Teoria delle dipendenze

Gabriele Frassi

Indice

1 Qualità delle relazioni

Abbiamo progettato il nostro sistema attraverso un linguaggio concettuale, abbiamo utilizzato il modello E-R, abbiamo verificato le ridondanze e l'efficienza ottenendo alla fine il modello logico, cioè una serie di schemi di relazione. Adesso dobbiamo verificare la *qualità delle relazioni*, cioè se presentano caratteristiche utili per il mantenimento della base di dati.

Obiettivo Valutare la qualità della progettazione di schemi relazionali, capire se un raggruppamento di attributi in uno schema di relazione sia migliore di un altro.

Approccio adottato *top-down.*

Obiettivi impliciti

- Conservazione dell'informazione, cioè il mantenimento di tutti i concetti espressi mediante il modello concettuale, inclusi tipi di attributi, tipi di entità e tipi di associazioni.
- La minimizzazione delle ridondanze, cioè l'evitare la memorizzazione ripetuta della stessa informazione e quindi la necessità di effettuare molteplici aggiornamenti al fine di mantenere la consistenza tra le diverse copie della medesima informazione.

Linee guida

- **Semplice è bello**! Uno schema di relazione deve essere progettato in modo da rendere semplice la spiegazione del suo significato. Non devo raggruppare attributi provenienti da più tipi di entità e tipi di relazione in un'unica relazione. Se uno schema corrisponde a un solo tipo di entità o a un solo tipo di relazione risulta semplice spiegarne il significato, evitando così l'emergere di ambiguità semantiche.
- **Niente anomalie**! Gli schemi vanno progettati in modo da evitare anomalie nell'inserimento, nella cancellazione o nella modifica. Se possono presentarsi anomalie vanno rilevate e dobbiamo fare in modo che i programmi legati alla base di dati operino in modo corretto.
- **Evitare frequenti valori nulli**! Evitare di porre in una relazione attributi i cui valori possono essere frequentemente nulli. Se questi sono inevitabili ci si assicuri che si presentino solo in casi eccezionali rispetto al numero di tuple di una relazione.

Esempio 1 Prendiamo la seguente relazione

`Fattura(CodF, CodProd, TotDaPagare, CostoNettoProd, IVA)`

Osservo che:

- Il codice della fattura (CodF) determina il prodotto comprato, quindi il codice del prodotto (CodProd) e il costo totale (TotDaPagare)
- Il codice del prodotto (CodProd) determina il costo netto (CostoNettoProd) e l'iva da pagare (IVA)
- CostoNettoProd e IVA determinano quanto bisogna pagare (TotDaPagare)

Devo mantenere il tutto consistente. TotDaPagare è determinato da altri attributi, per evitare anomalie conviene rimuoverlo e calcolare il costo totale direttamente nella query.

Esempio 2 Prendiamo la seguente relazione

`Anagrafe(CF, NomeP, Indirizzo, NomeC, NumAb)`

Osservo che :

- Il codice fiscale (CF) determina il nome della persona (NomeP), l'indirizzo (Indirizzo) e il nome della città (NomeC)
- Il nome della città (NomeC) determina il numero di abitanti (NumAb)

Il numero di abitanti è ripetuto tante volte quanti sono i residenti, questo valore deve essere mantenuto consistente (cioè uguale) per ogni persona di una certa città. Risulta ovvio che calcolare per ogni persona il numero di abitanti sia alquanto pesante. **Come risolviamo?** Trasformiamo Anagrafe in due schemi separati (con vincolo di integrità referenziale su NomeC e un vincolo aggiuntivo su NumAb)

```
Persona(CF, NomeP, Indirizzo, NomeC)
Residenza(NomeC, NumAb)
```

1.1 Approccio formale

Abbiamo visto che ci sono attributi che dipendono da altri attributi, devo verificare se ci sono troppe dipendenze tra gruppi di attributi all'interno di una tabella. A tal proposito parliamo di **dipendenze funzionali**, che esprimono legami semantici tra due gruppi di attributi di uno schema di relazione.

- Una dipendenza funzionale è una proprietà dello schema di relazione, non di un particolare stato valido
- Una dipendenza funzionale non può essere dedotta a partire da uno stato valido, ma deve essere definita esplicitamente da qualcuno che conosce la semantica degli attributi (quindi non determino proprietà osservando i valori delle istanze, potrei confondere proprietà valide solo per quell'istanza con proprietà dello schema di relazione)

Definizione Consideriamo la relazione r su $R(X)$ e due sottoinsiemi non vuoti Y e Z di X. Si dice che esiste in r una dipendenza funzionale da Y a Z se, per ogni coppia di ennuple t_1 e t_2 di r con gli stessi valori su Y risulta che t_1 e t_2 hanno gli stessi valori anche su Z.

