiamo richiedere la suriettività del mapping

Proprietà del mapping, nella decomposizione

- $R(X)$ con insieme di FD F
- $R_1(X_1), \ldots, R_n(X_n)$ (con $X_1 X_2 \ldots X_n = X$) ognuno con il relativo insieme di FD, $F_1, \ldots, F_n$
- Mapping
  - Proiezioni su $X_1, X_2, \ldots, X_n$
  - Join naturale delle n relazioni
- Proprietà
  - Totale? Sì, lo è sempre
  - Funzionale? Sì, lo è sempre
  - Iniettivo? Talvolta, e vorremmo imporlo
  - Suriettivo?

Ci serve un'altra proprietà

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato → Sede
Progetto → Sede

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato → Sede

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

nessuna FD

Il problema

- Supponiamo di voler inserire una nuova ennupla che specifica la partecipazione dell'impiegato Neri, che opera a Milano, al progetto Marte

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

Impiegato → Sede

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Neri	Marte	Milano

In altri termini

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Neri	Marte	Milano

non soddisfa F

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano
Neri	Milano

soddisfa F1

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

soddisfa F2

Proprietà del mapping

- Deve trasformare istanze legali di $R(X)$ in istanze legali di $R_1(X_1), \ldots, R_n(X_n)$
- Non deve trasformare istanze illegali di $R(X)$ in istanze legali di $R_1(X_1), \ldots, R_n(X_n)$ cioè
  - deve trasformare istanze illegali di $R(X)$ in istanze illegali di $R_1(X_1), \ldots, R_n(X_n)$

In conclusione, la definizione desiderata

- $R_1(X_1), \ldots, R_n(X_n)$ (con $F_1, \ldots, F_n$) è decomposizione adeguata di $R(X)$ (con F) se il mapping "PJ"
  - è una biiezione fra le istanze consistenti di $R(X)$ e quelle consistenti di $R_1(X_1), \ldots, R_n(X_n)$ ottenibili per proiezione da istanze di $R(X)$
  - trasforma istanze (non) consistenti di $R(X)$ istanze (non) consistenti di $R_1(X_1), \ldots, R_n(X_n)$

Decomposizione adeguata, con FD

- $R_1(X_1), \ldots, R_n(X_n)$ (con $F_1, \ldots, F_n$) è decomposizione adeguata di $R(X)$ (con F) se e solo se:
  - $r_1(X_1), \ldots, r_n(X_n)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - $F_1 \cup \ldots \cup F_n$ è equivalente a F
- Dimostrazione
  - La decomposizione senza perdita è cns per la biiezione del mapping
  - Il resto …

Decomposizione adeguata, con FD, 2

- Se il mapping PJ trasforma istanze (non) consistenti di $R(X)$ in istanze (non) consistenti di $R_1(X_1), \ldots, R_n(X_n)$ allora $F_1 \cup \ldots \cup F_n$ è equivalente a F
  - Se … allora $F_1 \cup \ldots \cup F_n$ implica F
    - Consideriamo una $X \rightarrow A \in F$ e mostriamo che è implicata da $F_1 \cup \ldots \cup F_n$; se così non fosse, esisterebbe una relazione r su $R(X)$ che soddisfa $F_1 \cup \ldots \cup F_n$ e non soddisfa F ; ma allora considerando le proiezioni di r avremmo una istanza non consistente (viola F) le cui proiezioni sono consistenti (se r soddisfa $F_1 \cup \ldots \cup F_n$ allora ciascuna proiezione soddisfa la rispettiva F_i)
  - Se … allora F implica $F_1 \cup \ldots \cup F_n$
    - Simile (scambiando F e $F_1 \cup \ldots \cup F_n$)

