6.5 Verifica della qualità di schemi relazionali

Il concetto di qualità per uno schema relazionale si riferisce alla possibilità di generare comportamenti anomali durante le operazioni di aggiornamento, dovuti alla presenza di dati ridondanti. Tali comportamenti possono essere evitati nel caso in cui lo schema relazionale soddisfi determinate proprietà, chiamate *forme normali*. Se uno schema non soddisfa le forme normali, è possibile, applicando la *teoria della normalizzazione*, generare un nuovo schema equivalente al precedente ma normalizzato. Questa teoria si basa sull'analisi di determinati vincoli di integrità noti come *dipendenze funzionali*.

È opportuno rimarcare che, se utilizziamo le metodologie di progettazione illustrate in questo capitolo e nel Capitolo 5, gli schemi relazionali generati spesso soddisfano una forma normale (in particolare, la *terza forma normale*, vedi Paragrafo 6.5.3.2) e quindi non presentano anomalie. In questo caso, la teoria della normalizzazione rappresenta un utile strumento di verifica, che può comunque suggerire modifiche nel caso in cui vogliamo ottenere forme normali differenti, magari più restrittive, come ad esempio la *forma normale di Boyce-Codd*. La normalizzazione è inoltre utile in fasi di revisione successiva degli schemi relazionali: vincoli di integrità sorti da nuovi requisiti applicativi possono infatti generare nuove anomalie, non presenti al momento della progettazione dello schema.

Nel seguito illustreremo dapprima le anomalie a cui gli schemi relazionali possono essere soggetti; dopodiché, descriveremo il processo di normalizzazione, evidenziandone le fasi principali.

6.5.1 Ridondanze ed anomalie

Le anomalie che vogliamo eliminare tramite il processo di normalizzazione dipendono dalla presenza di dati ridondanti in una base di dati relazionale. Per comprendere il problema, consideriamo la relazione `FilmInVideo` presentata nella Figura 6.15. Supponiamo che la chiave primaria della relazione sia `colloc`. Tale relazione contiene informazioni sui video e sui film ad essi corrispondenti. Queste informazioni, nello schema della Figura 6.14, sono rappresentate in due relazioni distinte: `Video` e `Film`. Osservando le tuple della relazione `FilmInVideo`, possiamo facilmente notare che le informazioni di ciascun film vengono ripetute per ogni video corrispondente a tale film. L'istanza della relazione contiene quindi valori *ridondanti*. Ad esempio, se esistessero 20 video per un certo film, i dati di questo film verrebbero ripetuti 20 volte, con conseguente spreco di spazio. A causa di tale ridondanza, si possono verificare anomalie quando si eseguono modifiche. In particolare:

- Supponiamo di voler modificare la valutazione del film "Pulp fiction" di Quentin Tarantino da 3.5 a 4. Tale valore deve essere modificato in tutte le tuple che rappresentano video relativi a tale film, quindi in tre tuple. Se non abbiamo la precauzione di modificare tale valore in tutte le tuple in cui appare il film da aggiornare, dopo la modifica, tuple relative allo stesso

`FilmInVideo`

colloc	titolo	regista	anno	genere	valutaz	tipo
1111	underground	emir kusturica	1995	drammatico	3.20	v
1112	underground	emir kusturica	1995	drammatico	3.20	d
1113	big fish	tim burton	2003	fantastico	3.10	v
1114	big fish	tim burton	2003	fantastico	3.10	d
1115	ed wood	tim burton	1994	drammatico	4.00	v
1116	pulp fiction	quentin tarantino	1994	thriller	3.50	v
1117	pulp fiction	quentin tarantino	1994	thriller	3.50	d
1118	pulp fiction	quentin tarantino	1994	thriller	3.50	d

Figura 6.15: Relazione con anomalie

film potrebbero avere valutazioni diverse. Questo problema viene chiamato *anomalia di modifica*.

- Supponiamo che tutti i video relativi a "Pulp fiction" di Quentin Tarantino abbiano dei problemi e debbano essere sostituiti. Le informazioni relative a tali video devono quindi essere cancellate dalla base di dati. In questo modo però, ogni informazione sul film "Pulp fiction" verrebbe persa. Potrebbe invece essere ragionevole mantenere tali informazioni sia per avere uno storico dei film proposti dalla videoteca sia per mantenere queste informazioni in previsione dell'arrivo di nuovi video relativi a questo film. Questo problema viene chiamato *anomalia di cancellazione*.

- Supponiamo infine di volere inserire l'elenco dei film per i quali verranno resi disponibili alcuni video a partire dal prossimo mese. Poiché la collocazione del video è attributo chiave della tabella `FilmInVideo`, le informazioni relative ad un film non possono essere inserite finché non è nota la collocazione di almeno un video relativo a quel film. Questo problema viene chiamato *anomalia di inserimento*.

Il problema legato alla presenza di anomalie dipende dall'avere utilizzato un'unica relazione per memorizzare informazioni relative a film e video, che rappresentano concettualmente due entità distinte che condividono un'associazione. È da notare infatti che tali anomalie non si verificano se utilizziamo le relazioni `Film` e `Video`, introdotte nel Paragrafo 6.4.3.

L'obiettivo della teoria della normalizzazione è, quindi, quello di:

- Identificare situazioni che possono generare anomalie, partendo dall'analisi di alcuni tipi di vincoli di integrità validi per lo schema di partenza. Tali vincoli sono noti come *dipendenze funzionali*.

- Identificare quali proprietà, rispetto alle dipendenze funzionali, uno schema deve soddisfare affinché non sia soggetto ad anomalie. Tali proprietà portano alla definizione di *forme normali* per gli schemi relazionali.

- Fornire strumenti per trasformare lo schema relazionale di partenza in uno schema equivalente al precedente (che rappresenta cioè le stesse informazioni) che soddisfi una forma normale e quindi non sia soggetto a determinati tipi di anomalie. Tale trasformazione prende il nome di *scomposizione* e prevede la sostituzione di una relazione che presenta anomalie con un insieme di relazioni, che non presentano tale problema. Il processo di scomposizione è guidato dalle dipendenze funzionali identificate.

È opportuno osservare che la normalizzazione, benché permetta di eliminare la ridondanza e quindi le anomalie, può portare ad inefficienze nelle prestazioni, nel caso in cui le interrogazioni richiedano frequentemente di combinare i dati suddivisi in relazioni distinte. Per esempio, lo schema presentato nella Figura 6.14 richiede l'esecuzione di un'operazione di join per determinare la valutazione del film corrispondente ad un certo video. Questo non è necessario se consideriamo la relazione `FilmInVideo` presentata nella Figura 6.15. Pertanto il processo di normalizzazione deve essere applicato con le dovute cautele.

6.5.2 Dipendenze funzionali

La presenza di ridondanza in uno schema, e quindi la possibilità di generare anomalie a fronte dell'esecuzione di operazioni di manipolazione, dipende da proprietà del dominio applicativo che possono essere rappresentate tramite vincoli di integrità chiamati *dipendenze funzionali*. Le dipendenze funzionali, come ogni altra tipologia di vincoli, devono essere identificate durante la fase di progettazione concettuale, a partire dal documento di specifica dei requisiti. Nel seguito introdurremo formalmente il concetto di dipendenza funzionale; discuteremo, quindi, aspetti connessi alla derivazione di dipendenze funzionali e chiariremo la relazione tra il concetto di dipendenza funzionale ed il concetto di chiave.

