

BD 2 - Decomposizione di Schemi

Stefano Calzavara

Università Ca' Foscari Venezia



Università
Ca' Foscari
Venezia

Anomalie

Schemi di scarsa qualità soffrono di **anomalie**, che vanno ad ostacolare le operazioni di inserimento, cancellazione ed aggiornamento dei dati.

<u>ID</u>	NomeUtente	<u>CodiceLibro</u>	Titolo	Data
8432	Rossi Carlo	XY188A	Decameron	07-07
6613	Pastine Maurizio	XY090C	Canzoniere	01-08
7115	Paolicchi Laura	XY101A	Vita Nova	05-08
6825	Paolicchi Luca	XY701B	Adelchi	14-01
6825	Paolicchi Luca	XY008C	Amleto	17-08

Decomposizione di Schemi

L'eliminazione di anomalie è tipicamente basata sulla **decomposizione** di schemi mal definiti in schemi più piccoli, equivalenti ma più disciplinati.

<u>ID</u>	NomeUtente
8432	Rossi Carlo
6613	Pastine Maurizio
7115	Paolicchi Laura
6825	Paolicchi Luca

ID	<u>CodiceLibro</u>	Titolo	Data
8432	XY188A	Decameron	07-07
6613	XY090C	Canzoniere	01-08
7115	XY101A	Vita Nova	05-08
6825	XY701B	Adelchi	14-01
6825	PXY008C	Amleto	17-08

Decomposizione di Schemi

Definition (Proiezione)

Dati uno schema $R(T, F)$ e $Z \subseteq T$, la **proiezione** di F su Z è definita come l'insieme $\pi_Z(F) = \{X \rightarrow Y \in F^+ \mid X \cup Y \subseteq Z\}$.

Definition (Decomposizione)

Dato uno schema $R(T, F)$, una sua **decomposizione** è un insieme di schemi $\rho = \{R_1(T_1, F_1), \dots, R_n(T_n, F_n)\}$ tale che $\bigcup_i T_i = T$, $\forall i : T_i \neq \emptyset$ e $\forall i : F_i = \pi_{T_i}(F)$.

Visto che gli F_i sono determinati da F e dai T_i per proiezione, per leggibilità indicheremo una decomposizione di $R(T, F)$ con la notazione più compatta $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$.

Proprietà delle Decomposizioni

Sebbene decomporre uno schema possa correggere le anomalie, non tutte le decomposizioni sono desiderabili:

- 1 **perdita di informazione**: la decomposizione va ad introdurre dei dati spuri, che possono inficiare la correttezza di alcune query
- 2 **perdita di dipendenze**: la decomposizione perde alcune dipendenze funzionali, andando ad alterare la semantica dei dati rappresentati

Proprietà desiderabili: una buona decomposizione dovrebbe eliminare le anomalie, ma preservare i dati e le dipendenze.

Decomposizioni con Perdita di Informazione

Prima della decomposizione: R

Proprietario	Telefono	Macchina
Mario Rossi	423567	CG153SE
Mario Rossi	542635	PT267MV

Dopo la decomposizione: R_1, R_2

Proprietario	Telefono
Mario Rossi	423567
Mario Rossi	542635

Proprietario	Macchina
Mario Rossi	CG153SE
Mario Rossi	PT267MV

Qual è il telefono del proprietario della macchina targata CG153SE?

- $\pi_{Telefono}(\sigma_{Macchina=CG153SE}(R)) = \{423567\}$
- $\pi_{Telefono}(\sigma_{Macchina=CG153SE}(R_1 \bowtie R_2)) = \{423567, 542635\}$

Decomposizioni che Preservano i Dati

In generale, un'operazione di decomposizione può **introdurre nuovi dati**, come formalizzato dal seguente teorema.

Theorem

Sia $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$ una decomposizione di $R(T, F)$, allora per ogni istanza r di $R(T, F)$ si ha $r \subseteq \pi_{T_1}(r) \bowtie \dots \bowtie \pi_{T_n}(r)$.

