

Capitolo 5

NORMALIZZAZIONE DI SCHEMI RELAZIONALI

Il modello dei dati relazionale adotta come unica struttura per la rappresentazione dei dati la relazione, prodotto cartesiano di domini elementari. Questa struttura, se da una parte attira per la sua estrema semplicità, dall'altra pone problemi nell'uso in caso di situazioni complesse, in cui è necessario ricorrere a degli artifici per modellare, ad esempio, attributi composti o multivalore, associazioni molti a molti, classi incluse in altre.

In questi casi esistono in genere diverse rappresentazioni possibili della stessa situazione, per cui sorge il problema di verificare se: (a) queste diverse rappresentazioni sono tra di loro equivalenti; (b) queste rappresentazioni sono di buona qualità. In particolare, la qualità di una rappresentazione viene qui valutata come l'assenza di determinate *anomalie* che sono definite nel capitolo. La teoria della normalizzazione si occupa per l'appunto di definire criteri formali per giudicare l'equivalenza di schemi e la qualità di tali schemi, e di definire algoritmi per trasformare uno schema in un altro equivalente ma privo di anomalie.

Come discuteremo al termine del capitolo, i risultati della teoria sono di particolare interesse per un progettista che deve partire da uno schema relazionale esistente e migliorarlo. Al progettista che parte da uno schema ad oggetti privo di anomalie, la teoria sviluppata in questo capitolo dimostra che lo schema relazionale ottenuto applicando le regole del Capitolo 4 sarà anch'esso privo di anomalie.

5.1 Le anomalie

Per mostrare i problemi che si presentano nella definizione di schemi relazionali, si supponga di rappresentare con un'unica relazione i dati relativi a prestiti di una biblioteca:

Biblioteca(NomeUtente, Residenza, Telefono, NumeroLibro, Autore, Titolo, Data)

I libri, di cui esiste solo una copia, sono identificati da NumeroLibro e possono essere presi in prestito da utenti dei quali interessano il NomeUtente, che li identifica, la Residenza e il Telefono. Un utente può avere più libri in prestito contemporaneamente e di ogni prestito interessa memorizzare la data. La chiave della relazione è NumeroLibro. Un esempio di istanza della relazione è:

Biblioteca

NomeUtente	Residenza	Telefono	NumerLibro	Autore
Rossi Carlo	Carrara	75444	XY188A	Boccaccio
Paolicchi Luca	Avenza	59729	XY256B	Verga
Pastine Maurizio	Dogana	66133	XY090C	Petrarca
Paolicchi Laura	Avenza	59729	XY101A	Dante
Paolicchi Luca	Avenza	59729	XY701B	Manzoni
Paolicchi Luca	Avenza	59729	XY008C	Moravia

Titolo	Data
Decameron	07-07
Novelle	07-08
Canzoniere	01-08
Vita Nova	05-08
Adelchi	14-01
La noia	17-08

Lo schema di relazione precedente presenta i seguenti inconvenienti o anomalie:

Ripetizione dell'informazione

Ogni volta che un utente prende in prestito un nuovo libro, vengono ripetuti la sua residenza e il suo numero telefonico; oltre allo spreco di spazio si complica l'aggiornamento della relazione, quando un utente cambia residenza o telefono.

Impossibilità di rappresentare certi fatti

Informazioni relative agli utenti della biblioteca possono essere memorizzate solo quando questi hanno un libro in prestito, non potendo lasciare la chiave NumerLibro non specificata.

Anche la scelta di utilizzare più relazioni potrebbe portare a degli inconvenienti, come nel caso seguente, dove si è “decomposto” la relazione Biblioteca in due relazioni:

Utenti(NomeUtente, Residenza, Telefono)
 Prestiti(NumerLibro, Autore, Titolo, Data, Telefono)

In questo caso l'associazione fra utenti e prestiti è modellata attraverso il numero di telefono. I dati delle relazioni Utenti e Prestiti si ottengono per proiezione dalla relazione Biblioteca sui rispettivi attributi:

Utenti = $\pi_{\text{NomeUtente}, \text{Residenza}, \text{Telefono}}(\text{Biblioteca})$

Utenti	NomeUtente	Residenza	Telefono
Rossi Carlo	Carrara	75444	
Paolicchi Luca	Avenza	59729	
Pastine Maurizio	Dogana	66133	
Paolicchi Laura	Avenza	59729	

Prestiti = $\pi_{\text{NumerLibro}, \text{Autore}, \text{Titolo}, \text{Data}, \text{Telefono}}(\text{Biblioteca})$

Prestiti	Numerolibro	Autore	Titolo	Data	Telefono
XY188A	Boccaccio	Decameron	07-07	75444	
XY256B	Verga	Novelle	07-07	59729	
XY090C	Petrarca	Canzoniere	01-08	66133	
XY101A	Dante	Vita Nova	05-08	59729	
XY701B	Manzoni	Adelchi	14-01	59729	
XY008C	Moravia	La Noia	17-08	59729	

Questa decomposizione permette di eliminare la duplicazione dei dati, ma presenta nuovi problemi. Supponiamo, ad esempio, di cercare tutti gli utenti che hanno prestiti da gennaio:

$$\pi_{\text{NomeUtente}, \text{Residenza}}(\text{Utenti} \bowtie \sigma_{\text{Data} \in [01-01, 31-01]}(\text{Prestiti}))$$

Il risultato dell'operazione è la relazione seguente:

NomeUtente	Residenza
Paolicchi Luca	Avenza
Paolicchi Laura	Avenza

Questa relazione è errata, perché Paolicchi Laura non ha preso in prestito un libro in gennaio. Infatti, la giunzione fra Utenti e Prestiti contiene più enneple della relazione originale Biblioteca. Una decomposizione che presenta questa anomalia è detta *con perdita di informazione* (*lossy decomposition*). Il motivo di questa perdita è la scelta di rappresentare l'associazione fra Utenti e Prestiti usando come chiave esterna il numero di telefono, che non identifica univocamente gli utenti. Una decomposizione senza perdita di informazione è invece la seguente:

$$\begin{aligned} &\text{Utenti}(\text{NomeUtente}, \text{Residenza}, \text{Telefono}) \\ &\text{Prestiti}(\text{Numerolibro}, \text{Autore}, \text{Titolo}, \text{Data}, \text{NomeUtente}) \end{aligned}$$

È lecito a questo punto chiedersi se questa decomposizione non introduce nuovi problemi o svantaggi. Ad esempio, volendo reperire la residenza dell'utente che ha in prestito un certo libro occorre fare una giunzione delle due relazioni, non necessaria nel caso della relazione unica. Per questi motivi, il problema di valutare se uno schema è "migliore" di un altro non è banale, in particolare se si vuol tener conto anche del costo delle operazioni.

Lo scopo principale della teoria della normalizzazione è quello di fornire strumenti formali per la progettazione di basi di dati che non presentino anomalie del tipo precedentemente mostrato, senza prendere in considerazione il costo delle operazioni. In particolare, nell'attività di progettazione, si parte da un qualche schema che modella la realtà di interesse e si cerca di ottenere uno schema che, in base a certi criteri, sia "migliore" di quello di partenza, ma "equivalente" ad esso, nel senso che contenga la stessa informazione. Per questo motivo si parla anche di *analisi di schemi* anziché di *progettazione di schemi*. Quindi, in sostanza, la teoria della normalizzazione si occupa dei seguenti problemi:

- definire quando due schemi sono equivalenti;
- definire criteri di bontà per schemi (cosa vuol dire che uno schema è migliore di un altro);
- trovare metodi algoritmici per ottenere da uno schema uno schema migliore ed equivalente.

Il primo problema è quello che in letteratura va sotto il nome di *problema della rappresentazione*, ossia quando e in che misura si può dire che uno schema *rappresenta* un altro. La formalizzazione del secondo problema invece ha portato, come vedremo, alla definizione di *forme normali* per schemi di relazioni, con caratteristiche desiderabili dal punto di vista della minimizzazione della ridondanza ed eliminazione delle anomalie. Per questo motivo l'attività di progettazione è anche detta *normalizzazione*, cioè riduzione a schemi in forma normale.

Prima di procedere nella descrizione della teoria della normalizzazione, è necessario discutere un assunto che ne è alla base: l'ipotesi che le informazioni da memorizzare in una base di dati siano descrivibili da uno *schema di relazione universale*.

Definizione 5.1 Lo *schema di relazione universale* U di una base di dati relazionale ha come attributi l'unione degli attributi di tutte le relazioni della base di dati.

Questa ipotesi comporta che tutti gli attributi con lo stesso nome in relazioni diverse abbiano lo *stesso* significato e quindi, in particolare, siano definiti sullo *stesso* dominio. Ad esempio, se vogliamo riferirci alle date di nascita e alle date di assunzione degli impiegati di una ditta, non possiamo usare due attributi con nome Data, neanche se sono in relazioni diverse, ma dovremo usare due nomi diversi (ad esempio, DataNascita e DataAssunzione). Questa ipotesi semplifica la trattazione della teoria perché permette di evitare operazioni di ridenominazione degli attributi, e permette di utilizzare sempre la giunzione naturale per collegare le ennuple correlate in due relazioni.

Nel seguito, verranno usate le seguenti notazioni:

- A, B, C, A_1, A_2 ecc. indicano singoli attributi.
- T, X, Y, X_1 ecc. indicano insiemi di attributi.
- XY è un'abbreviazione per $X \cup Y$, AB è un'abbreviazione per $\{A, B\}$, $A_1A_2 \dots A_n$ è un'abbreviazione per $\{A_1, A_2, \dots, A_n\}$, e XA è un'abbreviazione per $X \cup \{A\}$.
- $R(T)$ è uno schema di relazione, r una sua generica istanza e t è un'ennupla di r . Se $X \subseteq T$, allora $t[X]$ indica l'ennupla ottenuta da t considerando solo gli attributi in X .

5.2 Dipendenze funzionali

5.2.1 Definizione

Come mostrato nella sezione precedente, le anomalie che si incontrano in certi schemi relazionali provengono da una rappresentazione impropria dei fatti dell'universo del discorso. Per formalizzare la nozione di schema senza anomalie, occorre quindi, per prima cosa, introdurre nella teoria un modo per rappresentare formalmente informazioni sulle proprietà dei fatti che si modellano. Codd, che ha affrontato per primo questo problema, ha proposto a questo scopo un formalismo basato sulla nozione di *dipendenza fra dati* [Cod70]. Il primo tipo di dipendenza che si considera, chiamato *dipendenza funzionale*, formalizza la nozione di valore di un attributo determinato funzionalmente dal valore di altri.

Definizione 5.2 Una *dipendenza funzionale* fra un insieme di attributi X e un insieme di attributi Y di uno schema di relazione $R(T)$, con $XY \subseteq T$, è un vincolo d'integrità sulle istanze della relazione, espresso nella forma $X \rightarrow Y$, con il seguente significato: un'istanza r di $R(T)$ soddisfa la dipendenza $X \rightarrow Y$, o $X \rightarrow Y$ vale in

r , se per ogni coppia di ennuple t_1 e t_2 di r , se $t_1[X] = t_2[X]$ allora $t_1[Y] = t_2[Y]$, ovvero

$$X \rightarrow Y \Leftrightarrow \forall t_1, t_2 \in r, t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]^1 \quad (5.1)$$

In altre parole, una dipendenza funzionale $X \rightarrow Y$ è un vincolo di integrità che specifica che, per ogni istanza valida r di $R(T)$, $\pi_{XY}(r)$ è una *funzione* con dominio X e codominio Y .

Quando su $R(T)$ è definita la dipendenza funzionale $X \rightarrow Y$, si dice che X *determina* Y , o che Y è *determinato* da X . Se F è un insieme di dipendenze funzionali su $R(T)$, si dice che r è un'istanza *valida* di $R(T)$, rispetto ad F , se per ogni $X \rightarrow Y \in F$ vale la 5.1.

È importante notare due aspetti significativi relativi alle dipendenze funzionali:

- esse sono definite solo all'interno di *uno* schema di relazione, e non possono esistere, quindi, fra attributi appartenenti a relazioni diverse;
- sono proprietà intensionali, legate al significato dei fatti che si rappresentano; non è quindi possibile inferirle dall'osservazione di alcune istanze della relazione.

Esempio 5.1 Nello schema:

Biblioteca(NomeUtente, Residenza, Telefono, NumeroLibro, Autore, Titolo, Data)

valgono le seguenti dipendenze funzionali:

NomeUtente → Residenza, Telefono
NumeroLibro → Autore, Titolo
NumeroLibro → NomeUtente, Data

che sono rispettate nell'esempio d'istanza visto nella sezione precedente.

In conclusione, per specificare il significato dei fatti rappresentati si usano le dipendenze funzionali, che vengono utilizzate per verificare l'eventuale presenza di anomalie nel progetto e, se questo è il caso, per normalizzare lo schema. Data la loro importanza per la progettazione di uno schema relazionale, le dipendenze funzionali si considerano facenti parte dello schema di una relazione, che d'ora in poi si indicherà, in generale, con $R\langle T, F \rangle$, con F insieme di dipendenze funzionali su T .