6.5.2.1 Concetti di base

Come suggerisce il nome, una dipendenza funzionale descrive un legame di tipo funzionale esistente tra gli attributi di una singola relazione. Consideriamo nuovamente la relazione `FilmInVideo` presentata nella Figura 6.15. Notiamo che, dati il titolo ed il regista di un film, l'anno di produzione, il genere e la valutazione assegnati al film sono unici. Quindi possiamo dire che gli attributi `titolo` e `regista` *determinano funzionalmente* i valori degli attributi `anno`, `genere` e `valutaz`. Il concetto di dipendenza funzionale può essere definito formalmente come segue.

Definizione 6.1 (Dipendenza funzionale) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Siano X ed Y sottoinsiemi di U_R.*[5] *Sia r un'istanza di $R(A_1, ..., A_n)$. r soddisfa $X \rightarrow Y$ se, per ogni coppia di tuple t_1 e t_2 in r, vale la seguente condizione:*

[5] Ricordiamo che, dato uno schema relazionale $R(A_1, ..., A_n)$, U_R denota l'insieme $\{A_1, ..., A_n\}$ (vedi Capitolo 2).

se $t_1[X] = t_2[X]$, *allora* $t_1[Y] = t_2[Y]$.

X determina funzionalmente Y in $R(A_1, ..., A_n)$, indicato con $X \rightarrow_{R(A_1,...,A_n)} Y$ (o con $X \rightarrow Y$, in assenza di ambiguità) se qualunque istanza r di $R(A_1, ..., A_n)$ soddisfa $X \rightarrow Y$. $X \rightarrow_{R(A_1,...,A_n)} Y$ ($X \rightarrow Y$) è chiamata dipendenza funzionale su U_R. ◇

Le dipendenze funzionali esprimono vincoli di integrità sulla base dei dati e come tali dipendono dalla semantica dei dati e devono essere identificate durante le fasi di progettazione.

Nel seguito utilizzeremo le seguenti notazioni:

- $A_1A_2 \ldots A_n$ denota l'insieme $\{A_1, A_2, \ldots, A_n\}$;
- dati X e Y insiemi di attributi, XY denota $X \cup Y$.

Esempio 6.14 Consideriamo l'istanza della relazione `FilmInVideo` presentata nella Figura 6.15. Questa istanza soddisfa, tra le altre, le seguenti dipendenze funzionali:

a. `titolo regista` → `anno`

b. `titolo regista` → `valutaz`

c. `titolo` `regista` → `regista`

d. `colloc` → `titolo` `regista`

e. `colloc` → `valutaz`

f. `titolo` → `regista`

Notiamo che la dipendenza funzionale `titolo` → `colloc` non vale per questa istanza, in quanto esistono tuple con lo stesso valore per `titolo` e diverso valore per `colloc`. □

Se analizziamo le dipendenze funzionali identificate nell'Esempio 6.14, osserviamo che:

- Alcune di queste dipendenze dipendono dalla semantica dei dati (dipendenze (a), (b) e (d)), sono quindi vincoli di integrità che devono essere determinati durante la fase di progettazione concettuale; ad esempio, la dipendenza (b) rappresenta il vincolo che stabilisce che, dato un titolo di film ed un regista, la valutazione assegnata è unica; tali vincoli devono essere soddisfatti da qualunque istanza della relazione.
- Alcune di queste dipendenze dipendono dalla specifica istanza considerata (dipendenza (f)) e non rappresentano quindi vincoli di integrità per il dominio della videoteca (in generale, possono infatti esistere film con lo stesso titolo ma regista diverso).

- Altre sono sempre valide perché dipendono dalla definizione di dipendenza funzionale (dipendenza (c)). Tali dipendenze vengono chiamate *dipendenze funzionali banali*.

- Altre ancora non sono fondamentali per descrivere la semantica dei dati perché sono implicate da altre dipendenze funzionali note (la dipendenza (e) si può ottenere dalle dipendenze (b) e (d)). Tali dipendenze vengono chiamate *dipendenze funzionali derivate*.

Nel seguito discuteremo quest'ultimo tipo di dipendenza funzionale.

6.5.2.2 Dipendenze funzionali derivate

Il concetto di dipendenza funzionale derivata da altre dipendenze può essere formalizzato tramite la nozione di *implicazione logica* di un insieme di dipendenze funzionali. L'implicazione logica permette di determinare l'insieme massimale di dipendenze funzionali derivabili a partire da un insieme di dipendenze funzionali dato. Tale insieme viene chiamato *chiusura* di un insieme di dipendenze funzionali ed è formalmente definito come segue.

Definizione 6.2 (Chiusura di un insieme di dipendenze funzionali) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Sia F un insieme di dipendenze funzionali su U_R. Sia $X \rightarrow Y$ una dipendenza funzionale su U_R. F implica logicamente $X \rightarrow Y$, denotato con $F \models X \rightarrow Y$, se qualunque istanza r di $R(A_1, ..., A_n)$ che soddisfa tutte le dipendenze in F, soddisfa anche $X \rightarrow Y$. La chiusura di F, denotata con F^+, è l'insieme delle dipendenze funzionali logicamente implicate da F. Pertanto, $F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$.* ◇

Esempio 6.15 Consideriamo l'insieme di dipendenze funzionali F, composto dalle dipendenze illustrate nell'Esempio 6.14. Dalla Definizione 6.2, è immediato dimostrare che $F \models$ `colloc` $\rightarrow$ `anno`, $F \models$ `colloc` $\rightarrow$ `titolo`, $F \models$ `colloc` $\rightarrow$ `regista`. Quindi, escludendo le dipendenze banali e ridondanti, $F^+ = F \cup$ {`colloc` $\rightarrow$ `anno`, `colloc` $\rightarrow$ `titolo`, `colloc` $\rightarrow$ `regista`}. □

Come vedremo più avanti, il calcolo di F^+ è fondamentale per applicare la teoria della normalizzazione. A questo proposito, è stato definito un insieme di regole di inferenza, note come *regole di Armstrong* (dal nome della persona che le ha definite), che permette di calcolare F^+. Ogni regola di Armstrong ha la forma:

Se G allora $X \rightarrow Y$

dove G è un insieme (eventualmente vuoto) di dipendenze funzionali (le premesse) e $X \rightarrow Y$ è una dipendenza funzionale (la conclusione). Intuitivamente, la regola stabilisce che, avendo a disposizione le premesse, è possibile asserire la conclusione. Più precisamente, le regole di Armstrong sono le seguenti (nel seguito D è un insieme di attributi):

- **A1: riflessività**. Sia $Y \subseteq X \subseteq D$. Allora $X \rightarrow Y$.
- **A2: additività**. Sia $Z \subseteq D$. Se $X \rightarrow Y$, allora $XZ \rightarrow YZ$.
- **A3: transitività**. Se $X \rightarrow Y$ e $Y \rightarrow Z$, allora $X \rightarrow Z$.

Notiamo che la regola di riflessività genera dipendenze funzionali banali, cioè dipendenze in cui la parte destra è contenuta nella parte sinistra.

Dalle regole di Armstrong è possibile derivare le tre seguenti regole aggiuntive. Tali regole non aumentano il potere espressivo del sistema di deduzione in quanto possono essere ottenute applicando le regole di base A1, A2 ed A3; tuttavia sono molto utili per inferire con pochi passi di derivazione nuove dipendenze funzionali.

- **A4: unione**. Se $X \rightarrow Y$ e $X \rightarrow Z$, allora $X \rightarrow YZ$.
- **A5: pseudotransitività**. Se $X \rightarrow Y$ e $WY \rightarrow Z$, allora $XW \rightarrow Z$.
- **A6: scomposizione**. Sia $Z \subseteq Y$. Se $X \rightarrow Y$, allora $X \rightarrow Z$.

Le regole di Armstrong possono ovviamente essere concatenate in modo che le conclusioni di un certo insieme di regole diventino le premesse di un'altra regola. Tale concatenazione prende il nome di *derivazione*.