Notazione La notazione adottata è la seguente:

$$Y \longrightarrow Z$$

Attenzione Non è detto che se io ho $Y \longrightarrow Z$ abbia il contrario, cioè $Z \longrightarrow Y$

Forma normale Attraverso le dipendenze funzionali esprimo tutto quello che so riguardante la base di dati. In base a queste dipendenze vorrei costruirmi una caratteristica dello schema per cui valgono quelle dipendenze: ciò significa che se quello schema presenta quelle caratteristiche allora si ha consistenza.
Questo schema è detto **forma normale**, che garantisce l'assenza di particolari difetti dallo schema (determina un livello di qualità). In base alle proprietà che ho dato posso determinare se lo schema si trova in una certa forma normale o in nessuna.

Normalizzazione Se applico a un certo schema la definizione di forma normale e individuo che questo non è in tale forma, allora compio un processo di *normalizzazione*. La normalizzazione è utilizzata come tecnica di verifica dei risultati della progettazione, non costituisce metodologia di progetto.

Esempio 3 Vediamo la seguente tabella contenente le informazioni di una certa azienda

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

Abbiamo i cognomi, gli stipendi, i progetti con il loro bilancio e la funzione nel progetto di ciascun dipendente. Ogni impiegato può partecipare a più progetti, sempre con lo stesso stipendio, e con una sola funzione per progetto. Neri, per esempio, lavora su tre progetti differenti.
Potrei incorrere nelle seguenti anomalie effettuando modifiche in modo scorretto:

- **Anomalia di aggiornamento**: se lo stipendio di un impiegato varia è necessario andarne a modificare il valore più tuple
- **Anomalia di cancellazione**: se un impiegato si licenzia dobbiamo cancellarlo in diverse tuple (quindi rimuoverlo da tutti i progetti)
- **Anomalia di aggiornamento**: se un progetto varia il suo bilancio devo modificare l'attributo relativo in tutte le tuple dei dipendenti coinvolti nel progetto.

La causa sta, ovviamente, nella ripetizione dello stipendio di un impiegato e del bilancio di un progetto. L'errore è stato compiuto proprio nella fase di progettazione: abbiamo usato un'unica relazione per rappresentare gruppi di informazioni eterogenee. Trasformiamo le informazioni dette prima in dipendenze

- Ogni impiegato ha un solo stipendio: Impiegato $\longrightarrow$ Stipendio
- Ogni progetto ha un solo bilancio: Progetto $\longrightarrow$ Bilancio
- Ogni impiegato ha una sola funzione per progetto: Impiegato Progetto $\longrightarrow$ Funzione

Osservando la tabella individuiamo pieno rispetto di queste dipendenze. **Ma ne esistono anche altre?** Abbiamo le cosiddette **dipendenze banali**, che sono sempre soddisfatte (proprietà ovvie di una relazione). Si osserva

$$Impiegato\ Progetto \longrightarrow Progetto$$

Si dice che:

- $Y \to A$ è non banale se A non compare tra gli attributi di Y
- $Y \to Z$ è non banale se nessun attributo in Z appartiene a Y.

Legame tra dipendenze e anomalie Osserviamo che le prime due dipendenze comportano ripetizioni, mentre la terza no. Impiegato e Progetto non sono chiavi, Impiegato Progetto è chiave! La relazione contiene alcune informazioni legate alla chiave e altre ad attributi che non lo sono.

1.2 Teoria delle dipendenze

1.2.1 Implicazione

Sia F un insieme di dipendenze funzionali definite su $R(Z)$ e sia $X \longrightarrow Y$.

- Si dice che F implica la dipendenza $X \to Y$ ($F \Rightarrow X \to Y$) se per ogni istanza r di R che verifica tutte le dipendenze in F, risulta verificata anche $X \to Y$.
- Si dice anche che $X \to Y$ è implicata da F

1.2.2 Chiusura

Dato un insieme di dipendenze funzionali F definite su $R(Z)$ la chiusura di F è l'insieme di tutte le dipendenze funzionali implicate da F.

$$F^+ = \{X \to Y | F \Rightarrow X \to Y\}$$

Dato un insieme di dipendenze funzionali F definite su $R(Z)$, un'istanza r di R che soddisfa F soddisfa anche F^+.

