Decomposizioni che preservano le dipendenze

Index

Cosa significa preservare le dipendenze?

Uno schema tipicamente viene decomposto per due motivi:

- **non è in 3NF**
- per motivi di efficienza degli accessi → infatti più è piccola la taglia delle tuple maggiore è il numero che riusciamo a caricare in memoria nella stessa operazione di lettura; se le informazioni della tupla non vengono utilizzate dallo stesso tipo di operazioni nella base di dati meglio decomporre lo schema

Abbiamo visto che quando uno schema viene decomposto, non basta che i sottoschemi siano in 3NF

Decomposizione di uno schema di relazione

Definizione

Sia R uno schema di relazione. Una **decomposizione** di R è una famiglia $\rho\{R_1, R_2, \ldots, R_k\}$ di sottoinsiemi di R che ricopre R, ovvero che $\cup_{i=1}^{k} R_i = R$ (i sottoinsiemi possono avere intersezione non vuota)

In altre parole: se lo schema R è composto da un certo insieme di attributi, decomporlo significa definire dei sottoschemi che contengono ognuno un sottoinsieme degli attributi di R.
I sottoschemi possono avere attributi in comune, e la loro unione deve necessariamente contenere tutti gli attributi di R

Quindi R è un insieme di attributi, una decomposizione di R è una famiglia di insiemi di attributi

⚠ Warning

Decomporre una istanza di una relazione con un certo schema, in base alla decomposizione dello schema stesso, significa proiettare ogni tupla dell'istanza originaria sugli attributi dei singoli sottoschemi eliminando i duplicati che potrebbero essere generati dal fatto che due tuple sono distinte ma hanno una posizione comune che ricade nello stesso schema

Example

R

CF	Nome	Cognome	Matricola	Datan	Luogon	CorsoLaurea	Anno
DDD	Davide	Bigi	1111	12-12-90	Bari	Lettere	3
FFF	Gianni	Neri	1212	15-09-91	Milano	Legge	2
AAA	Antonio	Rossi	1313	09-08-92	Napoli	Matematica	1

R_1

CF	Nome	Cognome	Matricola	Datan	Luogon
DDD	Davide	Bigi	1111	12-12-90	Bari
FFF	Gianni	Neri	1212	15-09-91	Milano
AAA	Antonio	Rossi	1313	09-08-92	Napoli

← sottoschema della decomposizione di R

← proiezione dell'istanza sul sottoschema

R_2

CF	Nome	Cognome	Matricola	CorsoLaurea	Anno
DDD	Davide	Bigi	1111	Lettere	3
FFF	Gianni	Neri	1212	Legge	2
AAA	Antonio	Rossi	1313	Matematica	1

← sottoschema della decomposizione di R

← proiezione dell'istanza sul sottoschema

Equivalenza tra due insiemi di dipendenze funzionali

Definizione

Siano F e G due insiemi di dipendenze funzionali. F e G sono **equivalenti** ($F \equiv G$) se $F^+ = G^+$

Warning

F e G non contengono le stesse dipendenze, ma le loro chiusure si

---

Che si fa?

Verificare l'equivalenza di due insiemi F e G di dipendenze funzionali richiede che venga verificata l'uguaglianza di F^+ e G^+, cioè che $F^+ \subseteq G^+$ e che $F^+ \supseteq G^+$

Come detto in precedenza, calcolare la chiusura di un insieme di dipendenze funzionali richiede tempo esponenziale. Il seguente lemma ci permette tuttavia di verificare l'equivalenza dei due insiemi di dipendenze funzionali in tempo polinomiale

---

Lemma 2

Siano F e G due insiemi di dipendenze funzionali. Se $F \subseteq G^+$ allora $F^+ \subseteq G^+$

Dimostrazione

Sia $f \in F^+ - F$ (è una dipendenza in F^+ che non compare in F)

Ogni dipendenza in F è derivabile da G mediante gli assiomi di Armstrong (per ipotesi F si trova in G^+)
Inoltre $f \in F^+$ è derivabile dalle dipendenze in F mediante gli assiomi di Armstrong
Dunque si può dire che f è derivabile da G mediante gli assiomi di Armstrong

$$G \xrightarrow{A} F \xrightarrow{A} F^+$$

Preservare le dipendenze funzionali

Definizione

Sia R uno schema di relazione, F un insieme di dipendenze funzionali su R e $\rho\{R_1, R_2, \dots R_k\}$ una decomposizione di R.
Diciamo che ρ **preserva** F se $F \equiv \cup_{i=1}^{k} \pi_{R_i}(F)$ dove
$\pi_{R_i}(F) = \{X \to Y \text{ t.c. } X \to Y \in F^+ \wedge XY \subseteq R_i\}$