Dato un insieme di dipendenze funzionali F, diciamo che $X \rightarrow Y$ è derivabile da F, indicato con $F \vdash X \rightarrow Y$, se esiste una derivazione le cui premesse sono contenute in F e la cui conclusione coincide con $X \rightarrow Y$. È possibile provare che le regole di Armstrong sono *corrette*, nel senso che ogni derivazione con premesse in F può derivare solo conclusioni in F^+, e *complete*, nel senso che ogni dipendenza funzionale in F^+ può essere derivata a partire da F, applicando tali regole. Dalla correttezza e completezza delle regole, segue il seguente risultanto.

Teorema 6.1 *Sia F un insieme di dipendenze funzionali su un insieme di attributi D. Sia $X \subseteq D$, $Y \subseteq D$. Allora, $F \vdash X \rightarrow Y$ se e solo se $F \models X \rightarrow Y$.* ◇

La computazione di F^+ è in generale molto costosa (esponenziale nel numero di dipendenze). Tuttavia, come vedremo nel seguito, spesso non è necessario calcolare F^+ ma è sufficiente stabilire se una certa dipendenza funzionale appartiene ad F^+. Per questo problema, esiste un semplice e più efficiente algoritmo basato sulla nozione di *chiusura di un insieme di attributi* rispetto ad un insieme di dipendenze funzionali. Questa chiusura contiene tutti gli attributi che possono essere derivati da quelli di partenza, utilizzando le dipendenze funzionali date.

Definizione 6.3 (Chiusura di un insieme di attributi) *Sia F un insieme di dipendenze funzionali su un insieme di attributi D. Sia $X \subseteq D$. La chiusura di X rispetto ad F, denotata con X^+, è l'insieme $\{A \in D \mid F \vdash X \rightarrow A\}$.* ◇

Dalla definizione di chiusura di un insieme di attributi, è immediato provare il seguente risultato.

```
Input
    D: insieme finito di attributi
    F: insieme di dipendenze funzionali su D
    X: insieme di attributi tale che X ⊆ D
Output
    X⁺: chiusura di X rispetto ad F
Metodo
Begin
    X(0) := X
    Repeat
        U := ⋃_{((Y→Z)∈F ∧ Y⊆X(i))} Z
        X(i+1) := X(i) ∪ U
    Until X(i+1) = X(i)
    Return X(i)
End
```

Figura 6.16: Algoritmo per il calcolo della chiusura di un insieme di attributi

Proposizione 6.1 *Sia F un insieme di dipendenze funzionali. $F \vdash X \rightarrow Y$ se e solo se $Y \subseteq X^+$, con X^+ calcolata rispetto ad F.* ◇

L'algoritmo presentato nella Figura 6.16 specifica un procedimento per il calcolo della chiusura di un insieme di attributi. Partendo dall'insieme di attributi iniziale X, l'algoritmo applica ciascuna dipendenza funzionale a tale insieme per determinare quali attributi sono determinati funzionalmente dagli attributi correntemente considerati. Il nuovo insieme di attributi così determinato viene unito all'insieme corrente ed utilizzato nell'iterazione successiva. Il ciclo procede finché l'insieme di attributi non viene più modificato. Notiamo che, poiché $X = X(0) \subseteq \ldots \subseteq X(i) \subseteq \ldots \subseteq D$ e D è finito, dopo un numero finito di passi si raggiunge necessariamente un valore per l'indice i tale che $X(i) = X(i+1)$. Questa condizione garantisce la terminazione dell'algoritmo.

Esempio 6.16 Consideriamo lo schema della relazione `FilmInVideo` ed il seguente insieme di dipendenze funzionali:

a. `titolo regista` → `anno`

b. `titolo regista` → `genere`

c. `titolo regista` → `valutaz`

d. `colloc` → `titolo regista`

e. `colloc` → `tipo`

Dato $X =$ `colloc`, vogliamo calcolare X^+, applicando l'algoritmo presentato nella Figura 6.16.

$X(0) =$ `colloc`

$X(1)$ determiniamo tutte le dipendenze funzionali aventi sul lato sinistro `colloc`. Ne esistono due (dipendenze (d) e (e)); pertanto $X(1) =$ `colloc tipo titolo regista`.

$X(2)$ Determiniamo tutte le dipendenze funzionali aventi sul lato sinistro qualche attributo in $X(1)$. Tutte le dipendenze soddisfano questa condizione ma solo tre (dipendenze (a), (b) e (c)) permettono di generare nuovi attributi; si ottiene $X(2) =$ `colloc tipo titolo regista anno genere valutaz`.

$X(3)$ Determiniamo tutte le dipendenze funzionali aventi sul lato sinistro qualche attributo in $X(2)$. Tutte le dipendenze soddisfano questa condizione ma non permettono di generare nuovi attributi. Quindi $X(3) = X(2) = X^+$. □

6.5.2.3 Chiavi

Il concetto di chiave, già introdotto nella trattazione del modello relazionale e successivamente discusso nell'ambito del linguaggio SQL (vedi Capitoli 2 e 3), può essere definito anche in termini di dipendenze funzionali. Questa definizione alternativa, utile, come vedremo nel seguito, per la determinazione delle forme normali, permette anche di verificare se le chiavi identificate durante la progettazione logica sono correttamente definite rispetto alle dipendenze funzionali. Una chiave può essere definita come segue.

Definizione 6.4 (Chiave) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Sia F un insieme di dipendenze funzionali su U_R. Sia X un sottoinsieme di U_R. X è chiave di $R(A_1, ..., A_n)$ se verifica le seguenti condizioni:*

1. *$X \rightarrow A_1 A_2 \ldots A_n \in F^+$.*
2. *Non esiste $Y \subset X$ tale che $Y \rightarrow A_1 A_2 \ldots A_n \in F^+$.* ◇

La condizione *(1)* richiede che la dipendenza di tutti gli attributi di R da X segua logicamente dalle dipendenze funzionali date. La condizione *(2)* rappresenta una condizione di minimalità analoga a quanto visto per la definizione di chiave nel modello relazionale.

Sulla base della Definizione 6.4, per determinare se un insieme di attributi X è chiave per uno schema di relazione R rispetto ad un insieme di dipendenze funzionali F, è necessario determinare se la dipendenza che ha X sul lato sinistro e tutti gli attributi di R sul lato destro è conseguenza logica di F. Questa verifica equivale a determinare se una data dipendenza funzionale $X \rightarrow Y$ è in F^+ e quindi, per la Proposizione 6.1, è equivalente a stabilire se $Y \subseteq X^+$. Osserviamo che, se un attributo non compare mai a destra di una dipendenza funzionale in F, questo attributo farà certamente parte della chiave in quanto, in caso contrario, non sarebbe possibile derivarlo applicando le regole di Armstrong.

Esempio 6.17 Consideriamo lo schema di relazione `R(C,S,Z)` con dipendenze funzionali $\{\texttt{CS} \rightarrow \texttt{Z}, \texttt{Z} \rightarrow \texttt{C}\}$. Le chiavi della relazione sono `CS` e `SZ`. Infatti $\texttt{CS}^+ =$

$\texttt{SZ}^+ = \texttt{CSZ}$. La condizione di minimalità è soddisfatta in quanto né `C` né `Z` né `S` permettono di derivare `CSZ`. □

Il seguente esempio illustra come utilizzare le dipendenze funzionali per verificare se le chiavi identificate durante la progettazione logica sono effettivamente tali.

