Forme normali e normalizzazione

Paolo Atzeni
08/06/2005

(con materiale di M. Lenzerini)

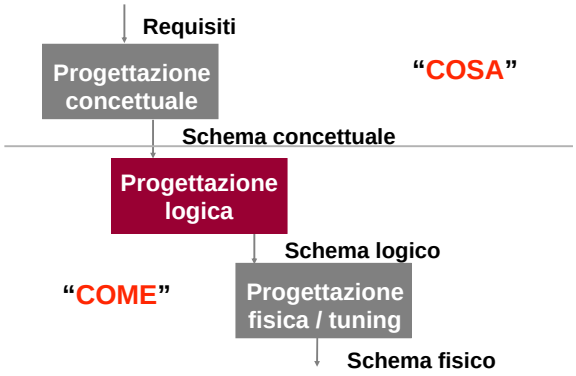
2. Complementi di metodologia di progettazione

2.1 Dipendenze funzionali, anomalie e normalizzazione
2.2 La forma normale di Boyce-Codd
2.3 La terza forma normale

2. Complementi di metodologia di progettazione

2.1 Dipendenze funzionali, anomalie e normalizzazione
2.2 La forma normale di Boyce-Codd
2.3 La terza forma normale

Fasi della progettazione



Fasi della progettazione logica

1. **Ristrutturazione dello schema ER**
2. **Traduzione diretta** dello schema ER ristrutturato nel modello relazionale
3. **Ristrutturazione dello schema relazionale**
   - richiede delle scelte da parte del progettista, tenendo conto delle prestazioni

Ristrutturazione dello schema relazionale

- **Decomposizione**
  - *Verticale* (sempre sulla chiave)
    - per facilitare l'accesso (con selezioni e proiezioni)
    - per normalizzazione (aspetto ignorato prima)
  - *Orizzontale*
    - per facilitare l'accesso (con selezioni)
  - *Mista*
    - per evitare valori nulli
- **Accorpamento**
  - per facilitare l'accesso (evita join)
  - per eliminare relazioni inutili

Forme normali di uno schema relazionale

- Una forma normale è una proprietà di una base di dati relazionale che ne garantisce (da un certo punto di vista) la "qualità"
- Quando una relazione non è normalizzata
  - presenta ridondanze
  - si presta a comportamenti poco desiderabili (anomalie) durante le operazioni di aggiornamento
- Sono di solito definite sul modello relazionale, ma possono essere applicate in altri contesti, ad esempio il modello ER

Normalizzazione

- Procedura che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale
- La normalizzazione ha avuto un ruolo importante nella teoria del progetto di basi di dati, al punto che per molto tempo, prima della diffusione della progettazione concettuale, ha rappresentato l'unico aspetto metodologico usato in pratica
- Nella nostra metodologia, forme normali e normalizzazione si utilizzano principalmente per verificare i risultati della progettazione di una base di dati, ed eventualmente migliorarli

Esempio di relazione con anomalie

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

Osservazioni

1. Lo stipendio per ciascun impiegato è unico ed è funzione del solo impiegato, indipendentemente dai progetti cui partecipa
2. Il bilancio per ciascun progetto è unico ed è funzione del solo progetto, indipendentemente dagli impiegati che vi partecipano
3. Ogni impiegato, in ciascun progetto cui partecipa, svolge una sola funzione, eventualmente diversa da progetto a progetto

Anomalie

- Lo stipendio di ciascun impiegato è ripetuto in più ennuple
  - **ridondanza**
- Se lo stipendio di un impiegato varia, è necessario andarne a modificare il valore in tutte le ennuple corrispondenti
  - **anomalia di aggiornamento**
- Se un impiegato interrompe la partecipazione a tutti i progetti, non è possibile conservare traccia del suo nome e del suo stipendio
  - **anomalia di cancellazione**
- Se si hanno informazioni su un nuovo impiegato, non è possibile inserirle finché non partecipa a un progetto
  - **anomalia di inserimento**

Perché questi fenomeni indesiderabili?

Abbiamo usato un'unica relazione per rappresentare informazioni eterogenee

- gli impiegati con i relativi stipendi
- i progetti con i relativi bilanci
- le partecipazioni degli impiegati ai progetti con le relative funzioni

Dipendenza funzionale: sintassi

Notazione: X,Y,Z denotano insiemi di attributi, mentre A,B,C denotano attributi. Inoltre, scriveremo

- A,B,C (o anche A B C) invece di {A,B,C}, e
- X Y invece di $X \cup Y$

Siano R(U) uno schema di relazione, e X,Y due sottoinsiemi non vuoti di U. La dipendenza funzionale in R da X a Y si denota con

$$X \rightarrow Y$$

Esempi:

Impiegato $\rightarrow$ Stipendio

Progetto $\rightarrow$ Bilancio

Impiegato Progetto $\rightarrow$ Funzione

Esercizio 1

Sia R(U) uno schema di relazione (ricordiamo che U denota l'insieme degli attributi di R)

Se *n* è la cardinalità di U, cioè il numero di attributi in U, quanti sono le possibili dipendenze funzionali che si possono definire in R?

