UNIVERSITÀ DEGLI STUDI DI VERONA
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Dispensa sulla

Teoria della Normalizzazione

Stefano Rossignoli e Luca Agha Mohammadi Saluth

Versione 1.0

a cura di Alberto Belussi

Sistemi Informativi Geografici

Anno accademico 2004-2005

Indice

Capitolo 1

Introduzione

Lo studio di una base di dati mediante la progettazione concettuale è strettamente guidata dalla descrizione delle proprietà che caratterizzano il caso in analisi, spesso indicato con il nome di *dominio applicativo*. Dalle proprietà caratterizzanti, dette solitamente *requisiti* si cerca poi di stabilire quale sia lo schema concettuale che meglio possa modellarle attraverso i costrutti di un modello concettuale (solitamente viene adottato il modello Entità Relazioni).

In molti casi inoltre, il progetto concettuale della base di dati deve prendere in considerazione le esigenze delle varie applicazioni che interagiranno con la base di dati stessa e le implicazioni sull'efficienza delle interrogazioni di tutte le possibili soluzioni che si possono adottare a livello concettuale. Notiamo quindi come l'abilità e l'esperienza nella stesura di schemi concettuali, unite all'approfondita conoscenza del dominio applicativo, siano fondamentali per poter ottenere un buon progetto concettuale e quindi un buon progetto logico/fisico di una base di dati.

In questa dispensa si presenta una parte della teoria della normalizzazione, focalizzando in particolare sugli strumenti formali che essa introduce nel modello relazionale per supportare la progettazione di una base di dati a livello logico. Vedremo in particolare come sia possibile specificare i requisiti del dominio applicativo direttamente nel modello relazionale attraverso il concetto di *dipendenza funzionale* e come riuscire, sulla base di tale specifica, a produrre un buon schema logico. Si studieranno pertanto alcune proprietà significative degli schemi relazionali (*forme normali* e *decomposizioni*) in modo che sia chiaramente

possibile valutare la bontà di uno schema relazionale.

Il Capitolo 2 presenta alcuni concetti preliminari sul modello relazionale utili per la comprensione dei capitoli successivi e discute i problemi prodotti da una cattiva progettazione della base di dati (detti *anomalie*), mettendo in evidenza come la dipendenza tra i dati rappresentati negli attributi delle relazioni (*dipendeze funzionali*) sia lo strumento formale che consente di distinguere una buona da una cattiva progettazione logica. A tale argomento viene quindi dedicato il Capitolo 3. Infine, Il Capitolo 4 chiude la trattazione con il processo, chiamato normalizzazione, mediante il quale è possibile identificare se un certo schema relazionale sia buono o meno e, in caso negativo, di manipolarlo affinchè lo possa diventare.

Capitolo 2

Preliminari

Questo capitolo presenta i concetti su cui si basa il modello relazionale. Tali concetti saranno utili alla comprensione dei capitoli seguenti. Il contenuto del capitolo è estratto dal libro [*Atzeni-Ceri-Paraboschi-Torlone. Basi di dati. McGraw-Hill*].

La maggior parte dei sistemi di basi di dati oggi sul mercato si fonda sul *modello relazionale* che fu proposto in una pubblicazione scientifica (*E.F.Codd. A relational model for large shared data banks*) nel 1970. L'introduzione di tale modello mirava al superamento delle limitazioni dei modelli all'epoca utilizzati a livello logico, che non permettevano di realizzare la proprietà di *indipendenza dei dati*, già riconosciuta fondamentale. Tale proprietà, se soddisfatta, permette ad utenti e programmi applicativi che utilizzano una base di dati di interagire ad alto livello di astrazione, che prescinde dai dettagli realizzativi utilizzati nella costruzione della base di dati. L'affermazione del modello relazionale è stata abbastanza lenta, a causa proprio dell'alto livello di astrazione. Il modello relazionale si basa su due concetti, *relazione* e *tabella*, di natura diversa ma facilmente riconducibili l'uno all'altro. La nozione di relazione proviene dalla matematica, in particolare dalla teoria degli insiemi, mentre il concetto di tabella è semplice ed intuitivo. La presenza contemporanea di questi due concetti, l'uno formale e l'altro intuitivo, è responsabile del grande successo ottenuto dal modello relazionale.