Una decomposizione **preserva i dati** (non perde informazione) quando ciò non si verifica.

Definition (Decomposizione che Preserva i Dati)

La decomposizione $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$ di $R(T, F)$ **preserva i dati** se per ogni istanza r di $R(T, F)$ si ha $r = \pi_{T_1}(r) \bowtie \dots \bowtie \pi_{T_n}(r)$.

Decomposizioni che Preservano i Dati

Come possiamo verificare se una decomposizione preserva i dati?

Theorem

Sia $\rho = \{R_1(T_1), R_2(T_2)\}$ una decomposizione di $R(T, F)$, si ha che ρ preserva i dati sse $T_1 \cap T_2 \rightarrow T_1 \in F^+$ oppure $T_1 \cap T_2 \rightarrow T_2 \in F^+$.

Questo permette di ricondurre il problema di determinare se una certa decomposizione binaria preserva i dati al problema dell'implicazione, che ha costo **polinomiale**.

Dimostrazione

Sia $\rho = \{R_1(T_1), R_2(T_2)\}$ una decomposizione di $R(T, F)$, dimostriamo che se $T_1 \cap T_2 \rightarrow T_1 \in F^+$ allora ρ preserva i dati.

Proof.

- 1 Sia r un'istanza valida di $R(T, F)$ e $s = (\pi_{T_1} r) \bowtie (\pi_{T_2} r)$, dobbiamo dimostrare che per ogni $t \in s$ abbiamo anche $t \in r$.
- 2 Per definizione di s esistono due tuple $u, v \in r$ con $u[T_1] = t[T_1]$, $v[T_2] = t[T_2]$ e $u[T_1 \cap T_2] = v[T_1 \cap T_2] = t[T_1 \cap T_2]$.
- 3 Poichè $T_1 \cap T_2 \rightarrow T_1 \in F^+$, da $u[T_1 \cap T_2] = v[T_1 \cap T_2]$ otteniamo $u[T_1] = v[T_1]$ e quindi $t = v \in r$.



Il caso $T_1 \cap T_2 \rightarrow T_2 \in F^+$ è analogo.

Esempio 1

Si consideri $R(A, B, C, D)$ con $F = \{A \rightarrow BC\}$.

La decomposizione binaria $\{R_1(A, B, C), R_2(A, D)\}$ preserva i dati:

- $T_1 = \{A, B, C\}$ e $T_2 = \{A, D\}$
- $T_1 \cap T_2 = \{A\}$
- $A_F^+ = \{A, B, C\} = T_1$, quindi $T_1 \cap T_2 \rightarrow T_1 \in F^+$

Visualizzazione di questa proprietà nella prossima slide.

Esempio 1 - Preservazione dei Dati

Dipendenze funzionali: $F = \{A \rightarrow BC\}$

Dopo la decomposizione: R_1, R_2

Prima della decomposizione: R

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_2	b_2	c_2	d_2

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

A	D
a_1	d_1
a_1	d_2
a_2	d_2

In effetti abbiamo $R_1 \bowtie R_2 = R$. Attenzione: è solo un esempio!

Esempio 2

Si consideri $R(A, B, C, D)$ con $F = \{A \rightarrow B, C \rightarrow D\}$.

La decomposizione binaria $\{R_1(A, B), R_2(C, D)\}$ non preserva i dati:

- $T_1 = \{A, B\}$ e $T_2 = \{C, D\}$
- $T_1 \cap T_2 = \emptyset$
- abbiamo quindi $\{T_1 \cap T_2 \rightarrow T_1, T_1 \cap T_2 \rightarrow T_2\} \cap F^+ = \emptyset$

Controesempio nella prossima slide.

