Lezione 17 - Copertura minimale di un insieme di dipendenze

Prof.ssa Maria De Marsico
demarsico@di.uniroma1.it

Sapienza Università di Roma

Parte del materiale è tratto dal materiale preparato dalla prof.ssa Moscarini per il canale A-L

Ricapitoliamo

- Fino ad ora abbiamo parlato del **perché** possa essere necessario **decomporre** uno schema di relazione ***R***, sui cui è definito un insieme di dipendenze funzionali ***F***, soprattutto in relazione a violazioni della 3NF che causano diversi tipi di anomalie.
- Abbiamo detto più volte che, qualunque sia il motivo che ci porta a decomporre lo schema, la decomposizione deve soddisfare tre requisiti fondamentali.
  - ogni sottoschema **deve essere 3NF**
  - la decomposizione **deve preservare le dipendenze funzionali**
  - deve essere **possibile ricostruire ogni istanza legale** dello schema originale **tramite join naturale** di istanze della decomposizione

Ricapitoliamo

- Nelle lezioni precedenti abbiamo visto come **verificare** che una decomposizione **data** (non ci interessa come sia stata prodotta) soddisfi tutte le condizioni, in particolare abbiamo parlato di come verificare:
  - se la decomposizione preserva le dipendenze funzionali
  - se sarà possibile ricostruire ogni istanza legale dello schema originale tramite join naturale di istanze della decomposizione

Che si fa ora?

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Ora affrontiamo il problema di **come** ottenere una decomposizione che soddisfi le nostre condizioni.
- Prima di tutto: **è sempre possibile ottenerla?**
- **La risposta è SI**: **è sempre possibile**, dato uno schema ***R*** su cui è definito un insieme di dipendenze funzionali ***F***, **decomporlo in modo da ottenere che**:
  – ogni sottoschema **è 3NF**
  – la decomposizione **preserva le dipendenze funzionali**
  – È **possibile ricostruire ogni istanza legale** dello schema originale **tramite join naturale** di istanze della decomposizione

– **Presenteremo un algoritmo che raggiunge questo scopo**

Attenzione!

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- La decomposizione che si ottiene dall'algoritmo che studieremo **non è l'unica possibile** che soddisfi le condizioni richieste
- **Lo stesso algoritmo**, a seconda **dell'input di partenza** (di cui parleremo in questa lezione) può fornire **risultati diversi** e tuttavia **corretti.**
- **Attenzione a non confondere** l'algoritmo per la decomposizione con quelli per la verifica.
- **Proprio perché <u>non esiste</u> <u>LA</u> decomposizione giusta,** ma ci sono diverse possibilità, potrebbe succedere che la decomposizione da verificare **non sia stata ottenuta** tramite l'algoritmo, quindi …
- **… usare l'algoritmo di decomposizione** per controllare **se** produce la decomposizione da verificare, e ottenerne invece **una diversa, non ci autorizza** a concludere che la decomposizione da verificare non possegga le proprietà richieste.

Prima di continuare …

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Prima di continuare dobbiamo introdurre il concetto di «*copertura minimale*» di un insieme F di dipendenze funzionali,
- Sarà proprio <u>**una**</u> **copertura minimale** di *F* a costituire l'input dell'algoritmo di decomposizione,
- Dato un insieme di dipendenze funzionali ***F***, possono esserci <u>**più coperture minimali equivalenti**</u> **(nel senso di avere tutte la la stessa chiusura, che poi è uguale anche a quella di *F*)**
- **E' proprio questo il motivo per cui l'algoritmo di decomposizione può produrre risultati diversi, ma tutti corretti**

Definizione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Definizione Sia F un insieme di dipendenze funzionali. Una *copertura minimale* di F è un insieme G di dipendenze funzionali equivalente ad F tale che:

- per ogni dipendenza funzionale in G la parte **destra** è un **singleton**, cioè è costituita da un unico attributo (ogni attributo nella parte destra **è non ridondante**)
- per **nessuna** dipendenza funzionale $\boldsymbol{X \rightarrow A}$ **in** $\boldsymbol{G}$ esiste $\boldsymbol{X' \subset X}$ tale che $\boldsymbol{G \equiv G - \{X \rightarrow A\} \cup \{X' \rightarrow A\}}$ (ogni attributo nella parte **sinistra** è non ridondante)
- per **nessuna** dipendenza funzionale $\boldsymbol{X \rightarrow A}$ **in** $\boldsymbol{G}$, $\boldsymbol{G \equiv G - \{X \rightarrow A\}}$ (ogni **dipendenza** è **non ridondante**).

Osservazioni

- Per ogni dipendenza funzionale in *G* la parte **destra** è un **singleton**, cioè è costituita da un unico attributo (ogni attributo nella parte destra **è non ridondante**)
  - sempre possibile con la regola della **decomposizione**

- Per **nessuna** dipendenza funzionale $X \rightarrow A$ **in** G esiste $X' \subset X$ tale che $G \equiv G - \{X \rightarrow A\} \cup \{X' \rightarrow A\}$ (ogni attributo nella parte **sinistra** è non ridondante)
  - non è possibile determinare funzionalmente A (in G o eventualmente in G^+) tramite un **sottoinsieme** di X

- per **nessuna** dipendenza funzionale $X \rightarrow A$ **in** G, $G \equiv G - \{X \rightarrow A\}$ (ogni **dipendenza** è **non ridondante**).
  - non è possibile determinare funzionalmente A (in G o eventualmente in G+) tramite altre dipendenze

Come si calcola

SAPIENZA
Università di Roma
Dipartimento di Informatica

Per ogni insieme di dipendenze funzionali *F* esiste una copertura minimale **equivalente ad *F*** che può essere ottenuta in tempo **polinomiale** in tre passi:

- usando la regola della decomposizione, le parti destre delle dipendenze funzionali vengono ridotte a singleton;
- ogni dipendenza funzionale $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A$ in F tale che

  $F \equiv F - \{ A_1A_2\ldots \mathbf{A_{i-1}A_iA_{i+1}}\ldots A_n \rightarrow A \} \cup \{ A_1A_2\ldots \mathbf{A_{i-1}A_{i+1}}\ldots A_n \rightarrow A \}$

  viene sostituita appunto da $A_1A_2\ldots \mathbf{A_{i-1}A_{i+1}}\ldots A_n \rightarrow A$; se quest'ultima **appartiene già** ad F la dipendenza originaria viene **semplicemente eliminata**, altrimenti il processo viene ripetuto **ricorsivamente** su $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A$; il passo termina quando **nessuna** dipendenza funzionale può più essere **ridotta**, cioè quando tutti gli attributi delle parti sinistre delle dipendenze funzionali risultano non ridondanti;
- ogni dipendenza funzionale $X \rightarrow A$ in F tale che
  - $F \equiv F - \{ X \rightarrow A \}$
  - viene **eliminata**, in quanto risulta ridondante; in questo modo minimizziamo il numero di dipendenze funzionali.

Verifica equivalenza

- I passi 2 e 3 richiedono come detto in precedenza di verificare l'equivalenza tra due insiemi di dipendenze funzionali.
- Per la verifica dell'equivalenza, vedremo che ci troviamo in casi particolari che possono essere risolti in maniera semplificata.