1.2.3 Superchiave

Dato $R(Z)$ ed un insieme F di dipendenze funzionali, un insieme di attributi K appartenenti a Z si dice superchiave di R se la dipendenza funzionale $K \to Z$ è logicamente implicata da F ($K \to Z$ è in F^+)

Chiave Se nessun sottoinsieme proprio di K è superchiave di R allora K si dice chiave di R.

1.3 Calcolo di F^+ (regole di Armstrong)

Utilizziamo un approccio costruttivo per arrivare all'algoritmo. Esistono una serie di regole, definite da *Armstrong*, che mi permettono di arrivare a trovare l'insieme di tutte le dipendenze funzionali presenti nella chiusura.

Regole di inferenza di Armstrong

- **Riflessività**: se $Y \subseteq X$, allora $X \to Y$
- **Additività (o espansione)**: se $X \to Y$, allora $XZ \to YZ$ per qualunque Z
- **Transitività**: se $X \to Y$ e $Y \to Z$ allora $X \to Z$

Regole derivate di Armstrong

- **Regola di unione**: $\{X \to Y, X \to Z\} \Rightarrow X \to YZ$
- **Regole di pseudotransitività (o aggiunta sinistra)**: $\{X \to Y, WY \to Z\} \Rightarrow XW \to Z$
- **Regola di decomposizione**: se $Z \subseteq Y, X \to Y \Rightarrow X \to Z$

Proprietà delle regole di Armstrong

- **Teorema correttezza**: le regole di inferenze sono corrette, cioè, applicandole a un insieme F di dipendenze funzionali, si ottengono solo dipendenze logicamente implicate da F.
- **Teorema completezza**: le regole di inferenza sono complete, cioè, applicandole ad un insieme F di dipendenze funzionali, si ottengono tutte le dipendenze logicamente implicate da F.
- **Teorema minimalità**: le regole di inferenza sono minimali, cioè, ignorando anche una sola di esse, l'insieme di regole che rimangono non è più completo.

Esempio di dimostrazione Dimostrare che per ogni istanza di relazione

$$X \to Y \Rightarrow XZ \to YZ$$

Supponiamo per assurdo che esista un'istanza r di R in cui valga $X \to Y$ ma non $XZ \to YZ$. Devono esistere due tuple t_1 e t_2 di r tali che:

1. $t_1[X] = t_2[X]$
2. $t_1[Y] = t_2[Y]$
3. $t_1[XZ] = t_2[XZ]$
4. $t_1[YZ] \neq t_2[YZ]$

Cioè è assurdo. Dalla prima e dalla terza deduciamo che

$$t_1[Z] = t_2[Z]$$

con questa e la seconda concludiamo che

$$t_1[YZ] = t_2[YZ]$$

in contraddizione con la quarta!

Riprendiamo l'esempio 3 Avevamo le seguenti dipendenze funzionali:

- Impiegato $\longrightarrow$ Stipendio
- Progetto $\longrightarrow$ Bilancio
- Impiegato Progetto $\longrightarrow$ Funzione

Utilizzando la *regola di attività* sulle prime due otteniamo

- Impiegato Progetto $\longrightarrow$ Stipendio Progetto
- Impiegato Progetto $\longrightarrow$ Bilancio Impiegato
- Impiegato Progetto $\longrightarrow$ Funzione (Uguale a prima)

Infine, con la *regola di unione*, avremo

- Impiegato Progetto $\longrightarrow$ Stipendio Progetto Impiegato Bilancio Funzione

Quindi Impiegato Progetto è chiave. Posso trovare altre dipendenze funzionali in F^+?

1.4 Equivalenza

Dato un insieme F di dipendenze funzionali è molto utile poter determinare se un insieme G sia equivalente ad F. Affermo che F e G sono equivalenti se

$$F^+ = G^+$$

cioè per ogni $X \to Y \in F$ deve essere

$$X \to Y \in G^+$$

e, viceversa, per ogni $X \to Y \in G$ deve essere

$$X \to Y \in F^+$$

Esempio Supponiamo di avere questi due insiemi

$$F = \{\ A \to C, AC \to D, E \to AD, E \to H\}$$
$$G = \{\ A \to CD, E \to AH\}$$

Voglio verificare l'equivalenza di F e G. Devo dimostrare che le dipendenze funzionali in F sono derivabili dalle dipendenze funzionali in G, e viceversa.