Warning

Ovviamente $\cup_{i=1}^{k} \pi_{R_i}(F)$ è un insieme di dipendenze funzionali
Ogni $\pi_{R_i}(F)$ è un insieme di dipendenze funzionali dato dalla proiezione dell'insieme di dipendenze funzionali F sul sottoschema R_i

Proiettare un insieme di dipendenze F su un sottoschema R_i non significa banalmente perdere le dipendenze dell'insieme F ed eliminare da queste dipendenze gli attributi che non sono in R_i, ma **prendere tutte e sole** le dipendenze **derivabili da** F tramite gli assiomi di Armstrong (quindi quelle in F^+) che hanno tutti gli attributi (dipendenti e determinanti) in R_i

---

Verificare validità di una decomposizione

Supponiamo di avere già una decomposizione e di voler verificare se preserva le dipendenze funzionali
Per fare ciò deve essere verificata l'equivalenza dei due insiemi di dipendenze funzionali F e $G = \cup_{i=1}^{k} \pi_{R_i}(F)$ e quindi la doppia inclusione $F^+ \subseteq G^+$ e $F^+ \supseteq G^+$

Per come è definito G in questo caso sarà sicuramente $F^+ \supseteq G^+$, infatti ogni proiezione di F che viene inclusa per definizione in G è un sottoinsieme di F^+, quindi F^+ contiene G e per il lemma 2 questo implica che $G^+ \subseteq F^+$

Dunque non dobbiamo verificare una delle due implicazioni, inoltre per il lemma 2 $F \subseteq G^+$ implica che $F^+ \subseteq G^+$ dunque ci basta verificare che: $F \subseteq G^+$ semplificandoci di molto il lavoro, cioè per il lemma 1: $\forall X \to Y \in F$, cerco se $Y \subseteq X_G^+$ cioè se è vero $X \to Y \in G^A (= G^+)$

Questa verifica può essere fatta con l'algoritmo che segue

$$
\begin{array}{l}
\textbf{Input} \to \text{due insiemi } F \text{ e } G \text{ di dipendenze funzionali su R} \\
\textbf{Output} \to \text{la variabile successo (true se } F \subseteq G^+) \\
\text{begin} \\
\quad \text{successo} := true \\
\quad \text{for every } X \to Y \in F \\
\quad \text{do} \\
\quad \text{begin} \\
\quad\quad \text{calcola } X^+ \\
\quad\quad \text{if } Y \not\subset X_G^+ \text{ then successo} := false \\
\quad \text{end} \\
\text{end}
\end{array}
$$

Il problema di questo algoritmo è che dovremmo prima calcolare G, ma per definizione di G ciò richiede il calcolo di F^+ che richiede tempo esponenziale.

Presentiamo un algoritmo che permette di calcolare X_G^+ a partire da F

$$
\begin{array}{l}
\textbf{Input} \to \text{uno schema } R, \text{ un insieme } F \text{ di dipendenze funzionali su } R, \\
\qquad\quad \text{una decomposizione } \rho = \{R_1, R_2, \ldots, R_k\} \text{ di } R, \text{un sottoinsieme } X \text{ di } R
\end{array}
$$

$$
\textbf{Output} \to \text{la chiusura di } X \text{ rispetto a } G = \cup_{i=1}^{k} \pi_{R_i}(F) \text{ (nella variabile } Z)
$$

$$
\begin{array}{l}
\text{begin} \\
\quad Z := X \\
\quad S := \varnothing \\
\quad \text{for } i := 1 \text{ to } k \\
\quad \text{do} \qquad S := S \cup (Z \cap R_i)_F^+ \cap R_i \\
\quad \text{while } S \not\subset Z \\
\quad\quad \text{do} \\
\quad\quad \text{begin} \\
\quad\quad\quad Z := Z \cup S \\
\quad\quad\quad \text{for } i := 1 \text{ to } k \\
\quad\quad\quad \text{do} \qquad S := S \cup (Z \cap R_i)_F^+ \cap R_i \\
\quad\quad \text{end} \\
\text{end}
\end{array}
$$