Esempio 6.18 Consideriamo lo schema della relazione `FilmInVideo` e l'insieme di dipendenze funzionali F presentate nell'Esempio 6.16. Dall'analisi dei requisiti, sappiamo che `colloc` è chiave della relazione. Le dipendenze funzionali ci permettono di verificare che questo vincolo è corretto. Infatti, poiché nell'Esempio 6.16 abbiamo dimostrato che $\texttt{colloc}^+ = U_{\texttt{FilmInVideo}}$, siamo certi che `colloc` è effettivamente una chiave per la relazione `FilmInVideo`. La condizione di minimalità è soddisfatta in quanto non esistono sottoinsiemi propri che soddisfano la condizione di chiave. □

Le dipendenze funzionali vengono in aiuto anche quando le chiavi di una relazione non sono note e devono essere determinate utilizzando le dipendenze funzionali, come illustrato dal seguente esempio.

Esempio 6.19 Supponiamo di voler determinare le chiavi della relazione `FilmInVideo`. Per prima cosa, osserviamo che `colloc` non compare mai a destra in una dipendenza funzionale, quindi `colloc` farà certamente parte della chiave (altrimenti non potrebbe mai essere derivato a partire dalle dipendenze funzionali date). Poiché, inoltre, `colloc` determina funzionalmente tutti gli altri attributi, possiamo concludere che `colloc` è l'unico attributo chiave. Infatti, ogni altro insieme di attributi per essere chiave deve contenere `colloc`, ma in questo caso non soddisferebbe la condizione di minimalità. □

6.5.3 Forme normali

Le forme normali sono state introdotte per stabilire condizioni che, se soddisfatte da uno schema relazionale, garantiscono l'assenza delle anomalie discusse in precedenza. Esistono vari tipi di forme normali. Esse differiscono per il tipo di anomalie che permettono di evitare. Le più note sono la forma normale di Boyce-Codd e la terza forma normale; quest'ultima è forse la più conosciuta ed usata nella pratica. Tuttavia, la forma normale di Boyce-Codd permette di evitare un insieme più ampio di anomalie. Il tipo di forma normale da imporre in una base di dati è quindi una scelta importante, in quanto diversi tipi di forme normali hanno un diverso impatto sulla ridondanza e sulle anomalie, e quindi sulle prestazioni delle applicazioni per la base di dati sviluppata. Nel seguito, presenteremo la forma normale di Boyce-Codd e la terza forma normale, discutendone le differenze.

6.5.3.1 Forma normale di Boyce-Codd

Per comprendere l'idea alla base della forma normale di Boyce-Codd, consideriamo le seguenti due tipologie di dipendenze funzionali per lo schema della relazione `FilmInVideo`, introdotte nell'Esempio 6.16.

- Le dipendenze `titolo regista` → `anno`, `titolo regista` → `genere` e `titolo regista` → `valutaz` generano ridondanza, in quanto molti video possono corrispondere allo stesso film, quindi avere lo stesso titolo e stesso regista. Ne consegue che tutte queste tuple avranno anche lo stesso valore per `anno`, `genere` e `valutaz`.
- La dipendenza `colloc` → `titolo regista` non genera invece ridondanza. Infatti, essendo `colloc` chiave per la relazione, non è possibile che due tuple abbiamo lo stesso valore per `colloc`.

Dall'esempio precedente segue che una dipendenza funzionale $X \rightarrow Y$, con X chiave o super-chiave per la relazione considerata, non potrà mai generare ridondanze e quindi anomalie. La forma normale di Boyce-Codd viene definita a partire da questa considerazione come segue.

Definizione 6.5 (Forma normale di Boyce-Codd) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Sia F un insieme di dipendenze funzionali su U_R. $R(A_1, ..., A_n)$ è in forma normale di Boyce-Codd (BCNF) rispetto ad F se per ogni dipendenza funzionale $X \rightarrow Y \in F$, con $Y \not\subseteq X$, allora X è una chiave o super-chiave di R.* ◇

Dalla definizione precedente segue che uno schema di relazione R è in BCNF se tutte le dipendenze funzionali non banali hanno sul lato sinistro una chiave od una super-chiave di R.

Esempio 6.20 Consideriamo lo schema della relazione `FilmInVideo` della Figura 6.15 e le dipendenze funzionali presentate nell'Esempio 6.16. Dall'Esempio 6.19, sappiamo che l'unica chiave di tale relazione è `colloc`. Tale schema non è in BCNF in quanto le dipendenze `titolo regista` → `anno`, `titolo regista` → `genere` e `titolo regista` → `valutaz` non soddisfano la condizione della Definizione 6.5 poiché `titolo regista` non è chiave della relazione. Al contrario, i seguenti schemi di relazione:

`Video(`<u>`colloc`</u>`,tipo,titolo`$^{\texttt{Film}}$`,regista`$^{\texttt{Film}}$`)`
`Film(`<u>`titolo,regista`</u>`,anno,genere,valutaz)`

con dipendenze funzionali:

`colloc` → `tipo titolo regista`	per `Video`
`titolo regista` → `anno genere valutaz`	per `Film`

sono in BCNF, poiché le uniche dipendenze funzionali sono quelle che hanno la chiave sul lato sinistro. □

6.5.3.2 Terza forma normale

La BCNF impone una condizione molto forte. Se da un lato questa condizione permette di evitare ogni tipo di anomalia, in quanto evita ogni tipo di ridondanza, potrebbe non permettere di modellare alcuni vincoli rilevanti per il dominio applicativo (tutti quelli che a sinistra non contengono una chiave od una superchiave). Inoltre, come vedremo successivamente, se una relazione non soddisfa la BCNF non sempre è possibile decomporla per ottenere uno schema equivalente in BCNF per il quale valgano tutte le dipendenze funzionali di partenza. Tale condizione, invece, può essere soddisfatta richiedendo una forma normale più debole, nota come *terza forma normale*. Il prezzo da pagare è che schemi in terza forma normale possono ammettere una certa ridondanza, benché limitata.

La definizione di terza forma normale dipende dal concetto di attributo primo, definito come segue.

Definizione 6.6 (Attributo primo) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. A_i, $1 \leq i \leq n$, è un attributo primo per $R(A_1, ..., A_n)$ se A_i è elemento di una qualche chiave di R.* ◇

La terza forma normale ammette ridondanza solo se questa dipende dalla replicazione di valori per attributi primi. Può quindi essere definita come segue.

Definizione 6.7 (Terza forma normale) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Sia F un insieme di dipendenze funzionali su U_R. $R(A_1, ..., A_n)$ è in terza forma normale (3NF) rispetto ad F se per ogni dipendenza funzionale $X \rightarrow Y \in F$, con $Y \not\subseteq X$, allora: (i) X è una chiave o super-chiave di R; oppure (ii) $\forall A \in Y$, A è un attributo primo per $R(A_1, ..., A_n)$.* ◇

Notiamo che, in base alla precedente definizione, uno schema in BCNF è anche in 3NF. Al contrario, uno schema in 3NF permette di modellare un insieme più ampio di dipendenze funzionali rispetto ad uno schema in BCNF. Tale insieme è costituito da tutte le dipendenze che non contengono una chiave o una super-chiave a sinistra della dipendenza e che contengono solo attributi primi a destra.

Esempio 6.21 Consideriamo lo schema di relazione presentato nell'Esempio 6.17. Le chiavi sono `CS` e `SZ`, quindi tutti gli attributi sono primi. Ne consegue che lo schema è in 3NF. Tuttavia, poiché `Z` non è né chiave né super-chiave, lo schema non è in BCNF. □

Esempio 6.22 Lo schema della relazione `FilmInVideo` con le dipendenze funzionali presentate nell'Esempio 6.16 non è in 3NF, in quanto gli attributi `anno`, `genere` e `valutaz` non sono primi (l'unica chiave è `colloc` quindi l'unico attributo primo è `colloc`). Ne consegue che le dipendenze funzionali (a), (b) e (c) non soddisfano la 3NF. □

6.5.4 Scomposizione di schemi relazionali

Una volta scelta una forma normale, la base di dati ottenuta tramite la progettazione logica non necessariamente soddisfa le proprietà imposte da tale forma normale. È quindi necessario trasformare lo schema di ogni relazione in uno o più schemi di relazioni che soddisfino tale forma normale. Questa operazione è chiamata *scomposizione*.

