

Appunti Basi di Dati

Luca Seggiani

8 Maggio 2024

Calcolare F^+ è molto costoso: l'algoritmo è a complessità esponenziale. Spesso però, quello che ci interessa veramente è verificare se F^+ contiene una certa dipendenza e non generare l'intera lista di chiusura. Per fare ciò basta calcolare X^+ per il teorema di chiusura degli attributi:

$$F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$$

Algoritmo per il calcolo di X^+

Vediamo quindi come calcolare la chiusura X^+ :

- Input: $R(T, F)$, $X \subseteq T$;
- Output: X^+ chiusura.

```
1 metti X in X+
2 while (X+ non cambia) do
3   foreach W -> V in F do
4     if W in X+ and V not in X+ then
5       metti V in X+
6 return X+
```

Chiavi

Dato uno schema $R(T, F)$, un'insieme di attributi $K \subseteq T$ si dice **superchiave** di R se la dipendenza funzionale $K \rightarrow T$ è implicata da F , ovvero se $K \rightarrow T \in F^+$. Un'insieme di attributi $K \subseteq T$ si dice a questo punto **chiave** di R se K è una superchiave di R e non esiste alcun sottoinsieme proprio di K che sia superchiave di R . Dato che in uno schema possono esserci più chiavi, di solito si identifica una chiave primaria che possa fare da identificatore per tutte le n-uple dello schema. Tutte le altre chiavi si dicono chiavi candidate.

Algoritmo per tutte le chiavi

Il problema di trovare tutte le chiavi di una relazione $R(Z)$ ha complessità esponenziale nel caso peggiore:

- Gli attributi che stanno solo a sinistra sono in tutte le chiavi. Si chiami N questo insieme;
- Gli attributi che stanno solo a destra non sono in nessuna chiave. Si chiami M questo insieme;
- Si aggiunge a N un'attributo alla volta tra quelli che non sono né in N né in M , poi una coppia di attributi, e così via. Chiamiamo X_i questo sottoinsieme di attributi: ad ogni aggiunta controlleremo se la dipendenza $N \cup X_i \rightarrow Z$ esiste.

Verifica di una chiave

Spesso è molto più conveniente verificare se una chiave è tale, piuttosto che trovare ogni possibile chiave. Per fare ciò, possiamo usare l'algoritmo per il calcolo della chiusura di un'insieme di attributi:

- $X \subseteq T$ è superchiave di $R(T, F)$ se e solo se $X \rightarrow T \in F^+$, ovvero $T \subseteq X^+$
- $X \subseteq T$ è chiave di $R(T, F)$ se e solo se $T \subseteq X^+$, e non esiste $Y \subset X$ tale che $T \subseteq Y^+$.

Equivalenza

Due insiemi di dipendenze funzionali F e G sugli attributi T di una relazione $R(T)$ sono equivalenti, in simboli $T \equiv G$, se e solo se $F^+ = G^+$. La relazione di equivalenza permette di stabilire se due insiemi di dipendenza rappresentano gli stessi fatti. Per verificare l'equivalenza è sufficiente che:

- Tutte le dipendenze di F appartengano a G^+ ;
- Tutte le dipendenze di G appartengano a F^+ .

Ridondanza

Sia F un insieme di dipendenze funzionali. Data $X \rightarrow Y \in F$, X contiene un **attributo estraneo** se e solo se:

$$(F - \{X \rightarrow Y\}) \cup (X - \{A \rightarrow\}Y) \equiv F$$

ovvero $X - \{A\} \rightarrow Y \in F^+$

$X \rightarrow Y$ è una **dipendenza ridondante** se e solo se $(F - \{X \rightarrow Y\}) \equiv F$, ovvero $X \rightarrow Y \in (F - \{X \rightarrow Y\})^+$. Le dipendenze che non contengono attributi estranei e la cui parte destra è un unico attributo sono dette **dipendenze elementari**.

Copertura minimale

Sia F un'insieme di dipendenze funzionali. F è una **copertura minimale** (detta anche *insieme minimale* e *copertura canonica*) se e solo se:

- Ogni parte destra di una dipendenza ha un unico attributo;
- Le dipendenze non contengono attributi estranei;
- Non esistono dipendenze ridondanti;

In soldoni, la copertura minimale rappresenta l'insieme di dipendenze funzionali F' più piccolo che implica tutte le dipendenze di F .

Algoritmo per il calcolo della copertura minimale

- Input: F ;
- Output G *copertura minimale*:

```

1 metti F in G
2 for each X -> G do
3   metti X in Z
4   for each A in X Do
5     if Y in (Z - {A})+ - F then
6       metti Z - {A} in Z
7       metti (G - {X -> Y}) U (Z -> Y) in G
8   foreach F in G do
9     if f in (G-{f})+ then
10      metti G - {f} in G
11 return G

```

Teorema della copertura minimale

Il precedente algoritmo dimostra il fatto che per ogni insieme di dipendenze funzionali F esiste una copertura minimale. Si noti che questo non afferma che la copertura minimale è unica: possono tranquillamente esistere coperture minimali equivalenti.

1 Normalizzazione

Eliminare le anomalie

La teoria che abbiamo sviluppato finora viene usata per identificare le anomalie in uno schema mal definito. Definiamo quindi il concetto di **forma normale**: una forma normale è una proprietà che deve essere soddisfatta dalla dipendenza fra attributi di schemi "ben fatti". Noi vedremo due esempi: la **forma normale di Boyce-Codd**, e un suo miglioramento, la **terza forma normale**.

Forma normale di Boyce-Codd

Uno schema $R(T, F)$ è in forma normale di Boyce-Codd (BCNF) se e solo se

per ogni dipendenza funzionale non banale $X \rightarrow Y \in F^+$, X è superchiave di R . Per definizione, il fatto che uno schema sia o meno in BCNF dipende dalla chiusura F^+ , non dalla specifica copertura F . Calcolare F^+ , come abbiamo visto, ha complessità esponenziale: fortunatamente esiste un'algoritmo di complessità polinomiale per valutare se uno schema è in forma BCNF. Si usa il corollario: uno schema $R(T, F)$ con F copertura minimale è in BCNF se e solo se per ogni dipendenza funzionale elementare $X \rightarrow A \in F$, X è superchiave.

- Input: $R(T, F)$;
- Output true se R è in BCNF, false altrimenti.

```

1 for each X -> Y in F do
2   if Y not in X and T not in X+ then
3     return false
4 return true

```