5.2.2 Dipendenze derivate

Dato uno schema di relazione $R\langle T, F \rangle$, è facile convincersi che le sue istanze valide soddisfano non solo le dipendenze espresse in F , ma anche altre derivabili da esse. Ad esempio, dato $R\langle T, \{X \rightarrow Y, X \rightarrow Z\} \rangle$, con $X, Y, Z \subseteq T$, e $W \subseteq X$, si può notare che in tutte le sue istanze valide valgono anche le dipendenze $X \rightarrow W$ e $X \rightarrow YZ$. Nel primo caso, infatti, se due ennuple coincidono su X coincideranno a maggior ragione su W che è un sottoinsieme di X .² Nel secondo caso se $e_1[X] = e_2[X]$, poiché e_1, e_2 soddisfano le dipendenze in F , si avrà che $e_1[Y] = e_2[Y]$ e $e_1[Z] = e_2[Z]$, e quindi $e_1[YZ] = e_2[YZ]$. L'implicazione di un insieme di dipendenze da altre viene definita nel modo seguente:

Definizione 5.3 Dato $R\langle T \rangle$ e dato F , diciamo che $F \models X \rightarrow Y$ (F implica logicamente $X \rightarrow Y$), se ogni istanza r di $R\langle T \rangle$ che soddisfa F soddisfa anche $X \rightarrow Y$.

1. Notare che gli insiemi X ed Y possono avere elementi in comune.
2. Le dipendenze $X \rightarrow W$, con $W \subseteq X$, sono dette *dipendenze banali*.

In base a questa definizione, per lo schema precedente valgono le seguenti implicazioni logiche:

$$\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$$

e

$$\{\} \models X \rightarrow X$$

La definizione precedente non fornisce un modo algoritmico per trovare le dipendenze funzionali implicate da un insieme F : per questo occorre un'assiomatizzazione delle dipendenze funzionali che fornisca un insieme *corretto* e *completo* di regole di inferenza, chiamate anche *assiomi*, che possono essere usate per derivare nuove dipendenze da un dato insieme di dipendenze.

Definizione 5.4 Sia RI un insieme di regole di inferenza per F . Indichiamo con $F \vdash X \rightarrow Y$ il fatto che $X \rightarrow Y$ sia derivabile da F usando RI .

L'insieme RI è *corretto* se: $F \vdash X \rightarrow Y \Rightarrow F \models X \rightarrow Y$.

L'insieme RI è *completo* se: $F \models X \rightarrow Y \Rightarrow F \vdash X \rightarrow Y$.

Il più noto insieme corretto e completo di regole di inferenza per le dipendenze funzionali è il seguente (*assiomi di Armstrong*) [Arm74]:

Riflessività: se $Y \subseteq X$, allora $X \rightarrow Y$.

Arricchimento: se $X \rightarrow Y$ e $W \subseteq T$, allora $XW \rightarrow YW$.

Transitività: se $X \rightarrow Y$ e $Y \rightarrow Z$, allora $X \rightarrow Z$.

Possiamo adesso definire formalmente il concetto di *derivazione* di una dipendenza.

Definizione 5.5 Una *derivazione* di f da F è una sequenza finita f_1, \dots, f_m di dipendenze, dove $f_m = f$ e ogni f_i è un elemento di F oppure è ottenuta dalle precedenti dipendenze f_1, \dots, f_{i-1} della derivazione usando una regola di inferenza.

Si noti che una sottosequenza f_1, \dots, f_k di una derivazione f_1, \dots, f_m è anche una derivazione, quindi $F \vdash f_k$ per ogni $k = 1, \dots, m$.

Alcune regole comunemente usate, derivabili a partire dagli assiomi di Armstrong, sono:

Unione: $\{X \rightarrow Y, X \rightarrow Z\} \vdash X \rightarrow YZ$.

Decomposizione: $\{X \rightarrow YZ\} \vdash X \rightarrow Y$.

Indebolimento: $\{X \rightarrow Y\} \vdash XZ \rightarrow Y$.

Identità: $\{\} \vdash X \rightarrow X$.

Dimostriamo la prima regola, lasciando al lettore la dimostrazione delle successive.

Teorema 5.1 $\{X \rightarrow Y, X \rightarrow Z\} \vdash X \rightarrow YZ$

Dimostrazione

1. $X \rightarrow Y$ (per ipotesi)
2. $X \rightarrow XY$ (per arricchimento da 1)
3. $X \rightarrow Z$ (per ipotesi)
4. $XY \rightarrow YZ$ (per arricchimento da 3)
5. $X \rightarrow YZ$ (per transitività da 2, 4)

□

Introduciamo ora la nozione di *chiusura* di un insieme di attributi rispetto ad un insieme di dipendenze funzionali.

Definizione 5.6 Dato uno schema $R\langle T, F \rangle$ con $X \subseteq T$, la *chiusura* di X rispetto a F è data da $\{A \in T \mid F \vdash X \rightarrow A\}$, e si indica con X_F^+ , o più semplicemente con X^+ se non vi sono ambiguità.

Teorema 5.2 $F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$

Dimostrazione (\Rightarrow) Sia $Y = A_1 \dots A_k$, $A_i \in T$. Per la regola di decomposizione si ha che

$$F \vdash X \rightarrow A_i, 1 \leq i \leq k$$

quindi, in base alla definizione di X^+ ,

$$A_i \in X^+, 1 \leq i \leq k$$

e quindi anche $Y \subseteq X^+$.

(\Leftarrow) Secondo la definizione di X^+ ,

$$F \vdash X \rightarrow A_i, 1 \leq i \leq k$$

e dalla regola di unione

$$F \vdash X \rightarrow A_1, \dots, A_k \text{ cioè } F \vdash X \rightarrow Y \quad \square$$

Il prossimo teorema dimostra l'equivalenza della nozione di implicazione logica (\models) e di quella di derivazione (\vdash) nel caso degli assiomi di Armstrong. Questo teorema stabilisce, in particolare, che se una dipendenza è derivabile con gli assiomi di Armstrong allora è anche implicata logicamente (correttezza degli assiomi), e viceversa se una dipendenza è implicata logicamente allora è anche derivabile dagli assiomi (completezza degli assiomi).

Teorema 5.3 Gli assiomi di Armstrong sono corretti e completi.

Dimostrazione

Supponiamo di avere un insieme di dipendenze funzionali F su T , e una dipendenza $X \rightarrow Y$.

Correttezza. Si deve dimostrare che se $F \vdash X \rightarrow Y$ allora $F \models X \rightarrow Y$. Si procede per induzione sulla lunghezza della derivazione. Sia f_1, \dots, f_m la derivazione di $X \rightarrow Y$ da F , e supponiamo che il teorema valga per tutte le derivazioni più corte. La dipendenza $f_m = X \rightarrow Y$ è un elemento di F , oppure è stata derivata usando un assioma di Armstrong. Nel primo caso è implicata logicamente in maniera banale. Supponiamo che f_m sia stata inferita con la regola riflessiva, allora $Y \subseteq X$. Quindi, f_m è implicata logicamente in maniera banale.

Supponiamo che f_m sia stata ottenuta da f_i usando la regola di arricchimento, allora f_i ha la forma $X' \rightarrow Y'$, con X' e Y' tali che $X = X'Z$ e $Y = Y'Z$ per qualche Z . Per l'ipotesi induttiva $F \models f_i$. Siano t e t' due ennuple con $t[X'Z] = t'[X'Z]$. Allora $t[X'] = t'[X']$, quindi, per f_i , $t[Y'] = t'[Y']$, per cui $t[Y'Z] = t'[Y'Z]$. Quindi f_m è implicata logicamente da F .

Infine, supponiamo che f_m sia stata ottenuta da $f_i = X \rightarrow W$ e $f_j = W \rightarrow Y$, per qualche W , usando la regola transitiva. Per l'ipotesi induttiva $F \models f_i$ e $F \models f_j$. Siano t e t' due ennuple con $t[X] = t'[X]$; per f_i , $t[W] = t'[W]$ e quindi, per f_j , $t[Y] = t'[Y]$. Dunque, $X \rightarrow Y$ è implicata logicamente da F .

Completezza. Si deve dimostrare che se $F \models X \rightarrow Y$, allora $F \vdash X \rightarrow Y$.

Si consideri una relazione r su T di due ennupe costruita come segue: $r = \{t, t'\}$ con $t[X^+] = t'[X^+]$ e $t[A] \neq t'[A]$ per ogni $A \in T - X^+$; r soddisfa F . Infatti, sia $V \rightarrow W$ una dipendenza di F . Se $V \not\subseteq X^+$, allora $t[V] \neq t'[V]$, ed r soddisfa ovviamente la dipendenza. Se invece $V \subseteq X^+$, si ha che $F \vdash X \rightarrow V$ e poiché $V \rightarrow W$, per transitività, $F \vdash X \rightarrow W$, da cui $W \subseteq X^+$ e quindi $t[W] = t'[W]$. Pertanto $V \rightarrow W$ è soddisfatta in r .

Mostriamo ora la tesi, cioè $F \models X \rightarrow Y \Rightarrow F \vdash X \rightarrow Y$. Da $F \models X \rightarrow Y$ segue che r soddisfa $X \rightarrow Y$. Poiché $X \subseteq X^+$, e $t[X^+] = t'[X^+]$, segue che $t[Y] = t'[Y]$ per cui $Y \subseteq X^+$ e $F \vdash X \rightarrow Y$ per il Teorema 5.2. \square

Il Teorema 5.3 permette di sostituire, nella trattazione precedente, tutte le occorrenze di \models con \vdash e viceversa; in particolare, si ha che $X^+ = \{A \in T \mid F \models X \rightarrow A\}$.

È infine interessante osservare che le regole di Armstrong sono indipendenti fra di loro (cioè nessuna di esse può essere derivata dalle altre due), e che non costituiscono l'unico insieme di regole con le proprietà di correttezza e completezza.

5.2.3 Chiusura di un insieme di dipendenze funzionali

L'esistenza di regole di inferenza per derivare nuove dipendenze funzionali da altre ci consente di definire la nozione di *chiusura* di un insieme di dipendenze funzionali.

Definizione 5.7 Dato un insieme F di dipendenze funzionali, si definisce la *chiusura* di F come $F^+ = \{X \rightarrow Y \mid F \vdash X \rightarrow Y\}$.

Un problema che si presenta spesso è quello di decidere se una dipendenza funzionale appartiene a F^+ (*problema dell'implicazione*); la sua risoluzione con l'algoritmo banale di generare F^+ applicando ad F ripetutamente ed esaustivamente gli assiomi di Armstrong ha una complessità esponenziale rispetto al numero di attributi dello schema. Infatti, se a è la dimensione di T , F^+ contiene almeno le $2^a - 1$ dipendenze funzionali banali $T \rightarrow X$, dove X è un sottoinsieme non vuoto di T .

Un metodo di complessità inferiore per risolvere il problema dell'implicazione scaturisce invece dalla seguente considerazione: per decidere se $X \rightarrow Y \in F^+$, si può controllare se $Y \subseteq X^+$. Questa riformulazione del problema è resa possibile dalla validità del Teorema 5.2.

Un semplice algoritmo per calcolare X^+ è il seguente:

```

Algoritmo CHIUSURA LENTA
  input      R(T, F), X ⊆ T
  output     XPIU
  begin
    XPIU := X
    while (XPIU è cambiato) do
      for each W → V ∈ F with W ⊆ XPIU and V ⊈ XPIU do
        XPIU := XPIU ∪ V
    end
  
```

Esempio 5.2 Sia $R(T, F)$ con $T = \{A, B, C, D, E, G, H, I\}$ ed $F = \{ADG \rightarrow GI, ACH \rightarrow ADG, BC \rightarrow AD, CE \rightarrow ACH\}$. Indicando con $XPIU^j$ il valore della variabile $XPIU$ alla fine della j -esima ripetizione del ciclo **while**, la chiusura di BCE è calcolata come segue:

1. $XPIU^0 = BCE$;
2. $XPIU^1 = BCE \cup AD \cup ACH = ABCDEH$;
3. $XPIU^2 = ABCDEH \cup ADG = ABCDEGH$;

$$4. XPIU^3 = ABCDEGH \cup GI = ABCDEGHI = T.$$

A questo punto l'algoritmo termina con $(BCE)^+ = T$ e quindi BCE è una superchiave.

Teorema 5.4 L'algoritmo CHIUSURA LENTA è corretto.

Dimostrazione

Terminazione. Ad ogni iterazione si aggiunge qualche attributo a $XPIU$ oppure si termina. Poiché il numero degli attributi è finito segue che l'algoritmo non può ciclare indefinitamente.

Correttezza. Per dimostrare che quando l'algoritmo termina $XPIU = X^+$, si dimostra che $XPIU \subseteq X^+$ e $X^+ \subseteq XPIU$.

Per dimostrare che $XPIU \subseteq X^+$ è sufficiente provare per induzione sul numero di iterazioni j che $XPIU^j \subseteq X^+$, per ogni j . Per $j = 0$ l'affermazione è ovvia essendo $XPIU^0 = X \subseteq X^+$ per riflessività. Per l'ipotesi induttiva, sia $XPIU^j \subseteq X^+$ e proviamo che $XPIU^{j+1} \subseteq X^+$. Sia A un attributo aggiunto alla $j + 1$ -esima iterazione. Per come è stato definito l'algoritmo si ha che $A \in V$, $W \rightarrow V \in F$, $W \subseteq XPIU^j$. Per l'ipotesi induttiva $W \subseteq X^+$, e dal Teorema 5.2, $X \rightarrow W$; per transitività $X \rightarrow V$ e quindi $X \rightarrow A$. Pertanto $A \in X^+$.

Dimostriamo ora che $X^+ \subseteq XPIU$. Si noti innanzitutto che se $V \subseteq W$, allora $V^+ \subseteq W^+$. Pertanto, poiché $X \subseteq XPIU$, $X^+ \subseteq XPIU^+$. Basta allora dimostrare che $XPIU^+ \subseteq XPIU$, ovvero che se $A \notin XPIU$ allora $A \notin XPIU^+$, cioè esiste un'istanza valida r di R che non soddisfa $XPIU \rightarrow A$. Si costruisce una relazione $r = \{t, t'\}$ di due ennuple distinte che dipendono dal risultato dell'algoritmo ponendo $t[A] = t'[A]$ per ogni $A \in XPIU$ e $t[B] \neq t'[B]$ per ogni $B \in T - XPIU$. Supponiamo di aver dimostrato che r soddisfa ogni dipendenza in F ; si dimostra che r non soddisfa $XPIU \rightarrow A$: t e t' coincidono su $XPIU$ per costruzione, mentre non coincidono su A perché per ipotesi $A \notin XPIU$.