La scomposizione di uno schema di relazione $R(A_1, A_2, \ldots, A_n)$ è la sua sostituzione con un insieme di schemi relazionali $\Sigma = \{R_1, R_2, \ldots, R_k\}$ tali che $U_R = U_{R_1} \cup U_{R_2} \cup \ldots \cup U_{R_k}$. Gli insiemi di attributi dei vari schemi R_i, $i = 1, ..., k$, non devono necessariamente essere disgiunti; pertanto uno stesso attributo di R può apparire in più di uno schema in Σ.

Nell'eseguire la scomposizione è importante garantire due proprietà, dette, rispettivamente, di *lossless join* (join senza perdite) e di *preservazione delle dipendenze funzionali*. Tali proprietà verranno discusse nel paragrafo seguente. Successivamente, presenteremo due algoritmi di scomposizione, uno per ogni forma normale introdotta.

6.5.4.1 Proprietà delle scomposizioni

Quando eseguiamo una scomposizione di uno schema relazionale in più schemi è necessario fare attenzione a non perdere informazioni. In particolare, se una relazione viene scomposta, è importante che sia possibile riottenere esattamente la stessa relazione eseguendo il join naturale delle relazioni in cui è stata scomposta. Questa proprietà, detta di lossless join (join senza perdite), è introdotta dalla definizione seguente.

Definizione 6.8 (Scomposizione lossless join) *Sia $R(A_1, ..., A_n)$ uno schema di relazione e sia F un insieme di dipendenze funzionali su U_R. Sia $\Sigma = \{R_1, R_2, \ldots, R_k\}$ una scomposizione di $R(A_1, ..., A_n)$. Σ è lossless join rispetto ad F se per ogni istanza r di $R(A_1, ..., A_n)$ che soddisfa F, vale:*

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_k}(r)$$

cioè r è il join naturale delle sue proiezioni sugli schemi R_i, $i = 1, \ldots, k$. ◇

Esempio 6.23 Consideriamo lo schema di relazione `R(A,B,C,D,E,G,H,I)` e l'insieme di dipendenze funzionali $F = \{\texttt{A} \rightarrow \texttt{HI}, \texttt{B} \rightarrow \texttt{DE}\}$. Le istanze di `R` non sono scomponibili senza perdite rispetto ad `ABDEI` ed `ACGH`. Infatti, consideriamo la seguente istanza r di `R`:

A	B	C	D	E	G	H	I
a1	b1	c1	d1	e1	g1	h1	i1
a1	b2	c2	d2	e2	g2	h1	i1

Consideriamo le proiezioni della relazione r sugli schemi con attributi, rispettivamente, `ABDEI` ed `ACGH`:

A	B	D	E	I
a1	b1	d1	e1	i1
a1	b2	d2	e2	i1

A	C	G	H
a1	c1	g1	h1
a1	c2	g2	h1

Se eseguiamo il join naturale delle proiezioni precedenti otteniamo la seguente relazione:

A	B	C	D	E	G	H	I
a1	b1	c1	d1	e1	g1	h1	i1
a1	b2	c2	d2	e2	g2	h1	i1
a1	b1	c2	d1	e1	g2	h1	i1
a1	b2	c1	d2	e2	g1	h1	i1

Come possiamo notare, la relazione ottenuta dall'operazione di join non è uguale ad r, quindi la scomposizione non è lossless join. □

Benché esista un algoritmo generale per determinare se una data scomposizione verifica la proprietà di lossless join, che non presentiamo per ragioni di spazio, il seguente teorema presenta un semplice test applicabile nel caso in cui la scomposizione contenga solo due schemi.

Teorema 6.2 *Sia $\Sigma = \{R_1, R_2\}$ una scomposizione per uno schema di relazione $R(A_1, ..., A_n)$. Sia F un insieme di dipendenze funzionali su U_R. Σ ha la proprietà di lossless join rispetto ad F se e solo se F implica logicamente una delle seguenti dipendenze funzionali: (i) $(U_{R_1} \cap U_{R_2}) \rightarrow (U_{R_1} - U_{R_2})$; (ii) $(U_{R_1} \cap U_{R_2}) \rightarrow (U_{R_2} - U_{R_1})$.* ◇

Dal teorema precedente è immediato dimostrare che, se gli schemi di una scomposizione sono disgiunti, la scomposizione non è lossless join.

Esempio 6.24 Consideriamo la relazione `R` e l'insieme di dipendenze F, presentati nell'Esempio 6.23. Le istanze di `R` sono scomponibili senza perdite rispetto ad `ABDEI` e `ABCGH`. Infatti:

- `ABDEI` $\cap$ `ABCGH` = `AB`;
- `ABDEI` $\setminus$ `ABCGH` = `DEI`;
- `AB` $\rightarrow$ `DEI` $\in F^+$. □

Esempio 6.25 Consideriamo la relazione `FilmInVideo`. La scomposizione di `FilmInVideo` nelle relazioni:

R_1(<u>colloc</u>,tipo,titolo^{R_2},regista^{R_2})
R_2(<u>titolo</u>,<u>regista</u>,anno,genere,valutaz)

soddisfa la proprietà di lossless join in quanto:

- $U_{R_1} \cap U_{R_2} = \{\texttt{titolo}, \texttt{regista}\}$;
- $U_{R_2} - U_{R_1} = \{\texttt{anno}, \texttt{genere}, \texttt{valutaz}\}$;

- `titolo regista` → `anno genere valutaz` vale nell'insieme iniziale di dipendenze funzionali.

La proprietà di lossless join non è invece verificata dalla scomposizione:

R_1(colloc,tipo)
R_2(titolo,regista,anno,genere,valutaz)

in quanto gli schemi sono disgiunti. □

Un'altra importante proprietà di una scomposizione di uno schema R in $\Sigma = \{R_1, \ldots, R_k\}$ è che l'insieme delle dipendenze funzionali F definito per R sia implicato dalle "proiezioni" di F sugli schemi $R_1, \ldots, R_k$; la nozione di proiezione permette di restringere un insieme di dipendenze funzionali ad un sottoinsieme di attributi dello schema per il quale sono state definite, come illustrato dalla seguente definizione.

Definizione 6.9 (Proiezione di insiemi di dipendenze funzionali) *Sia D un insieme di attributi. Sia F un insieme di dipendenze funzionali su D. Sia $Z \subseteq D$. La proiezione di F su Z, denotata con $\Pi_Z(F)$, è l'insieme $\{X \rightarrow Y | X \rightarrow Y \in F^+ \mid XY \subseteq Z\}$.* ◇

Una scomposizione $\Sigma = \{R_1, ..., R_k\}$ preserva un insieme di dipendenze funzionali F se l'unione di tutti gli insiemi di dipendenze $\Pi_{R_i}(F)$, per $i = 1, \ldots, k$, implica logicamente tutte le dipendenze in F, come formalizzato dalla definizione seguente.

Definizione 6.10 (Scomposizione che preserva le dipendenze) *Sia $R(A_1, ..., A_n)$ uno schema di relazione. Sia $\Sigma = \{R_1, \ldots, R_k\}$ una scomposizione per $R(A_1, ..., A_n)$. Sia F un insieme di dipendenze funzionali su U_R. Σ preserva le dipendenze in F se $\cup_{i=1,...,k} \Pi_{R_i}(F) \models F$.* ◇

La proprietà precedente assicura che, se sono verificate tutte le dipendenze funzionali "locali" ad ogni schema R_i, $i = 1, \ldots, k$, allora anche le dipendenze funzionali "non-locali", cioè le dipendenze che coinvolgono attributi di schemi diversi, sono verificate. Dato che le dipendenze funzionali sono degli importanti vincoli di integrità semantica, la proprietà di preservazione delle dipendenze assicura che se ogni relazione verifica i vincoli locali, automaticamente verifica i vincoli non locali.

