Appunti Basi di Dati

Luca Seggiani

15 Maggio 2024

Decomposizione di schemi
Dato unos schema $R(T)$, l'insieme di schemi $\rho = \{R_1(T_1), ..., R_k(T_k)\}$ è una **decomposizione** di R solo se $\cup_i T_i = T$ (l'unione degli schemi dà lo schema di partenza). Si noti che questo non richiede che gli schemi R_i siano disgiunti. Una decomposizione equivale allo schema di partenza in quanto

- Preserva i dati;
- Preserva le dipendenze funzionali.

Teorema della perdita di dati
Si ha che se $\rho = \{R_1(T_1)...., R_k(T_k)\}$ è una decomposizione di $R(T, F)$, allora per ogni relazione r che soddisfa $R(T, F)$:

$$r \subseteq \pi_{T_1}(r) \bowtie ... \bowtie \pi_{T_k}(r)$$

Ciò significa che una decomposizione ha perdita di dati quando, ricostruendo una relazione, otteniamo più n-uple della relazione originaria.

Decomposizioni che preservano i dati
Dato uno schema $R(T, F)$ e una decomposizione $\rho\{R_1(T_1), ..., R_k(T_k)\}$, ρ preserva i dati se e solo se, per ogni relazione r che soddisfa $R(T, F)$ si ha:

$$r = \pi_{T_1}(r) \bowtie ... \bowtie \pi_{T_k}(r)$$

Ciò significa che, per una decomposizione che preserva i dati, ogni istanza valida r della relazione di partenza deve essere identica al join naturale delle sue proiezioni sui T_1 (cioè ricostruendo la relazione si ottiene la relazione di partenza e nulla di più).

Teorema di preservazione dei dati
Sia $\rho = \{R_1(T_1), R_2(T_2)\}$ una decomposizione di $R(T, F)$. Essa preserva i dati se e solo se $T_1 \cap T_2 \to T_1 \in F^+$ oppure $T_1 \cap T_2 \to T_2 \in F^+$. In altre

parole, gli attributi comuni alle due relazioni devono essere chiave in una delle due tabelle (relazioni).

Proiezione di un insieme di dipendenze

Dato $R(T, F)$ e $T_1 \subseteq T$, la proiezione dell'insieme di dipendenze F sull'insieme di attributi T_i è:

$$\pi_{T_i} = \{X \to Y \in F^+ | X, Y \subseteq T_i\}$$

Nota bene che la proiezione si costruisce sulle dipendenze di F^+, non di F soltanto.

Algoritmo per il calcolo di $\pi_{T_i}(F)$

Si presenta un'algoritmo per il calcolo della proiezione dell'insieme di dipendenze appena definita.

- Input: $R(T, F)$ e $T_i \subseteq T$
- Output: $\pi_{T_i}(F)$

```
Z vuoto
for each Y in T do
  metti Y - Y in W
  metti Z unito (Y -> (W intersecato T_i)) in Z
return Z
```

Questo algoritmo ha, nel caso pessimo, complessità esponenziale.

Decomposizioni che preservano le dipendenze

Dato uno schema $R(T, F)$, e una decomposizione $\rho = \{R_1(T_1), ..., R_k(T_k)\}$, ρ è una decomopsizione di $R(T, F)$ che preserva le dipendenze se e solo se $\cup_i \pi_{T_i}(F) \equiv F$ In altre parole, la decomposizione di $R(T, F)$ in due relazioni con attributi X e Y preserva le dipendenze se $\pi_X(F) \cup \pi_Y(F) \equiv F$, cioè se $(\pi_X(F) \cup \pi_Y(F)) = F^+$. Dovremo verificare quest'ultima uguaglianza per dimostrare che una decomposizione di $R(T, F)$ preserva le dipendenze. Fare ciò significa:

- Calcolare la proiezione di un insieme di dipendenze funzionali su un insieme di attributi, cosa che richiede un algoritmo a complessità esponenziale;
- Determinare l'equivalenza di due insiemi di dipendenze funzionali X e G, cosa che richiede un'algoritmo a complessità poliniomiale:
  - Per ogni $X \to Y \in F$, calcoliamo X_G^+ e verifichiamo se $Y \in X_G^+$;
  - Per ogni $X \to Y \in G$, calcoliamo X_F^+ e verifichiamo se $Y \in X_F^+$;