Verifica equivalenza

- Richiamiamo intanto alcuni risultati di lemmi e teoremi utili in questo caso:
  - $F \equiv G$ se e solo se $F^+ = G^+$, cioè se e solo se $F^+ \subseteq G^+$ **e** $G^+ \subseteq F^+$
  - se $F \subseteq G$ banalmente $F \subseteq G^+$
  - $F \subseteq G^+$ implica che $F^+ \subseteq G^+$

- Per verificare nel caso non banale se $F \subseteq G^+$, per ogni $X \rightarrow Y \in F$ controlliamo se $X \rightarrow Y \in G^+$, cioè se $Y \in (X)^+_G$, cioè se Y appartiene alla chiusura di X rispetto all'insieme di dipendenze funzionali G. A tale scopo usiamo l' algoritmo visto.
- **Nota**: abbiamo a che fare sempre **con lo stesso schema di relazione *R* non ancora** decomposto

Verifica equivalenza

- Analizziamo separatamente i passi 2 e 3.

- Come anticipato ci troviamo di fronte a due casi **particolari** del problema **dell’equivalenza tra insiemi di dipendenze**.

Passo 2

- Nel passo 2, **ogni volta** che vogliamo verificare la ridondanza di un attributo nella parte di una dipendenza, assumiamo di indicare con F l'insieme che contiene la dipendenza originaria

  $A_1A_2\ldots\mathbf{A_{i-1}A_iA_{i+1}}\ldots A_n \to A$ e con G l'insieme che contiene al suo posto la dipendenza $A_1A_2\ldots\mathbf{A_{i-1}A_{i+1}}\ldots A_n \to A$. Notiamo prima di tutto che i due insiemi differiscono **esattamente** in una dipendenza. **Le altre sono uguali**, e quindi **banalmente** appartengono **alla chiusura di entrambi gli insiemi**. Per stabilire quindi l'equivalenza **resta da verificare che**

  $$\boxed{\mathbf{A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \to A \in G^+ \wedge A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \to A \in F^+}}$$

Passo 2

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Per come procede l'algoritmo che calcola la chiusura di un insieme di attributi, è facile concludere che **è superfluo controllare** se

  $\mathbf{A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A \in G^+}$**, cioè se** $\mathbf{A \in (A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n)^+_G}$**.** Basta considerare che partiamo con un insieme iniziale Z più ampio di quello presente a sinistra della dipendenza di $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A \in G$, cioè al primo passo poniamo $Z = A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$. Di conseguenza, in base al passo

  $S := \{ A \mid Y \rightarrow V \in G \wedge A \in V \wedge Y \subseteq Z \}$, ed in particolare alla condizione $Y \subseteq Z$, inseriremmo immediatamente A in S proprio per la presenza in **G** della dipendenza $A_1A_2\ldots \mathbf{A_{i-1}A_{i+1}}\ldots A_n \rightarrow A$.
- La giustificazione teorica di questo fatto risiede negli assiomi di Armstrong della riflessività e della transitività:
- Poiché $A_1A_2\ldots \mathbf{A_{i-1}A_{i+1}}\ldots An \subseteq A_1A_2\ldots \mathbf{A_{i-1}A_iA_{i+1}}\ldots A_n$, avremo per **riflessività** la dipendenza banale $A_1A_2\ldots \mathbf{A_{i-1}A_iA_{i+1}}\ldots A_n \rightarrow A_1A_2\ldots \mathbf{A_{i-1}A_{i+1}}A_n$ e quindi poiché in G abbiamo $A_1A_2\ldots A_{i-1}\mathbf{A_{i+1}}\ldots A_n \rightarrow A$, per **transitività** varrà anche
- $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A$.

Esempio

Abbiamo $F = \{AB \rightarrow C,\ A \rightarrow D, D \rightarrow C\}$

Per verificare se possiamo eliminare la B in $AB \rightarrow C$ dobbiamo verificare

1. se $A \rightarrow C \in \mathbf{F^+}$
   - Nell'algoritmo si parte con Z=A e si applicano le dipendenze in F
2. se $AB \rightarrow C \in \mathbf{G^+}$ con $G = \{A \rightarrow C, A \rightarrow D, D \rightarrow C\}$
   - Nell'algoritmo si parte con Z=AB e si applicano le dipendenze in G