Decomposizione adeguata, con FD, 3

- Se $F_1 \cup \ldots \cup F_n$ è equivalente a F allora il mapping PJ trasforma istanze (non) consistenti di $R(X)$ istanze (non) consistenti di $R_1(X_1), \ldots, R_n(X_n)$
  - Se $F_1 \cup \ldots \cup F_n$ è equivalente a F allora PJ trasforma istanze consistenti in istanze consistenti
    - Consideriamo una istanza consistente r e mostriamo che la sua proiezione $r_1, \ldots, r_n$ è consistente; supponiamo che una r_i violi una dipendenza in F_i; ma allora i valori che causano la violazione compaiono anche in r e quindi r viola tale dipendenza e quindi, poiché $F_1 \cup \ldots \cup F_n$ è equivalente a F allora r viola F, contro l'ipotesi
  - Se $F_1 \cup \ldots \cup F_n$ è equivalente a F allora PJ trasforma istanze non consistenti in non istanze consistenti
    - Simile

Decomposizione adeguata, con FD

- Abbiamo la caratterizzazione
- $R_1(X_1), \ldots, R_n(X_n)$ (con $F_1, \ldots, F_n$) è decomposizione adeguata di $R(X)$ (con F) se e solo se:
  - $r_1(X_1), \ldots, r_n(X_n)$ è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - $F_1 \cup \ldots \cup F_n$ è equivalente a F
- C'è ancora una cosa che non sappiamo verificare

Decomposizione senza perdita, con FD

- Cominciamo con il caso binario
  - $R(X)$ con F
  - $X_1 X_2 = X$

Teorema

- $r_1(X_1), r_2(X_2)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - se e solo se F implica almeno una delle FD (dove $X_{12} = X_2 \cap X_2$):
    - $X_{12} \rightarrow X_1$
    - $X_{12} \rightarrow X_2$
  - se e solo se X_{12} è chiave per una delle due relazioni decomposte
  - se e solo se almeno una fra X_1 e X_2 è superchiave per R cioè se F implica almeno una delle FD:
    - $X_1 \rightarrow X$
    - $X_2 \rightarrow X$

Decomposizione senza perdita, esempi

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Rossi	Venere	Roma
Verdi	Venere	Roma
Bruni	Giove	Milano

Impiegato → Sede
Progetto → Sede

Impiegato	Progetto
Rossi	Marte
Rossi	Venere
Verdi	Venere
Bruni	Giove

Impiegato	Sede
Rossi	Roma
Verdi	Roma
Bruni	Milano

OK

Impiegato	Sede
Rossi	Roma
Verdi	Roma
Bruni	Milano

Progetto	Sede
Marte	Roma
Venere	Roma
Giove	Milano

NO!

Attenzione

- $r_1(X_1), r_2(X_2)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - se e solo se X_{12} è chiave per una delle due relazioni decomposte

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Bruni	Venere	Roma
Rossi	Venere	Roma
Bruni	Marte	Roma

Impiegato → Sede
Progetto → Sede

Impiegato	Sede
Rossi	Roma
Bruni	Roma

Progetto	Sede
Marte	Roma
Venere	Roma

OK

Attenzione, 2

- ~~Non~~ è ~~quindi~~ vero che
- $r_1(X_1), r_2(X_2)$ r è decomposizione senza perdita di $r(X)$
  - se ~~e solo se~~ r soddisfa almeno una delle FD:
    - $X_{12} \rightarrow X_1$
    - $X_{12} \rightarrow X_2$

Decomposizione senza perdita, dimostrazione

- $r_1(X_1),r_2(X_2)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F se e solo se F implica almeno una delle FD:
  - $X_{12} \rightarrow X_1$, $X_{12} \rightarrow X_2$
- Dimostrazione (solo se):
  - se $r_1(X_1),r_2(X_2)$ è dsp per tutte le r allora F implica $X_{12} \rightarrow X_1$ o $X_{12} \rightarrow X_2$
  - supponiamo che F non le implichi e mostriamo una r che soddisfa F e non si decompone senza perdita

r	$X_1 - (X_{12})^+_F$	$(X_{12})^+_F$	$X_2 - (X_{12})^+_F$
	0 0 … 0	0 0 … 0	1 1 … 1
	1 1 … 1	0 0 …0	0 0 … 0

  - r soddisfa F (solita dimostrazione)
  - r non si decompone senza perdita (il join contiene la ennupla con tutti 0)