- $A \to CD \Rightarrow \boxed{A \to C}, A \to D$
- $A \to CD, CCD \to CD \Rightarrow AC \to CD \Rightarrow AC \to C, \boxed{AC \to D}$
- $E \to AH \Rightarrow E \to A, \boxed{E \to H}$
- $E \to A, A \to D \Rightarrow E \to D$
- $E \to A, E \to D \Rightarrow \boxed{E \to AD}$

Ovviamente dovremo fare la stessa cosa al contrario. Da tutto questo capiamo quanto sia costoso il calcolo di F^+. Generalmente ci interessa sapere se F^+ contiene una certa dipendenza.

Chiusura transitiva Come alternativa posso utilizzare la *chiusura transitiva* di un insieme di attributi X rispetto a F. Data una relazione $R(U)$, un insieme di attributi $X \subseteq U$, la chiusura detta consiste nell'insieme degli attributi che dipendono da X (esplicitamente o implicitamente).

$$X^{+F} = \{A \mid A \in U \ \wedge \ F \Longrightarrow X \to A\}$$

Posso dimostrare che $X \to Y$ è in F^+ se $Y \subseteq X^+$

Riprendiamo l'esempio Invece di verificare se $X \to Y$ in F è anche in G^+ verifico se $Y \subseteq (X)^{+G}$ (chiusura di X rispetto a G). Le varie chiusure qua sotto sono state trovate analizzando l'insieme di dipendenze funzionali G.

- per $A \to C$ risulta $(A)^{+G} = ACD$. Ok ho $C \subseteq (A)^{+G}$
- per $AC \to D$ risulta $(AC)^{+G} = ACD$. Ok ho $D \subseteq (AC)^{+G}$
- per $E \to AD$ risulta $(E)^{+G} = EADCH$. Ok ho $AD \subseteq (E)^{+G}$
- per $E \to H$ risulta $(E)^{+G} = EHADC$. Ok ho $H \subseteq (E)^{+G}$

E viceversa per ogni dipendenza funzionale in G.

Esempio 2 Supponiamo di avere il seguente insieme

$$F = \{A \to B, BC \to D, B \to E, E \to C\}$$

Calcoliamo A^+, cioè l'insieme di attributi dipendenti da A. Trovo che

- $A^+ = A$
- $A^+ = AB$ poichè $A \to B$ e $A \subseteq A^+$
- $A^+ = ABE$ poichè $B \to E$ e $B \subseteq A^+$
- $A^+ = ABEC$ poichè $E \to C$ e $E \subseteq A^+$
- $A^+ = ABECD$ poichè $BC \to D$ e $BC \subseteq A^+$

Quindi da A dipendono tutti gli attributi dello schema, ovvero A è superchiave oltre che chiave.

1.5 Definizione aggiornata di equivalenza

Abbiamo modificato la definizione di equivalenza. Dati F e G, essi sono equivalenti se

- per ogni $X \to Y \in F, Y \in X^{+G}$, e
- per ogni $Z \to W \in G, W \in Z^{+F}$.

1.6 Importanza della chiusura in un insieme di attributi

Dato $R(Z)$ con le sue dipendenze F, la chiusura di un insieme $X \subseteq Z$ di attributi è fondamentale per diversi scopi:

- Possiamo verificare se una dipendenza funzionale è logicamente implicata da F

$$X \to Y \in F^+ \iff Y \subseteq X^{+F}$$

- Possiamo verificare se un insieme di attributi è chiave o superchiave:
  - X è superchiave di R se e solo se $X \to Z \in F^+$, cioè se e solo se $Z \subseteq X^{+F}$
  - X è chiave di R se e solo se $X \to Z \in F^+$ e non esiste alcun sottoinsieme $Y \subset X$ eliminando almeno un elemento, tale che $Z \subseteq Y^{+F}$

1.7 Ridondanze di un insieme di dipendenze funzionali

Alcuni attributi di una dipendenza funzionale possono essere ridondanti. L'obiettivo è individuare forme equivalenti più semplici! Posso semplificare rimuovendo attributi sul lato sinistro di una DF (poichè non essenziali per determinare la parte destra) o rimuovere intere DF poichè ridondanti. Ricorriamo alle regole di Armstrong:

- Applicazione della *Regola di unione*: $\{A \to B, B \to C, A \to CD\}$ può essere semplificata in $\{A \to B, B \to C, A \to C, A \to D\}$
- Attributi estranei a sx: $\{A \to B, B \to C, AC \to D\}$ può essere semplificata in $\{A \to B, B \to C, A \to D\}$

Un insieme F di dipendenze funzionali può contenere dipendenze ridondanti, ovvero ottenibili tramite altre dipendenze di F. Ricorriamo alla transitività: $A \to C$ è ridondante in $\{A \to B, B \to C, \boxed{A \to C}\}$. Ricordiamo che la regola fondamentale è semplificare al massimo la rappresentazione dell'informazione.