Dipendenze funzionali: semantica

Una istanza r di R soddisfa la dipendenza funzionale

$X \rightarrow Y$

se, per ogni coppia di ennuple t_1 e t_2 di r con $t_1[X] = t_2[X]$, si ha che $t_1[Y] = t_2[Y]$

Osservazioni

1. per mostrare che r vìola la dipendenza funzionale $X \rightarrow Y$ occorre mostrare due ennuple t_1 e t_2 di r con $t_1[X] = t_2[X]$ e $t_1[Y] \neq t_2[Y]$
2. ad ogni chiave K di R corrisponde una dipendenza funzionale in R da K verso tutti gli attributi della relazione

Dipendenze funzionali banali

- Sulla base della definizione, in ogni relazione R è sempre soddisfatta la seguente dipendenza funzionale, comunque presi gli attributi A e B di R:

  A B $\rightarrow$ B

- Si tratta però di una dipendenza "banale" (sempre soddisfatta)
- Più precisamente
  - $X \rightarrow A$ è banale se A appartiene a X
  - $X \rightarrow Y$ è banale se tutti gli attributi di Y appartengono a X, cioè se $Y \subseteq X$

Dipendenze funzionali implicate

- Sia (R,F) uno schema di relazione con dipendenze funzionali, dove R è lo schema di relazione ed F è l'insieme delle dipendenze funzionale definite in R
- Una istanza r di R si dice legale rispetto ad F (ovvero, una istanza r di (R,F) si dice legale) se r soddisfa F, cioè se r soddisfa tutte le dipendenze funzionali in F
- In una istanza legale r di (R,F) sono soddisfatte non solo le dipendenze funzionali in F, ma anche tutte le dipendenze funzionali logicamente implicate da F

Dipendenze funzionali implicate

- Una dipendenza funzionale f si dice logicamente implicata da F se ogni istanza di R che soddisfa tutte le dipendenze funzionali in F soddisfa anche f
- Se F è un insieme di dipendenze funzionali, allora l'insieme delle dipendenze funzionali logicamente implicate da F si indica con F^+, e si chiama la chiusura di F.

Esercizio 2

Si consideri (R,F) con

- R(A,B,C,D)
- F = { A → B, B → C }

e si dimostri che A → C è soddisfatta in ogni istanza legale di (R,F), cioè si verifichi che A → C è logicamente implicata da F

Suggerimento: si consideri una qualunque istanza legale r di (R,F), e si verifichi che è impossibile che la dipendenza funzionale A → C sia violata in r

Dipendenze funzionali e chiavi

Si consideri uno schema di relazione R(U)

Un insieme di attributi X appartenenti ad U si dice superchiave di (R,F), o semplicemente superchiave di R, se la dipendenza funzionale X → U è logicamente implicata da F, cioè se X → U è in F^+. Inoltre, se nessun sottoinsieme proprio di X è superchiave di R, allora X si dice chiave di R

Attenzione: non si confonda "chiave" con "chiave primaria"

Attenzione: stiamo ragionando su come si può dedurre dalle sole dipendenze funzionali che un insieme di attributi è chiave

Attenzione: dichiarare che X è superchiave di (R,F) equivale ad asserire che la dipendenza funzionale X → U è in F

Regole di inferenza di Armstrong

Dato F, come possiamo calcolare F^+, cioè tutte le dipendenze funzionali logicamente implicate da F?

Mediante le regole di inferenza di Armstrong:

1. Riflessività: Se Y ⊆ X, allora X → Y
2. Aggiunta: Se X → Y, allora X Z → Y Z, per qualunque Z
3. Transitività: Se X → Y e Y → Z, allora X → Z

Proprietà delle regole di Armstrong

Teorema (correttezza): Le regole di inferenza di Armstrong sono corrette, cioè applicando tali regole ad un insieme F di dipendenze funzionali si ottengono solo dipendenze funzionali logicamente implicate da F.

Teorema (completezza): Le regole di inferenza di Armstrong sono complete, cioè applicando tali regole ad un insieme F di dipendenze funzionali si ottengono tutte le dipendenze funzionali logicamente implicate da F.

Teorema (minimalità): Le regole di inferenza di Armstrong sono minimali, cioè ignorando anche una sola di esse, l'insieme di regole che rimangono non è più completo.

Esercizio 3

1. Dimostrare il teorema della correttezza delle regole di inferenza di Armstrong (suggerimento: si faccia riferimento all'esercizio 1)
   Commento: dimostrare il teorema della completezza è molto più difficile!
2. Basandosi sulle regole di inferenza di Armstrong, dimostrare che esiste un algoritmo per calcolare F^+ a partire da F (suggerimento: si provi a definire un algoritmo "generativo" che applica le regole di inferenza finché può)
3. Se n è la cardinalità di F, qual è nel caso peggiore la cardinalità di F^+?