Decomposizione senza perdita, dimostrazione 2

- $r_1(X_1),r_2(X_2)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F se e solo se F implica almeno una delle FD:
  - $X_{12} \rightarrow X_1$, $X_{12} \rightarrow X_2$
- Dimostrazione (se)
  - se F implica $X_{12} \rightarrow X_1$ allora $r_1(X_1),r_2(X_2)$ è dsp per ogni r che soddisfa F
  - consideriamo r che soddisfa F e mostriamo si decompone senza perdita, cioè che ogni ennupla nel join delle proiezioni appartiene ad r
  - consideriamo una qualunque ennupla t nel join delle proiezioni; essa proviene da t_1 in $\text{PROJ}_{X_1}(r)$ e t_2 in $\text{PROJ}_{X_2}(r)$ che a loro volta vengono da t'_1 t'_2 in r per definizione di join e proiezione:
    - $t'_1[X_1] = t_1 = t[X_1]$ e $t'_2[X_2] = t_2 = t[X_2]$
  - Quindi $t'_1[X_{12}] = t'_2[X_{12}]$ s, poiché F implica $X_{12} \rightarrow X_1$ abbiamo $t'_1[X_1] = t'_2[X_1]$ e quindi $t'_2[X_1] = t'_1[X_1] = t[X_1]$ e $t'_2[X_2] = t[X_2]$ e cioè $t'_2[X_{12}] = t[X_{12}]$ e cioè $t'_2 = t$ e quindi t appartiene ad r

Decomposizione senza perdita, con FD

- Cominciamo con il caso binario
  - $R(X)$ con F
  - $X_1 X_2 = X$

Teorema

- $r_1(X_1), r_2(X_2)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - se e solo se F implica almeno una delle FD (dove $X_{12} = X_2 \cap X_2$):
    - $X_{12} \rightarrow X_1$
    - $X_{12} \rightarrow X_2$
  - se e solo se X_{12} è chiave per una delle due relazioni decomposte
  - se e solo se almeno una fra X_1 e X_2 è superchiave per R cioè se F implica almeno una delle FD:
    - $X_1 \rightarrow X$
    - $X_2 \rightarrow X$

Decomposizione n-aria senza perdita, con FD

- $R(X)$ con F
- $X_1 X_2 \ldots X_n = X$
- $F_1, F_2, \ldots, F_n = F$ FD rispettivamente su $X_1, X_2, \ldots, X_n$ (FD immerse)

Teorema

- $r_1(X_1), r_2(X_2) \ldots r_n(X_n)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - se e solo se almeno una fra $X_1, X_2, \ldots, X_n$ è superchiave per R cioè se e solo se F implica almeno una delle FD:
    - $X_1 \rightarrow X$, $X_2 \rightarrow X$, … , $X_n \rightarrow X$

- Nota:
  - Il teorema vale anche se le FD non sono immerse, ma con tale ipotesi (che è ok nei casi reali) la dimostrazione è più semplice

Decomposizione n-aria senza perdita, con FD dimostrazione

- $r_1(X_1), r_2(X_2) \ldots r_n(X_n)$ r è decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
  - se e solo se almeno una fra $X_1, X_2, \ldots, X_n$ è superchiave per R cioè se F implica almeno una delle FD:
    - $X_1 \rightarrow X$, $X_2 \rightarrow X$, ... , $X_n \rightarrow X$
- Dimostrazione (solo se)
  - Generalizzazione del caso binario:
    - Una relazione su n ennuple, la i-esima delle quali ha 0 sugli attributi in $(X_i)^+_F$ e i sugli altri
      - Non si decompone senza perdita (facendo proiezioni e join ottengo la ennupla con tutti 0, che non appartiene alla relazione originaria visto che nessuno degli X_1 è superchiave)
      - Soddisfa F (solita dimostrazione)