1.8 Dipendenze funzionali semplici

Possiamo portare un insieme di dipendenze funzionali F in forma standard, quella in cui sulla destra c'è un solo attributo.
Il seguente insieme

$$F = \{AB \to CD, AC \to DE\}$$

Possiamo riscriverlo così

$$F = \{AB \to C, AB \to D, AC \to D, AC \to E\}$$

Quanto fatto è solo un primo step.

1.9 Attributi estranei

In alcune dipendenze funzionali è possibile che ci siano attributi inutili (appunto estranei) sul lato sinistro. Come li identifico?

- Supponiamo di avere $F = \{AB \to C, A \to B\}$
- Ho $A^+ = A$ e $B^+ = B$
- $A^+ = AB$ poichè $A \to B$ e $A \subseteq A^+$
- $A^+ = ABC$ poichè $AB \to C$ e $AB \subseteq A^+$

C dipende solo da A, e in $AB \to C$ l'attributo B è estraneo poichè a sua volta dipende da A. Possiamo riscrivere l'insieme di dipendenze funzionali nel seguente modo

$$F^+ = \{A \to C, A \to B\}$$

Morale della favola In una dipendenza funzionale del tipo $AX \to B$ l'attributo A è estraneo se X^+ include B (ovvero X da solo determina B).

1.10 Algoritmo per la ridondanza di una dipendenza funzionale

Dopo aver eliminato gli attributi estranei si deve verificare se sono presenti intere dipendenze funzionali inutili, cioè se intere dipendenze funzionali sono implicate da altre. Come faccio a capire se una dipendenza del tipo $X \to A$ è ridondante?

- La elimino dall'insieme F
- Calcolo la chiusura di X (X^+)
- Verifico se include A, cioè se con le dipendenze funzionali che restano riusciamo ancora a dimostrare che $X \to A$

1.11 Copertura minimale

Un insieme di dipendenze funzionali F è minimale se:

- Nella parte destra di ogni dipendenza funzionale c'è un solo attributo
- Non si possono togliere attributi nella parte sinistra di qualche FD senza perdere l'equivalenza nell'insieme ottenuto con F
- Non si possono togliere intere dipendenze funzionali senza perdere l'equivalenza nell'insieme ottenuto con F

Segue che una copertura minimale di un insieme F è un insieme equivalente a F, ma di complessità minore.

NB La copertura minimale non è unica.

Esempio di individuazione di copertura minimale

- Abbiamo il seguente insieme: $F = \{AB \to C, B \to A, C \to A\}$
- A è estraneo in $AB \to C$, quindi: $F = \{B \to C, B \to A, C \to A\}$
- $B \to A$ è ridondante, quindi ottengo: $F = \{B \to C, C \to A\}$

Osserviamo che la dipendenze funzionale ridondante non può essere eliminata se prima non eliminiamo l'attributo estraneo. Segue l'importanza dello svolgere i passi spiegati nell'ordine mostrato!

2 Forme normali e normalizzazione

2.1 Forma normale di Boyce-Codd (BCNF)

Uno schema, se rispetta questa forma normale, è privo di informazione duplicata. Una relazione r è in forma normale di Boyce-Codd se per ogni dipendenza funzionale (non banale) $X \to Y$ definita su di essa, X è superchiave di r. Questa forma normale richiede che i concetti in una relazione siano omogenei (cioè che tutte le proprietà siano direttamente associate alla chiave)

Presenza di dipendenze non banali Se un insieme F di dipendenze per R non è in BCNF, allora in F c'è almeno una dipendenza $X \to Y$ non banale con X non superchiave di R.

Teorema Dato uno schema R e un insieme F di dipendenze funzionali, se F non contiene alcune $X \to Y$ non banale con X non superchiave di R, allora neanche F^+ la contiene.

Metodo di verifica Analizzo una ad una le dipendenze non banali in F verificando se ognuna ha una superchiave come membro sinistro.

- Occorre saper verificare se F è minimale
- Occorre saper verificare se un insieme di attributi è superchiave di una relazione.