Altre regole di inferenza

Altre regole di inferenza si possono derivare dalle regole di Armstrong. In particolare

4. Unione: Se X → Y e X → Z, allora X → Y Z
5. Decomposizione: Se X → Y Z, allora X → Y e X → Z
6. Aggiunta sinistra: Se X → Y, allora X Z → Y per qualunque Z

Dimostrazione che la regola dell'unione è corretta

Dobbiamo dimostrare che, mediante le regole di Armstrong, si ottiene $X \rightarrow Y\ Z$ a partire da $X \rightarrow Y$ e $X \rightarrow Z$.

Partiamo quindi da

(a) $X \rightarrow Y$

(b) $X \rightarrow Z$

- per la regola 2 applicata a *(a)* otteniamo
  (c) $X\ Z \rightarrow Y\ Z$
- per la regola 2 applicata a *(b)* otteniamo $X\ X \rightarrow X\ Z$, che equivale a
  (d) $X \rightarrow X\ Z$
- per la regola 3 applicata a *(d)* e *(c)* otteniamo
  (e) $X \rightarrow Y\ Z$

Esercizio 4

- Dimostrare che le regole
  4. Decomposizione
  5. Aggiunta sinistra

  sono corrette.

- Dimostrare anche che dato uno schema di relazione (R,F), esiste un insieme di dipendenze G tale che:
  1. tutte le dipendenze in G hanno un singleton come membro destro
  2. la chiusura di G è uguale alla chiusura di F, cioè $G^+ = F^+$

  Si noti che questo implica che ci si può sempre restringere a dipendenze funzionali il cui membro destro è un unico attributo.

Torniamo all'esempio

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

Impiegato → Stipendio
Progetto → Bilancio
Impiegato Progetto → Funzione

Osservazione

- Alcune dipendenze funzionali causano anomalie
  - gli impiegati hanno un unico stipendio
    Impiegato → Stipendio
  - i progetti hanno un unico bilancio
    Progetto → Bilancio
- Altre no:
  - In ciascun progetto, un impiegato svolge una sola funzione
    Impiegato Progetto → Funzione
- Perché?

Anomalie

Impiegato → Stipendio
Progetto → Bilancio
Impiegato Progetto → Funzione

- **Impiegato Progetto** è chiave
- **Impiegato** non è chiave
- **Progetto** non è chiave

- Le anomalie sono causate dalla presenza contemporanea di concetti eterogenei:
  - proprietà dei soli impiegati (lo stipendio)
  - proprietà dei soli progetti (il bilancio)
  - proprietà associate alla chiave **Impiegato Progetto**

2. Complementi di metodologia di progettazione

Forma normale di Boyce e Codd

Uno schema di relazione con dipendenze funzionali (R,F) è in forma normale di Boyce e Codd (BCNF) se, per ogni dipendenza funzionale $X \rightarrow Y$ in F, si ha che:

- **$X \rightarrow Y$ è banale**

  oppure

- **X è una superchiave di R**

La forma normale BCNF richiede in sostanza che i concetti rappresentati nella relazione (R,F) siano omogenei, ovvero che (R,F) rappresenti solo proprietà direttamente associate alla chiave

Forma normale di Boyce e Codd: osservazioni

- Se (R,F) non è in BCNF allora in F c'è almeno una dipendenza funzionale $X \rightarrow Y$ che non è banale, e tale che X non è una superchiave di R. Chiamiamo una tale dipendenza funzionale un'avversaria della BCNF
- È facile verificare che un'avversaria della BCNF è una dipendenza funzionale che è responsabile di anomalie nelle istanze della relazione

Forma normale di Boyce e Codd: osservazioni

Chi mi assicura che, data una (R,F) tale che F non contiene alcuna avversaria di BCNF, non ci sia un'avversaria di BCNF in F^+? Se ci fosse, la BCNF non sventerebbe il pericolo di anomalie

Teorema (avversaria sconfitta): Data (R,F), se in F non c'è alcuna avversaria di BCNF, allora non c'è alcuna avversaria nemmeno in F^+.

Esercizio 5

Dimostrare il teorema dell'avversaria sconfitta.

Suggerimento: si ricordi che ogni dipendenza funzionale in F^+ viene generata ad un certo passo durante l'esecuzione dell'algoritmo generativo di calcolo di F^+ a partire da F.

Esercizio 5: soluzione

Si ricordi che ogni dipendenza funzionale in F^+ viene generata ad un certo passo durante l'esecuzione dell'algoritmo generativo di calcolo di F^+. A questo punto, basta dimostrare il seguente lemma.

Lemma Se nel generare F_{i+1} da F_i (con $i \geq 0$) l'algoritmo generativo introduce un'avversaria, allora F_i conteneva già un'avversaria.