- Il punto 2. è banale perché se calcolo $(AB)^+_G$ ho immediatamente $A \rightarrow C$ (parte sinistra contenuta in Z) che mi permette di aggiungere C alla variabile S

Passo 2

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Resta da verificare se $A_1A_2\ldots\mathbf{A_{i-1}A_{i+1}}\ldots A_n \rightarrow A \in \mathbf{F^+}$.
- Per fare questo usiamo l'algoritmo per verificare se

$$A \in (A_1A_2\ldots\mathbf{A_{i-1}A_{i+1}}\ldots A_n)^+_F$$

- In questo caso infatti potrebbe verificarsi che stiamo tentando di imporre un vincolo **estraneo** alla definizione dello schema.

- Possiamo incontrare una situazione particolare. Se infatti la dipendenza $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A$ appartiene **essa stessa** all'insieme F, **banalmente** apparterrà ad F^+. In questo caso, **oltre a non effettuare la verifica, eliminiamo del tutto la dipendenza originaria.** (Esempio, se F contiene sia $AB \rightarrow C$ che $A \rightarrow C$ allora $AB \rightarrow C$ non solo si può ridurre ma anche eliminare)

- In ogni caso, se dimostriamo l'equivalenza di F e G, possiamo assumere G come insieme di riferimento **per le verifiche successive** (G diventa il nostro nuovo F).

Passo 2: osservazioni

- Nelle verifiche successive, relative alla riduzione di ulteriori dipendenze **in questo passo**, **è inutile ricalcolare le chiusure** per attributi o gruppi di attributi per i quali tale calcolo sia già stato effettuato.

- Dato per esempio X, siccome $(X)^{+}_{F} = \{ A \mid X \rightarrow A \in F^{+} \}$, e siccome il **verso** della verifica (se $A_1A_2\ldots\mathbf{A_{i-1}A_{i+1}}\ldots A_n \rightarrow A \in \mathbf{F^{+}}$) porta a calcolare le chiusure rispetto ad un insieme di dipendenze che è F iniziale oppure un insieme G che è stato **già dimostrato equivalente** ad F (per cui $F^{+} = G^{+}$), se $X \rightarrow A \in F^{+}$ allora vale anche $X \rightarrow A \in G^{+}$, cioè $(X)^{+}_{F} = (X)^{+}_{G}$.

Passo 2: osservazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Notiamo anche che:

- se $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A \in F$ ma **non esiste** $Y \rightarrow A \in F$ con $Y \neq A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$, oppure {sottoinsieme1 di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$}$\rightarrow$ {sottoinsieme2 di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$} $\in F^+$, allora sarebbe **inutile** provare ad eliminare attributi a sinistra della dipendenza, in quanto, per come viene definita la chiusura, e da come lavora l'algoritmo, non potremo comunque inserire A in alcuna chiusura che non sia quella generata dalla dipendenza $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A$ (non esistono sottoinsiemi di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$ che determinano A né combinazioni $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow Y$ + $Y \rightarrow A$ che permettano di applicare la transitività, né {sottoinsieme1 di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$}$\rightarrow$ {sottoinsieme2 di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$} $\in F^+$, che permetterebbe di inserire comunque A nella chiusura di {sottoinsieme1 di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$}); la seconda condizione però non è immediatamente evidente, quindi … meglio verificare!
- se $A_1A_2\ldots A_n \rightarrow A \in F$ e $Y \rightarrow A \in F$ con $Y \subset A_1A_2\ldots A_n$, eliminiamo $A_1A_2\ldots A_n \rightarrow A$ senza bisogno di ulteriori verifiche; infatti nella verifica di equivalenza per le successive sostituzioni di $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A$ con $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A$, purché $Y \subset A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n$ avremo sempre $A \in S$ già all'inizio dell'algoritmo grazie alla dipendenza $Y \rightarrow A$, fino ad arrivare a provare la sostituzione proprio con $Y \rightarrow A$ che però è un duplicato.