Decomposizione n-aria senza perdita, con FD dimostrazione, 2

- Dimostrazione (se)
  - Supponiamo X_i superchiave e scriviamo (per comodità) $K = X_i$
  - Consideriamo una qualunque r che soddisfa F e una t nel join delle sue proiezioni e dimostriamo che t è in r
  - Come nel caso binario, esistono $t'_1\, t'_2 \ldots t'_n$ in r con $t'_i[X_i] = t[X_i]$ per $i = 1,\ldots,n$. Indichiamo t'_i con t_0 e dimostriamo che $t_0 = t$ e quindi t è in r
  - Se K è superchiave, allora esiste una sequenza di FD $Y_1 \rightarrow B_1,\ Y_2 \rightarrow B_2,\ \ldots\ ,\ Y_h \rightarrow B_h$ in F che, nell'esecuzione dell'algoritmo di chiusura portano a ottenere $X = KB_1B_2\ldots B_h$ da K. Indichiamo $Z_0 = K,\ Z_j = Z_{j-1}B_j$ (per $j=1,\ldots,h$) e quindi $Z_h=X$.
  - Procediamo mostrando (per $j=0,\ldots,h$) $t_0[Z_j] = t[Z_j]$ (e così $t_0 = t$ per $j=h$)
    - Base: $t_0[Z_0] = t[Z_0]$, per ipotesi ($Z_0 = K = X_i$ e $t'_i = t_0$)
    - Passo induttivo: supponiamo $t_0[Z_{j-1}] = t[Z_{j-1}]$, e dimostriamo $t_0[Z_j] = t[Z_j]$, quindi ci basta dimostrare che $t_0[B_j] = t[B_j]$
    - Come visto sopra, esiste $Y_j \rightarrow B_j$ in F con $Y_j \subseteq Z_{j-1}$ e quindi $t_0[Y_j] = t[Y_j]$. Dato che la dipendenza è immersa, esiste $X_q \supseteq Y_jB_j$ e quindi $t'_q[X_q] = t[X_q]$. Ma allora, poiché $Y_j \subseteq X_q$ e $Y_j \subseteq Z_{j-1}$ abbiamo $t'_q[Y_j] = t_0[Y_j]$ e quindi (poiché r soddisfa F e quindi $Y_j \rightarrow B_j$) $t'_q[B_j] = t_0[B_j]$. Ma $B_j \in X_q$ e quindi $t'_q[B_j] = t[B_j]$. Ne segue che $t_0[B_j] = t[B_j]$

Decomposizione in BCNF

- Se una relazione non soddisfa la BCNF la decomponiamo
  - vogliamo una decomposizione adeguata e cioè,
  - data $R(X)$ (con F) vogliamo $R_1(X_1), \ldots, R_n(X_n)$ (con $F_1, \ldots, F_n$) con
    - $r_1(X_1), \ldots, r_n(X_n)$ decomposizione senza perdita di $r(X)$ per tutte le r che soddisfano F
    - $F_1 \cup \ldots \cup F_n$ equivalente a F

Algoritmo per la decomposizione in BCNF

Decomponi(R,F)

{ if esiste $X \rightarrow A$ che viola la BCNF per **(R,F)**

then { sostituisci **R** con

una relazione R_1 con attributi **U-A** e

una relazione R_2 con attributi **XA;**

Decomponi(R_1,F_{U-A});

Decomponi(R_2,F_{XA})

}

}

- Nota:
  - F_Y è la proiezione di **F** su **Y,** cioè l'insieme delle dipendenze implicate da **F** che coinvolgono solo attributi in **Y**
  - Si calcola per "enumerazione", cioè in modo inefficiente; tralasciamo

Algoritmo per la decomposizione in BCNF, 2

- Teorema:
  - L'algoritmo termina e produce uno schema che è in BCNF e che è decomposizione senza perdita di quello dato

- Però l'algoritmo non garantisce la conservazione delle dipendenze!