Dimostrazione Supponiamo che nel generare F_{i+1} da F_i l'algoritmo introduca l'avversaria $X \rightarrow Y$. Poiché per definizione $X \rightarrow Y$ non è banale, non è stata la regola 1 a generare $X \rightarrow Y$, ed inoltre X non è superchiave. Ci sono due casi. Nel primo caso, $X \rightarrow Y$ viene introdotta mediante la regola 2 a partire da $W \rightarrow V$ (non banale, perché altrimenti $X \rightarrow Y$ sarebbe banale): questo significa che X=WZ e Y=VZ, e perciò W non è superchiave (perché altrimenti lo sarebbe X), cioè $W \rightarrow V$ è un'avversaria in F_i. Nel secondo caso, $X \rightarrow Y$ viene introdotta mediante la regola 3 a partire da $X \rightarrow W$ e $W \rightarrow Y$. Se $X \rightarrow W$ non è banale, allora, poiché X non è superchiave, $X \rightarrow W$ è un'avversaria in F_i. Se $X \rightarrow W$ è banale, allora $W \rightarrow Y$ non è banale (perché altrimenti $X \rightarrow Y$ sarebbe banale), e W, essendo un sottoinsieme di X che non è superchiave, non é una superchiave, cioé $W \rightarrow Y$ è un'avversaria in F_i.

Forma normale di Boyce e Codd: verifica

Come si fa a verificare che (R,F) è in BCNF?

Grazie al teorema dell'avversaria sconfitta, è sufficiente analizzare una ad una le dipendenze funzionali in F (e non F^+), e per ognuna verificare se non è avversaria della BCNF, cioè se è banale o se è tale da avere una superchiave come membro sinistro

Ciò implica che *occorre saper verificare se un insieme di attributi è superchiave di una relazione*

Il metodo per verificare se un insieme di attributi è superchiave di una relazione si basa sulla nozione di chiusura di un insieme di attributi rispetto ad un insieme di dipendenze funzionali

Chiusura di un insieme di attributi rispetto a F

Sia (R,F) uno schema di relazione con dipendenze funzionali e sia X un sottoinsieme non vuoto di U. La chiusura di X rispetto a F, indicata con X^+, è l'insieme di attributi

$$\{ A \mid X \rightarrow A \text{ è in } F^+ \}$$

Algoritmo per il calcolo di X^+ rispetto ad F a partire da X

```
CalcolaChiusura(X,F): insieme di attributi
{  chiusura = X;
   repeat
        if c'è una Z → Y in F tale che Z ⊆ chiusura
        then chiusura = chiusura ∪ Y
   until chiusura non è cambiato nell'ultima esecuzione
        delle istruzioni del ciclo;
   return chiusura // chiusura è uguale a X+
}
```


Esercizio 6

- Dimostrare che l'algoritmo CalcolaChiusura per il calcolo di X^+ rispetto a F a partire da X è corretto
- Calcolare la complessità di tempo dell'algoritmo CalcolaChiusura

Importanza della chiusura di un insieme di attributi

Consideriamo uno schema di relazione con dipendenze funzionali (R,F), dove U è l'insieme di tutti gli attributi di R

La chiusura di un insieme X di attributi, e quindi l'algoritmo CalcolaChiusura è fondamentale per diversi scopi:

- Si può utilizzare per verificare se una dipendenza funzionale è logicamente implicata da F. Infatti, $X \rightarrow Y$ è in F^+ se e solo se $Y \subseteq X^+$
- Si può utilizzare per verificare se un insieme di attributi è superchiave o chiave:
  - X è superchiave di (R,F) se e solo se $X \rightarrow U$ è in F^+, cioè se e solo se $U \subseteq X^+$
  - X è chiave di (R,F) se e solo se $X \rightarrow U$ è in F^+ e non esiste alcun sottoinsieme Y di X ottenuto da X eliminando un solo elemento, tale che $U \subseteq Y^+$

Algoritmo per la verifica della BCNF

Sia (R,F) uno schema di relazione con dipendenze funzionali.

Algoritmo per verificare che (R,F) è in BCNF

```
VerificaBCNF(R,F): boolean
{  repeat
        considera una dipendenza funzionale X → Y in F;
        if  (Y ⊄ X) and (U ⊄ CalcolaChiusura(X,F))
        then return false // (R,F) non è in BCNF
   until  tutte le dipendenze funzionali in F sono state
          considerate;
   return true // (R,F) è in BCNF
}
```


Esercizio 7

- Dimostrare che l'algoritmo VerificaBCNF per verificare che (R,F) è in BCNF è corretto
- Calcolare la complessità di tempo dell'algoritmo VerificaBCNF

Se una relazione non soddisfa la BCNF?