2.1 Relazioni e tabelle

Definizione 2.1 ***(Prodotto cartesiano)***
Dati due insiemi D_1 e D_2, si chiama ***prodotto cartesiano*** *di D_1 e D_2 e si indica con $D_1 \times D_2$ l'insieme:*
$D_1 \times D_2 = \{(v_1, v_2) \mid v_1 \in D_1 \wedge v_2 \in D_2\}$

Definizione 2.2 ***(Relazione matematica)***
Dati due insiemi D_1 e D_2, chiamati domini di relazione, si definisce una ***relazione matematica*** *R su D_1 e D_2 come:*
$R : R \subseteq D_1 \times D_2$

Le precedenti definizioni di prodotto cartesiano e relazione matematica fanno riferimento a due insiemi, ma possono essere estese in maniera ovvia ad un numero n arbitrario di insiemi. Il numero n delle componenti del prodotto cartesiano viene detto *grado* del prodotto cartesiano e della relazione. Il numero di elementi della relazione viene chiamato, conformemente alla teoria degli insiemi, *cardinalità* di una relazione.
Le relazioni possono essere utilizzate per rappresentare i dati di interesse per una qualche applicazione, rappresentandole eventualmente in forma tabellare. Sulle relazioni e sulle loro rappresentazioni tabellari possiamo quindi fare varie osservazioni. In base a quanto detto, una relazione matematica è un insieme di n-uple ordinate $v_1, v_2, \ldots, v_n$ con $v_1 \in D_1, v_2 \in D_2, \ldots, v_n \in D_n$. Con riferimento all'uso che facciamo delle relazioni per organizzare i dati nelle nostre basi di dati, possiamo dire che ciascuna n-upla contiene dati fra loro collegati, anzi stabilisce un legame tra loro. Ciascuna n-upla è in oltre *ordinata* ovvero l'i-esimo valore proviene dall'i-esimo dominio. È definito cioè un ordinamento fra i domini, che è significativo ai fini dell'interpretazione dei dati nelle relazioni.

Affinchè si possa astrarre dall'ordinamento dei domini di una relazione viene associata a ciascuna occorrenza di dominio nella relazione un nome, detto *attributo*, che descrive inoltre il ruolo giocato dal dominio stesso nella relazione. Modificando la definizione di relazione con l'introduzione degli attributi possiamo vedere come l'ordinamento degli attributi (e delle colonne nella rappresentazione tabellare) risulta irrilevante. Infatti, mentre precedentemente un dominio era

identificato tramite la posizione nella relazione, ora esso è determinato dal nome che gli abbiamo attribuito.

Per formalizzare tali concetti, stabiliamo una corrispondenza fra attributi e domini per mezzo di una funzione $DOM : X \rightarrow D$, che associa a ciascun attributo $A \in X$ un dominio $DOM(A) \in D$. A questo punto introduciamo la seguente definitione di tupla:

Definizione 2.3 ***(Tupla)***
Una tupla su un insieme di attributi X è una funzione t che associa a ciascun attributo $A \in X$ un valore del dominio $DOM(A)$.

Possiamo dare quindi una nuova definizione di relazione come segue.

Definizione 2.4 ***(Relazione)***
Una relazione su X è un insieme di tuple t su X.

La differenza fra questa definizione e la Definizione 2.2 risiede solo nell'astrazione introdotta rispetto alla posizione assunta dagli elementi all'interno della relazione stessa. A questo punto introduciamo la notazione che utilizzeremo in seguito. Se t è una tupla su X e $A \in X$, allora con $t[A]$ indichiamo il valore di t su A.

Come già sottolineato, una relazione può essere utilizzata per organizzare dati rilevanti nell'ambito di una applicazione di interesse. Peraltro, di solito non è sufficiente allo scopo una singola relazione: una base di dati è in generale costituita da più relazioni, le cui tuple contengono valori comuni, quando è necessario per stabilire legami logici. La ripetizione di tali valori viene detta *ridondanza utile*, ovvero necessaria al fine di poter modellare i suddetti legami logici (o relazioni concettuali). I riferimenti fra tuple di relazioni diverse vengono quindi rappresentati per mezzo dei valori comuni che compaiono negli attributi delle tuple; tale caratteristica viene spesso indicata dicendo che il modello relazionale è *basato sui valori* (o value-based) in contrapposizione con la caratteristica *object-based* di molti modelli usati in altri contesti.