Info

Con $S := S \cup (Z \cap R_i)_F^+ \cap R_i$ sostanzialmente si calcola la chiusura in F degli elementi (di cui cerchiamo di calcola la chiusura in G) rispetto al sottoschema R_i, infine facciamo l'intersezione con R_i in modo tale da avere al massimo tutti gli attributi contenuti di R_i.
In questo modo rispettiamo la definizione di $G = \cup_{i=1}^{k} \pi_{R_i}(F)$

Osservazione

Il while serve per determinare le dipendenze a cavallo tra gli schemi che sono contenute in F, infatti rifacendo la chiusura otterrò anche la transitività (se ho $A \to B$ in un sottoschema e $B \to C$ in un altro sottoschema allora nella chiusura finale avrò $A \to C$ che è una dipendenza di F)

Warning

L'algoritmo termina sempre, infatti ciò non vuol dire che una dipendenza $X \to Y$ è preservata

Per verificare se $X \to Y$ è preservata, in base al Lemma 2 e in base al teorema sull'uguaglianza $F^+ = F^A$, dobbiamo controllare se Y è contenuto nella copia finale della variabile Z (che conterrà la chiusura di X rispetto a G, X_G^+)

Teorema

Sia R uno schema di relazione, F un insieme di dipendenze funzionali su R, $\rho = \{R_1, R_2, \ldots, R_k\}$ una decomposizione di R e X un sottoinsieme di R. L'algoritmo dato calcola correttamente X_G^+, dove $G = \cup_{i=1}^{k} \pi_{R_i}(F)$.

Dimostrazione

Dobbiamo dimostrare che $A \in Z^{(f)} \Leftrightarrow A \in X_G^+$

Parte $\Rightarrow$

Mostreremo per induzione su i che $Z^{(i)} \subseteq X_G^+$, per ogni i (e in particolare per $i = f$)

- Base dell'induzione ($i = 0$): poiché $Z^{(0)} = X$ e $X \subseteq X^+$, si ha $Z^{(0)} \subseteq X_G^+$
- Ipotesi induttiva ($i > 0$): $Z^{(i-1)} \subseteq X_G^+$
- Passo induttivo: $Z^{(i)}$

Sia A un attributo in $Z^{(i)} - Z^{(i-1)}$ (elemento aggiunto al passo i) allora deve esistere R_j tale che $A \in (Z^{(i-1)} \cap R_j)_F^+ \cap R_j$.

Poiché $A \in (Z^{(i-1)} \cap R_j)_F^+$ vuol dire che $(Z^{(i-1)} \cap R_j) \to A \in F^+$

Possiamo quindi dire che siccome:

- $(Z^{(i-1)} \cap R_j) \to A \in F^+$
- $A \in R_j$
- $A \in Z^{(i-1)}$

allora $(Z^{(i-1)} \cap R_j) \to A \in G$ ($G = \cup_{i=1}^{k} \pi_{R_i}(F)$)

Per l'ipotesi induttiva ho $Z^{(i-1)} \subseteq X_G^+ \overset{\text{Lemma 1}}{\implies} X \to Z^{(i-1)} \in G^+ (= G^A)$
Per la regola di decomposizione si ha anche che $X \to (Z^{(i-1)} \cap R_j) \in G^+$ poiché $R_j \subseteq Z^{(i-1)}$
Dunque per transitività $X \to A \in G^+$ e quindi $A \in X_G^+$

Parte $\Rightarrow$

Vedi dispensa associata al corso (non necessario)

Esercizi

Example

$$R = (A, B, C, D)$$

$$F = \{AB \to C, D \to C, D \to B, C \to B, D \to A\}$$

Dire se la decomposizione $\rho = \{ABC, ABD\}$ preserva le dipendenze in F

In base a quanto visto basta verificare che $F \subseteq G^+$ cioè che ogni dipendenza funzionale in F si trova in G^+

Warning

In effetti è inutile controllare che vengano preservate le dipendenze tali che l'unione delle parti destra e sinistra è contenuta interamente in un sottoschema, perché secondo la definizione
$\pi_{R_i}(F) = \{X \to Y \mid X \to Y \in F^+ \wedge XY \subseteq R_j\}$

Tali dipendenze fanno parte per definizione di G

⚠ Warning

Per come è strutturato l'algoritmo, a Z possono solo venire aggiunti elementi (cioè non succede mai che un'attributo venga eliminato da Z), quindi quando Z arriva a contenere la **parte destra della dipendenza** possiamo essere sicuri che la dipendenza stessa è preservata e sospendere il seguito del procedimento (in un compito scritto questo va giustificato)