- La rimpiazziamo con altre relazioni che soddisfano la BCNF
- Come?
  - Decomponendo verticalmente la relazione sulla base delle dipendenze funzionali, al fine di separare i concetti
  - Assicurandoci che le relazioni che rimpiazzano quella originaria rappresentino tutte e sole le informazioni rilevanti rappresentate dalla relazione originaria

Schema decomposto per l'esempio di pag. 26

Impiegato	Stipendio
Rossi	20
Verdi	35
Neri	55
Mori	48
Bianchi	48

Progetto	Bilancio
Marte	2
Giove	15
Venere	15

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Verdi	Giove	progettista
Verdi	Venere	progettista
Neri	Venere	direttore
Neri	Giove	consulente
Neri	Marte	consulente
Mori	Marte	direttore
Mori	Venere	progettista
Bianchi	Venere	progettista
Bianchi	Giove	direttore

Attenzione: decomposizione con perdita nel join

Impiegato → Sede
Progetto → Sede

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

diversa dalla relazione di partenza →

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

Decomposizione senza perdita nel join

- Consideriamo una relazione con dipendenze funzionali (R,F)
- La decomposizione di R in due relazioni con attributi X e Y è una decomposizione senza perdita nel join se per ogni istanza legale r di (R,F) il join naturale di r[X] e r[Y] è uguale a r
- Si noti che r è sempre incluso nel join naturale di r[X] e r[Y], mentre non è sempre vero che il join naturale di r[X] e r[Y] sia incluso in r, come dimostrato dall'esempio precedente

Decomposizione senza perdita nel join

Teorema (decomposizione senza perdita nel join) La decomposizione di (R,F) in due relazioni R_1 e R_2 è una decomposizione senza perdita nel join se e solo se

- $R_1 \cap R_2 \rightarrow R_1$ è in F^+
  oppure
- $R_1 \cap R_2 \rightarrow R_2$ è in F^+

Ancora una volta, si rivela utile la capacità di calcolare la chiusura di un insieme di attributi, ovvero l'algoritmo CalcolaChiusura

Si noti che il teorema non esclude che, se non è verificata la condizione $R_1 \cap R_2 \rightarrow R_1$ in F^+ oppure $R_1 \cap R_2 \rightarrow R_2$ è in F^+, esista una istanza r di R tale che il join naturale di r[X] e r[Y] è uguale a r. Tuttavia, il teorema assicura che se la condizione non è verificata, esista una istanza r' di R tale che il join naturale di r'[X] e r'[Y] non è uguale a r'.

Un altro problema

Consideriamo questa nuova decomposizione per lo schema di pag. 41 (che è senza perdita nel join):

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

e supponiamo di voler inserire una nuova tupla che specifica la partecipazione dell'impiegato Neri al progetto Marte

- l'operazione sembra corretta se si guarda solo alle relazioni decomposte
- ma se riformuliamo l'operazione sulla relazione originaria, ci accorgiamo che l'operazione viola la dipendenza funzionale da Progetto a Sede (si aggiunga la tupla, si calcoli il join delle due relazioni, e si verifichi che il join non soddisfa più la dipendenza funzionale)

Decomposizione senza perdita di dipendenze

- Sia (R,F) uno schema di relazione con dipendenze funzionali, e sia X un sottoinsieme di attributi di R
- La **proiezione di F su X**, denotata da F_X, è l'insieme di dipendenze funzionali $Z \rightarrow Y$ in F^+ che coinvolgono solo attributi in X, cioè tali che $Z \subseteq X$ e $Y \subseteq X$
- La decomposizione di (R,F) in due relazioni con attributi X e Y è una **decomposizione senza perdita di dipendenze** se $(F_X \cup F_Y)$ è equivalente a F, cioè se $(F_X \cup F_Y)^+ = F^+$
- Nell'esempio precedente, Progetto → Sede non è conservata

La verifica di decomposizione senza perdita di dipendenze

La definizione di decomposizione senza perdita di dipendenze è basata sulla verifica se $(F_X \cup F_Y)$ è equivalente a F.

Per applicare la definizione,

- è necessario sapere calcolare se un insieme di dipendenze funzionali è equivalente ad un altro
- è necessario saper calcolare la proiezione di un insieme di dipendenze funzionali su un insieme di attributi

Verifica di equivalenza

Per verificare se un insieme di dipendenze funzionali F è equivalente ad un altro G, un modo ovvio è quello di calcolare F^+ e G^+, e poi verificarne l'uguaglianza. Poiché questo metodo è molto dispendioso, è importante il seguente teorema, che ci fornisce direttamente un algoritmo polinomiale per la verifica di equivalenza

Teorema (verifica di equivalenza) Un insieme di dipendenze funzionali F è equivalente ad un altro G (ovvero $F^+=G^+$), se e solo se

1. per ogni $X \rightarrow Y$ in F, si ha che $X \rightarrow Y$ è in G^+ , cioè si ha che Y è in CalcolaChiusura(X,G)
2. per ogni $X \rightarrow Y$ in G, si ha che $X \rightarrow Y$ è in F^+ , cioè si ha che Y è in CalcolaChiusura(X,F)

