

BD 2 - Chiavi & Coperture Canoniche

Luca Cosmo

Università Ca' Foscari Venezia



Università
Ca' Foscari
Venezia

Chiavi ed Attributi Primi

Definition (Superchiave)

Dato uno schema di relazione $R(T, F)$, un insieme di attributi $X \subseteq T$ è una **superchiave** di R se e solo se $X \rightarrow T \in F^+$.

Definition (Chiave)

Una **chiave** è una superchiave minimale, cioè una superchiave tale che nessuno dei suoi sottoinsiemi propri sia a sua volta una superchiave.

Definition (Attributi Primi)

Un attributo è **primo** se e solo se appartiene ad almeno una chiave.

Verifica di Superchiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **superchiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Calcola la chiusura X_F^+
- 2 Verifica se $X_F^+ = T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è superchiave:

Verifica di Superchiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **superchiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Calcola la chiusura X_F^+
- 2 Verifica se $X_F^+ = T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è superchiave:

- 1 $ABD_0^+ = ABD$

Verifica di Superchiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **superchiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Calcola la chiusura X_F^+
- 2 Verifica se $X_F^+ = T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è superchiave:

- 1 $ABD_0^+ = ABD$
- 2 $ABD_1^+ = ABCD$ (tramite $AB \rightarrow C$)

Verifica di Superchiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **superchiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Calcola la chiusura X_F^+
- 2 Verifica se $X_F^+ = T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è superchiave:

- 1 $ABD_0^+ = ABD$
- 2 $ABD_1^+ = ABCD$ (tramite $AB \rightarrow C$)
- 3 $ABD_2^+ = ABCDE$ (tramite $A \rightarrow E$)

Verifica di Superchiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **superchiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Calcola la chiusura X_F^+
- 2 Verifica se $X_F^+ = T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è superchiave:

- 1 $ABD_0^+ = ABD$
- 2 $ABD_1^+ = ABCD$ (tramite $AB \rightarrow C$)
- 3 $ABD_2^+ = ABCDE$ (tramite $A \rightarrow E$)
- 4 $ABD_3^+ = ABCDEF$ (tramite $B \rightarrow F$)

Verifica di Chiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **chiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Verifica se X è una superchiave. Se non lo è, non è una chiave
- 2 Verifica che per ogni $A \in X$ si abbia $(X \setminus \{A\})_F^+ \neq T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è chiave, perchè esso è una superchiave (come dimostrato) ed inoltre abbiamo che:

Verifica di Chiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **chiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Verifica se X è una superchiave. Se non lo è, non è una chiave
- 2 Verifica che per ogni $A \in X$ si abbia $(X \setminus \{A\})_F^+ \neq T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è chiave, perchè esso è una superchiave (come dimostrato) ed inoltre abbiamo che:

- 1 A non può essere rimosso: $BD_G^+ = BDF$

Verifica di Chiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **chiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Verifica se X è una superchiave. Se non lo è, non è una chiave
- 2 Verifica che per ogni $A \in X$ si abbia $(X \setminus \{A\})_F^+ \neq T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è chiave, perchè esso è una superchiave (come dimostrato) ed inoltre abbiamo che:

- 1 A non può essere rimosso: $BD_G^+ = BDF$
- 2 B non può essere rimosso: $AD_G^+ = ADE$

Verifica di Chiave

Dato $R(T, F)$, possiamo verificare se $X \subseteq T$ è una **chiave** tramite il seguente algoritmo di costo **polinomiale**:

- 1 Verifica se X è una superchiave. Se non lo è, non è una chiave
- 2 Verifica che per ogni $A \in X$ si abbia $(X \setminus \{A\})_F^+ \neq T$

Example

Si consideri la relazione $R(T, G)$, con $T = ABCDEF$ e $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. ABD è chiave, perchè esso è una superchiave (come dimostrato) ed inoltre abbiamo che:

- 1 A non può essere rimosso: $BD_G^+ = BDF$
- 2 B non può essere rimosso: $AD_G^+ = ADE$
- 3 D non può essere rimosso: $AB_G^+ = ABCEF$

Trovare una Chiave

Dato $R(T, F)$, è possibile trovare una sua chiave in tempo **polinomiale**.
L'idea dell'algoritmo è di partire da T e rimuovere uno ad uno tutti gli attributi che non sono indispensabili per derivare T .