Rimane da provare che r soddisfa ogni dipendenza in F . Sia $W \rightarrow V$ una dipendenza di F . Se $W \not\subseteq XPIU$, allora $t[W] \neq t'[W]$, ed r soddisfa ovviamente la dipendenza. Se $W \subseteq XPIU$, si ha che $V \subseteq XPIU$ poiché l'algoritmo termina con $XPIU$. Ma $V \subseteq XPIU$ implica $t[V] = t'[V]$ e quindi r soddisfa la dipendenza. \square

Sia a il numero degli attributi di T e p il numero delle dipendenze funzionali in F . Il ciclo **while** viene eseguito al più p volte (perché ogni dipendenza funzionale dà il suo contributo alla chiusura al più una volta) e al più a volte (perché si possono aggiungere al più a attributi). Per ogni ciclo **while**, il ciclo interno viene eseguito al più p volte e ogni volta si controllano due inclusioni di insiemi. Poiché il test di contenuto tra insiemi ordinati di a elementi ha una complessità di tempo $O(a)$, si conclude che l'algoritmo CHIUSURA LENTA ha complessità di tempo, nel caso peggiore, di $O(ap \min\{a, p\})$.

L'algoritmo CHIUSURA LENTA ha il pregio della semplicità, ma adotta una tecnica molto inefficiente per verificare, per ogni iterazione ed ogni dipendenza, se la dipendenza può essere usata per aggiungere attributi ad X^+ . Questa verifica può essere resa molto più efficiente con l'impiego di opportune strutture dati, ottenendo un algoritmo con complessità di tempo di $O(ap)$ [BB79].

L'algoritmo CHIUSURA VELOCE ha un ruolo fondamentale nelle applicazioni della teoria della normalizzazione, perché il calcolo della chiusura di un insieme di attributi è richiesto in molti altri problemi, come vedremo nel seguito. Inoltre, con un'opportuna rappresentazione dei dati si può mostrare come l'algoritmo CHIUSURA VELOCE possa essere usato per risolvere altri problemi ben noti, come il *problema della raggiungibilità* in un grafo diretto $G = (V, E)$ (dato un nodo del grafo, trovare tutti gli altri nodi raggiungibili da esso). Si pone $T = V$ e $F = \{A \rightarrow B | A, B \in V \wedge (A, B) \in E\}$. Dato un nodo $v \in V$, la chiusura di $\{v\}$ contiene i nodi raggiungibili.

5.2.4 Chiavi

Usando le nozioni di dipendenza funzionale e di chiusura di un insieme di dipendenze, si possono definire in modo formale le nozioni di *superchiave*, *chiave* e *attributo primo* di uno schema.

Definizione 5.8 Dato uno schema $R\langle T, F \rangle$, un insieme di attributi $W \subseteq T$ è una *superchiave* di R se $W \rightarrow T \in F^+$.

Definizione 5.9 Dato uno schema $R\langle T, F \rangle$, un insieme di attributi $W \subseteq T$ è una *chiave* di R se W è una superchiave e non esiste un sottoinsieme stretto di W che sia una superchiave di R .

Definizione 5.10 Dato uno schema $R\langle T, F \rangle$, un attributo $A \in T$ si dice *primo* se e solo se appartiene ad almeno una chiave, altrimenti si dice *non primo*.

Dato che in uno schema ci possono essere più chiavi, di solito ne viene scelta una (generalmente con il minimo numero di attributi), la *chiave primaria*, come identificatore delle ennuple delle istanze dello schema.

In generale, il problema di trovare tutte le chiavi di uno schema richiede un algoritmo di complessità esponenziale nel caso peggiore e il problema di sapere se un attributo è primo è NP-completo [LO78].

Come trovare tutte le chiavi

Vedremo più avanti come in alcuni casi occorra stabilire se un attributo di uno schema $R\langle T, F \rangle$ sia primo e, quindi, trovare tutte le chiavi di R .

Osserviamo per prima cosa che valgono le seguenti proprietà:

1. Se un attributo A di T non appare a destra di alcuna dipendenza in F , allora A appartiene ad ogni chiave di R (si veda l'Esercizio 3).
2. Se un attributo A di T appare a destra di qualche dipendenza in F , ma non appare a sinistra di alcuna dipendenza non banale, allora A non appartiene ad alcuna chiave.

Sia X l'insieme degli attributi che non appaiono a destra di alcuna dipendenza in F . Dalla proprietà (1), segue che se $X^+ = T$, allora X è una chiave di R ed è anche l'unica possibile.

Per esempio, sia $R\langle T, F \rangle$ con $T = \{A, B, C, D, E, G\}$ ed $F = \{BC \rightarrow AD, D \rightarrow E, CG \rightarrow B\}$. C e G non appaiono a destra delle dipendenze, quindi devono far parte di ogni chiave. Poiché $CG^+ = T$, CG è l'unica chiave di R .

Se invece $X^+ \neq T$, allora occorre aggiungere a X altri attributi. Per la proprietà (2), basta aggiungere quegli attributi W di T che appaiono a destra di qualche dipendenza e a sinistra di qualche altra, uno alla volta. Ad ogni passo occorre evitare di aggiungere attributi che siano già nella chiusura di X , poiché tali attributi sono chiaramente ridondanti, oppure attributi che producono un insieme X' che contiene una chiave trovata in precedenza. Poi si calcola la chiusura di ogni X' , fino a che questa non coincide con T , il che garantisce che X' sia una chiave.

Per esempio, sia $R\langle T, F \rangle$ con $T = \{A, B, C, D, E, G\}$ ed $F = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CG \rightarrow B\}$.

G non appare a destra delle dipendenze e $G^+ = G$. Aggiungendo un attributo di $W = \{A, B, C, D\}$ a G si trova che

$$\begin{aligned}GA^+ &= GA \neq T. \\GB^+ &= GB \neq T. \\GC^+ &= T. \quad GC \text{ è una chiave di } R. \\GD^+ &= GDE \neq T.\end{aligned}$$

Si prova ad aggiungere ad GA , GB e GD un altro attributo di W , considerando solo insiemi di attributi che non contengono la chiave GC , e si trova che

$$\begin{aligned}GAB^+ &= T. \quad GAB \text{ è una chiave di } R. \\GAD^+ &= GADE \neq T. \\GBD^+ &= GBDE \neq T.\end{aligned}$$

Si prova infine ad aggiungere ad GAD e GBD un altro attributo di W , ma non si trovano insiemi di attributi che non contengono le chiavi GC e GAB , e quindi si conclude che non ne esistono altre.

In generale, una soluzione si trova con il seguente algoritmo che analizza tutti i “candidati”, ovvero i sottinsiemi di T che potrebbero essere chiavi.

Algoritmo TROVA TUTTE LE CHIAVI

```

input       $R(T, F)$ 
output     Chiavi l'insieme di tutte le chiavi di  $R(T, F)$ 
begin
   $NoDes := T - \cup_{X \rightarrow A \in F} A;$ 
   $SinDes := \cup_{X \rightarrow A \in F} X \cap \cup_{X \rightarrow A \in F} A;$ 
   $Candidati := [NoDes:(SinDes)];$ 
   $Chiavi := [];$ 
  while ( $Candidati$  non vuoto) do
    begin
       $X::(Y) := \text{first}(Candidati);$ 
       $Candidati := \text{rest}(Candidati);$ 
      if not some  $K$  in  $Chiavi$  with  $K \subset X$ 
      then if  $X^+ = T$ 
        then  $Chiavi := Chiavi + X;$ 
        else begin
           $A_1 \dots A_n := Y - X^+;$ 
          for  $i$  in  $1..n$  do  $Candidati := Candidati$ 
                           append  $[XA_i::(A_{i+1} \dots A_n)]$ 
        end
      end
    end

```

L'algoritmo memorizza in $Chiavi$ le chiavi già trovate e nella lista $Candidati$ i candidati ancora da analizzare.³ Ogni candidato è un insieme di sottinsiemi di T rappresentato in una forma compatta $X::(Y)$, che denota tutti gli insiemi formati dagli attributi X uniti ad un qualsiasi sottinsieme degli attributi Y . Ad esempio, $AB::(CD)$ rappresenta $\{AB, ABC, ABD, ABCD\}$. Se $NoDes$ sono gli attributi che non appaiono a destra di nessuna dipendenza e $SinDes$ sono quelli che appaiono sia a sinistra che a destra, per le osservazioni precedenti tutte le chiavi appartengono all'insieme $NoDes:(SinDes)$, per cui inizialmente $Candidati = [NoDes:(SinDes)]$.

Gli insiemi in $X::(Y)$ sono analizzati a partire da X . Se X è chiave, allora tutti gli altri insiemi in $X::(Y)$ sono scartati. Altrimenti, poiché gli elementi di X^+ non possono apparire in una chiave che contiene X , dato $Y - X^+ = \{A_1 \dots A_n\}$, si

3. Su una lista $L = [x_1, \dots, x_n]$ si usano i seguenti operatori: $\text{first}(L)$ che ritorna il primo elemento di una lista non vuota L ; $\text{rest}(L)$ che ritorna la lista non vuota L priva del primo elemento; $L + x_i$ che ritorna una lista i cui primi elementi sono quelli di L e l'ultimo x_i ; $L_1 \text{ append } L_2$ che ritorna una lista i cui primi elementi sono quelli di L_1 ed i successivi quelli di L_2 .

mettono in *Candidati* i nuovi candidati $X A_1 :: (A_2, \dots, A_n)$, $X A_2 :: (A_3, \dots, A_n)$, ..., $X A_n :: ()$, i quali coprono $(X :: (A_1 \dots A_n)) - \{X\}$.

Il test $X^+ = T$ assicura che X è superchiave. Per essere certi che X sia anche chiave, si controlla che X non contenga chiavi già trovate in precedenza. Le chiavi trovate in seguito invece non potranno mai essere contenute strettamente in X perché tutti i candidati analizzati dopo X avranno lunghezza maggiore o uguale. Questa invariante è assicurata mantenendo *Candidati* ordinata per lunghezze crescenti, inserendo i nuovi candidati sempre in coda alla lista, con l'operatore append.

Esemplichiamo l'applicazione dell'algoritmo su $R \langle T, F \rangle$ con $T = \{A, B, C, D, E\}$ ed $F = \{AC \rightarrow B, C \rightarrow D, D \rightarrow E, ABD \rightarrow C, B \rightarrow E\}$, mostrando l'effetto su *Chiavi* e *Candidati* di ciascuna esecuzione del ciclo while, con le variabili inizializzate a

$$NoDes = A; \quad Chiavi = []; \quad SinDes = BCD; \quad Candidati = [A :: (BCD)]$$

1. $X :: (Y) := \text{first}(Candidati) = A :: (BCD);$
 $Candidati := \text{rest}(Candidati) = []$
 $X^+ = A^+ = A \Rightarrow A$ non chiave
 $Y - X^+ = BCD - A = BCD$ e quindi
 $Candidati := [] + AB :: (CD) + AC :: (D) + AD :: ()$
2. $X :: (Y) := \text{first}(Candidati) = AB :: (CD)$
 $Candidati := \text{rest}(Candidati) = [AC :: (D), AD :: ()]$
 $X^+ = AB^+ = ABE \Rightarrow AB$ non chiave
 $Y - X^+ = CD - ABE = CD$ e quindi
 $Candidati := [AC :: (D), AD :: ()] + ABC :: (D) + ABD :: ()$
3. $X :: (Y) := \text{first}(Candidati) = AC :: (D)$
 $Candidati := \text{rest}(Candidati) = [AD :: (), ABC :: (D), ABD :: ()]$
 $X^+ = AC^+ = ACBDE \Rightarrow AC$ chiave
 $Chiavi := [AC];$
4. $X :: (Y) := \text{first}(Candidati) = AD :: ()$
 $Candidati := \text{rest}(Candidati) = [ABC :: (D), ABD :: ()]$
 $X^+ = AD^+ = ADE \Rightarrow AD$ non chiave
 $Y = (),$ per cui non si genera nessun nuovo candidato.
5. $X :: (Y) := \text{first}(Candidati) = ABC :: (D)$
 $Candidati := \text{rest}(Candidati) = [ABD :: ()][]$
 esiste $AC \in Chiavi$ con $AC \subset ABC$, per cui $ABC :: (D)$ è scartato
6. $X :: (Y) := \text{first}(Candidati) = ABD :: ()$
 $Candidati := \text{rest}(Candidati) = []$
 $X^+ = ABD^+ = ABDEC \Rightarrow ABD$ chiave
 $Chiavi := [AC, ABD];$

Candidati è vuoto, per cui l'algoritmo si ferma e restituisce le due chiavi trovate.

5.2.5 Copertura di un insieme di dipendenze

Per operare su insiemi di dipendenze, è comodo portarli in una forma “minima”. Per arrivare ad una definizione formale di minimalità, si introduce per prima cosa il concetto di *copertura*.

Definizione 5.11 Due insiemi di dipendenze F e G sugli attributi T di una relazione R sono *equivalenti*, ($F \equiv G$), se e solo se $F^+ = G^+$. Se $F \equiv G$ allora F è una *copertura* di G e viceversa.

La relazione di equivalenza fra insiemi di dipendenze permette di stabilire quando due schemi di relazione rappresentano gli stessi fatti: basta controllare se gli attributi sono gli stessi e le dipendenze equivalenti. Per controllare l'equivalenza delle dipendenze è sufficiente controllare che tutte le dipendenze di F appartengano a G^+ e che tutte quelle di G appartengano a F^+ .