Esempio 2 - Perdita di Informazione

Dipendenze funzionali: $F = \{A \rightarrow B, C \rightarrow D\}$

Dopo la decomposizione: R_1, R_2

Prima della decomposizione: R

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2

A	B
a_1	b_1
a_2	b_2

C	D
c_1	d_1
c_2	d_2

Abbiamo che $(a_1, b_1, c_2, d_2) \in R_1 \bowtie R_2$, ma $(a_1, b_1, c_2, d_2) \notin R$.

Decomposizioni con Perdita di Dipendenze

Prima della decomposizione: R

Proprietario	Telefono	Macchina
Mario Rossi	423567	CG153SE
Mario Rossi	423567	PT267MV
Mario Rossi	542635	PT267MV

Dopo la decomposizione: R_1, R_2

Proprietario	Telefono
Mario Rossi	423567
Mario Rossi	542635

Telefono	Macchina
423567	CG153SE
423567	PT267MV
542635	PT267MV

Supponiamo di voler inserire (Luca Bianchi, 421448, CG153SE)

- nel primo caso violerei la dipendenza Macchina \rightarrow Proprietario
- nel secondo caso non me ne posso accorgere, se non dopo giunzione

Decomposizioni che Preservano le Dipendenze

Una decomposizione **preserva le dipendenze** sse l'unione delle dipendenze indotte sui singoli schemi equivale alle dipendenze dello schema originale.

Definition (Decomposizione che Preserva le Dipendenze)

La decomposizione $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$ di $R(T, F)$ **preserva le dipendenze** se e solo se $\bigcup_i \pi_{T_i}(F) \equiv F$.

Come verificarlo alitmicamente? Appliciamo la definizione:

- 1 Calcoliamo le proiezioni $\pi_{T_i}(F) = \{X \rightarrow Y \in F^+ \mid X \cup Y \subseteq T_i\}$
- 2 Verifichiamo se $\bigcup_i \pi_{T_i}(F) \equiv F$

Al momento non sappiamo risolvere nessuno dei due problemi! Iniziamo a ragionare sul secondo, che è più semplice...

Verificare l'Equivalenza

Theorem

$F \equiv G$ se e solo se $F \subseteq G^+$ e $G \subseteq F^+$.

Proof.

- Sia $F \equiv G$, allora $F^+ = G^+$ per definizione. Dato che si ha $F \subseteq F^+$ e $G \subseteq G^+$, ottengo $F \subseteq G^+$ e $G \subseteq F^+$ come desiderato.
- Poichè $F \subseteq G^+$, osservo che $F^+ \subseteq (G^+)^+ = G^+$. Analogamente da $G \subseteq F^+$ ottengo $G^+ \subseteq (F^+)^+ = F^+$. Concludo che $F^+ = G^+$.



Verificare l'Equivalenza

Theorem

$F \equiv G$ se e solo se $F \subseteq G^+$ e $G \subseteq F^+$.

Sia $G = \bigcup_i \pi_{T_i}(F)$, per dimostrare che $F \equiv G$ osserviamo che:

- 1 $F \subseteq G^+$ è verificabile tramite il problema dell'implicazione, perchè equivale a verificare che per ogni $X \rightarrow Y \in F$ abbiamo $Y \subseteq X_G^+$
- 2 $G \subseteq F^+$ vale per definizione, quindi non serve neppure verificarlo

Ci manca quindi solo da calcolare $G = \bigcup_i \pi_{T_i}(F)$ per avere un algoritmo che verifica se le dipendenze sono preservate o meno.

Calcolo delle Proiezioni

Purtroppo non è possibile calcolare $G = \bigcup_i \pi_{T_i}(F)$ in modo efficiente, perchè il calcolo delle singole proiezioni $\pi_{T_i}(F)$ ha costo **esponenziale**.