Come normalizziamo? Se una relazione non è in BCNF la rimpiazziamo con altre relazioni BCNF: faccio questo decomponendo sulla base delle dipendenze funzionali, separando i concetti.

Esempio Prendiamo la seguente tabella, che presenta certe dipendenze funzionali.

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

$$\text{Impiegato} \rightarrow \text{Sede}$$
$$\text{Progetto} \rightarrow \text{Sede}$$

Procediamo alla decomposizione, ottenendo quanto segue

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Un buon indicatore della correttezza di quanto stiamo facendo è la ricomposizione della tabella iniziale dopo aver effettuato le divisioni: se non posso riottenere la relazione iniziale allora qualcosa non va. Si individua che alcune composizioni di Boyce-Codd potrebbero presentare questo problema

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

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

se io faccio JOIN con la sede ottengo tuple che precedentemente non esistevamo. Segue che dobbiamo trovare composizioni BCNF particolari!

Decomposizione senza perdita Una istanza r di una relazione R si decompone senza perdita su X_1 e X_2 se il join naturale delle proiezione di r su X_1 e X_2 è uguale ad r stessa.

Algoritmo per la decomposizione in BCNF Assumiamo che tutte le dipendenze funzionali in F abbiano un unico attributo come membro destro, e che U sia l'insieme di tutti gli attributi di R.

- $Decomponi(R, F)$
  - Se esiste $X \rightarrow A$ in F con X non superchiave di R:
    - Sostituisco R con una relazione R_1 con attributi $U - A$ ed una relazione R_2 con attributi $X \cup A$

- $Decomponi(R_1, F_{U-A})$
- $Decomponi(R_2, F_{X \cup A})$

Esempio Riprendiamo la solita tabella di prima. Abbiamo

$$F = \{Impiegato \to Sede, Progetto \to Sede\}$$

Applico l'algoritmo e ottengo

$$Decomponi(R, F) = R_1(Impiegato, Progetto), R_2(Impiegato, Sede)$$

Teorema, correttezza dell'algoritmo della decomposizione Qualunque sia l'input, l'esecuzione dell'algoritmo su tale input termina e produce una decomposizione della relazione originaria tale che:

- Ogni relazione ottenuta è in BCNF
- La decomposizione è senza perdita nel JOIN

Proiezione delle dipendenze funzionali di $R(U)$ su $X \subseteq U$ La proiezione di F su X, denotata da F_x è l'insieme delle dipendenze funzionali $Z \to Y$ in F^+ che coinvolgono solo attributi in X, cioè tali che $Z \subseteq X$ e $Y \subseteq X$.

Calcolare la proiezione delle DF su X

- CalcolaProiezione(F, X) :=
  - Pongo $result = \{\emptyset\}$
  - Per ogni sottoinsieme proprio $S \subset X$, per ogni attributo $A \in X$ tale che $A \notin S$ e tale che non esiste alcun sottoinsieme S' di S tale che

    $$S' \to A$$

    è in result, allora dico che se $A \subseteq S^+$ allora

    $$result = result \cup \{S \to A\}$$

Il metodo è funzionante ma in alcuni casi la proiezione di F su X può avere dimensione esponenziale rispetto alla dimensione di F ed X. Si può dimostrare che nessun insieme equivalente a quello delle proiezioni non puiò avere cardinalità minore.

Esempio da internet per capire una spiegazione arzigogolata Supponiamo di avere $R(A, B, C)$ e il seguente gruppo di dipendenze funzionali

$$F = \{A \to B, B \to C, C \to A\}$$

individuo le seguenti proiezioni

- $F_{AB} = \{A \to B, B \to A\}$.

- $F_{AC} = \{A \to C, C \to A\}$

La prima proiezione è interessante perchè ci fa capire che non possiamo limitarci a verificare solo che $X \subseteq AB$ ed $Y \subseteq AB$ con $X \to Y$. Per comodità dobbiamo trovare, dato un insieme F, la sua copertura minimale. A quel punto dobbiamo stare attenti agli attributi sinistri, cioè controllare se essi non dipendono a loro volta dagli elementi su cui stiamo facendo la proiezione. In questo caso:

- Pongo da $B \to C$ e $C \to A$

  $$BC \to A$$

  Ovviamente C dipende da B e per questo può essere rimosso.