La definizione seguente ci fornisce il tipo di copertura di insiemi di dipendenze funzionali che cerchiamo.

Definizione 5.12 Sia F un insieme di dipendenze funzionali.

1. Dato $X \rightarrow Y \in F$, X contiene un *attributo estraneo* A se e solo se $(F - \{X \rightarrow Y\}) \cup \{X - \{A\} \rightarrow Y\} \equiv F$, ovvero se e solo se $X - \{A\} \rightarrow Y \in F^+$.
2. $X \rightarrow Y$ è una *dipendenza ridondante* se e solo se $(F - \{X \rightarrow Y\}) \equiv F$, ovvero se e solo se $X \rightarrow Y \in (F - \{X \rightarrow Y\})^+$.
3. F è una *copertura canonica* se e solo se:
 - (a) ogni parte destra di una dipendenza ha un unico attributo;
 - (b) le dipendenze non contengono attributi estranei;
 - (c) non esistono dipendenze ridondanti.

La terminologia qui adottata è quella di [Mai83]; in [UW01] questo insieme viene chiamato insieme minimale.

Le dipendenze che non contengono attributi estranei, e il cui determinato è un unico attributo, sono dette *dipendenze elementari*.

Il seguente teorema, infine, ci permetterà d'ora in poi di utilizzare, quando ci serviranno, insiemi di dipendenze che sono coperture canoniche.

Teorema 5.5 Per ogni insieme di dipendenze F esiste una copertura canonica.

Questo teorema verrà dimostrato fornendo un algoritmo per calcolare la copertura canonica di un qualsiasi insieme di dipendenze. Assumeremo che le dipendenze abbiano un unico attributo nella parte destra; se questo non fosse il caso è semplice trasformarle (ad esempio, $A \rightarrow BC$ diventa $A \rightarrow B$ e $A \rightarrow C$).

Algoritmo Calcolo della copertura canonica

```

 F insieme di dipendenze funzionali
 G copertura canonica di F
begin
  G := F;
  for each X → Y ∈ G with |X| > 1 do
    begin
      Z := X;
      for each A ∈ X do
        if Y ⊆ [Z - {A}]_F^+ then Z := Z - {A};
        G = (G - {X → Y}) ∪ {Z → Y}
      end ;
      for each f ∈ G do
        if f ∈ (G - {f})^+ then G = G - {f}
    end

```

Questo algoritmo prima elimina gli attributi estranei dalle dipendenze, quindi elimina le dipendenze ridondanti. In entrambi i casi viene utilizzato il calcolo della chiusura di un insieme di attributi, con uno degli algoritmi già discussi. Esso ha complessità $O(a^2 p^2)$, perché il test $Y \subseteq [X - A]_F^+$ ha complessità $O(ap)$. Un algoritmo più complicato con un costo medio inferiore è stato dato da [DM88].

Si osservi che la verifica “ $Y \subseteq [Z - \{A\}]_F^+$ ” dell’estraneità dell’attributo A va fatta rispetto all’insieme F che contiene la dipendenza sotto esame, poiché tale dipendenza potrebbe essere utilizzata nella dimostrazione del fatto che $Z - \{A\} \rightarrow Y$. Viceversa, la verifica della ridondanza “ $f \in (G - \{f\})^+$ ” va fatta senza considerare f , perché altrimenti sarebbe banalmente verificata.

È importante eliminare prima gli attributi estranei e dopo le dipendenze ridondanti, come dimostra il seguente esempio: $F = \{BA \rightarrow D, B \rightarrow A, D \rightarrow A\}$. La dipendenza $B \rightarrow A$ non è riconosciuta come ridondante finché l’attributo estraneo A in $BA \rightarrow D$ non viene eliminato.

Notare che il Teorema 5.5 afferma l’esistenza, ma non l’univocità di una copertura canonica. Il seguente esempio mostra che in generale F può avere più coperture canoniche.

Esempio 5.3 Per $F = \{AB \rightarrow C, A \rightarrow B, B \rightarrow A\}$, sia $\{A \rightarrow C, A \rightarrow B, B \rightarrow A\}$ che $\{B \rightarrow C, A \rightarrow B, B \rightarrow A\}$ sono coperture canoniche.

In [Mai83] sono presentati altri tipi di coperture e i relativi algoritmi di calcolo.

Definizione 5.13 Un insieme F di dipendenze funzionali è *minimo* se ha meno dipendenze di ogni insieme a lui equivalente.

Esempio 5.4 L’insieme $G = \{A \rightarrow BC, B \rightarrow A, AD \rightarrow E, BD \rightarrow I\}$ non è ridondante ma nemmeno minimo poiché $F = \{A \rightarrow BC, B \rightarrow A, AD \rightarrow EI\}$ è equivalente a G ma ha meno dipendenze. F è una copertura minima di G .

Definizione 5.14 Un insieme F di dipendenze funzionali è *ottimo* se ha meno simboli di attributi di ogni insieme a lui equivalente.

Esempio 5.5 L’insieme $F = \{EC \rightarrow D, AB \rightarrow E, E \rightarrow AB\}$ è una copertura ottima per $G = \{ABC \rightarrow D, AB \rightarrow E, E \rightarrow AB\}$. Notare che G è ridotto e minimo ma non ottimo.

5.3 Decomposizione di schemi

Come discusso inizialmente, l’approccio da seguire per eliminare anomalie da uno schema mal definito, è quello di decomporlo in schemi più piccoli che godono di particolari proprietà (*forme normali*), ma sono in qualche senso equivalenti allo schema originale. Il concetto di decomposizione viene definito come segue.

Definizione 5.15 Dato uno schema $R(T)$, $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$ è una *decomposizione* di R se e solo se $\bigcup_i T_i = T$.

Si noti che la definizione non richiede che gli schemi R_i siano disgiunti.

Per ciò che riguarda l’equivalenza tra lo schema originario e la sua decomposizione, si richiede in genere che questa soddisfi due condizioni, indipendenti fra di loro: *preservi i dati (decomposizione senza perdite o lossless join)* e *preservi le dipendenze*.

5.3.1 Decomposizioni che preservano i dati

Vediamo con un esempio cosa vuol dire che una decomposizione preservi i dati, e perché tale proprietà sia importante.

Esempio 5.6 Si consideri la relazione con schema $R(P, T, C)$ che memorizza fatti relativi ai proprietari (P) di case (C), che possono avere più telefoni (T) intestati al loro proprietario. Supponiamo che valgano le dipendenze $T \rightarrow C$ e $C \rightarrow P$:

R	P	T	C
	p1	t1	c1
	p1	t2	c2
	p1	t3	c2

Supponiamo che si decida di decomporre lo schema e la relazione come segue:

$R_1 = \pi_{P,T}(R) =$	P	T	$R_2 = \pi_{P,C}(R) =$	P	C
	p1	t1		p1	c1
	p1	t2		p1	c2
	p1	t3			

Per ricostruire la relazione di partenza bisogna ricombinare le due relazioni della decomposizione effettuando una giunzione naturale:

$R_1 \bowtie R_2 =$	P	T	C
	p1	t1	c1
	p1	t1	c2
	p1	t2	c1
	p1	t2	c2
	p1	t3	c1
	p1	t3	c2

Il risultato ottenuto non coincide però con la tabella iniziale: ha più ennuple e viola le dipendenze.

L'esempio ha mostrato come nella decomposizione di una relazione e successiva giunzione si possa perdere parte dell'informazione. Quando ciò si verifica si dice che la decomposizione *non preserva i dati*.

Definizione 5.16 $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$ è una *decomposizione di $R(T, F)$ che preserva i dati* se e solo se per ogni relazione r che soddisfa $R(T, F)$:

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

La precedente definizione asserisce che, per una decomposizione che preserva i dati, ogni istanza valida r della relazione di partenza deve essere uguale alla giunzione naturale della sua proiezione sui T_i , mentre, per una decomposizione qualunque, ne è solo un sottoinsieme:

Teorema 5.6 Se $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$ è una (qualunque) decomposizione di $R(T, F)$, allora per ogni istanza r di $R(T)$ si ha:

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

Dimostrazione Poiché $\bigcup_i T_i = T$, anche la relazione $(\pi_{T_1}r) \bowtie \dots \bowtie (\pi_{T_k}r)$ è definita su T . Data un'ennupla $t \in r$, $t[T_i] \in \pi_{T_i}r$ e dalla definizione di giunzione naturale segue che $t \in (\pi_{T_1}r) \bowtie \dots \bowtie (\pi_{T_k}r)$. \square

Questo teorema chiarisce che cosa sia una perdita di informazione: proiettando una relazione sui sottoschemi e poi facendo la giunzione si ottengono più ennupla di quante ce ne fossero nella relazione originaria.

Le definizione precedente è quantificata sull'insieme generalmente infinito di tutte le istanze valide di una relazione, e quindi non è utilizzabile direttamente per controllare la proprietà di preservazione dei dati di una decomposizione. Per questo, ci viene in aiuto il seguente teorema.

Teorema 5.7 Sia $\rho = \{R_1(T_1), R_2(T_2)\}$ una decomposizione di $R\langle T, F \rangle$. ρ è una decomposizione che preserva i dati se e solo se $T_1 \cap T_2 \rightarrow T_1 \in F^+$ oppure $T_1 \cap T_2 \rightarrow T_2 \in F^+$.

Dimostrazione

(\Leftarrow) Supponiamo che $T_1 \cap T_2 \rightarrow T_1 \in F^+$. Sia r un'istanza valida di $R\langle T, F \rangle$ e $s = (\pi_{T_1}r) \bowtie (\pi_{T_2}r)$. Sia $t \in s$, bisogna dimostrare che $t \in r$.

Per come è stata definita s , esistono due ennupla u e v in r tali che $u[T_1] = t[T_1]$, $v[T_2] = t[T_2]$ e $u[T_1 \cap T_2] = v[T_1 \cap T_2] = t[T_1 \cap T_2]$. Poiché $T_1 \cap T_2 \rightarrow T_1 \in F^+$, ne segue che $u[T_1] = v[T_1]$ e quindi $t = v$.

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

(\Rightarrow) Supponiamo che per ogni istanza valida r di $R\langle T, F \rangle$, $r = (\pi_{T_1}r) \bowtie (\pi_{T_2}r)$; bisogna dimostrare che $T_1 \cap T_2 \rightarrow T_1 \in F^+$ oppure $T_1 \cap T_2 \rightarrow T_2 \in F^+$. Procederemo per assurdo, supponendo che nessuna delle due dipendenze funzionali sia implicata da F .

Sia $W = (T_1 \cap T_2)^+$, $Y_1 = T_1 - W$ e $Y_2 = T_2 - W$. Y_1 e Y_2 sono non vuoti per ipotesi, e W , Y_1 e Y_2 sono una partizione di T .

Per ogni $A_i \in T$, $1 \leq i \leq k$, consideriamo due valori $a_i, a'_i \in \text{dom}(A_i)$, con $a_i \neq a'_i$. Costruiamo la relazione r con attributi WY_1Y_2 formata dalle due ennupla:

$$e_1[A_i] = a_i, 1 \leq i \leq k$$

$$e_2[A_i] = \begin{cases} a_i & \text{se } A_i \in W \\ a'_i & \text{se } A_i \in Y_1Y_2 \end{cases}$$

r soddisfa ogni dipendenza $V \rightarrow Z \in F$. Infatti, se $V \not\subseteq W$, allora $e_1[V] \neq e_2[V]$, ed r soddisfa ovviamente la dipendenza. Se $V \subseteq W$, si ha che $(T_1 \cap T_2) \rightarrow V$, quindi $(T_1 \cap T_2) \rightarrow Z$ per transitività, quindi $Z \subseteq W$, per cui $e_1[Z] = e_2[Z]$, e quindi r soddisfa la dipendenza. Inoltre, poiché Y_1 e Y_2 non sono vuoti, $\pi_{T_1}r$ e $\pi_{T_2}r$ contengono due ennupla e la loro giunzione naturale ne contiene quattro, più di quelle di r :

$$(\pi_{T_1}r) \bowtie (\pi_{T_2}r) \neq r$$

contraddicendo l'ipotesi. \square

Questo risultato è stato generalizzato con un algoritmo, che non sarà qui presentato, che controlla se una decomposizione con un numero qualsiasi di schemi di relazione preserva i dati (si veda, ad esempio, [AA93]).

5.3.2 Decomposizioni che preservano le dipendenze

L'altra proprietà interessante di una decomposizione è che preservi le dipendenze. Intuitivamente, questo significa che l'unione delle dipendenze dei sottoschemi è "uguale" alle dipendenze dello schema originario.

Riprendiamo l'esempio precedente per mostrare cosa vuol dire che una decomposizione preservi le dipendenze e perché anche questa proprietà sia importante.

Esempio 5.7 Supponiamo di decomporre la relazione con schema $R(P, T, C)$ e dipendenze $T \rightarrow C$ e $C \rightarrow P$ come segue:

$R_1 = \pi_{P,T}(R) =$	$R_2 = \pi_{T,C}(R) =$		
P	T	T	C
p1	t1	t1	c1
p1	t2	t2	c2
p1	t3	t3	c2

La decomposizione preserva i dati, per il teorema visto, ma non conserva la dipendenza $C \rightarrow P$, perché gli attributi C e P sono finiti in relazioni diverse. La conseguenza negativa di questo fatto è questa: supponiamo che si voglia inserire nella base di dati il fatto che la casa $c1$ ha un nuovo telefono $t4$ intestato alla persona $p2$. Se l'operazione si fa nella relazione con schema $R(P, T, C)$, l'operazione verrebbe proibita perché si viola il vincolo $C \rightarrow P$, ma se l'operazione si fa con le relazioni con schemi $R_1(P, T)$ ed $R_2(T, C)$ essa andrebbe a buon fine perché non viola le dipendenze dei singoli schemi, a meno di non fare i controlli sulla loro giunzione perdendo i benefici della decomposizione.