Algoritmo

```
function PROJECT( $F, T_i$ )  
   $proj \leftarrow \emptyset$   
  for all  $X \subset T_i$  do  
     $Y \leftarrow X_F^+ \setminus X$   
     $proj \leftarrow proj \cup \{X \rightarrow Y \cap T_i\}$   
  return  $proj$ 
```

Riassunto

Alla luce di quanto discusso, possiamo verificare se la decomposizione $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$ di $R(T, F)$ preserva le dipendenze tramite il seguente algoritmo:

- 1 Calcola le proiezioni $\pi_{T_i}(F)$ per ogni $i \in [1, n]$
- 2 Calcola $G = \bigcup_i \pi_{T_i}(F)$
- 3 Verifica che per ogni $X \rightarrow Y \in F$ si abbia $Y \subseteq X_G^+$

Tale algoritmo ha costo **esponenziale** a causa del calcolo delle proiezioni.

Esempio (1/2)

Siano $R(A, B, C)$ e $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$. Vogliamo verificare se la decomposizione $\rho = \{R_1(A, B), R_2(B, C)\}$ preserva le dipendenze.

Calcoliamo $\pi_{AB}(F)$, considerando i due sottoinsiemi propri A e B :

- $A_F^+ = ABC$, quindi $A \rightarrow B \in \pi_{AB}(F)$
- $B_F^+ = BCA$, quindi $B \rightarrow A \in \pi_{AB}(F)$

Concludiamo quindi $\pi_{AB}(F) = \{A \rightarrow B, B \rightarrow A\}$.

Calcoliamo ora $\pi_{BC}(F)$, considerando i due sottoinsiemi propri B e C :

- $B_F^+ = BCA$, quindi $B \rightarrow C \in \pi_{BC}(F)$
- $C_F^+ = CAB$, quindi $C \rightarrow B \in \pi_{BC}(F)$

Concludiamo quindi $\pi_{BC}(F) = \{B \rightarrow C, C \rightarrow B\}$.

Esempio (2/2)

A questo punto possiamo calcolare:

$$G = \pi_{AB}(F) \cup \pi_{BC}(F) = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}.$$

Iteriamo sulle dipendenze $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ e verifichiamo che siano tutte derivabili da G :

- $A \rightarrow B$: abbiamo $B \in A_G^+ = ABC$
- $B \rightarrow C$: abbiamo $C \in B_G^+ = BAC$
- $C \rightarrow A$: abbiamo $A \in C_G^+ = CBA$

Concludiamo che la decomposizione in esame preserva le dipendenze.

Ottimizzare la Verifica

Per fortuna non ci interessa davvero calcolare $G = \bigcup_i \pi_{T_i}(F)$, ma ci basta verificare che per ogni $X \rightarrow Y \in F$ abbiamo $Y \subseteq X_G^+$. In effetti esiste un algoritmo che calcola X_G^+ in tempo **polinomiale** senza calcolare G .

Algoritmo

```
function FC( $X, F, \rho$ )  
   $res_{old} \leftarrow \emptyset$   
   $res_{new} \leftarrow X$   
  while  $res_{new} \neq res_{old}$  do  
     $res_{old} \leftarrow res_{new}$   
    for all  $R_i(T_i) \in \rho$  do  
       $res_{new} \leftarrow res_{new} \cup ((res_{new} \cap T_i)_F^+ \cap T_i)$   
  return  $res_{new}$ 
```

Ottimizzare la Verifica

Dati uno schema $R(T, F)$ e $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$, è quindi possibile verificare se ρ preserva le dipendenze tramite il seguente algoritmo.

Algoritmo

```
function PRESERVEDEPS( $R(T, F), \rho$ )  
  for all  $X \rightarrow Y \in F$  do  
    if  $Y \not\subseteq FC(X, F, \rho)$  then return False  
  return True
```

La complessità dell'algoritmo ottimizzato è quindi **polinomiale**.

Esempio (1/3)

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ e $\rho = \{R_1(A, B), R_2(B, C)\}$.