Una decomposizione problematica

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto Sede → Dirigente
Dirigente → Sede

Una decomposizione problematica, 2

- Progetto Sede $\rightarrow$ Dirigente coinvolge tutti gli attributi e quindi nessuna decomposizione può preservare tale dipendenza
- quindi in alcuni casi la BCNF “non è raggiungibile”
- Per questo si studia anche la terza forma normale (3NF)

Una nuova forma normale

- Una relazione r è in terza forma normale se, per ogni FD (non banale) X $\rightarrow$ Y definita su r, è verificata almeno una delle seguenti condizioni:
  - X contiene una chiave K di r
  - ogni attributo in Y è contenuto in almeno una chiave di r

3NF: definizione formale

- Uno schema di relazione $R(U)$ con le dipendenze F
  - è in BCNF
    - se per ogni FD (non banale) $X \rightarrow A$ in F
      - il primo membro X contiene una chiave di R
  - è in 3NF
    - se per ogni FD (non banale) $X \rightarrow A$ in F
      - il primo membro X contiene una chiave di R
      - oppure A appartiene ad una chiave di R

3NF: proprietà

- Teoremi (dimostrazioni per esercizio, salvo l'ultima)
  - La BCNF è strettamente più restrittiva della 3NF
    - se **R(U)** è in BCNF rispetto a **F** allora è in 3NF rispetto a **F** (ma non necessariamente viceversa)
  - La 3NF è indipendente dalla copertura
    - se **F** è equivalente a **G** allora **R(U)** è in 3NF rispetto a **F** se e solo se lo è rispetto a **G**
  - Se c'è una sola chiave, le due forme normali coincidono
    - se **R(U)** con **F** ha una sola chiave, allora è in BCNF se e solo se è in 3NF
  - La verifica della 3NF è un problema NP-completo

3NF: un’osservazione

- La 3NF risolve i nostri problemi solo fino ad un certo punto
  - Lo schema “problematico” presenta anomalie e soddisfa la 3NF
- Però ci possiamo accontentare, soprattutto perché la 3NF è sempre raggiungibile

Normalizzazione in 3NF

- Procedere per “decomposizione”, come suggerito per la BCNF, può essere inefficiente, perché la verifica della 3NF è complessa (NP-completa)
- Esiste un algoritmo di “sintesi”:
  - Costruisce schemi di relazione a partire dalle dipendenze, separandole le une dalle altre

Algoritmo di sintesi

- **R(U)** con **F**
  - Calcola una copertura ridotta **G** di **F**
  - Partiziona **G** in insiemi $G_1, G_2, \ldots, G_k$ sulla base della chiusura dei primi membri ($X \rightarrow A$ e $Y \rightarrow B$ appartengono alla stessa classe se e solo se $X^+_F = Y^+_F$)
  - Costruisci lo schema decomposto con uno schema di relazione per ogni classe della partizione, con
    - tutti gli attributi che compaiono nelle relative dipendenze
    - le dipendenze di **G** (quindi non solo quelle della partizione) che hanno tutti gli attributi inclusi nello schema
  - Se uno schema prodotto al passo precedente ha tutti gli attributi contenuti in un altro, eliminalo
  - Se fra gli schemi prodotti nessuno contiene una chiave per la relazione originaria, allora trova una chiave **K** e aggiungi uno schema di relazione su **K** (con insieme vuoto di FD)

Copertura ridotta

- $\mathbf{G}$ è copertura ridotta di $\mathbf{F}$ se
  - $\mathbf{G}$ è equivalente ad $\mathbf{F}$
  - le FD in $\mathbf{G}$ hanno la forma $\mathbf{X \rightarrow A}$
  - $\mathbf{G}$ è non ridondante
    - per nessuna $\mathbf{X \rightarrow A}$ in $\mathbf{G}$ si ha che $\mathbf{G}$ è equivalente a $\mathbf{G - \{ X \rightarrow A \}}$
  - i primi membri in $\mathbf{G}$ sono minimali
    - per nessuna $\mathbf{X \rightarrow A}$ in $\mathbf{G}$ si ha che $\mathbf{G}$ è equivalente a $\mathbf{G - \{ X \rightarrow A \} \cup \{ Y \rightarrow A \}}$ con $\mathbf{Y \subset X}$