Per le ragioni che vedremo più avanti, una decomposizione di $R(P, T, C)$ che preservi dati e dipendenze è $R_1(T, C)$ ed $R_2(C, P)$.

Per definire formalmente cosa vuol dire che una decomposizione preservi le dipendenze, ci serve la nozione di *proiezione di un insieme di dipendenze*:

Definizione 5.17 Dato $R\langle T, F \rangle$, e $T_i \subseteq T$, la *proiezione* di F su T_i è:

$$\pi_{T_i}(F) = \{X \rightarrow Y \in F^+ \mid X, Y \subseteq T_i\}.$$

È importante notare che in questa definizione la proiezione viene costruita considerando le dipendenze presenti in F^+ , non quelle in F .

Esempio 5.8 Si consideri lo schema $R\langle T, F \rangle$, con $T = \{ABC\}$, $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$. Allora $\pi_{AB}(F) = \{A \rightarrow B, B \rightarrow A\}$ e $\pi_{AC}(F) = \{A \rightarrow C, C \rightarrow A\}$.

Vediamo un algoritmo banale di complessità esponenziale per il calcolo di $\pi_{T_i}(F)$.

Algoritmo Calcolo della proiezione di un insieme di dipendenze
 $R\langle T, F \rangle$ e $T_i \subseteq T$
 Una copertura della proiezione di F su T_i
begin
 for each $Y \subseteq T_i$ **do**
 begin
 $Z := Y_{F^+}$
 ritorna $Y \rightarrow (Z \cap T_i)$
 end
end

La nozione di proiezione di un insieme di dipendenze ci permette di formalizzare l'equivalenza fra le dipendenze dello schema originale e quelle degli schemi decomposti:

Definizione 5.18 $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$ è una *decomposizione* di $R\langle T, F \rangle$ che preserva le dipendenze se e solo se $\cup \pi_{T_i}(F) \equiv F$.

Esempio 5.9 Si consideri lo schema di relazione $R\langle T, F \rangle$, con $T = \{ABC\}$, $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$. Potrebbe sembrare che la decomposizione $\rho = \{R_1(A, B), R_2(B, C)\}$ non preservi le dipendenze perché A e C non appaiono insieme in uno schema della decomposizione. Invece $\pi_{AB}(F) = \{A \rightarrow B, B \rightarrow A\}$ e $\pi_{BC}(F) = \{B \rightarrow C, C \rightarrow B\}$ e $\pi_{AB}(F) \cup \pi_{BC}(F) \vdash C \rightarrow A$.

Potrebbe sembrare che il controllo che una decomposizione preservi le dipendenze, ovvero che ogni dipendenza $X \rightarrow Y \in F$ appartenga a G^+ , con $G = \cup \pi_{T_i}(F)$, richieda un algoritmo di complessità di tempo esponenziale in quanto occorre calcolare le proiezioni di F sui T_i . In [Ull83] viene invece presentato un algoritmo di complessità di tempo polinomiale per risolvere il problema. L'idea su cui si basa l'algoritmo è di stabilire se $X \rightarrow Y \in G^+$ trovando la chiusura X_G^+ di X rispetto a G senza calcolare G in maniera esplicita:

Algoritmo CHIUSURA X_G^+

```

  $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$  decomposizione di  $R\langle T, F \rangle$ ,  $X \subseteq T$ 
  $X_G^+, \text{ con } G = \cup \pi_{T_i}(F)$ 
begin
   $X_G^+ := X;$ 
  while ( $X_G^+$  è cambiato) do
    for each  $i := 1$  to  $k$  do
       $X_G^+ := X_G^+ \cup ((X_G^+ \cap T_i)^+ \cap T_i)$ 
end

```

dove la chiusura $(X_G^+ \cap T_i)^+$ viene calcolata usando l'algoritmo visto in precedenza. Ad ogni iterazione il passo $X_G^+ := X_G^+ \cup ((X_G^+ \cap T_i)^+ \cap T_i)$ aggiunge a X_G^+ gli attributi A_i tali che $(X_G^+ \cap T_i) \rightarrow A_i \in \pi_{T_i}(F)$.

Si dimostra che il seguente algoritmo determina correttamente se una decomposizione preservi le dipendenze.

Algoritmo Controllo se una decomposizione conserva le dipendenze

```

  $\rho = \{R_1(T_1), \dots, R_k(T_k)\}$  decomposizione di  $R\langle T, F \rangle$ 
 "Sì" se  $\rho$  preserva le dipendenze di  $R$ 
begin
  for each  $X \rightarrow Y \in F$  do
    if  $Y \not\subseteq X_G^+$  then Termina con "No";
    Termina con "Sì"
end

```

Esempio 5.10 Si consideri lo schema di relazione $R\langle T, F \rangle$, con $T = \{ABCD\}$, $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$ e quindi tale che in F^+ ogni attributo determina tutti gli altri. Vogliamo controllare se la decomposizione $\rho = \{R_1(AB), R_2(BC), R_3(CD)\}$ preserva le dipendenze di R . Intuitivamente potrebbe sembrare che ciò non sia vero perché la dipendenza $D \rightarrow A$ non viene proiettata su nessuna relazione R_i . Dobbiamo però considerare che in realtà è F^+ che viene proiettata sugli R_i , ed F^+ contiene anche le dipendenze $\{D \rightarrow C, C \rightarrow B, B \rightarrow A\}$, che rimangono nelle proiezioni e che, una volta rimesse insieme, implicano logicamente $D \rightarrow A$.

Verifichiamo come l'algoritmo stabilisca correttamente che $D \rightarrow A \in G^+$, con $G = \pi_{AB}(F) \cup \pi_{BC}(F) \cup \pi_{CD}(F)$, ovvero che $A \in D_G^+$.

Iniziamo il calcolo di D_G^+ ponendo $D_G^+ := \{D\}$. Il corpo del ciclo eseguito per R_1 non modifica D_G^+ perché $\{D\} \cup ((\{D\} \cap \{A, B\})^+ \cap \{A, B\})$ è ancora uguale a $\{D\}$. In maniera analoga il caso per R_2 , mentre per R_3 abbiamo:

$$\begin{aligned}
 D_G^+ &= \{D\} \cup ((\{D\} \cap \{C, D\})_F^+ \cap \{C, D\}) \\
 &= \{D\} \cup (\{D\}_F^+ \cap \{C, D\}) \\
 &= \{D\} \cup (\{A, B, C, D\} \cap \{C, D\}) \\
 &= \{C, D\}
 \end{aligned}$$

Analogamente, nel passo successivo, eseguendo il corpo del ciclo per R_2 su $D_G^+ = \{C, D\}$ produce $D_G^+ = \{B, C, D\}$ ed infine, nel terzo passo, otteniamo $D_G^+ = \{A, B, C, D\}$, da cui risulta vero che $A \in D_G^+$.

È importante osservare che le due proprietà delle decomposizioni, di preservare i dati e le dipendenze, sono del tutto indipendenti: esistono cioè delle decomposizioni che preservano i dati ma non le dipendenze, e viceversa. Il seguente risultato collega tuttavia le due cose, e fornisce una condizione *sufficiente* (anche se non necessaria) per stabilire che una decomposizione che preserva le dipendenze preserva anche i dati.

Teorema 5.8 Sia $\rho = \{R_i \langle T_i, F_i \rangle\}$ una decomposizione di $R \langle T, F \rangle$ che preserva le dipendenze e tale che T_j , per qualche j , è una superchiave per $R \langle T, F \rangle$. Allora ρ preserva i dati.

Dimostrazione

Supponiamo, senza perdita di generalità, che T_1 sia la superchiave. Dobbiamo dimostrare che, per ogni r che soddisfa $R \langle T, F \rangle$, $\pi_{T_1} r \bowtie \pi_{T_2} r \dots \bowtie \pi_{T_n} r \subseteq r$. Sia $e \in \pi_{T_1} r \bowtie \pi_{T_2} r \dots \bowtie \pi_{T_n} r$. Per definizione di giunzione esistono $e_i \in \pi_{T_i} r$ tali che, per i in $1, \dots, n$, $e_i[T_i] = e[T_i]$. Per definizione di proiezione esistono $e'_i \in r$ tali che, per i in $1, \dots, n$, $e'_i[T_i] = e_i[T_i]$, da cui $e'_i[T_i] = e[T_i]$. Vogliamo ora dimostrare che in realtà $e = e'_1$; la tesi segue, poiché $e'_1 \in r$. A tale scopo basta dimostrare che la relazione s , con schema $R(T)$, composta solo da e e da e'_1 , soddisfa F ; questo implicherebbe $e = e'_1$, poiché le due ennuple coincidono sulla superchiave T_1 .

Poiché la scomposizione preserva le dipendenze, è sufficiente dimostrare che s soddisfa $\pi_{T_i} F$ per ogni i . Si osservi che, per ogni dipendenza $X \rightarrow Y$ e per ogni $T' \subseteq T$ che includa sia X che Y , una relazione $r : R \langle T, F \rangle$ soddisfa la dipendenza $X \rightarrow Y$ se e solo se la soddisfa $\pi_{T'} r$. Quindi anche s soddisfa $\bigcup_i \pi_{T_i} F$ se e solo se $\pi_{T_1} s$, ovvero $\{e[T_i], e'_1[T_i]\}$, soddisfa $\pi_{T_i} F$ per ogni i . $\{e[T_i], e'_1[T_i]\}$, soddisfa $\pi_{T_1} F$ poiché $\{e[T_i], e'_1[T_i]\} \subseteq \pi_{T_1} r$: $e[T_i] = e'_1[T_i]$, ed e'_1 ed e'_i stanno entrambi in r . \square

Nel seguito, discutendo gli algoritmi di decomposizioni di schemi in forme normali, avremo come obiettivo l'ottenimento di decomposizioni con entrambe le proprietà. Vedremo, però, che questo non sarà sempre possibile, e che in alcuni casi otterremo degli schemi non completamente equivalenti a quelli di partenza.

5.4 Forme normali

Con l'aiuto dei concetti introdotti, siamo ora in grado di affrontare l'obiettivo principale della normalizzazione: il passaggio da schemi "anomali" a schemi "ben fatti". Questa teoria, nata dai lavori di Codd, insieme al modello relazionale, si è via via affinata, sia negli strumenti che negli obiettivi perseguiti. Durante il suo sviluppo, sono state individuate diverse categorie di "anomalie" dovute a cattiva progettazione, e questo ha portato alla definizione di diverse *forme normali*, intese come proprietà che devono essere soddisfatte dalle dipendenze fra attributi di schemi "ben fatti".

Tra le forme normali proposte alcune hanno solo un interesse storico, come la *prima forma normale* e la *seconda forma normale*. La prima forma normale richiede che i valori di tutti i domini di una relazione siano atomici, proprietà oggi considerata un vincolo implicito del modello dei dati. Recentemente, sono stati introdotti dei modelli relazionali che non soddisfano il vincolo dell'atomicità dei domini, estendendo sia i linguaggi relazionali che la teoria, per trattare anche gli operatori sui valori dei domini (*modelli relazionali estesi*, *modelli relazionali non in prima forma normale*, *modelli relazionali a oggetti*). Questi nuovi modelli relazionali prevedono che il valore di un attributo in una ennupla possa essere un valore elementare, un'ennupla, oppure una relazione.

Le forme normali più interessanti sono la *forma normale di Boyce-Codd (BCNF)* e la *terza forma normale (3NF)*, meno restrittiva della BCNF, che vedremo in questo ordine anche se la BCNF è stata proposta successivamente alla 3NF.

5.4.1 Forma Normale di Boyce-Codd

L'idea su cui si basa la nozione di forma normale di Boyce-Codd è che una dipendenza funzionale $X \rightarrow A$ (in cui X non contiene attributi estranei) indica che, nel dominio che si modella, esiste una collezione C di entità che sono univocamente identificate da X . Ad esempio, facendo riferimento allo schema

Biblioteca(NomeUtente, Residenza, Telefono, NumeroLibro, Autore, Titolo, Data)

presentato all'inizio del capitolo, la dipendenza $\text{NomeUtente} \rightarrow \text{Residenza}$, Telefono , indica l'esistenza nel dominio modellato di entità identificate da un NomeUtente e con proprietà Residenza e Telefono . Quindi, se una relazione ben disegnata contiene X , vi sono solo due possibilità:

1. se la relazione modella la collezione C , allora X deve essere una chiave per tale relazione;
2. se la relazione non modella la collezione C , allora X deve apparire solo come chiave esterna, per cui nessuno degli altri attributi delle entità della collezione deve apparire nella relazione.

Ne segue che, in ogni relazione, o X è una chiave (caso 1), oppure non deve apparire nessuna $A \notin X$ tale che $X \rightarrow A$ (caso 2).

Più precisamente, vale la seguente definizione.

Definizione 5.19 Uno schema $R\langle T, F \rangle$ è in BCNF se e solo se per ogni dipendenza funzionale non banale $X \rightarrow Y \in F^+$, X è una superchiave.

In seguito, le dipendenze $X \rightarrow Y$ non banali tali che X non è una superchiave saranno chiamate *dipendenze anomalie*.

Questa definizione indica chiaramente come il fatto di essere in BCNF dipenda solo dalla chiusura F^+ delle dipendenze, e non dalla specifica copertura F che si è scelta, ma non è utilizzabile in pratica, perché la generazione di tutto F^+ è un'operazione troppo costosa. Il teorema seguente fornisce invece un criterio per stabilire se uno schema di relazione sia in BCNF con un algoritmo di complessità polinomiale.