Algoritmo

```
function FINDKEY( $R(T, F)$ )  
   $K \leftarrow T$   
  for all  $A \in T$  do  
    if  $(K \setminus \{A\})_F^+ = T$  then  
       $K \leftarrow K \setminus \{A\}$   
  return  $K$ 
```

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

1 Inizializziamo $K_0 = ABCDEF$

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto
- 4 Rimuoviamo C da K_1 : $BDEF_G^+ = ABCDEF$, quindi C deve essere rimosso ed aggiorniamo la chiave a $K_2 = BDEF$

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto
- 4 Rimuoviamo C da K_1 : $BDEF_G^+ = ABCDEF$, quindi C deve essere rimosso ed aggiorniamo la chiave a $K_2 = BDEF$
- 5 Rimuoviamo D da K_2 : $BEF_G^+ = ABCEF$, quindi D va tenuto

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto
- 4 Rimuoviamo C da K_1 : $BDEF_G^+ = ABCDEF$, quindi C deve essere rimosso ed aggiorniamo la chiave a $K_2 = BDEF$
- 5 Rimuoviamo D da K_2 : $BEF_G^+ = ABCEF$, quindi D va tenuto
- 6 Rimuoviamo E da K_2 : $BDF_G^+ = BDF$, quindi E va tenuto

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto
- 4 Rimuoviamo C da K_1 : $BDEF_G^+ = ABCDEF$, quindi C deve essere rimosso ed aggiorniamo la chiave a $K_2 = BDEF$
- 5 Rimuoviamo D da K_2 : $BEF_G^+ = ABCEF$, quindi D va tenuto
- 6 Rimuoviamo E da K_2 : $BDF_G^+ = BDF$, quindi E va tenuto
- 7 Rimuoviamo F da K_2 : $BDE_G^+ = ABCDEF$, quindi F deve essere rimosso ed aggiorniamo la chiave a $K_3 = BDE$

Trovare una Chiave: Esempio

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$. Costruiamo una chiave:

- 1 Inizializziamo $K_0 = ABCDEF$
- 2 Rimuoviamo A da K_0 : $BCDEF_G^+ = ABCDEF$, quindi A deve essere rimosso ed aggiorniamo la chiave a $K_1 = BCDEF$
- 3 Rimuoviamo B da K_1 : $CDEF_G^+ = ACDEF$, quindi B va tenuto
- 4 Rimuoviamo C da K_1 : $BDEF_G^+ = ABCDEF$, quindi C deve essere rimosso ed aggiorniamo la chiave a $K_2 = BDEF$
- 5 Rimuoviamo D da K_2 : $BEF_G^+ = ABCEF$, quindi D va tenuto
- 6 Rimuoviamo E da K_2 : $BDF_G^+ = BDF$, quindi E va tenuto
- 7 Rimuoviamo F da K_2 : $BDE_G^+ = ABCDEF$, quindi F deve essere rimosso ed aggiorniamo la chiave a $K_3 = BDE$

L'algoritmo ritorna la chiave $K_3 = BDE$. E' l'unica chiave?

Trovare l'Insieme delle Chiavi

Dato $R(T, F)$, trovare **tutte** le chiavi ha costo **esponenziale**, perchè ogni sottoinsieme di T è potenzialmente una chiave. Esiste però un algoritmo piuttosto ottimizzato per la ricerca di tutte le chiavi.

Intuizione

- Generiamo le possibili chiavi dalle più piccole alle più grandi
- Rappresentiamo i candidati da testare nella forma $X :: (Y)$, dove X è l'insieme degli attributi da testare come chiave, e Y l'insieme dei possibili attributi da aggiungere a X qualora $X_F^+ \neq T$.
- Se $X_F^+ = T$, allora X è una chiave e possiamo scartare $X :: (Y)$
- Altrimenti calcoliamo $Y \setminus X_F^+ = \{A_1, \dots, A_n\}$ e generiamo i nuovi candidati $XA_1 :: (A_2, \dots, A_n), XA_2 :: (A_3, \dots, A_n), \dots, XA_n :: ()$
- Nota che se un attributo non compare mai alla destra di una dipendenza funzionale, allora esso deve fare parte di tutte le chiavi. L'insiemi di tali attributi sarà il primo che testeremo.