Calcolo della proiezione di F su X

- Abbiamo definito F_X, cioè la proiezione di F su X, come l'insieme di dipendenze funzionali $Z \rightarrow Y$ in F^+ che coinvolgono solo attributi in X, cioè tali che $Z \subseteq X$ e $Y \subseteq X$
- Ovviamente, se vogliamo utilizzare F_X, non è necessario calcolare esattamente F_X, ma è sufficiente calcolare un qualunque insieme di dipendenze funzionali equivalenti ad F_X
- Infatti, vista la definizione di equivalenza, se G è equivalente ad H, utilizzare G oppure H non fa alcuna differenza, perché, essendo G^+ uguale ad H^+, entrambi "rappresentano" lo stesso insieme di dipendenze funzionali
- Nel seguito, quando ci riferiremo a F_X intenderemo un qualunque insieme di dipendenze funzionali equivalente alla proiezione di F su X

Calcolo della proiezione di F su X

Per calcolare F_X, cioè la proiezione di F su X, possiamo procedere per enumerazione (purtroppo non si può fare meglio), evitando però di generare dipendenze funzionali "inutili"

Algoritmo per calcolare la proiezione di F su X
CalcolaProiezione(F,X): insieme di dipendenze

```
{  result = ∅;
   per ogni sottoinsieme proprio S di X
        per ogni attributo A in X tale che A non è in S, e tale che
                non esiste alcun sottoinsieme S' di S tale che
                S' → A è in result
           if A è in CalcolaChiusura(S,F)
           allora result = result ∪ { S → A };
   return result   // result è F_X
}
```


Dimensione della proiezione di F su X

Ci sono casi un cui la proiezione di F su X ha dimensione esponenziale rispetto alla dimensione di F e X, come mostrato dal seguente esempio

Consideriamo $R(A_1,A_2,\ldots,A_n,B_1,B_2,\ldots,B_n,C_1,C_2,\ldots,C_n,D)$ e $F = \{ A_i \rightarrow C_i, B_i \rightarrow C_i \mid 1 \leq i \leq n \} \cup \{ C_1C_2\ldots C_n \rightarrow D \}$

La proiezione di F su $\{ A_1,A_2,\ldots,A_n,B_1,B_2,\ldots,B_n,D \}$ è $P = \{ X_1X_2\ldots X_n \rightarrow D \mid X_i = A_i$ oppure $X_i = B_i$ per $1 \leq i \leq n \}$, la cui dimensione è ovviamente esponenziale rispetto alla dimensione dello schema R e delle dipendenze funzionali F. Si noti che si può dimostrare che nessun insieme equivalente a P ha cardinalità minore.

Esempio

La relazione (R,F), con

- R(A,B,C), e
- $F = \{ A \rightarrow B, B \rightarrow C, A \rightarrow C \}$

non è in BCNF. Se la decomponiamo in $R_1(A,B)$ con $A \rightarrow B$ e $R_2(B,C)$ con $B \rightarrow C$, otteniamo due relazioni in BCNF, con la proprietà che la decomposizione è senza perdita nel join. La decomposizione è anche senza perdita di dipendenze, perchè la dipendenza funzionale $A \rightarrow C$ è logicamente implicata dalle due dipendenze funzionali che valgono in R_1 e R_2.

Nota: se la definizione di conservazione delle dipendenze non considerasse $(F_X \cup F_Y)^+$ ma solo $(F_X \cup F_Y)$, allora la decomposizione sembrerebbe perdere la dipendenza funzionale $A \rightarrow C$, che non è esprimibile direttamente né in R_1 (cioè mediante F_{AB}) nè in R_2 (cioè mediante F_{BC}).

Qualità delle decomposizioni

Idealmente, una decomposizione dovrebbe soddisfare le due proprietà che abbiamo discusso, e cioè:

1. la proprietà della **decomposizione senza perdita nel join**, che garantisce la ricostruibilità dei dati rappresentabili nella relazione originaria
2. la **conservazione delle dipendenze**, che garantisce la possibilità di verificare in modo efficiente (senza ricostruire il join) che i vincoli di integrità originari siano soddisfatti

La prima proprietà è più importante della seconda. Infatti, anche se la seconda non è soddisfatta, le dipendenze funzionali potrebbero comunque essere verificate attraverso l'applicazione del join (accettando, però, il costo del join).