Passiamo quindi riassumere le definizioni relative al modello relazionale distinguendo il livello degli schemi da quello delle istanze.

Definizione 2.5 ***(schema di relazione)***
Uno ***schema di relazione*** *è costituito da un simbolo R, detto nome della relazione, e da un insieme di attributi $X = \{A_1, A_2, \ldots, A_n\}$, il tutto di solito indicato con $R(X)$. A ciascun attributo è associato un dominio, come visto in precedenza.*

Definizione 2.6 ***(schema di base di dati)***
Uno ***schema di base di dati*** *è un insieme di schemi di relazione $R = \{R_1(X_1), R_2(X_2), \ldots, R_n(X_n)\}$ dove i nomi di relazione hanno come scopo principale quello di distinguere le varie relazioni nella base di dati.*

Definizione 2.7 ***(istanza di relazione)***
Una ***istanza di relazione*** *(o semplicemente relazione) su uno schema $R(X)$ è un insieme r di tuple su X.*

Definizione 2.8 ***(istanza di base di dati)***
*Un'****istanza di base di dati*** *(o semplicemente base di dati) su uno schema $R = \{R_1(X_1), R_2(X_2), \ldots, R_n(X_n)\}$ è un insieme di relazioni $r = \{r_1, r_2, \ldots, r_n\}$, dove ogni r_i per $1 \leq i \leq n$ è una relazione sullo schema $R_i(X_i)$.*

2.2 Informazione incompleta e valori nulli

La struttura del modello relazionale, come discussa precedentemente, è molto semplice e potente. Nonostante ciò essa impone un certo grado di rigidità in quanto le informazioni debbono essere rappresentate per mezzo di tuple di dati omogenee: in particolare, in ogni relazione possiamo rappresentare solo tuple corrispondenti allo schema della relazione stessa. In effetti, in molti casi, i dati disponibili possono non corrispondere esattamente al formato previsto.
Per rappresentare in modo semplice, ma al tempo stesso comodo, la non disponibilità di valori, il concetto di relazione viene esteso prevedendo che una tupla possa assumere, su ciascun attributo, o un valore del dominio oppure un valore speciale detto *valore nullo*, che denota appunto l'assenza di informazione. Nelle rappresentazioni tabellari utilizzeremo per il valore nullo il simbolo *NULL*.

2.3 Vincoli di integrità

Le strutture del modello relazionale ci permettono di organizzare le informazioni di interesse per le nostre applicazioni. In molti casi, però, non è vero che qualsiasi insieme di tuple sullo schema rappresenti informazioni corrette per l'applicazione. Ad esempio, la presenza di valori nulli può in alcuni casi non essere ammissibile ove una determinata informazione è da considerarsi essenziale.

In una base di dati è quindi opportuno evitare situazioni indesiderate introducendo nello schema relazionale alcune proprietà aggiuntive che si impone siano soddisfatte da tutte le istanze della base di dati. Tali proprietà vengono dette *vincoli di integrità* e possono essere viste come *predicati* che associano ad ogni istanza della base di dati il valore *vero* o *falso*, a seconda che l'istanza soddisfi il vincolo o meno. In generale, ad uno schema di base di dati associamo un insieme di vincoli e consideriamo *corrette* (o *lecite*, o *ammissibili*) le istanze che soddisfano tutti i vincoli.

È possibile classificare i vincoli a seconda degli elementi di una base di dati che ne sono coinvolti. Diremo quindi che un vincolo è *intrarelazionale* se la sua soddisfazione è definita rispetto alle tuple di una relazione della base di dati. Chiameremo invece vincolo *interrelazionale* un vincolo che coinvolge più relazioni. Tra i vincoli intrarelazionali riconosciamo in particolare i *vincoli di tupla* - il cui soddisfacimento dipende esclusivamente dai valori presenti in una singola tupla - ed i *vincoli di dominio* (o *vincoli su valori*) il cui soddisfacimento dipende dal valore assunto da un'attributo di una singola tupla.