Esempio 6.26 Consideriamo lo schema `R(C,S,Z)` con dipendenze funzionali $\{\mathtt{CS} \rightarrow \mathtt{Z}, \mathtt{Z} \rightarrow \mathtt{C}\}$. La scomposizione di `R` in $\mathtt{R}_1$`(S,Z)` ed $\mathtt{R}_2$`(C,Z)` ha la proprietà di lossless join; infatti $\{\mathtt{S}, \mathtt{Z}\} \cap \{\mathtt{C}, \mathtt{Z}\} \rightarrow \{\mathtt{C}, \mathtt{Z}\} - \{\mathtt{S}, \mathtt{Z}\}$.

D'altra parte la proiezione di $\{\mathtt{CS} \rightarrow \mathtt{Z}, \mathtt{Z} \rightarrow \mathtt{C}\}$ su $\mathtt{R}_1$`(S,Z)` genera solo le dipendenze funzionali banali che seguono dalla regola di riflessività (cioè $\mathtt{S} \rightarrow \mathtt{S}$ e $\mathtt{Z} \rightarrow \mathtt{Z}$), mentre la proiezione su $\mathtt{R}_2$`(C,Z)` genera le dipendenze funzionali banali più la dipendenza $\mathtt{Z} \rightarrow \mathtt{C}$. Quindi, la dipendenza $\mathtt{CS} \rightarrow \mathtt{Z}$ non è preservata poiché le dipendenze banali e la dipendenza $\mathtt{Z} \rightarrow \mathtt{C}$ non implicano la dipendenza $\mathtt{CS} \rightarrow \mathtt{Z}$. □

Esempio 6.27 Consideriamo nuovamente lo schema della relazione `FilmInVideo` e l'insieme di dipendenze funzionali F formato dalle dipendenze presentate nell'Esempio 6.16 e la dipendenza (f) `anno` $\rightarrow$ `tipo`, che esprime il fatto che il tipo di un video (vhs o dvd) dipende dall'anno di produzione del film corrispondente. Consideriamo adesso la seguente scomposizione:

$\mathtt{R_1}$ = `colloc titolo regista tipo`
$\mathtt{R_2}$ = `titolo regista anno genere valutaz`

È facile dimostrare che $\Pi_{\mathtt{R_1}}(F)$ contiene le dipendenze (d) ed (e) mentre $\Pi_{\mathtt{R_2}}(F)$ contiene le dipendenze (a), (b) e (c). Poiché non è possibile ottenere la dipendenza (f) a partire dalle dipendenze (a), (b), (c), (d) ed (e) (infatti, la chiusura di `anno` rispetto a queste dipendenze contiene solo l'attributo `anno`) questa scomposizione non preserva le dipendenze. □

La Definizione 6.10 fornisce anche un algoritmo per determinare se una scomposizione $\Sigma = \{R_1, \ldots, R_k\}$ preserva un insieme di dipendenze funzionali F. Tale algoritmo è basato però sul calcolo di F^+ per il calcolo delle proiezioni ed è quindi esponenziale nella dimensione dell'insieme delle dipendenze. Esistono tuttavia altri algoritmi polinomiali, che non presentiamo nel testo per motivi di spazio, basati sul concetto di chiusura di un insieme di attributi.

Un semplice approccio per il calcolo della proiezione $\Pi_{R_i}(F)$ consiste nell'osservare che, dato $X \subseteq R_i$, la dipendenza funzionale ottenuta da $X \rightarrow X^+$ (con X^+ calcolato rispetto ad F) eliminando gli attributi a destra che non compaiono in R_i appartiene a $\Pi_{R_i}(F)$. Inoltre, è facile dimostrare che ogni dipendenza funzionale che appartiene a $\Pi_{R_i}(F)$ può essere calcolata in questo modo. Il seguente esempio illustra l'utilizzo di questo approccio.

Esempio 6.28 Consideriamo lo schema di relazione `R(A,B,C,D)` e l'insieme di dipendenze funzionali $F = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{D}, \mathtt{D} \rightarrow \mathtt{A}\}$. Consideriamo adesso la scomposizione $\Sigma = \{\mathtt{AB}, \mathtt{BC}, \mathtt{CD}\}$. Per calcolare $\Pi_{\mathtt{AB}}(F)$ osserviamo che, calcolando le chiusure rispetto ad F, $\mathtt{A}^+ = \mathtt{B}^+ = \mathtt{ABCD}$, quindi, escludendo le dipendenze banali, otteniamo $\Pi_{\mathtt{AB}}(F) = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{A}\}$. Analogamente, possiamo dimostrare che $\Pi_{\mathtt{BC}}(F) = \{\mathtt{B} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{B}\}$ e $\Pi_{\mathtt{CD}}(F) = \{\mathtt{C} \rightarrow \mathtt{D}, \mathtt{D} \rightarrow \mathtt{C}\}$. Quindi, solo la dipendenza funzionale $\mathtt{D} \rightarrow \mathtt{A} \in F$ non è contenuta in $G = \Pi_{\mathtt{AB}}(F) \cup \Pi_{\mathtt{BC}}(F) \cup \Pi_{\mathtt{CD}}(F)$. Per stabilire se tale dipendenza è preservata dalla scomposizione, è sufficiente determinare se $G \models \mathtt{D} \rightarrow \mathtt{A}$. Poiché, calcolando la chiusura rispetto a G, $\mathtt{D}^+ = \mathtt{ABCD}$, anche $\mathtt{D} \rightarrow \mathtt{A}$ è preservata dalla scomposizione. □

6.5.4.2 Scomposizione in BCNF

Una questione importante nella scomposizione di uno schema di relazione in BCNF riguarda l'esistenza o meno di una tale scomposizione che abbia le proprietà di lossless join e preservazione delle dipendenze. È possibile dimostrare che qualsiasi schema di relazione ammette una scomposizione BCNF che soddisfa la proprietà di lossless join mentre è possibile che non esista, per un dato schema di relazione,

```
Input
    R(A_1, ..., A_n): schema di relazione
    F: insieme di dipendenze funzionali su U_R
Output
    Σ: scomposizione lossless join in BCNF di R(A_1, ..., A_n)
Metodo
Begin
    Σ := {R}
    calcola le chiavi di R
    While almeno uno schema S ∈ Σ non è in BCNF Do
        scegli X → Y ∈ Π_S(F) tale che X non è super-chiave di S e Y ⊈ X
        S_1 := X ∪ Y
        S_2 := S − Y
        calcola Π_{S_1}(F) e Π_{S_2}(F)
        calcola le chiavi di S_1 e S_2
        Σ := (Σ − S) ∪ {S_1, S_2}
    EndWhile
    Return Σ
End
```

Figura 6.17: Algoritmo per il calcolo di una scomposizione in BCNF

una scomposizione in BCNF che preservi le dipendenze. Il prezzo da pagare per ottenere una forma normale che elimini completamente le anomalie, come la BCNF, è quindi una potenziale perdita di vincoli di integrità.