Algoritmo per la decomposizione in BCNF

Assumiamo (senza perdita di generalità) che ogni volta che chiamiamo l'algoritmo descritto sotto con parametri R e F, ovvero uno schema di relazione con dipendenze funzionali, ogni dipendenza funzionale in F abbia un unico attributo come membro destro, e che U sia l'insieme di tutti gli attributi di R

Decomponi(R,F)
{ if esiste un'avversaria $X \rightarrow A$ di BCNF in (R,F)
  then { sostituisci R con una relazione R_1 con attributi U-A, ed una relazione R_2 con attributi $X \cup A$;
  Decomponi(R_1,F_{U-A});
  Decomponi(R_2,$F_{X \cup A}$)
  }
}

Proprietà dell'algoritmo

- **Teorema** (correttezza dell'algoritmo della decomposizione) Qualunque sia l'input, l'esecuzione dell'algoritmo su tale input termina, e produce una decomposizione della relazione originaria tale che:
  - ogni relazione ottenuta è in BCNF
  - la decomposizione è senza perdita nel join
- Nota: A seconda dell'ordine con cui si considerano le dipendenze funzionali, il risultato della decomposizione può cambiare
- Nota: non è assicurato che la decomposizione ottenuta sia senza perdita di dipendenze

Esercizio 8

Si consideri

- R(C,S,J,D,P,Q,V)
- F = { $C \rightarrow$ SJDPQV, JP $\rightarrow$ C, SD $\rightarrow$ P, J $\rightarrow$ S }

e si dimostri che

1. (R,F) non è in BCNF
2. a seconda che si esegua l'algoritmo Decomponi partendo da un'avversaria, oppure da un'altra avversaria, si ottengono decomposizioni diverse

Una relazione non in BCNF

Consideriamo la seguente relazione DPS:

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

con dipendenze funzionali:

Progetto Sede $\rightarrow$ Dirigente
Dirigente $\rightarrow$ Sede

La decomposizione di DPS è problematica

- Progetto Sede $\rightarrow$ Dirigente coinvolge tutti gli attributi quindi nessuna decomposizione che prevede proiezioni della relazione originaria può preservare tale dipendenza
- Si può concludere che ci sono casi in cui la BCNF "non è raggiungibile" senza perdere qualche dipendenza (o meglio, senza essere costretto a trattare la verifica di qualche dipendenza calcolando il join delle relazioni ottenute dalla decomposizione)
- Questo ha motivato l'interesse verso un'altra forma normale, che adesso introduciamo

2. Complementi di metodologia di progettazione

2.1 Dipendenze funzionali, anomalie e normalizzazione
2.2 La forma normale di Boyce-Codd
2.3 La terza forma normale

Una nuova forma normale

Uno schema di relazione con dipendenze funzionali (R,F) è in **terza forma normale (3NF)** se, per ogni dipendenza funzionale $X \rightarrow Y$ in F, è verificata almeno una delle seguenti condizioni:

- $X \rightarrow Y$ è banale
- X è una superchiave di R
- ogni attributo in Y è contenuto in almeno una chiave di R

BCNF e terza forma normale

- la terza forma normale è meno restrittiva della forma normale di Boyce e Codd (infatti, ogni relazione che è in BCNF è anche in 3NF)
- il problema di verificare se una relazione è in 3NF è NP-completo (perciò il miglior algoritmo deterministico conosciuto ha complessità esponenziale nel caso peggiore)
- ha il vantaggio però di essere sempre "raggiungibile"

Perché verificare la 3NF è NP-completo

- Dato un (R,F), e dato un attributo A
  1. si genera non deterministicamente un sottoinsieme S degli attributi di R che contiene A,
  2. si controlla se S è una chiave (non una superchiave)
- Il controllo di cui al punto 2 può essere eseguito in tempo polinomiale rispetto a R e F (si veda la slide a pag. 37), e quindi l'algoritmo ha complessità NP
- Si può poi dimostrare che il problema è NP-hard

Cenno alla seconda forma normale

Supponiamo che $X \rightarrow A$ sia un'avversaria della 3NF. Ci sono due casi

1. X è un sottoinsieme proprio di una chiave, e allora $X \rightarrow A$ viene chiamata una dipendenza parziale
2. X non è un sottoinsieme proprio di alcuna chiave, e allora abbiamo una catena $K \rightarrow X \rightarrow A$, dove K è una chiave, e in questo caso $X \rightarrow A$ viene chiamata una dipendenza transitiva

Uno schema di relazione con dipendenze funzionali è in seconda forma normale se non contiene dipendenze funzionali parziali (mentre può contenere dipendenze funzionali transitive)

Una relazione non in BCNF ma in 3NF

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto Sede $\rightarrow$ Dirigente
Dirigente $\rightarrow$ Sede

Decomposizione in terza forma normale

- L'algoritmo Decomponi per la decomposizione in BCNF fornisce anche una decomposizione in 3NF senza perdita nel join
- Come si fa ad ottenere una decomposizione in 3NF che preserva le dipendenze?
  - si introduce il concetto di copertura minimale di un insieme di dipendenze funzionali
  - si usa questo concetto per aggiungere un passo all'algoritmo Decomponi

Copertura minimale di un insieme di dipendenze

Un insieme G di dipendenze funzionali si dice copertura minimale di un insieme di dipendenze funzionali F se

1. Ogni dipendenza funzionale in G ha un singleton come membro destro
2. La chiusura di G è uguale alla chiusura di F
3. Se H è l'insieme ottenuto togliendo una qualunque dipendenza funzionale da G, allora la chiusura di H è diversa dalla chiusura di F
4. Se L è l'insieme ottenuto togliendo un qualunque attributo dal membro sinistro di una qualunque dipendenza funzionale di G, allora la chiusura di L è diversa dalla chiusura di F

Nota: la definizione fornisce direttamente un metodo per calcolare la copertura minimale di un insieme di dipendenze funzionali

Nota: un insieme di dipendenze funzionali si dice minimale se esso è una copertura minimale di se stesso.