Esaminiamo ora le tre categorie più importanti di vincoli di integrità: una particolare classe di vincoli di tupla; i *vincoli di chiave* (che sono i più importanti vincoli intrarelazionali) ed i *vincoli di integrità referenziale* (ovvero i vincoli interrelazionali di maggiore interesse).

2.3.1 Vincoli di tupla

Come è stato detto, i vincoli di tupla esprimono condizioni sui valori di ciascuna tupla, indipendentemente dalle altre tuple. Una possibile sintassi per esprimere vincoli di questo tipo è quella che permette di definire espressioni booleane (cioè mediante connettivi AND, OR e NOT) con atomi che confrontano (con

gli operatori di uguaglianza, disuguaglianza e ordinamento) valori di attributo ed espressioni su valori di attributo.

2.3.2 Chiavi

I vincoli di chiave sono i più importanti vincoli presenti nel modello relazionale. Potremmo infatti affermare che senza di essi il modello stesso non avrebbe senso. Tali vincoli permettono di individuare quali attributi possono determinare i valori dei rimanenti attributi all'interno di una relazione. Una *chiave* è quindi un insieme di attributi utilizzato per *identificare univocamente* le tuple di una relazione. Per formalizzare la definizione, procediamo in due passi:

- un insieme K di attributi è *superchiave* di una relazione r se r non contiene due tuple distinte t_1 e t_2 con $t_1[K] = t_2[K]$;
- K è chiave di r se è una superchiave minimale di r (cioè non esiste un'altra superchiave K' di r che sia contenuta in K come sottinsieme proprio).

Possiamo ora fare alcune riflessioni sul concetto di chiave, che giustificano l'importanza a esso attribuita. Possiamo innanzitutto notare come ogni relazione ed ogni schema di relazione abbiano sempre una chiave, in quanto nel caso peggiore la chiave è rappresentata dalla totalità degli attributi in essi coinvolti. Il fatto che su ciascuno schema di relazione possa essere definita almento una chiave garantisce la accessibilità a tutti i valori di una base di dati e la loro univoca identificazione. I noltre, permette di stabilire efficacemente quei legami logici fra dati contenuti in relazioni diverse che caratterizzano il modello relazionale come "modello basato sui valori". Notiamo però che, in presenza di valori nulli, non è più vero che i valori delle chiavi permettono di identificare univocamente le tuple delle relazioni. Per tale motivo si sente la necessità di limitare la presenza di valori nulli nelle chiavi delle relazioni. In pratica, si adotta una soluzione semplice: su una delle chiavi, detta la *chiave primaria*, si vieta la presenza di valori nulli; sulle altre, i valori nulli sono in genere ammessi.

2.3.3 Vincoli di integrità referenziale

I vincoli che andremo ora ad introdurre rappresentano la più importante classe di vincoli interrelazionali. Un *vincolo di integrità referenziale* fra un insieme di

attributi X di una relazione R_1 e un'altra relazione R_2 è soddisfatto se i valori su X di ciascuna tupla dell'istanza di R_1 compaiono come valori della chiave (primaria) dell'istanza di R_2.

La definizione precisa richiede un po' di attenzione, in particolare nel caso in cui vi siano più chiavi. Procediamo gradualmente, vedendo prima il caso in cui la chiave di R_2 è unica e composta di un solo attributo B (e quindi l'insieme X è a sua volta costituito da un solo attributo A): allora, il vincolo di integrità referenziale fra l'attributo A di R_1 e la relazione R_2 è soddisfatto se, per ogni tupla t_1 in R_1 per cui $t_1[A]$ non è nullo, esiste una tupla t_2 in R_2 tale che $t_1[A] = t_2[B]$. Nel caso più generale, dobbiamo fare attenzione al fatto che ciascuno degli attributi in X deve corrispondere ad un preciso attributo della chiave primaria K di R_2. Allo scopo, è necessario specificare un ordinamento sia nell'inisieme X sia in K. Indicando gli attributi in ordine, $X = A_1A_2 \dots A_p$ e $K = B_1B_2 \dots B_p$ il vincolo è soddisfatto se per ogni tupla t_i in R_1 senza valori nulli su X esiste una tupla t_2 in R_2 con $t_1[A_i] = t_2[B_i]$, per ogni i compreso fra 1 e p.

2.4 Ridondanza e Anomalie

Si consideri il seguente esempio.