Teorema 5.9 Uno schema $R\langle T, F \rangle$ è in BCNF se e solo se per ogni dipendenza funzionale non banale $X \rightarrow Y \in F$, X è una superchiave.

Corollario 5.1 Uno schema $R\langle T, F \rangle$, con F copertura canonica, è in BCNF se e solo se per ogni dipendenza funzionale elementare $X \rightarrow A \in F$, X è una superchiave (ovvero è una chiave).

Da questo risultato scaturisce che un algoritmo per controllare se uno schema di relazione è in BCNF ha complessità $O(ap^2)$.

Esempio 5.11 Si consideri il seguente schema di relazione:

Magazzini($\{\text{Articolo}, \text{Magazzino}, \text{Quantità}, \text{Indirizzo}\}$,
 $\{\text{Articolo} \rightarrow \text{Magazzino} \rightarrow \text{Quantità}$,
 $\text{Magazzino} \rightarrow \text{Indirizzo}\})$

che descrive gli articoli presenti in un certo numero di magazzini, insieme con la quantità disponibile. Lo schema non è in forma normale perché l'attributo *Indirizzo* dipende dall'attributo *Magazzino*, che non è una chiave per la relazione, come si può verificare dalla seguente istanza valida.

Magazzini	Articolo	Magazzino	Quantità	Indirizzo
	Flauto	Roma	10	Via Cavour, 7
	Oboe	Roma	5	Via Cavour, 7
	Arpa	Torino	1	Via Mazzini, 11

Se ne può dedurre che la relazione mescola informazioni relative a quelle entità che sono identificate dal campo *Magazzino* con informazioni relative ad altre entità (gli articoli), per cui lo schema è mal disegnato. Più concretamente, dalla violazione della forma normale scaturiscono le seguenti anomalie:

1. ridondanza:

- (a) inserendo un nuovo articolo in un magazzino già presente occorre replicare l'indirizzo del magazzino;
- (b) la modifica di indirizzo di un magazzino deve essere riportata in tutte le ennuple dove è presente quel magazzino, altrimenti si ha una inconsistenza.

Questa ridondanza scaturisce direttamente dalla presenza di dipendenze anomale, in modo del tutto indipendente da ogni altro aspetto del mondo modellato;

2. difficoltà a gestire l'esistenza di magazzini senza merce:

- (a) per inserire un nuovo magazzino è necessario che vi sia almeno un articolo;
- (b) eliminando l'ultimo esemplare di un certo articolo (come "arpa" nel magazzino di "Torino"), se l'articolo era l'unico presente va perduto anche l'indirizzo del magazzino.

Queste anomalie non scaturiscono direttamente dalle dipendenze anomale, ma piuttosto dall'esistenza nel mondo modellato di una nozione di "magazzino" che è indipendente da quella delle merci immagazzinate.

5.4.2 Normalizzazione di schemi in BCNF

La BCNF fornisce un criterio per stabilire in modo formale se uno schema è privo di anomalie. In caso contrario esistono algoritmi, detti di *normalizzazione*, per trasformare opportunamente lo schema.

Gli algoritmi di normalizzazione di schemi in BCNF sono detti di *analisi* (dal greco, *scomposizione*) in quanto partono da uno schema contenente tutti gli attributi e lo dividono fino a che non si ottiene uno schema in BCNF. Più precisamente, se $R(XAZ, F)$ non è in BCNF a causa della dipendenza anomala $X \rightarrow A$, allora R si decomponete nei due schemi $R_1(X, A)$ ed $R_2(X, Z)$, si calcolano le proiezioni delle dipendenze di R su R_1 e R_2 e si ripete il procedimento. La decomposizione trovata non è l'unica possibile ma dipende dall'ordine in cui si prendono in considerazione le dipendenze anomale dello schema.

Algoritmo Decomposizione per forma normale di Boyce-Codd

```

  $R_1 \langle T_1, F_1 \rangle$ , con  $F$  copertura canonica (non necessario, ma utile)
  $\rho = \{R_1, \dots, R_m\}$  decomposizione di  $R$  in BCNF che preserva i dati
begin
   $\rho := \{R_1 \langle T_1, F_1 \rangle\}; n := 1;$ 
  while esiste  $R_i \langle T_i, F_i \rangle \in \rho$  non in BCNF per la dipendenza  $X \rightarrow A$  do
    begin
       $n := n + 1;$ 
       $T' := X^+;$ 
       $F' := \pi_{T'}(F_i);$ 
       $T'' := T_i - (T' - X);$ 
       $F'' := \pi_{T''}(F_i);$ 
       $\rho := \rho - R_i \langle T_i, F_i \rangle + \{R_i \langle T', F' \rangle, R_n \langle T'', F'' \rangle\}$ 
    end;
  end ;

```

L'algoritmo ha complessità esponenziale, a causa del calcolo della proiezione delle dipendenze funzionali, e produce una decomposizione che preserva i dati per la seguente proprietà delle decomposizioni.

Teorema 5.10 Sia $\rho = \{R_1, \dots, R_m\}$ una decomposizione di $R \langle T, F \rangle$ che preserva i dati e sia $\sigma = \{S_1, S_2\}$ una decomposizione di R_1 che preserva i dati rispetto a $\pi_{T_1}(F)$. Allora anche la decomposizione $\rho = \{S_1, S_2, R_2, \dots, R_m\}$ preserva i dati rispetto a F .

Purtroppo non esiste alcuna garanzia che la decomposizione generata, oltre a preservare i dati, preservi anche le dipendenze, come mostra l'esempio seguente.

Esempio 5.12 Partiamo dalla relazione:

```

Telefoni \langle \{Prefisso, Numero, Località\},
          \{Prefisso, Numero → Località
            Località → Prefisso\} \rangle

```

Inizialmente, $\rho = \{\text{Telefoni}\}$. Nella prima iterazione viene scoperto che la dipendenza Località → Prefisso viola la BCNF, e quindi si rimpiazza ρ con $R_1 = \{\{\text{Località}, \text{Prefisso}\}, \{\text{Località} \rightarrow \text{Prefisso}\}\}$ e con $R_2 = \{\{\text{Numero}, \text{Località}\}\}$. Nella seconda iterazione, poiché le relazioni ottenute sono in BCNF, l'algoritmo termina. La scomposizione prodotta preserva i dati, ma la dipendenza Prefisso, Numero → Località va perduta. In termini concreti, supponiamo di avere assegnato, per sbaglio, lo stesso numero 504050 a due abbonati che risiedono in due diverse località Pisa e Pontedera con lo stesso prefisso 050. Questo errore verrebbe prevenuto da un sistema che gestisse la dipendenza funzionale Prefisso, Numero → Località, ma non viola nessuno dei vincoli relativi allo schema decomposto.

Questo esempio mostra anche che non possono esistere algoritmi di decomposizione in BCNF che preservano sempre le dipendenze. Infatti, una decomposizione in BCNF dello schema Telefoni non può mantenere tutti e tre gli attributi nella stessa relazione, per cui la dipendenza Prefisso, Numero → Località non appare nella proiezione di F sugli schemi della decomposizione. In queste condizioni, dato che né Prefisso né Numero, presi separatamente, determinano la località, la chiusura di Prefisso, Numero secondo le dipendenze proiettate non potrà contenere la località.

L'algoritmo di analisi presentato ha complessità esponenziale, a causa del costo dell'operazione di proiezione delle dipendenze. Sono stati proposti anche algoritmi polinomiali ([TF82]), ma non sono utilizzati nella pratica poiché producono schemi poco naturali ed eccessivamente decomposti.

Per essere certi di poter sempre ottenere uno schema normalizzato che preservi dati e dipendenze occorre abbandonare la BCNF e limitarsi ad una forma normale meno restrittiva, la *terza forma normale*.

5.4.3 Terza forma normale

La terza forma normale (3NF) è un criterio per valutare la qualità di uno schema che è meno restrittivo della BCNF, e che gode della seguente proprietà: ogni schema $R\langle T, F \rangle$ ammette una decomposizione che preserva i dati, preserva le dipendenze, ed è in 3NF; una tale decomposizione può essere ottenuta in tempo polinomiale. Lo svantaggio della 3NF sta nel fatto che, essendo meno restrittiva della BCNF, accetta anche schemi che presentano delle anomalie.

Come nel caso della BCNF, sono state date diverse definizioni equivalenti di 3NF. Una semplice definizione è:

Definizione 5.20 Uno schema $R\langle T, F \rangle$ è in 3NF se e solo se, per ogni dipendenza funzionale non banale $X \rightarrow A \in F^+$, allora X è una superchiave oppure A è primo.

Come si vede da questa definizione, se una relazione è in forma normale di Boyce-Codd allora è anche in terza forma normale.

Questa definizione non è utilizzabile in pratica per verificare se uno schema sia in 3NF a causa delle dimensioni esponenziali di F^+ . Esiste un teorema, analogo a quello visto per la BCNF, che indica la possibilità di effettuare il test in modo molto più efficiente, anche se non in tempo polinomiale.

Teorema 5.11 Uno schema $R\langle T, F \rangle$ è in 3NF se e solo se, per ogni dipendenza funzionale $X \rightarrow A_1, \dots, A_n \in F$, e per ogni $i \in \{1 \dots n\}$, $A_i \in X$ oppure X è una superchiave oppure A_i è primo.

Corollario 5.2 Uno schema $R\langle T, F \rangle$, con F copertura canonica, è in 3NF se e solo se, per ogni dipendenza funzionale $X \rightarrow A \in F$, allora X è una superchiave (ovvero una chiave) oppure A è primo.

In ogni caso, per stabilire se uno schema è in 3NF occorre conoscere gli attributi primi. Per questo motivo, dati gli analoghi risultati per la ricerca delle chiavi, vale la seguente:

Proposizione 5.1 Il problema di decidere se uno schema di relazione è in 3NF è NP-completo.

La terza forma normale, dato che ammette la presenza di dipendenze anomale, non elimina tutte le anomalie, come mostra l'esempio seguente.

Esempio 5.13 Consideriamo il seguente schema di relazione:

```
Telefoni⟨{Prefisso, Numero, Località, NomeAbbonato, Via},
          {Prefisso, Numero → Località,
           Prefisso, Numero → NomeAbbonato,
           Prefisso, Numero → Via,
           Località → Prefisso}⟩
```

che descrive gli abbonati al telefono. Le chiavi della relazione sono (Prefisso, Numero) e (Località, Numero). La relazione è in 3NF, ma non in BCNF, come si può facilmente verificare, e infatti esiste una ridondanza dovuta al fatto che ogni volta che si inserisce un nuovo numero telefonico di una certa località bisogna ripetere l'informazione sul prefisso.

5.4.4 Normalizzazione di schemi in 3NF

L'interesse per la terza forma normale deriva dal fatto che esiste un algoritmo semplice ed efficiente che permette di decomporre qualunque schema in un insieme di schemi di relazione in 3NF preservando dati e dipendenze. L'algoritmo si basa sulla seguente idea: dato un insieme di attributi T ed una copertura canonica G , si partiziona G in gruppi G_i tali che tutte le dipendenze in ogni G_i hanno la stessa parte sinistra. Quindi, da ogni G_i , si definisce uno schema di relazione composto da tutti gli attributi che vi appaiono, la cui chiave, detta *chiave sintetizzata*, è la parte sinistra comune.

Algoritmo Algoritmo *intuitivo* per la sintesi di schemi in 3NF

```

 Un insieme  $R$  di attributi e un insieme  $F$  di dipendenze su  $R$ .
 Una decomposizione  $\rho = \{S_i\}_{i=1..n}$  di  $R$  tale che
  $\rho$  preservi dati e dipendenze e ogni  $S_i$  sia in 3NF,
 rispetto alle proiezioni di  $F$  su  $S_i$ .
begin
{Passo 1} Trova una copertura canonica  $G$  di  $F$  e pon  $\rho = \{\}$ .
{Passo 2} Sostituisci in  $G$  ogni insieme  $X \rightarrow A_1, \dots, X \rightarrow A_h$  di dipendenze
         con lo stesso determinante, con la dipendenza  $X \rightarrow A_1 \dots A_h$ .
{Passo 3} Per ogni dipendenza  $X \rightarrow Y$  in  $G$ , metti uno schema con attributi  $XY$ 
         in  $\rho$ .
{Passo 4} Elimina da  $\rho$  ogni schema che sia contenuto in un altro schema di  $\rho$ .
{Passo 5} Se la decomposizione non contiene alcuno schema i cui attributi
         costituiscono una superchiave per  $R$ , aggiungi ad essa lo schema
         con attributi  $W$ , con  $W$  una chiave di  $R$ .
end
```

L'algoritmo è chiamato di *sintesi* perché opera raggruppando attributi per formare schemi. Vale la seguente proprietà.

Teorema 5.12 Dato uno schema $R(T, F)$, l'algoritmo elementare produce una decomposizione $\rho = \{S_i\}_{i=1..n}$ di R che preserva dati e dipendenze.

Dimostrazione

L'algoritmo produce una decomposizione perché tutti gli attributi di T appartengono allo schema generato; in particolare, se ci sono attributi che non partecipano a nessuna dipendenza, questi fanno parte di ogni chiave di R , per cui compaiono nella relazione aggiunta al passo 5. La decomposizione preserva le dipendenze poiché per ogni dipendenza $X \rightarrow A_i$ in G esiste uno schema che contiene XA_i . Infine, la decomposizione preserva i dati poiché, grazie al passo 5, soddisfa la condizione espressa dal Teorema 5.8. \square

La complessità dell'algoritmo dipende da quella del calcolo della copertura canonica, pari a $O(a^2 p^2)$.