L'algoritmo della Figura 6.17, dato uno schema di relazione ed un insieme di dipendenze funzionali, genera una scomposizione in BCNF di tale schema che verifica la proprietà di lossless join. Ad ogni passo, l'algoritmo scompone lo schema S in due schemi S_1 ed S_2 che rappresentano una scomposizione di S con proprietà di lossless join, in quanto soddisfano le ipotesi del Teorema 6.2. Poiché tale algoritmo si basa sul calcolo delle proiezioni di F, la sua complessità dipende dalla complessità dell'algoritmo utilizzato per il calcolo delle proiezioni. Il seguente esempio illustra un'applicazione dell'algoritmo presentato nella Figura 6.17.

Esempio 6.29 Consideriamo lo schema di relazione `R(C,S,Z)` con dipendenze funzionali $\{\texttt{CS} \rightarrow \texttt{Z}, \texttt{Z} \rightarrow \texttt{C}\}$. Le chiavi della relazione sono `CS` e `SZ`, quindi lo schema non è in BCNF. Consideriamo la dipendenza $\texttt{Z} \rightarrow \texttt{C}$ che viola la BCNF. Applicando l'algoritmo presentato nella Figura 6.17, scomponiamo `R` in due schemi $\texttt{S}_1(\texttt{Z}, \texttt{C})$ ed $\texttt{S}_2(\texttt{S}, \texttt{Z})$. È facile osservare che (escludendo le dipendenze banali) $\Pi_{\texttt{S}_1}(F) = \{\texttt{Z} \rightarrow \texttt{C}\}$ mentre $\Pi_{\texttt{S}_2}(F) = \emptyset$. Lo schema risultante è in BCNF, e quindi in 3NF, ma non preserva le dipendenze (vedi Esempio 6.26). □

Esempio 6.30 Consideriamo lo schema della relazione `FilmInVideo` con il seguente insieme di dipendenze funzionali F:

- `titolo regista` → `anno genere valutaz`

- `colloc` → `titolo regista tipo`

La chiave della relazione è `colloc`, quindi lo schema non è in BCNF. Per scomporre `FilmInVideo` in BCNF, consideriamo la dipendenza `titolo regista` → `anno genere valutaz` che viola la BCNF.

Applicando l'algoritmo presentato nella Figura 6.17, scomponiamo la relazione `FilmInVideo` in due schemi $\mathtt{R}_1(\mathtt{titolo}, \mathtt{regista}, \mathtt{anno}, \mathtt{genere}, \mathtt{valutaz})$ ed $\mathtt{R}_2(\mathtt{colloc}, \mathtt{titolo}, \mathtt{regista}, \mathtt{tipo})$.

Calcoliamo adesso le proiezioni di F su $\mathtt{R}_1$ ed $\mathtt{R}_2$, omettendo le dipendenze banali e quelle derivate:

- $\Pi_{\mathtt{R}_1}(F) = \{\mathtt{titolo\ regista} \rightarrow \mathtt{anno\ genere\ valutaz}\}$, con chiave (`titolo`, `regista`);
- $\Pi_{\mathtt{R}_2}(F) = \{\mathtt{colloc} \rightarrow \mathtt{titolo\ regista\ tipo}\}$, con chiave `colloc`.

Entrambe le relazioni sono in BCNF quindi la scomposizione, che in questo caso preserva le dipendenze, è data da $\{\mathtt{R}_1, \mathtt{R}_2\}$. Per definizione, tale scomposizione è anche in 3NF. □

6.5.4.3 Scomposizione in 3NF

Analogamente a quanto visto per la BCNF, esistono algoritmi per il calcolo della 3NF. L'algoritmo che vedremo può essere applicato solo sotto opportune ipotesi. Se queste ipotesi sono soddisfatte, l'algoritmo genera una scomposizione in 3NF lossless join che preserva le dipendenze.

La proprietà che l'insieme di dipendenze funzionali deve soddisfare è la *minimalità*; l'insieme non deve quindi contenere ridondanze. Un insieme di dipendenze funzionali si definisce minimale se valgono le seguenti proprietà:

1. **Unicità degli attributi a destra**. Il lato destro di ogni dipendenza in F corrisponde ad un singolo attributo.

2. **Assenza di dipendenze funzionali ridondanti**. Per nessuna dipendenza $X \rightarrow A \in F$, l'insieme di dipendenze $G = F - \{X \rightarrow A\}$ è equivalente ad F, cioè vale $F^+ = G^+$.

   Per determinare se questa condizione è soddisfatta, si calcola X^+ rispetto a $F - \{X \rightarrow A\}$; se A è in X^+, allora $F^+ = G^+$ e quindi $X \rightarrow A$ è ridondante.

3. **Assenza di attributi a sinistra ridondanti**. Per nessuna dipendenza $X \rightarrow A \in F$ e nessun sottoinsieme Z di X, l'insieme di dipendenze $G = F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ è equivalente ad F, cioè vale $F^+ = G^+$.

   In particolare, Z è ridondante in $X \rightarrow A$ se $(X - Z)^+$, calcolata rispetto ad F, contiene A.

Osserviamo che la condizione (1) richiede una sorta di forma semplificata per le dipendenze di un insieme minimale; per la regola di scomposizione è sempre possibile ricondursi a tale forma (vedi Paragrafo 6.5.2.2).

È possibile dimostrare che per ogni insieme di dipendenze funzionali F esiste un insieme di dipendenze funzionali minimale G tale che $F^+ = G^+$. L'insieme G può essere calcolato applicando le suddette condizioni in sequenza, considerando una dipendenza funzionale alla volta. Ovviamente, l'ordine in cui le varie dipendenze sono esaminate può generare insiemi minimali diversi.

Esempio 6.31 Consideriamo lo schema relazionale `R(A,B,C,D)` e l'insieme di dipendenze funzionali $F = \{\texttt{AB} \rightarrow \texttt{D}, \texttt{A} \rightarrow \texttt{B}, \texttt{A} \rightarrow \texttt{C}, \texttt{B} \rightarrow \texttt{C}, \texttt{C} \rightarrow \texttt{B}\}$. Vogliamo generare un insieme di dipendenze funzionali equivalente ad F ma minimale. Notiamo che ogni dipendenza a destra contiene un solo attributo. Quindi partiamo con l'analisi delle dipendenze ridondanti. Consideriamo la dipendenza $\texttt{A} \rightarrow \texttt{C}$. Notiamo che $\texttt{A}^+$ calcolato rispetto ad $F - \{\texttt{A} \rightarrow \texttt{C}\}$ è `ABCD`, quindi la dipendenza $\texttt{A} \rightarrow \texttt{C}$ è ridondante e può essere eliminata. Sia $F_1 = F - \{\texttt{A} \rightarrow \texttt{C}\}$. Se applichiamo lo stesso ragionamento a tutte le dipendenze rimanenti, notiamo che nessun'altra dipendenza può essere eliminata. Determiniamo adesso gli eventuali attributi ridondanti a sinistra. Consideriamo la dipendenza $\texttt{AB} \rightarrow \texttt{D}$. Poiché $\texttt{A}^+$ calcolata rispetto ad F_1 equivale ad `ABCD`, `B` può essere eliminata da $\texttt{AB} \rightarrow \texttt{D}$. L'insieme di dipendenze minimale è quindi $F_2 = \{\texttt{A} \rightarrow \texttt{D}, \texttt{A} \rightarrow \texttt{B}, \texttt{B} \rightarrow \texttt{C}, \texttt{C} \rightarrow \texttt{B}\}$.