Esercizio 9

Considerare il seguente insieme di dipendenze funzionali

$F = \{ A \rightarrow B, ABCD \rightarrow E, EF \rightarrow GH, ACDF \rightarrow EG \}$

e dimostrare che

$\{ A \rightarrow B, ACD \rightarrow E, EF \rightarrow G, EF \rightarrow H \}$

è una copertura minimale di F.

Algoritmo per la decomposizione in 3NF

Sia (R,F) uno schema di relazione con dipendenze funzionali, e sia F minimale, e tale che al membro destro di ogni dipendenza ci sia un attributo.

1. Eseguiamo l'algoritmo Decomponi su (R,F), ed otteniamo gli schemi $R_1,R_2,\ldots,R_n$ (indichiamo con F_i la proiezione di F sugli attributi di R_i) tali che
   - $R_1,R_2,\ldots,R_n$ sono in BCFN e quindi anche in 3NF
   - la decomposizione $R_1,R_2,\ldots,R_n$ è senza perdita nel join
2. Identifichiamo l'insieme N di dipendenze funzionali in F che non sono preservate in $R_1,R_2,\ldots,R_n$, cioè che non sono incluse nella chiusura dell'unione dei vari F_i
3. Per ogni dipendenza funzionale $X \rightarrow A$ in N, aggiungiamo lo schema relazionale X A alla decomposizione, con la corrispondente dipendenza funzionale

Teorema (decomposizione per 3NF) La decomposizione ottenuta dall'algoritmo è senza perdita di join, senza perdita di dipendenze, e produce uno schema in cui tutte le relazioni sono in 3NF.

La normalizzazione nella nostra metodologia

Ricordiamo che nella nostra metodologia la normalizzazione è utilizzata nella fase di ristrutturazione logica, come metodo di verifica dello schema.

In questa fase si analizzano le dipendenze funzionali di ogni relazione, e per ogni relazione che non è in BCNF:

1. si esegue l'algoritmo Decomponi sulla relazione
2. se la decomposizione ottenuta è con perdita di dipendenze, allora si ottiene una decomposizione in 3NF senza perdita di dipendenze, eseguendo gli ulteriori passi illustrati prima

Riprendiamo questo schema

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Dirigente → Sede
Progetto Sede → Dirigente

Questo schema non è decomponibile in BCNF senza perdere dipendenze funzionali. Alcuni sostengono che la relazione è progettata male dal punto di vista concettuale

Una possibile riorganizzazione

Dirigente	Progetto	Sede	Reparto
Rossi	Marte	Roma	1
Verdi	Giove	Milano	1
Verdi	Marte	Milano	1
Neri	Saturno	Milano	2
Neri	Venere	Milano	2

Dirigente → Sede Reparto
Sede Reparto → Dirigente
Progetto Sede → Reparto

Decomposizione in BCNF

Dirigente	Sede	Reparto
Rossi	Roma	1
Verdi	Milano	1
Neri	Milano	2

Progetto	Sede	Reparto
Marte	Roma	1
Giove	Milano	1
Marte	Milano	1
Saturno	Milano	2
Venere	Milano	2

Questa è in effetti una decomposizione senza perdita di dipendenze.

Tuttavia, la riorganizzazione descritta, e la decomposizione qui riportata soffrono di un problema: è stato introdotto il concetto di Reparto, che è un concetto artificiale, non rilevante nella applicazione.

Esercizio 10

Si consideri la seguente relazione

LIBRO(titolo, autore, tipo, prezzo, affiliazioneautore, editore)

con le dipendenze funzionali

{ titolo → editore tipo,
  tipo → prezzo,
  autore → affiliazioneautore }

Dire se la relazione LIBRO è in BCNF, motivando la risposta. Se necessario, effettuare una opportuna decomposizione della relazione.

Esercizio 11

Dare la definizione di chiave, superchiave, dipendenza funzionale, e relazione in BCNF.

Considerare poi lo schema di relazione R(A,B,C,D,E) con le dipendenze funzionali

{ AB → C, B → D, A → E }

e dire se è in BCNF, motivando la risposta. In caso negativo, fornire una adeguata ristrutturazione dello schema in un nuovo schema in cui tutte le relazioni sono in BCNF.

Esercizio 12

Considerare lo schema di relazione R(A,B,C,D) con le dipendenze funzionali

{ A → C, C → B, B → A, C → D }

e dire se è in BCNF, motivando la risposta. In caso negativo, fornire una adeguata ristrutturazione dello schema in un nuovo schema in cui tutte le relazioni sono in BCNF.