Partiamo dalla dipendenza $A \rightarrow B$:

- 1 Partiamo inizializzando $FC(A, F, \rho) = \{A\}$
- 2 Consideriamo $R_1(\{A, B\})$, abbiamo:
 $(\{A\} \cap \{A, B\})_F^+ \cap \{A, B\} = A_F^+ \cap \{A, B\} = \{A, B\}$, quindi aggiungiamo B a $FC(A, F, \rho)$
- 3 Consideriamo $R_2(\{B, C\})$, abbiamo:
 $(\{A, B\} \cap \{B, C\})_F^+ \cap \{B, C\} = B_F^+ \cap \{B, C\} = \{B, C\}$, quindi aggiungiamo C a $FC(A, F, \rho)$
- 4 Otteniamo quindi $B \in FC(A, F, \rho) = \{A, B, C\}$

Esempio (2/3)

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ e $\rho = \{R_1(A, B), R_2(B, C)\}$.

Passiamo alla dipendenza $B \rightarrow C$:

- 1 Partiamo inizializzando $FC(B, F, \rho) = \{B\}$
- 2 Consideriamo $R_1(\{A, B\})$, abbiamo:
 $(\{B\} \cap \{A, B\})_F^+ \cap \{A, B\} = B_F^+ \cap \{A, B\} = \{A, B\}$, quindi aggiungiamo A a $FC(B, F, \rho)$
- 3 Consideriamo $R_2(\{B, C\})$, abbiamo:
 $(\{A, B\} \cap \{B, C\})_F^+ \cap \{B, C\} = B_F^+ \cap \{B, C\} = \{B, C\}$, quindi aggiungiamo C a $FC(B, F, \rho)$
- 4 Otteniamo quindi $C \in FC(B, F, \rho) = \{A, B, C\}$

Esempio (3/3)

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ e $\rho = \{R_1(A, B), R_2(B, C)\}$.

Passiamo infine alla dipendenza $C \rightarrow A$:

- 1 Partiamo inizializzando $FC(C, F, \rho) = \{C\}$
- 2 Consideriamo $R_1(\{A, B\})$, abbiamo: $(\{C\} \cap \{A, B\})_F^+ \cap \{A, B\} = \emptyset$, quindi per ora non aggiungiamo niente a $FC(C, F, \rho)$
- 3 Consideriamo $R_2(\{B, C\})$, abbiamo:
 $(\{C\} \cap \{B, C\})_F^+ \cap \{B, C\} = C_F^+ \cap \{B, C\} = \{B, C\}$, quindi aggiungiamo B a $FC(C, F, \rho)$
- 4 Consideriamo $R_1(\{A, B\})$, abbiamo:
 $(\{B, C\} \cap \{A, B\})_F^+ \cap \{A, B\} = B_F^+ \cap \{A, B\} = \{A, B\}$, quindi aggiungiamo A a $FC(C, F, \rho)$
- 5 Otteniamo quindi $A \in FC(C, F, \rho) = \{A, B, C\}$

Teorema

In generale la preservazione dei dati è **indipendente** dalla preservazione delle dipendenze. Esiste però un teorema che collega le due proprietà, che è particolarmente utile perchè applicabile a decomposizioni non binarie.

Theorem

Sia $\rho = \{R_1(T_1), \dots, R_n(T_n)\}$ una decomposizione di $R(T, F)$ che preserva le dipendenze e tale che almeno uno degli insiemi di attributi T_j sia una superchiave per $R(T, F)$, allora ρ preserva anche i dati.

Per gli studenti interessati: la dimostrazione è riportata nel testo.

Checkpoint

Punti Chiave

- La definizione di decomposizione
- Decomposizioni che preservano i dati e le dipendenze
- Tecniche algoritmiche per dimostrare che una certa decomposizione preserva i dati e le dipendenze (con costo polinomiale)

Materiale Didattico

Fondamenti di Basi di Dati: Sezione 5.3