Algoritmo per la decomposizione in BCNF
Possiamo adesso vedere un'algoritmo che decompone uno schema $R(T, F)$ nella sua forma normale di Boyce-Codd:

- Input: $R(T, F)$ con F in forma $X \to A$, non normalizzata
- Output: ρ che preserva i dati

```
metti R(T,F) in P
while esiste R_i(T_i, F_i) in P che non e in BCNF do
  for each X -> A in F_i do
    if A not in X and T_i not in X+ then
      metti R_i(T_i - A, proiezione su T_1 - A di F_i) in R_1 #questi sono tutti gli attributi di R - A
      metti R_i(X + A, proiezione su X + A di F_i) in R_2 # questi sono tutti gli attributi della dipendenza funzionale in esame
      metti P - (R_i unito a {R_1,R_2}) in P
      break
return P
```

Questo algoritmo termina e produce una decomposizione della relazione tale che:

- La decomposizione prodotta è in BCNF;
- La decomposizione prodotta preserva i dati.

Non è detto che la decomposizione preservi le dipendenze!

Qualità delle decomposizioni
Una decomposizione dovrebbe sermpe garantire:

- Di essere in BCNF;
- Di non presentare perdite di dati, in modo da permettere la ricostruzione dell'informazione originale attraverso join naturali;
- Di conservare le dipendenze funzionali, in modo da mantenere i vincoli di integrità originari.

Quando non si riesce a raggiungere una forma normale di Boyce-Codd, probabilmente c'è stato un cattivo processo di progettazione prima della normalizzazione. In ogni caso, esiste una forma normale alternativa (e meno restrittiva) alla Boyce-Codd:

Terza forma normale
Una relazione $R(T, F)$ è in terza forma normale (3NF) se e solo se, per ogni dipendenza funzionale non banale $X \to A \in F^+$, è verificata almeno una delle due condizioni:

- X è una superchiave di R (come nella Boyce-Codd);
- A è contenuto in almeno una chiave di R (si dice che A è un attributo primo).

Vediamo che la BCNF implica la 3NF: uno schema in BCNF è automaticamente in 3NF, ma non tutti gli schemi in 3NF sono in BCNF. Il problema della 3NF è la sua **verifica**: il problema di decisione è NP-completo. Il miglior algoritmo deterministico di risoluzione ha complessità esponenziale nel caso peggiore. Questo è dato dalla ricerca degli attributi primi, cioè le chiavi, che ha complessità esponenziale. Tuttavia, si può sempre ottenere una decomposizione in 3NF che preserva dati e dipendenze funzionali.

Algoritmo per la decomposizione in 3NF

Ci basiamo su un intuizione: dato un insieme di attributi T e una copertura minimale G, si divide G in gruppi G_i in modo che tutte le dipendenze funzionali di ogni gruppo G_i abbiano la stessa parte sinistra. Da ogni gruppo G_i si definisce uno schema di relazione composto da tutti gli attributi che appaioni in G_i, la cui chiave (detta **chiave sintetizzata**) è la parte sinistra comune. L'algoritmo risulta quindi:

- Input: $R(T, F)$
- Output: ρ che preserva dati, dipendenze ed è in 3NF

```
metti la copertura minimale di F in G
poni P vuoto
#sostituisci ogni insieme di dipendenze {X -> A_1, ..., X -> A_h} con X -> A_1, ..., A_h
for each X -> Y in G do
  #crea uno schema con attributi XY in P
#elimina da P ogni schema che sia contenuto in un'altro schema di P
#se P non contiene nessuno schema i cui attributi costituiscono una superchiave di R, aggiungere a P uno schema con attributi W, dove W e' una chiave di R
```

L'esecuzione dell'algoritmo termina e produce una decomposizione tale che:

- La decomposizione prodotta è in 3NF;
- La decomposizione prodotta preserva i dati e le dipendenze funzionali.

Questo algoritmo ha complessità polinomiale: paradossalmente, trovare una soluzione è polinomiale, ma verificarla è esponenziale