Calcolo della copertura ridotta

- Algoritmo
  - Inizializziamo G con F
  - decomponiamo ogni FD in G nella forma $X \rightarrow A$ (non banale)
  - per ogni $XA \rightarrow B$ in G, se G è equivalente a $G - \{ XA \rightarrow B \} \cup \{ X \rightarrow B \}$ allora sostituiamo $XA \rightarrow B$ con $X \rightarrow B$
    - Nota: G è equivalente a $G - \{ XA \rightarrow B \} \cup \{ X \rightarrow B \}$ se e solo se G implica $X \rightarrow B$ (calcoliamo la chiusura di X, se contiene B, possiamo eliminare A dal primo membro)
  - per ogni $X \rightarrow A$ in G , se essa è ridondante, la eliminiamo
- Terminazione: ciascun passo prevede un numero finito di passi e, se modifica, riduce il numero degli attributi o delle diepndenze
- Correttezza:
  - Ogni passo porta ad una delle proprietà e non introduce violazioni delle proprietà ottenute in precedenza

Copertura ridotta, esempio

- $E \rightarrow NS$
- $NL \rightarrow EMD$
- $EN \rightarrow LCD$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $EPD \rightarrow AE$
- $NLCP \rightarrow A$

- $E \rightarrow N$
- $E \rightarrow S$
- $NL \rightarrow E$
- $NL \rightarrow M$
- $NL \rightarrow D$
- $EN \rightarrow L$
- $EN \rightarrow C$
- $EN \rightarrow D$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $EPD \rightarrow A$
- $NLCP \rightarrow A$

- $E \rightarrow N$
- $E \rightarrow S$
- $NL \rightarrow E$
- $NL \rightarrow M$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$
- $E \rightarrow D$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $EP \rightarrow A$
- $NLP \rightarrow A$

- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $EN \rightarrow L$
- $EN \rightarrow C$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $NLP \rightarrow A$

Algoritmo di sintesi

- **R(U)** con **F**
  - Calcola una copertura ridotta **G** di **F**
  - Partiziona **G** in insiemi $G_1, G_2, \ldots, G_k$ sulla base della chiusura dei primi membri ($X \rightarrow A$ e $Y \rightarrow B$ appartengono alla stessa classe se e solo se $X^+_F = Y^+_F$)
  - Costruisci lo schema decomposto con uno schema di relazione per ogni classe della partizione, con
    - tutti gli attributi che compaiono nelle relative dipendenze
    - le dipendenze di **G** (quindi non solo quelle della partizione) che hanno tutti gli attributi inclusi nello schema
  - Se uno schema prodotto al passo precedente ha tutti gli attributi contenuti in un altro, eliminalo
  - Se fra gli schemi prodotti nessuno contiene una chiave per la relazione originaria, allora trova una chiave **K** e aggiungi uno schema di relazione su **K** (con insieme vuoto di FD)

Sintesi, esempio

- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $NLP \rightarrow A$

- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$
- $C \rightarrow S$
- $D \rightarrow M$
- $M \rightarrow D$
- $NLP \rightarrow A$

$R_1(ENLCD)$

- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$

$R_2(CS)$

- $C \rightarrow S$

$R_3(DM)$

- $D \rightarrow M$
- $M \rightarrow D$

$R_4(NLPA)$

- $NLP \rightarrow A$

Sintesi, commenti

- La copertura ridotta non è unica e quindi ci possono essere diverse soluzioni

R_1(ENLCD)
- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$

R_2(CS)
- $C \rightarrow S$

R_3(DM)
- $D \rightarrow M$
- $M \rightarrow D$

R_4(NLPA)
- $NLP \rightarrow A$

R_1(ENLCD)
- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow D$
- $E \rightarrow L$
- $E \rightarrow C$

R_2(CS)
- $C \rightarrow S$

R_3(DM)
- $D \rightarrow M$
- $M \rightarrow D$

R_4(EPA)
- $EP \rightarrow A$

R_1(ENLCM)
- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow M$
- $E \rightarrow L$
- $E \rightarrow C$