Un risultato diverso sarebbe stato ottenuto se, anziché considerare la dipendenza $\texttt{A} \rightarrow \texttt{C}$ (risultata ridondante e quindi eliminata da F), avessimo considerato la dipendenza $\texttt{A} \rightarrow \texttt{B}$. Poiché $\texttt{A}^+$ calcolato rispetto ad $F - \{\texttt{A} \rightarrow \texttt{B}\}$ coincide con `ABCD`, la dipendenza $\texttt{A} \rightarrow \texttt{B}$ è ridondante e può essere eliminata. Dopo questa eliminazione, non ci sono altre dipendenze ridondanti. Sia $F_3 = F - \{\texttt{A} \rightarrow \texttt{B}\}$. È facile determinare che anche in questo caso l'attributo `B` è ridondante in $\texttt{AB} \rightarrow \texttt{D}$ e può essere eliminato. Quindi, l'insieme $F_4 = \{\texttt{A} \rightarrow \texttt{D}, \texttt{A} \rightarrow \texttt{C}, \texttt{B} \rightarrow \texttt{C}, \texttt{C} \rightarrow \texttt{B}\}$ è un altro insieme di dipendenze minimale equivalente ad F. □

A partire da un insieme di dipendenze funzionali minimale, la scomposizione in 3NF può essere calcolata applicando l'algoritmo nella Figura 6.18. Come già anticipato, è possibile dimostrare che le scomposizioni risultanti sono lossless join e preservano le dipendenze.

Esempio 6.32 Consideriamo lo schema presentato nell'Esempio 6.31 e l'insieme di dipendenze minimale F_2. Poiché $\texttt{A}^+ = \texttt{ABCD}$, `A` è chiave. Inoltre, poiché A non compare mai a destra, non possono esistere altre chiavi. Lo schema non è quindi in 3NF in quanto, ad esempio, nella dipendenza $\texttt{C} \rightarrow \texttt{B}$, `C` non è né chiave né super-chiave e `B` non è primo. Generiamo una scomposizione in 3NF dello schema come segue:

1. Generiamo gli schemi $\texttt{R}_1\texttt{(A,D)}$, $\texttt{R}_2\texttt{(A,B)}$ ed $\texttt{R}_3\texttt{(B,C)}$.
2. Gli schemi $\texttt{R}_1\texttt{(A,D)}$ e $\texttt{R}_2\texttt{(A,B)}$ possono essere combinati ottenendo il nuovo schema $\texttt{R}_{1,2}\texttt{(A,B,D)}$.

```
Input
    R(A_1, ..., A_n): schema di relazione
    F: insieme minimale di dipendenze funzionali su U_R
Output
    Σ: scomposizione lossless join in 3NF di R(A_1, ..., A_n), che preserva le dipendenze
Metodo
Begin
    i := 0; Σ := ∅
    For ogni dipendenza funzionale X → B ∈ F Do
        i := i + 1
        R_i := XB
        Σ := Σ ∪ {R_i}
    EndFor
    siano R_h = XB_h e R_k = XB_k, R_h, R_k ∈ Σ
        ottenuti al passo precedente da dipendenze X → B_h e X → B_k
    Σ := (Σ − {R_h, R_k}) ∪ {XB_hB_k}
    If nessuno schema R_j ∈ Σ, 1 ≤ j ≤ i, contiene una chiave di R Then
        i := i + 1
        R_i := una qualsiasi chiave di R
        Σ := Σ ∪ {R_i}
    EndIf
    For ogni R_h ∈ Σ Do
        For ogni R_k ∈ Σ, h ≠ k, tale che U_{R_k} ⊂ U_{R_h} Do
            Σ := Σ − {R_k}
        EndFor
    EndFor
    Return Σ
End
```

Figura 6.18: Algoritmo per la scomposizione in 3NF

3. Controlliamo che esista almeno uno schema che contiene una chiave. Lo schema esiste ed è $\mathtt{R}_{1,2}$.

4. Poiché non esistono relazioni di contenimento tra gli schemi, la scomposizione non viene ulteriormente modificata.

Gli schemi ottenuti al passo (2) rappresentano pertanto una scomposizione lossless join in 3NF della relazione iniziale, che preserva le dipendenze. È facile dimostrare che questa scomposizione è anche in BCNF.

Supponiamo adesso di applicare l'algoritmo per il calcolo della scomposizione in BCNF, partendo dalla dipendenza $\mathtt{C} \rightarrow \mathtt{B}$, che viola la condizione di BCNF. Otteniamo gli schemi $\mathtt{R}_1\mathtt{(B,C)}$ e $\mathtt{R}_1\mathtt{(A,D,C)}$ tali che:

- $\Pi_{\mathtt{R}_1}(F) = \{\mathtt{B} \rightarrow \mathtt{C}, \mathtt{C} \rightarrow \mathtt{B}\}$, con chiavi B e C;
- $\Pi_{\mathtt{R}_2}(F) = \{\mathtt{A} \rightarrow \mathtt{D}, \mathtt{A} \rightarrow \mathtt{C}\}$, con chiave A.

Gli schemi sono in BCNF e le dipendenze sono preservate. Infatti, l'unica dipendenza che appartiene ad F_2 e non compare nelle proiezioni è $\mathtt{A} \rightarrow \mathtt{B}$. Tuttavia,

$\texttt{A}^+$, calcolato rispetto all'unione delle proiezioni, coincide con `ADBC`. Quindi, anche `A` $\rightarrow$ `B` è preservata. □

Note conclusive

La progettazione logica, insieme alla normalizzazione, permette di generare schemi relazionali di qualità, a partire da schemi ER. Come abbiamo già ribadito, se utilizziamo le metodologie di progettazione basate sui diagrammi ER (vedi Capitolo 5), gli schemi relazionali ottenuti sono già in 3NF. Il motivo per cui tali metodologie generano dei "buoni" schemi è che in realtà gli schemi non buoni sono quelli in cui viene usata un'unica relazione per rappresentare entità ed associazioni distinte. Poiché tali metodologie generano una relazione per ogni tipo di entità e per ogni associazione, gli schemi ottenuti sono corretti ed aderiscono al semplice principio di rappresentare ogni tipo di entità e di associazione con uno schema relazionale apposito. Per esempio, la relazione `FilmInVideo` può essere generata solo da uno schema in cui esiste una singola entità che contiene sia gli attributi dell'entità `Film` sia gli attributi dell'entità `Video`. L'applicazione corretta delle metodologie di progettazione riduce, quindi, la necessità di ricorrere alla normalizzazione per migliorare lo schema relazionale ottenuto. Tuttavia, la normalizzazione continua a rappresentare un utile strumento di verifica di qualità in quanto:

- non sempre le basi di dati esistenti sono state progettate seguendo i criteri illustrati in questo capitolo e nel Capitolo 5;
- lo schema di una base di dati può evolvere nel tempo; a questa evoluzione non sempre si affianca un raffinamento della documentazione. Modifiche applicate a schemi normalizzati, se effettuate con scarsa attenzione, possono portare alla generazione di ridondanze.

In entrambi i casi, la normalizzazione permette di identificare eventuali ridondanze e di scomporre lo schema, riducendo il verificarsi di potenziali anomalie. È inoltre un utile strumento per verificare la scelta degli identificatori, e di conseguenza delle chiavi, determinate nelle fasi di progettazione concettuale e logica.

La teoria della normalizzazione è stata ampiamente studiata. Oltre alle dipendenze funzionali introdotte in questo capitolo, esistono molti altri tipi di dipendenze funzionali che permettono di identificare nuove forme di ridondanza e di definire nuove forme normali. Tra gli ulteriori tipi di dipendenze considerate in letteratura, ricordiamo le dipendenze di inclusione e le dipendenze multi-valore. Tra le forme normali non considerate in questo capitolo, alcune, come la seconda forma normale (2NF), sono più deboli rispetto alla terza forma normale nel senso che ammettono più tipi di ridondanza; altre, come la quarta e la quinta forma normale, vengono definite considerando insiemi più ampi di dipendenze e permettono di restringere ulteriormente l'insieme delle potenziali anomalie. È importante notare tuttavia che la forma normale più usata nella pratica è la 3NF.