Trovare l'Insieme delle Chiavi

```
function FINDALLKEYS( $R(T, F)$ )  
   $Z = \{B \in T \mid \forall X \rightarrow Y \in F : B \notin Y\}$   
   $Cand = [Z :: (T \setminus Z)]$   
   $Keys = []$   
  while  $Cand \neq []$  do  
     $X :: (Y) = \text{HEAD}(Cand)$   
     $Cand = \text{TAIL}(Cand)$   
    if  $\nexists K \in Keys : K \subset X$  then  
      if  $X_F^+ = T$  then  
         $Keys = Keys + X$   
      else  
         $A_1, \dots, A_n = Y \setminus X_F^+$   
        for  $i \in 1, \dots, n$  do  
           $Cand = Cand + XA_i :: (A_{i+1}, \dots, A_n)$ 
```

Trovare l'Insieme delle Chiavi: Esempio (1/5)

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$:

- $Cand = [BD :: (ACEF)]$
- $Keys = []$
- $X :: (Y) = BD :: (ACEF)$
- $X_G^+ = BD_G^+ = BDF$, quindi BD non è una chiave
- $Y \setminus X_G^+ = ACE$

Aggiornamento:

- $Cand = [BDA :: (CE), BDC :: (E), BDE :: ()]$
- $Keys = []$

Trovare l'Insieme delle Chiavi: Esempio (2/5)

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$:

- $Cand = [BDA :: (CE), BDC :: (E), BDE :: ()]$
- $Keys = []$
- $X :: (Y) = BDA :: (CE)$
- $X_G^+ = BDA_G^+ = BDACEF$, quindi BDA è una chiave

Aggiornamento:

- $Cand = [BDC :: (E), BDE :: ()]$
- $Keys = [BDA]$

Trovare l'Insieme delle Chiavi: Esempio (3/5)

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$:

- $Cand = [BDC :: (E), BDE :: ()]$
- $Keys = [BDA]$
- $X :: (Y) = BDC :: (E)$
- $X_G^+ = BDC_G^+ = BDCF$, quindi BDC non è una chiave
- $Y \setminus X_G^+ = E$

Aggiornamento:

- $Cand = [BDE :: (), BDCE :: ()]$
- $Keys = [BDA]$

Trovare l'Insieme delle Chiavi: Esempio (4/5)

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$:

- $Cand = [BDE :: (), BDCE :: ()]$
- $Keys = [BDA]$
- $X :: (Y) = BDE :: ()$
- $X_G^+ = BDE_G^+ = BDEACF$, quindi BDE è una chiave

Aggiornamento:

- $Cand = [BDCE :: ()]$
- $Keys = [BDA, BDE]$

Trovare l'Insieme delle Chiavi: Esempio (5/5)

Sia $G = \{AB \rightarrow C, E \rightarrow A, A \rightarrow E, B \rightarrow F\}$:

- $Cand = [BDCE :: ()]$
- $Keys = [BDA, BDE]$
- $X :: (Y) = BDCE : ()$
- X contiene la chiave BDE , quindi non viene considerato

Poichè $Cand = []$, l'algoritmo termina.

Verifica di Primalità

Dato $R(T, F)$, il problema di verificare se un attributo $A \in T$ è **primo** ha complessità **esponenziale**:

- più precisamente, si può dimostrare che è un problema **NP-completo**
- ciò implica che non esistono soluzioni significativamente più efficienti dell'approccio ovvio di generare tutte le possibili chiavi!
- questo è l'approccio che useremo per trovare l'insieme degli attributi primi quando sarà necessario all'interno del corso

Forma Canonica

Abbiamo visto vari algoritmi che operano sull'insieme delle dipendenze funzionali. Per questo motivo è utile portare tale insieme in una forma più “disciplinata”, detta **forma canonica**, equivalente all'originale.