- BC è l'insieme S che dicevamo nella spiegazione dell'algoritmo. In questo caso esiste un sottoinsieme S' formato solo da C... Infatti possiamo dire che

  $$C \to A$$

Proprietà dell'algoritmo di decomposizione A seconda dell'ordine con cui si considerano le dipendenze funzionali il risultato della decomposizione può cambiare. Riprendiamo l'esempio di prima e consideriamo le dipendenze funzionali in ordine diverso. Otteniamo uno schema diverso

```
Rx(Impiegato, Sede)
Ry(Impiegato, Progetto)
```

Supponiamo di voler inserire un nuovo impiegato, che sta a Milano, che lavorerà sul progetto Marte, con sede a Roma.

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

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

L'inserimento è legittimo ma emerge un'inconsistenza relativa alla sede della persona e la sede del progetto

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

Non capiamo più se la sede indicata è quella dell'utente o del progetto! Non ho perdita di informazione attraverso il JOIN (le tuple che già esistevano prima della divisione della relazione esistono), ma la tupla nuova rappresenta qualcosa di inconsistente (la sede di Marte è a Milano, quando in realtà il progetto ha sede a Roma). Segue che quanto detto ci garantisce il JOIN ma non le dipendenze (nell'esempio sono state perse).

Verifica di decomposizione senza perdita di dipendenze La definizione di decomposizione senza perdita di dipendenze (considerando due relazioni $R1(X)$ ed $R2(Y)$ ottenute dalla decomposizione) è basata sul verificare che

$$(F_X \cup F_Y)^+ = F^+$$

cioè la chiusura dell'insieme F di dipendenze funzionali equivale alla chiusura dell'insieme ottenuto dall'unione della proiezione delle DF di F su X con la proiezione delle DF di F su Y. Per applicare questa definizione dobbiamo saper calcolare se un insieme di dipendenze funzionali è equivalente ad un altro. Inoltre dobbiamo saper calcolare la proiezione di un insieme di dipendenze funzionali su un insieme di attributi.

Qualità delle decomposizioni Una decomposizione dovrebbe sempre garantire:

- La BCNF
- L'assenza di perdite, in modo da poter ricostruire le informazioni originarie.
- La conservazione delle dipendenze, in modo da mantenere i vincoli di integrità originari.

Esempio di verifica della BCNF Supponiamo che ogni dirigente abbia una sede e che ogni progetto possa essere diretto da più persone, ma in sedi diverse. In questo caso la chiave sarà *Progetto Sede*.

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

Il progetto da solo non può determinarmi il dirigente poichè questo può essere collocato in più sedi. Risulta valido dire che

$$Progetto\ Sede \rightarrow Dirigente$$

poichè ho un solo dirigente per ogni sede di progetto. Quanto segue, tuttavia, non è una BCNF

$$Dirigente \rightarrow Sede$$

poichè i dirigenti possono trovarsi in sedi diverse (Dirigente da solo non è chiave). Si osserva che la prima dipendenza coinvolge tutti gli attributi e che quindi decomporla comporta perderla!

Morale della favola Risulta difficile trovare un BCNF che conservi le dipendenze e il JOIN insieme. Seguono ulteriori forme normali!

2.2 Terza forma normale (3NF)

Una relazione r è in terza forma normale (3NF) se per ogni dipendenza funzionale non banale

$$X \to Y$$

definita su r, è verificata almeno una delle seguenti condizioni:

- X è superchiave di r
- Ogni attributo in Y è contenuto in almeno una chiave di r

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

Progetto Sede → Dirigente
Dirigente → Sede
L'attributo Sede è contenuto nella chiave

Si individua una ridondanza nella ripetizione della sede del dirigente per i vari progetti che dirige.

2.2.1 Confronto tra BCNF e 3NF

- La terza forma normale è meno restrittiva rispetto alla BCNF (ammette relazioni con alcune anomalie e ridondanze).
- Il problema di verificare se una relazione è 3NF è NP-completo (quindi il miglior algoritmo deterministico conosciuto ha complessità esponenziale nel caso peggiore).
- Tuttavia si ha un vantaggio: ottengo sempre una decomposizione senza perdite e con conservazione delle dipendenze.

2.2.2 Algoritmo di decomposizione

Nelle diapositive sono presentati due algoritmi di decomposizione:

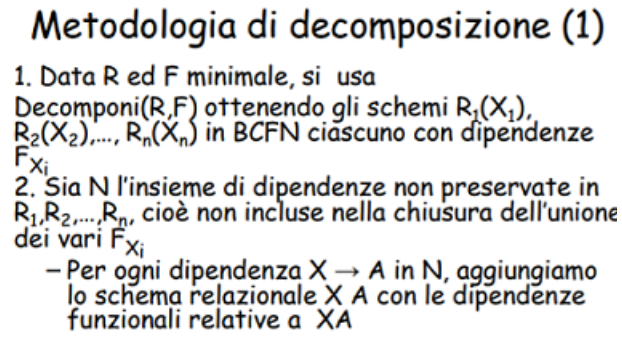
- Una che garantisce l'assenza di perdita sul JOIN e poi conserva le dipendenze
- Una che garantisce le dipendenze e poi risolve l'eventuale perdita sul JOIN

Generalmente si verifica che lo schema 3NF sia anche BCNF. Se la relazione ha una sola chiave allora le due forme normali coincidono.