Menzionando esplicitamente l'osservazione fatta sopra, basta verificare che sia preservata la dipendenza $D \to C$

$Z = D$
$S = \varnothing$

Ciclo esterno sui sottoschemi ABC e ABD
$S = S \cup (D \cap ABC)_F^+ \cap ABC = \varnothing \cup (\varnothing)_F^+ \cap ABC = \varnothing \cup \varnothing \cap ABC = \varnothing$
$S = S \cup (D \cap ABD)_F^+ \cap ABD = S \cup (D)_F^+ \cap ABD = \varnothing \cup DCBA \cap ABD = A$

$ABD \not\subset D$ quindi entriamo nel ciclo while
$Z = Z \cup S = ABD$
Ciclo for interno al while sui sottoschemi ABC e ABD

$$S = S \cup (ABD \cap ABC)_F^+ \cap ABC = S \cup (AB)_F^+ \cap ABC = \\ = ABD \cup ABC \cap ABC = ABCD$$

$$S = S \cup (ABD \cap ABD)_F^+ \cap ABD = S \cup (ABD)_F^+ \cap ABD = \\ = ABCD \cup ABCD \cap ABD = ABCD \cup ABD = ABCD$$

$ABCD \not\subset ABD$ quindi rientriamo nel ciclo while
$Z = Z \cup S = ABCD$
Ciclo for interno al while sui sottoschemi ABC e ABD

$$S = S \cup (ABCD \cap ABC)_F^+ \cap ABC = S \cup (ABC)_F^+ \cap ABC = \\ = ABCD \cup ABC \cap ABC = ABCD$$

$$S = S \cup (ABCD \cap ABD)_F^+ \cap ABD = S \cup (ABD)_F^+ \cap ABD =$$
$$= ABCD \cup ABCD \cap ABD = ABCD \cup ABD = ABCD$$

$S \subset Z$ quindi STOP
L'algoritmo si ferma, ma va controllato il contenuto di Z
$Z = (D)_G^+ = ABCD$ e quindi $C \in (D)_G^+$, quindi la dipendenza è preservata

In base alle osservazioni sulle dipendenze sicuramente contenute in G e al fatto di aver verificato che $D \to C$ è preservata (era l'unica in dubbio), possiamo già dire che la decomposizione preserva le dipendenze

Example

$$R = (A, B, C, D, E)$$
$$F = \{AB \to E, B \to CE, ED \to C\}$$

Dire se la decomposizione $\rho = \{ABE, CDE\}$ preserva le dipendenze in F

In base a quanto visto basta verificare che $F \subseteq G^+$ cioè che ogni dipendenza funzionale in F si trova in G^+

Info

Come abbiamo verificato è inutile controllare che vengano preservate le dipendenze tali che l'unione delle parti destra e sinistra è contenuta interamente in un sottoschema, perché secondo la definizione
$\pi_{R_i}(F) = \{X \to Y \mid X \to Y \in F^+ \wedge XY \subseteq R_j\}$

In questo esempio vale per $AB \to E$ e per $ED \to C$
Quindi verifichiamo solo che venga preservata la dipendenza $B \to CE$

Verifichiamo che sia preservata $B \to CE$

$Z = B$
$S = \varnothing$
Ciclo esterno sui sottoschemi ABE e CDE

$$S = S \cup (B \cap ABE)_F^+ \cap ABE = \varnothing \cup (B)_F^+ \cap ABE = \varnothing \cup BCE \cap ABE = BE$$

$$S = BE \cup (B \cap CDE)_F^+ \cap CDE = BE \cup (\varnothing)_F^+ \cap CDE = BE$$

$BE \not\subset B$ quindi entriamo nel ciclo while
$Z = Z \cup S = B \cup BE = BE$

Ciclo for interno al while sui sottoschemi ABE e CDE

$S = BE \cup (BE \cap ABE)_F^+ \cap ABE = BE \cup (BE)_F^+ \cap ABE = BE \cup BCE \cap AB$

$S = BE \cup (BE \cap CDE)_F^+ \cap CDE = S \cup (E)_F^+ \cap CDE = BE \cup E \cap CDE = E$

$BE = BE(S \subset Z)$ quindi STOP
L'algoritmo si ferma, ma va controllato il contenuto di Z
$Z = (B)_G^+ = BE$
$E \in (B)_G^+$ ma $C \notin (B)_G^+$

Quindi la dipendenza $B \to CE$ non è preservata (nella chiusura manca uno degli attributi che dovrebbero essere determinati funzionalmente da B)