Definition (Equivalenza)

Due insiemi di dipendenze funzionali F e G sono **equivalenti**, indicato con $F \equiv G$, se e solo se $F^+ = G^+$.

Se $F = G$ allora $F \equiv G$, ma in generale non vale il contrario.

Forma Canonica

Definition (Attributo Estraneo)

Sia $X \rightarrow Y \in F$. L'attributo $A \in X$ è **estraneo** sse $X \setminus \{A\} \rightarrow Y \in F^+$.

Definition (Dipendenza Ridondante)

La dipendenza $X \rightarrow Y \in F$ è **ridondante** sse $X \rightarrow Y \in (F \setminus \{X \rightarrow Y\})^+$.

Definition (Forma Canonica)

F è in **forma canonica** se e solo se per ogni $X \rightarrow Y \in F$:

- 1 $|Y| = 1$;
- 2 X non contiene attributi estranei;
- 3 $X \rightarrow Y$ non è ridondante.

Copertura Canonica

Definition (Copertura Canonica)

G è una **copertura canonica** di F sse $F \equiv G$ e G è in forma canonica.

Theorem

Per ogni insieme di dipendenze F esiste una copertura canonica.

La dimostrazione è costruttiva: definiamo un algoritmo per produrre una copertura canonica. Si osservi che uno stesso insieme di dipendenze può avere più coperture canoniche.

Copertura Canonica

L'algoritmo applica i seguenti 3 passi:

- 1 Decomporre tutte le dipendenze funzionali che hanno più attributi sulla destra. $X \rightarrow Y \Rightarrow \{X \rightarrow A \mid A \in Y\}$
- 2 Togliere dalla parte sinistra delle dipendenze gli attributi che non impediscono di derivare la dipendenza stessa.
 $X \rightarrow A \Rightarrow X \setminus Z \mid A \in (X \setminus Z)_F^+$
- 3 Togliere le dipendenze funzionali non essenziali per derivare la dipendenza stessa. $X \rightarrow A \Rightarrow F^{new} = F \setminus \{X \rightarrow A\} \mid A \in X_{F \setminus \{X \rightarrow A\}}^+$

Copertura Canonica

```
function CANONIZE( $F$ )  
   $G \leftarrow \{X \rightarrow A \mid \exists Y : X \rightarrow Y \in F \wedge A \in Y\}$   
  for all  $X \rightarrow A \in G$  such that  $|X| > 1$  do  
     $Z \leftarrow X$   
    for all  $B \in X$  do  
      if  $A \in (Z \setminus \{B\})_G^+$  then  
         $Z \leftarrow Z \setminus \{B\}$   
     $G \leftarrow (G \setminus \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$   
  for all  $X \rightarrow A \in G$  do  
    if  $A \in X_{G \setminus \{X \rightarrow A\}}^+$  then  
       $G \leftarrow G \setminus \{X \rightarrow A\}$   
  return  $G$ 
```

Copertura Canonica: Esempio

Sia $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$:

- 1 decomponiamo $A \rightarrow BC$ in $A \rightarrow B$ e $A \rightarrow C$. Dato che $A \rightarrow B$ era già presente, otteniamo $G = \{A \rightarrow C, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$
- 2 l'unica dipendenza che può contenere attributi estranei è $AB \rightarrow C$. A è estraneo, perchè $C \in B_G^+ = BC$, quindi la dipendenza diventa $B \rightarrow C$. Dato che $B \rightarrow C$ era già presente in G , otteniamo il nuovo insieme $H = \{A \rightarrow C, B \rightarrow C, A \rightarrow B\}$
- 3 è facile verificare che l'unica dipendenza ridondante è $A \rightarrow C$, dato che $C \in A_{H \setminus \{A \rightarrow C\}}^+ = ABC$. Pertanto una copertura canonica di F è l'insieme $\{B \rightarrow C, A \rightarrow B\}$

Checkpoint

Punti Chiave

- I concetti di chiave, superchiave ed attributo primo
- I principali algoritmi relativi a tali concetti e la loro complessità
- Il concetto di copertura canonica ed un algoritmo per calcolarla

Materiale Didattico

Fondamenti di Basi di Dati: Sezioni 5.2.4 e 5.2.5