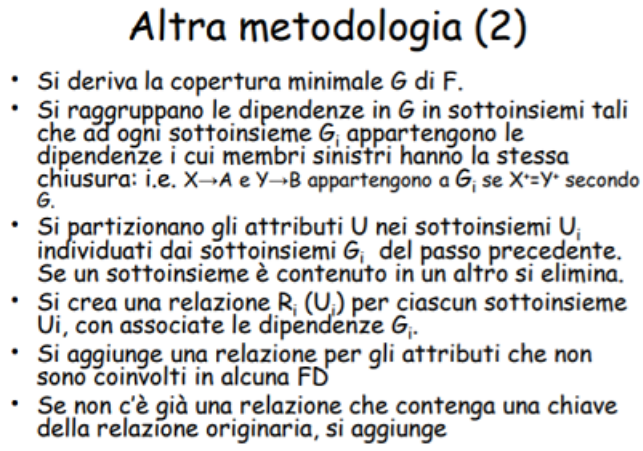
3 Conclusioni

Una decomposizione dovrebbe sempre garantire

- BCNF o 3NF
- L'assenza di perdite (in modo da poter ricostruire le informazioni originarie)



(a) Primo metodo, utilizzato in tutti gli esercizi della Prof



(b) Secondo metodo

- La conservazione delle dipendenze (in modo da mantenere i vincoli di integrità originari)

Il fatto che non si possa ottenere la BCNF è questione di cattiva progettazione. Tutta questa teoria serve per verificare la qualità dello schema logico: quanto spiegato può essere utilizzato anche nella progettazione concettuale per ottenere uno schema di buona qualità (verifica delle ridondanze, partizionamento di entità e relazioni....)

4 Normalizzazione e progetto

4.1 Alcune definizioni aggiuntive

- Se in uno schema di relazione c'è più di una chiave, ognuna di esse è detta chiave candidata. Una delle chiavi è detta chiave primaria
- Un attributo di R è detto attributo primo di R se è membro di una qualche chiave candidata di R. Un attributo è detto non-primo se non è membro di alcuna chiave candidata

4.2 Dipendenze funzionali complete e parziali

- **DF completa**: una DF $X \to Y$ si dice dipendenza funzionale completa (DFC) se la rimozione di qualsiasi attributo A da X comporta il venir meno della sussistenza della DF
- **DF parziale**: una DF $X \to Y$ si dice dipendenza funzionale parziale (DFP) se si possono rimuovere da X certi attributi A e la dipendenza continua a sussistere. Intendiamo che

$$X - \{A\} \to Y$$

4.3 Forme normali

Il processo di normalizzazione sottopone uno schema di relazione a una serie di test per verificare se sono soddisfatte le proprietà tipiche di una forma normale. Individuiamo le seguenti forme:

- Prima forma normale (1NF)

- Seconda forma normale (2NF)
- Terza forma normale (3NF)
- Forma normale di Boyce e Codd (BCNF)
- 4NF e 5NF (Non importante)

4.3.1 Prima forma normale (1NF)

La 1NF richiede che il dominio di un attributo comprenda solo valori atomici (semplici e indivisibili) e che il valore di qualsiasi attributo in una tupla sia un valore singolo del dominio. La forma normale 1NF è già parte integrante della definizione formale di relazione nel modello relazionale.

4.3.2 Seconda forma normale (2NF)

Uno schema di relazione R è in forma 2NF se ogni attributo non-primo A di R è funzionalmente dipendente in modo completo da ogni chiave di R. Si osserva che possono esistere dipendenze tra attributi non-primi.

4.3.3 Confronto tra forme normali

La BCNF implica la 3NF[1], che implica la 2NF. La 4NF e la 5NF riguardano dipendenza di tipo diverso, multivalore.

[1]Se tutte le volte X in $X \to Y$ è superchiave allora soddisfo tutte le volte una delle due condizioni possibili dalla 3NF