Esempio 5.14 Si considerino gli attributi Prefisso, Numero e Località e le dipendenze Prefisso, Numero \rightarrow Località e Località \rightarrow Prefisso. Le dipendenze sono una copertura canonica e l'algoritmo elementare produce la decomposizione con schema $\{\text{Prefisso}, \text{Numero}, \text{Località}\}$. Notare che la decomposizione è in 3NF ma non in BCNF.

Esempio 5.15 Applicando solo i primi quattro passi dell'algoritmo elementare si produrrebbero decomposizioni che preservano le dipendenze ma non i dati. Si considerino gli attributi A, B, C e D e le dipendenze $A \rightarrow B$ e $C \rightarrow D$. Con i primi quattro passi si produce la decomposizione con schemi $R_1(A, B)$ e $R_2(C, D)$ che non preserva i dati. Applicando il passo 5 si aggiunge alla decomposizione anche lo schema $R_3(A, C)$ e la decomposizione preserva anche i dati.

Teorema 5.13 Dato uno schema $R\langle T, F \rangle$, l'algoritmo elementare produce una decomposizione $\rho = \{S_i\}_{i=1..n}$ di R in schemi in 3NF.

Dimostrazione

Supponiamo, per assurdo, che nella decomposizione ci sia uno schema S non in 3NF, con attributi $XA_1 \cdots A_h$, dove X è la “chiave sintetizzata”. X è una chiave di S : X implica tutti gli altri attributi perché $X \rightarrow A_i$ appartiene a G per ogni A_i , e non contiene attributi ridondanti perché altrimenti G non sarebbe una copertura canonica.

Poiché S non è in 3NF, esiste una dipendenza $V \rightarrow C$ in $(\pi_S G)^+$, e quindi in G^+ , in cui V non è una superchiave di S , C non è primo in S e $C \notin V$.

Poiché X è una chiave di S , si ha che $C \notin X$ e quindi $C = A_i$ per qualche i . Si prendono in esame due casi, a seconda che $V \subset X$ oppure no.

Se $V \subset X$, allora l'esistenza in G della dipendenza $X \rightarrow A_i$ è in contraddizione col fatto che G è una copertura canonica.

Se $V \not\subset X$, allora $V = YW$ per qualche $Y \subset X$ e $W \subseteq A_1 \cdots A_{i-1} A_{i+1} \cdots A_h$, con $W \neq \emptyset$. Sappiamo che la dipendenza $X \rightarrow A_i$ appartiene alla copertura canonica; raggiungiamo ora l'assurdo dimostrando invece che essa è ridondante e quindi può essere derivata dall'insieme $G' = G - \{X \rightarrow A_i\}$. A questo scopo mostriamo che $G' \vdash X \rightarrow YW$ e $G' \vdash YW \rightarrow A_i$.

$G' \vdash X \rightarrow YW$, segue dal fatto che $Y \subseteq X$, mentre W contiene solo attributi A_j con $j \neq i$, e le dipendenze $X \rightarrow A_j$ ($j \neq i$) appartengono a G' .

$G' \vdash YW \rightarrow A_i$: poiché $G \vdash X \rightarrow S$ mentre $G \not\vdash YW \rightarrow S$ ($YW = V$ non è una superchiave per S), si ha che $X \not\subseteq V_G^+$; quindi $V_{G'}^+ = V_G^+$, poiché nel calcolo della chiusura di V in G la dipendenza $X \rightarrow A_i$ non interviene. Avendo supposto $G \vdash YW \rightarrow A_i$, ciò implica che $G' \vdash YW \rightarrow A_i$. \square

L'algoritmo di sintesi produce una decomposizione $\rho = \{S_i\}_{i=1..n}$ di R tale che ρ preserva dati e dipendenze e ogni S_i è in 3NF, rispetto alle proiezioni di F su S_i , ma non fornisce per ogni S_i una copertura delle proiezioni di F su S_i .

Esempio 5.16 Sia $R = \{ABCD\}$ e $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow B\}$.

Le dipendenze sono una copertura canonica e l'algoritmo intuitivo produce la decomposizione con schemi:

- $R_1(ABC)$ con chiave sintetizzata AB ;
- $R_2(CD)$ con chiave sintetizzata C ;
- $R_3(DB)$ con chiave sintetizzata D .

La proiezione di F su R_2 è $\{C \rightarrow D\}$, su R_3 è $\{D \rightarrow B\}$, su R_1 è $\{AB \rightarrow C, C \rightarrow B\}$ e non $\{AB \rightarrow C\}$.

In [Ber76] è dato un altro algoritmo di sintesi di complessità $O(a^2 p^2)$ che produce una decomposizione con il minor numero possibile di schemi. Infatti, l'algoritmo intuitivo non ha questa proprietà, come mostrato nell'esempio che segue.

Esempio 5.17 Sia $R = \{ABCDEH\}$ e $F = \{EH \rightarrow A, EH \rightarrow D, CD \rightarrow E, CD \rightarrow H, AE \rightarrow B, BH \rightarrow C, C \rightarrow A\}$.

Le dipendenze sono una copertura canonica e l'algoritmo intuitivo produce la decomposizione con schemi:

- $R_1(EHAD)$ con chiave sintetizzata EH ;
- $R_2(CDEH)$ con chiave sintetizzata CD ;
- $R_3(AEB)$ con chiave sintetizzata AE ;
- $R_4(BHC)$ con chiave sintetizzata BH ;
- $R_5(CA)$ con chiave sintetizzata C .

La decomposizione è senza perdite di dati perché le chiavi di R sono EH , CD e BDH , con EH e CD chiavi sintetizzate di R_1 e R_2 .

L'algoritmo dato in [Ber76] produce invece una decomposizione con uno schema in meno:

- $R_1(EHCD)$ con chiavi sintetizzate EH e CD ;
- $R_2(AEB)$ con chiave sintetizzata AE ;
- $R_3(BHC)$ con chiave sintetizzata BH ;
- $R_4(CA)$ con chiave sintetizzata C .

L'idea usata per ridurre il numero degli schemi della decomposizione è di controllare dopo il passo 3 se esistono gruppi con determinanti X e Y equivalenti (cioè se vale $X \rightarrow Y$ e $Y \rightarrow X$); in tal caso si fondono i due gruppi in uno solo. Tuttavia non basta aggiungere solo questo passo all'algoritmo intuitivo per ottenere un algoritmo corretto, come mostrato nel seguito.

Alla fine del passo 3 si hanno i seguenti gruppi di dipendenze:

- $G_1 = \{EH \rightarrow A, EH \rightarrow D\};$
- $G_2 = \{CD \rightarrow E, CD \rightarrow H\};$
- $G_3 = \{AE \rightarrow B\};$
- $G_4 = \{BH \rightarrow C\};$
- $G_5 = \{C \rightarrow A\}.$

I determinanti EH e CD sono equivalenti, infatti:

- da $EH \rightarrow A$ e $AE \rightarrow B$ si deduce $EH \rightarrow B$;
- da $EH \rightarrow B$ e $BH \rightarrow C$ si deduce $EH \rightarrow C$;
- da $EH \rightarrow C$ e $EH \rightarrow D$ si deduce $EH \rightarrow CD$.

Pertanto si fondono G_1 e G_2 ottenendo i gruppi:

- $G_{12} = \{EH \rightarrow A, EH \rightarrow D, CD \rightarrow E, CD \rightarrow H\};$
- $G_3 = \{AE \rightarrow B\};$
- $G_4 = \{BH \rightarrow C\};$
- $G_5 = \{C \rightarrow A\}.$

Se si definissero gli schemi della decomposizione a partire da questi gruppi si otterebbe:

- $R_1(EHACD)$ con chiavi sintetizzate EH e CD ;
- $R_2(AEB)$ con chiave sintetizzata AE ;
- $R_3(BHC)$ con chiave sintetizzata BH ;
- $R_4(CA)$ con chiave sintetizzata C .

con R_1 non in forma normale per la dipendenza $C \rightarrow A$.

Per risolvere questo problema, viene previsto un altro passo che ha l'effetto di rimuovere l'attributo che provoca l'anomalia (nel nostro esempio A da R_1); questo attributo si trova sicuramente in un altro schema della decomposizione e si dimostra che la sua rimozione non altera le proprietà dell'algoritmo.

In conclusione, dovendo scegliere tra la 3NF e la BCNF, la strategia di controllare prima se esiste una decomposizione in BCNF che preservi le dipendenze e in caso negativo ripiegare sulla 3NF non può essere usata, perché richiede un algoritmo di complessità esponenziale. È compito del progettista quindi scegliere in anticipo fra i due tipi di normalizzazione: la 3NF, con un algoritmo di complessità polinomiale che produce decomposizioni che preservano i dati e le dipendenze, ma con la possibilità di ottenere schemi che contengono ancora delle anomalie, oppure la BCNF, con un algoritmo di complessità esponenziale (quello polinomiale ha scarso interesse pratico) che produce decomposizioni che non preservano necessariamente le dipendenze.

5.5 Dipendenze multivalore

La BCNF risolve tutte le anomalie connesse alle dipendenze funzionali. In una base di dati relazionali, d'altra parte, possono essere presenti altre anomalie non imputabili alle dipendenze funzionali. Consideriamo il seguente esempio:

Esempio 5.18 Consideriamo la relazione *Impiegati*(Nome, Figlio, Stipendio) che contiene informazioni relative ai figli e alla storia degli stipendi percepiti di un impiegato.

Una possibile estensione è la seguente:

<i>Impiegati</i>	<i>Nome</i>	<i>Figlio</i>	<i>Stipendio</i>
Bragazzi	Maurizio	100	
Bragazzi	Maurizio	120	
Bragazzi	Maurizio	140	
Bragazzi	Marcello	100	
Bragazzi	Marcello	120	
Bragazzi	Marcello	140	
Fantini	Maria	160	
Fantini	Maria	180	
Fantini	Maria	190	

La relazione dell'esempio precedente è in BCNF, infatti non vi sono dipendenze funzionali non banali, ma è presente una notevole ridondanza. Ci si può accorgere facilmente che una ridondanza di questo tipo si verifica tutte le volte che in una relazione si rappresentano proprietà multivalore indipendenti, come le proprietà figli a carico e storia dello stipendio degli impiegati.

Notare che le anomalie non dipendono esclusivamente dal fatto che esista una proprietà multivalore, ma dal fatto che una tale proprietà coesiste nello schema con altre proprietà semplici o multivalore indipendenti. Ad esempio, decomponendo lo schema in due sottoschemi in modo da modellare separatamente le proprietà multivalori indipendenti (come è stato fatto nell'algoritmo di trasformazione di uno schema ad oggetti in relazionale), si avrebbe una base di dati priva di anomalie:

<i>Stipendio</i> <i>Impiegati</i>	<i>Nome</i>	<i>Stipendio</i>
Bragazzi	100	
Bragazzi	120	
Bragazzi	140	
Fantini	160	
Fantini	180	
Fantini	190	

<i>Figli</i> <i>Impiegati</i>	<i>Nome</i>	<i>Figlio</i>
Bragazzi	Maurizio	
Bragazzi	Marcello	
Fantini	Maria	

Notare anche che se si cambia la semantica dell'attributo *Stipendio*, immaginando che non sia più lo stipendio percepito da un impiegato, ma quello percepito attualmente da ogni figlio, allora nello schema sarebbe modellata un'unica proprietà multivalore costituita da un insieme di coppie (*Figlio*, *Stipendio*), e non più due indipendenti, e lo schema sarebbe di nuovo privo di anomalie:

<i>Impiegati</i>	<i>Nome</i>	<i>Figlio</i>	<i>Stipendio</i>
Bragazzi	Maurizio	100	
Bragazzi	Marcello	120	
Fantini	Maria	180	

Per evitare le anomalie dovute alla coesistenza di proprietà multivalore indipendenti la teoria della normalizzazione vista finora è stata estesa nel modo seguente:

1. è stata definita una nuova dipendenza fra i dati, detta *dipendenza multivalore (MVD)*;
2. è stata estesa alle dipendenze multivalore la nozione di dipendenza derivata;
3. è stato dato un nuovo insieme di regole di inferenza corretto e completo per dipendenze funzionali e multivalore;
4. è stata estesa la nozione di decomposizione che preserva dati e dipendenze;
5. è stata data una nuova definizione di forma normale, la *quarta forma normale*, che generalizza la forma normale BCNF;
6. è stato dato un algoritmo di normalizzazione di schemi non in quarta forma normale che generalizza quello visto per BCNF, ed ha le stesse proprietà di conservare i dati ma non le dipendenze.

5.6 La denormalizzazione

L'eliminazione di ogni dipendenza anomala da uno schema va perseguita per produrre un buon schema concettuale ed un buon schema logico della realtà modellata.

Quando poi si passa ad effettuare la progettazione fisica, e quindi si prendono in considerazione i problemi legati all'efficienza dell'esecuzione delle applicazioni, può essere opportuno riintrodurre nello schema fisico qualche dipendenza anomala. Si immagini, ad esempio, una situazione in cui si deve gestire un archivio di studenti e di esami da loro superati, in cui gli aggiornamenti siano estremamente rari, vi sia a disposizione una grande quantità di spazio disco, e in cui si desideri effettuare la giunzione di uno studente e i suoi esami con la massima efficienza. In questo caso, è opportuno memorizzare le informazioni relative a studenti ed esami in un'unica relazione (*denormalizzazione*), evitando così la necessità di effettuare giunzioni, anche se a prezzo di un maggior costo delle operazioni di modifica, e di una maggiore occupazione di memoria.