Passo 3

- Passiamo ora ad esaminare il passo 3 dell'algoritmo per la copertura minimale.
- Assumiamo di indicare con F l'insieme che contiene la dipendenza $X \to A$, e con G l'insieme in cui tale dipendenza è stata eliminata.
- Anche in questo caso, i due insiemi **differiscono per una sola dipendenza**, anzi si verifica che $\mathbf{G \subseteq F}$, quindi **sappiamo già che $\mathbf{G^+ \subseteq F^+}$.**
- Resta quindi da verificare **che sia $\mathbf{F^+ \subseteq G^+}$,** che **sarà sicuramente vero** se $F \subseteq G^+$.
- In particolare, **basta verificare se $\mathbf{X \to A \in G^+}$** (le altre sono comuni ai due insiemi, e quindi alle loro chiusure) e cioè se $\mathbf{A \in (X)^+_G}$.

Passo 3: osservazioni

- In questo caso però le chiusure di attributi o gruppi di attributi **vanno ricalcolate**, perché il **verso** della verifica porta a calcolare le chiusure rispetto ad un insieme di dipendenze **che non è stato ancora dimostrato essere equivalente ad F** (anzi, è proprio quello che vorremmo dimostrare).

- Notiamo inoltre che se $X \rightarrow A \in F$ ma **non esiste** $Y \rightarrow A \in F$ con $Y \neq X$, allora **è inutile** provare ad eliminare $X \rightarrow A$, in quanto eliminando tale dipendenza non saremmo più in grado di determinare funzionalmente A.

Riassumendo ...

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Per ogni dipendenza esaminata al passo 2 dell'algoritmo per la copertura minimale:

- assumendo di indicare con F l'insieme che contiene la dipendenza originaria $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A$ e con G l'insieme che contiene al suo posto la dipendenza $A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A$, per verificare se F $\equiv$ G basta verificare se

  $$A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n \rightarrow A \in F^+ \text{ (se } A \in (A_1A_2\ldots A_{i-1}A_{i+1}\ldots A_n)^+_{F;}\text{)}$$
- se A1A2…Ai-1Ai+1…An $\rightarrow$ A$\in$ F, allora eliminiamo la dipendenza che stavamo esaminando
- se $A_1A_2\ldots\ldots A_n \rightarrow A \in F$ e $Y \rightarrow A \in F$ con $Y \subseteq A_1A_2\ldots A_n$, allora eliminiamo la dipendenza $A_1A_2\ldots\ldots A_n \rightarrow A \in F$
- se $A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n \rightarrow A \in F$ ma **non esiste** $Y \rightarrow A \in F$ con $Y \neq A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$, allora è inutile provare ad eliminare attributi a sinistra della dipendenza;
- **non è necessario ricalcolare le chiusure** transitive degli attributi o di gruppi di attributi;

Riassumendo ...

Per ogni dipendenza esaminata al passo 3 dell'algoritmo per la copertura minimale:

- assumendo di indicare con F l'insieme che contiene la dipendenza originaria $X \rightarrow A$ e con G l'insieme che non la contiene, per verificare se $F \equiv G$ basta verificare se $X \rightarrow A \in G^+$ (se $A \in (X)^+_G$)
- se $X \rightarrow A \in F$ ma **non esiste** $Y \rightarrow A \in F$ con $Y \neq X$, allora è inutile provare ad eliminare $X \rightarrow A$
- le chiusure transitive degli attributi o di gruppi di attributi **vanno ricalcolate**

Riassumendo ...

- Possono esserci **più** coperture minimali per un dato insieme di dipendenze funzionali.

- Usando l'algoritmo si può **sempre** trovare **almeno una** copertura minimale per **qualunque** insieme *F* di dipendenze.

- Può anche succedere che *F* **sia già** in forma minimale, e quindi **non ci sia da fare alcuna riduzione**.