R_2(CS)
- $C \rightarrow S$

R_3(DM)
- $D \rightarrow M$
- $M \rightarrow D$

R_4(EPA)
- $EP \rightarrow A$

R_1(ENLCM)
- $E \rightarrow N$
- $NL \rightarrow E$
- $NL \rightarrow M$
- $E \rightarrow L$
- $E \rightarrow C$

R_2(CS)
- $C \rightarrow S$

R_3(DM)
- $D \rightarrow M$
- $M \rightarrow D$

R_4(NLPA)
- $NLP \rightarrow A$

Algoritmo di sintesi

- **R(U)** con **F**
  - Calcola una copertura ridotta **G** di **F**
  - Partiziona **G** in insiemi $G_1, G_2, \ldots, G_k$ sulla base della chiusura dei primi membri ($X \rightarrow A$ e $Y \rightarrow B$ appartengono alla stessa classe se e solo se $X^+_F = Y^+_F$)
  - Costruisci lo schema decomposto con uno schema di relazione per ogni classe della partizione, con
    - Tutti gli attributi che compaiono nelle relative dipendenze
    - Le dipendenze di **G** (quindi non solo quelle della partizione) che hanno tutti gli attributi inclusi nello schema
  - Se uno schema prodotto al passo precedente ha tutti gli attributi contenuti in un altro, eliminalo
  - Se fra gli schemi prodotti nessuno contiene una chiave per la relazione originaria, allora trova una chiave **K** e aggiungi uno schema di relazione su **K** (con insieme vuoto di FD)

Nell'esempio problematico

- Se non avessimo questo passo, otterremmo
  - R_1(Progetto, Sede, Dirigente) Progetto Sede $\rightarrow$ Dirigente
  - R_2(Dirigente, Sede) Dirigente $\rightarrow$ Sede
- Quindi, non solo dobbiamo avere
  - $\mathbf{F_1 \cup \ldots \cup F_n}$ equivalente a $\mathbf{F}$
- Ma anche:
  - Se una FD $\mathbf{Y \rightarrow A}$ è definita in una relazione $\mathbf{R_i(X_i)}$ e definibile in un'altra $\mathbf{R_j(X_j)}$ (cioè $\mathbf{YA \subseteq X_j}$) allora $\mathbf{Y \rightarrow A}$ deve appartenere anche a $\mathbf{F_j}$

Algoritmo di sintesi, altro esempio

- **R(ABCDEHKL) AB → DC, BC → A, DE → H, H → EL**
- Copertura ridotta: **AB → C, AB → D, BC → A, DE → H, H → E, H → L**
- Partizioni:
  - **AB → C, AB → D, BC → A**
  - **DE → H**
  - **H → E, H → L**
- Schemi di relazione
  - **R_1(ABCD)** **AB → C, AB → D, BC → A,**
  - **R_2(DEH)** **DE → H, H → E**
  - **R_3(HEL)** **H → E, H → L**
- Nessuno schema include una chiave, ne aggiungiamo un altro
  - **R_0 (ABEK)**

Algoritmo di sintesi sulla decomposizione problematica

- In pratica, non decompone!

Algortitmo di sintesi, correttezza

- Terminazione, ok
- Decomposizione adeguata:
  - $F_1 \cup \ldots \cup F_n$ equivalente a F , ok
  - Decomposizione senza perdita, ok, grazie all'ultimo passo
  - 3NF:
    - In ciascuna classe della partizione, i primi membri sono tutti equivalenti, e quindi sono chiave
    - Le uniche violazioni possono venire dalle dipendenze che sono incluse in F_i pur non appartenendo alla classe:
      - $X \rightarrow A \in G - G_i$; dimostriamo che A appartiene ad una chiave di R_i (cioè ad un primo membro della classe); se così non fosse, A sarebbe il secondo membro di una $Y \rightarrow A$ della classe, ma allora la $Y \rightarrow A$ sarebbe ridondante, perché $Y \rightarrow X$ è implicata da G