È tuttavia di grande importanza non anticipare queste considerazioni dalla fase di progettazione fisica a quella di progettazione logica. Questo, non solo perché è importante che lo schema logico sia della massima qualità, ma anche perché, quando si produce lo schema fisico, ci possono essere dei mezzi migliori per ottenere lo stesso effetto. In particolare, quasi tutti i sistemi per la gestione di basi di dati permettono di memorizzare due diverse relazioni in modo *aggregato (clustered)*, senza modificare lo schema logico, ma solo indicando la volontà di aggregare le due relazioni all'interno dello schema fisico. La memorizzazione aggregata significa, nel nostro esempio, memorizzare ogni studente accanto a tutti gli esami da lui superati. Questa modalità di memorizzazione rende estremamente efficiente l'esecuzione della giunzione, a scapito dell'efficienza di altre operazioni (ad esempio, la scansione di tutti gli studenti), senza appesantire troppo l'occupazione di memoria.

La denormalizzazione è molto comune quando i dati sono gestiti usando non un DBMS ma un sistema di archiviazione oppure un foglio elettronico (*spreadsheet*). In questi casi, la denormalizzazione si utilizza non solo per ragioni di efficienza,

ma anche per velocizzare la produzione delle applicazioni, evitando la codifica di algoritmi di giunzione.

5.7 Uso della teoria della normalizzazione

L'uso delle nozioni definite in questo capitolo da parte di un progettista dipende dallo scopo che egli sta cercando di perseguire, distinguendo in particolare tra:

1. traduzione relazionale di un progetto concettuale ad oggetti;
2. analisi di una base di dati, o archivio, già esistente;
3. progettazione secondo l'approccio della relazione universale.

Nel primo caso, il progettista ha imparato, da questo capitolo, che, durante la progettazione concettuale (anche ad oggetti), le dipendenze anomale devono essere guardate con sospetto, chiedendosi se una dipendenza anomala $X \rightarrow Y$ non nasconde in realtà un'entità con attributi XY che non è stata individuata. La teoria sviluppata in questo capitolo gli permette anche di sapere che l'eliminazione totale delle dipendenze anomale può, in certi casi, comportare la perdita di alcune di esse. I risultati contenuti nel capitolo possono essere inoltre usati per dimostrare che, se si traduce uno schema ad oggetti privo di dipendenze anomale usando le regole definite nel Capitolo 4, si ottiene uno schema relazionale in BCNF.

La teoria qui sviluppata è molto più utile in una situazione in cui il progettista deve analizzare una base di dati, o un insieme di archivi, già esistente, per verificare la bontà dello schema. In questo caso, la tecnica di verificare, relazione per relazione o archivio per archivio, l'esistenza di dipendenze funzionali anomale è di decomporre lo schema secondo tali dipendenze, e di semplice applicazione e porta in genere a buoni risultati.

Chi, infine, decida di progettare un sistema seguendo l'approccio della relazione universale può sfruttare a pieno la teoria sviluppata in questo capitolo, ma deve tenere conto del fatto che questo approccio ha un potere espressivo minore rispetto all'utilizzo di un modello ad oggetti per la progettazione concettuale, e del fatto che uno schema realistico non può essere modellato senza utilizzare, accanto alle dipendenze funzionali, almeno le dipendenze multivалore.

5.8 Conclusioni

In questo capitolo sono stati presentati i principali risultati della teoria della normalizzazione. L'obiettivo di questa teoria è la costruzione di strumenti formali per la definizione di schemi di relazioni che non presentino anomalie. Per raggiungere questo obiettivo sono stati introdotti alcuni tipi di vincoli, le dipendenze fra i dati, che esprimono formalmente aspetti significativi dell'universo del discorso. Questi vincoli sono stati utilizzati per descrivere proprietà di schemi di relazioni senza anomalie (schemi normalizzati). Sono stati descritti alcuni algoritmi che permettono di trasformare schemi non normalizzati in schemi normalizzati, mantenendo, per quanto possibile, il significato originario del modello (decomposizioni che preservano i dati e decomposizioni che preservano le dipendenze). Sono stati inoltre discussi alcuni risultati negativi di tale teoria, dovuti sia alla intrinseca complessità computazionale di alcuni algoritmi, che è esponenziale, sia al fatto che è in generale impossibile decomporre uno schema in forma normale di Boyce-Codd preservando nello stesso tempo i dati e le dipendenze.

La teoria della normalizzazione non si limita a considerare le dipendenze funzionali e multivalore, come è stato fatto in questo capitolo, e ha studiato altri tipi di

dipendenze fra i dati (ad esempio, le dipendenze di giunzione, *join dependencies*, le dipendenze multivalore racchiuse, *embedded multivalued dependencies*), altre forme normali (ad esempio, 5NF e DKNF) ed altri interessanti problemi. Per approfondire questa tematica si rinvia ai testi specifici sull'argomento.

Esercizi

1. Usando gli assiomi di Armstrong si dimostri che:
 - (a) se $X \rightarrow YZ$, allora $X \rightarrow Y$ e $X \rightarrow Z$;
 - (b) se $X \rightarrow Y$ e $WY \rightarrow Z$, allora $XW \rightarrow Z$;
 - (c) se $X \rightarrow YZ$ e $Z \rightarrow BW$, allora $X \rightarrow YZB$.
2. Sia $F = \{A \rightarrow B, C \rightarrow B, D \rightarrow ABC, AC \rightarrow D\}$ un insieme di dipendenze funzionali. Trovare un insieme di dipendenze G in forma canonica equivalente a F .
3. Sia $R(T, F)$ uno schema relazionale. Dimostrare che se un attributo $A \in T$ non appare a destra di una dipendenza in F , allora A appartiene ad ogni chiave di R .
4. Sia $R(A, B, C, D, E)$ uno schema di relazione su cui siano definite le dipendenze funzionali:

$$F = \{AB \rightarrow CDE, AC \rightarrow BDE, B \rightarrow C, C \rightarrow B, C \rightarrow D, B \rightarrow E\}$$

Si richiede di:

- (a) portare F in forma canonica;
- (b) determinare le possibili chiavi;
- (c) mostrare che lo schema non è in terza forma normale;
- (d) portare lo schema in terza forma normale.
5. Mostrare (a) che se uno schema relazionale è in BCNF allora è anche in 3NF e (b) che se uno schema non è in 3NF allora non è neanche in BCNF.
6. Si consideri l'insieme di attributi $T = ABCDEGH$ e l'insieme di dipendenze funzionali $F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$. Per ognuno dei seguenti insiemi di attributi X , (a) trovare una copertura della proiezione di F su X , e (b) dire qual è la forma normale più forte soddisfatta da X :
 - (a) $X = ABC$;
 - (b) $X = ABCD$;
 - (c) $X = ABCEG$;
 - (d) $X = ABCH$;
 - (e) $X = ABCDEGH$.
7. Dire se è più costoso garantire il vincolo $A \rightarrow C$ su una relazione con schema $R(A, B, C)$ oppure su una base di dati con schema $R_1(A, B)$ ed $R_2(B, C)$.
8. Si dica sotto quali condizioni le regole date per trasformare uno schema grafico in uno schema relazionale producono schemi in BCNF.
9. Si rappresentino con uno schema relazionale i seguenti fatti riguardanti impiegati: codice fiscale, cognome, numeri del telefono (possono essere più di uno), nomi dei figli a carico (possono essere più di uno). Dare una decomposizione dello schema che sia in 3NF, ma non in 4NF e un'altra che sia in 4NF.

10. Si consideri la relazione $R(A, B, C, D)$ e si supponga che l'unica istanza possibile di R sia:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_1	c_1	d_3
a_2	b_1	c_3	d_4

Si trovi una copertura canonica delle dipendenze funzionali soddisfatte da R .

11. Si consideri il seguente schema relazionale:

Impiegati(Nome, Livello, Stipendio)

per il quale valgono le seguenti dipendenze funzionali

Nome \rightarrow Livello, Stipendio

Livello \rightarrow Stipendio

(a) Lo schema è in 3NF o in BCNF?

(b) Se lo schema non è in 3NF, si applichi l'algoritmo di sintesi per trovare una decomposizione che preservi dati e dipendenze.

(c) Se lo schema non è in BCNF, si applichi l'algoritmo di decomposizione per trovare uno decomposizione in BCNF e dire se conserva le dipendenze.

12. Si supponga che per archiviare dati sull'inventario delle apparecchiature di un'azienda sia stata usata una tabella con la seguente struttura:

Inventario(NInventario, Modello, Descrizione, NSerie, Costo, Responsabile, Telefono)

<i>NInventario</i>	<i>Modello</i>	<i>Descrizione</i>	<i>NSerie</i>	<i>Costo</i>
...
111	SUN3	Stazione Sun	ajk0785	25000
112	PB180	Notebook Mac	a908m	6000
113	SUN3	Stazione Sun	ajp8907	27000
...

<i>Responsabile</i>	<i>Telefono</i>
...	...
Caio	576
Tizio	587
Tizio	587
...	...

Il numero di inventario identifica un'apparecchiatura. Un'apparecchiatura ha un costo, un modello e un numero di serie. Apparecchiature dello stesso modello possono avere costi differenti, perché acquistate in momenti diversi, ma hanno la stessa descrizione. Il numero di serie è una caratteristica dell'apparecchiatura e due diverse apparecchiature dello stesso modello hanno numero di serie diverso. Ogni apparecchiatura ha un responsabile, che può avere più apparecchiature, ma un unico numero di telefono. I responsabili sono identificati dal cognome.

Definire le dipendenze funzionali e dire se lo schema proposto presenta anomalie, giustificando la risposta.

Trasformare la rappresentazione in uno schema relazionale in 3NF.

13. Una palestra ospita diversi corsi appartenenti a diverse tipologie (aerobica, danza moderna, ...). Ogni corso ha una sigla, che lo identifica, un insegnante e alcuni allievi. Un insegnante offre in generale più corsi, anche con diverse tipologie, e

anche un allievo può essere iscritto a più corsi. Di ogni insegnante interessano il nome (che lo identifica) e l'indirizzo. Di ogni allievo interessano il nome (che lo identifica) e il numero di telefono. Per ogni allievo interessa sapere, per ogni corso che frequenta, quanto ha già versato finora. La palestra gestisce attualmente i dati con un foglio elettronico con tante colonne quanti sono i fatti elementari da trattare.

Si chiede di:

- (a) Definire le dipendenze funzionali.
- (b) Dare una copertura canonica delle dipendenze in tale schema ed elencare le chiavi.
- (c) Applicare allo schema l'algoritmo di sintesi per portarlo in 3NF, e dire se lo schema così ottenuto è anche in BCNF.
- (d) Applicare allo schema l'algoritmo di decomposizione per portarlo in BCNF, e dire se tale decomposizione preserva le dipendenze.

14. Quali di questi test ammettono algoritmi di complessità polinomiale:

- (a) Dato lo schema di relazione $R\langle T, F \rangle$, $A \in T$, A è primo?
- (b) Dati due insiemi di dipendenze F e G , $F \equiv G$?
- (c) Dato lo schema di relazione $R\langle T, F \rangle$, e $X \subseteq T$, X è una superchiave?
- (d) Dato lo schema di relazione $R\langle T, F \rangle$, e $X \subseteq T$, X è una chiave?

15. Dato lo schema $R\langle T, F \rangle$, discutere la complessità dei seguenti problemi:

- (a) Trovare una chiave di R .
- (b) Trovare tutte le chiavi di R .
- (c) $F - \{X \rightarrow Y\} \equiv F$.

16. Discutere la complessità dei seguenti test:

- (a) *TEST 3NF*: dato lo schema di relazione $R\langle T, F \rangle$, R è in 3NF rispetto ad F ?
- (b) *TEST BCNF*: dato lo schema di relazione $R\langle T, F \rangle$, R è in BCNF rispetto ad F ?
- (c) *TEST BCNF DI SOTTOSCHEMA*: dato lo schema di relazione $R\langle T, F \rangle$, dato $X \subseteq T$, X è in BCNF rispetto alla proiezione di F su X ?
- (d) *TEST COPERTURA CANONICA*: dato lo schema di relazione $R\langle T, F \rangle$, F è in forma canonica?

17. Si supponga che una dipendenza funzionale $X \rightarrow Y$ sia soddisfatta da un'istanza di relazione r . Sia $s \subseteq r$ (quindi s è una relazione con gli stessi attributi di r). s soddisfa $X \rightarrow Y$? Se sì, dire perché, altrimenti dare un controsenso.

18. Si supponga che una dipendenza funzionale $X \rightarrow Y$ sia soddisfatta da due istanze di relazione r ed s con gli stessi attributi. $r \cap s$ soddisfa $X \rightarrow Y$? $r \cup s$ soddisfa $X \rightarrow Y$? Se sì, dire perché, altrimenti dare un controsenso.

19. Sia $R\langle T, F \rangle$ uno schema relazionale, con F una copertura canonica. Si dimostri che se lo schema ha una sola chiave ed è in 3NF allora è anche in BCNF. (Suggerimento: si inizi con lo scrivere la definizione di 3NF e di BCNF, e si dia un nome, ad esempio Y , all'insieme di attributi che forma la sola chiave di $R\langle T, F \rangle$. Si ragioni per assurdo, supponendo che $R\langle T, F \rangle$ sia in 3NF ma non in BCNF).

20. Dimostrare il Teorema 5.9.

21. Dimostrare il Teorema 5.11.

22. Sia $R\langle T, F \rangle$ uno schema relazionale. Dimostrare che se un attributo A di T appare a destra di qualche dipendenza in F , ma non appare a sinistra di alcuna dipendenza non banale, allora A non appartiene ad alcuna chiave.

Note bibliografiche

Un'introduzione alla teoria della normalizzazione è riportata in ogni libro sulle basi di dati. Una discussione approfondita della teoria può essere trovata in volumi specifici sull'argomento [AA93], [Mai83], [MR92], [AHV95].