Esempio 2.1 *Vogliamo memorizzare i risultati ottenuti dagli studenti di Informatica (COGNOME, NOME, MATRICOLA, DATA DI NASCITA) negli esami (INSEGNAMENTO, DATA APPELLO) del corso di laurea. Avremo perciò il seguente insieme di attributi: {Cognome, Nome, Matricola, Data_Nascita, Voto, Insegnamento, Data_Appello}. Il nostro obiettivo sarà quello di ottenere un buon schema logico nel modello relazionale. Supponiamo quindi di dover utilizzare un'unica tabella che contiene tutti gli attributi sopra elencati. Possiamo vederne un'istanza in Tabella 2.1.*

MATR	INSEGN	COGN	NOME	DATA_N	VOTO	DATA_AP
00100	Basi	**Rossi**	**Paolo**	**1.1.1980**	25	17.7.03
00200	Progr.	Bianchi	Mario	10.10.1981	27	18.06.04
00100	Reti	**Rossi**	**Paolo**	**1.1.1980**	30	5.9.04

Tabella 2.1: Tabella RISULTATO

Notiamo che la prima e la terza riga riportano gli stessi dati relativi allo studente Rossi Paolo, viene introdotta quindi della ridondanza all'interno della tabella, ovvero vengono replicate inutilmente delle informazioni. Uno schema logico per essere di buona qualità dovrà contenere la minima ridondanza possibile.

La presenza di ridondanza produce non solo spreco di memoria, ma ha conseguenze ben più gravi sull'attività di aggiornamento della base di dati. Tali conseguenze di dicono anomalie e si presentano a fronte delle operazioni che modificano il contenuto della base di dati quali: aggiornamento di attributi, inserimento e cancellazione di tuple. Le anomalie si differenziano quindi a seconda dell'operazione che le produce:

- **Anomalia di aggiornamento:**
  Questa anomalia è presente quando, per aggiornare un'informazione atomica contenuta in un attributo della base di dati, si è obbligati a modificare il valore di tale attributo su più di una tupla.

- **Anomalia di inserimento:**
  L'anomalia di inserimento nasce quando per inserire una nuova tupla è necessario assegnare valori nulli (non disponendo di altri valori) anche ad attributi che appartengono ad una chiave candidata della tablella.

- **Anomalia di cancellazione:**
  L'anomalia di cancellazione è presente quando l'eliminazione di un insieme di dati porta o all'eliminazione di altri valori presenti nella stessa tupla che dovrebbero invece rimanere nella base di dati o all'inserimento di valori nulli anche in attributi che appartengono ad una chiave candidata della tabella.

Per studiare in maniera sistematica i concetti appena introdotti è necessario far uso di uno specifico strumento di lavoro: la dipendenza funzionale. Si tratta di un particolare vincolo di integrità per il modello relazionale che descrive legami di tipo funzionale tra gli attributi di una relazione. Il capitolo 3 introduce formalmente tale concetto e fornisce gli strumenti necessari per ragionare su schemi relazionari proprio a partire da tali vincoli di integrità. Andremo in particolar modo a capire come i legami che intercorrono tra le relazioni di uno schema e le dipendenze funzionali valide su di esse permettano di distinguere se lo schema presenti delle ridondanze (inutili) o meno.

Capitolo 3

Dipendenze funzionali

In questo capitolo si presenta il concetto di dipendenza funzionale e si illustrano le sue principali proprietà, in particolare si introdurranno alcuni strumenti formali per la manipolazione di insiemi di dipendenze funzionali e per l'eliminazione di ridondanza da tali insiemi (calcolo della **copertura minima**).

Definizione 3.1 ***(Dipendenza funzionale)***
Sia data una relazione r sullo schema $R(X)$ e siano $Y, Z \subseteq X$. Si dice che r soddisfa la dipendenza funzionale $Y \rightarrow Z$ se è verificata la seguente condizione:

$$\forall t, t' \in r : t[Y] = t'[Y] \Rightarrow t[Z] = t'[Z]$$

Si dice inoltre che la dipendenza funzionale $Y \rightarrow Z$ è valida sullo schema $R(X)$ se per ogni istanza r di $R(X)$ risulta che r soddisfa $Y \rightarrow Z$.

La dipendenza funzionale è un vincolo di integrità, ovvero é soddisfatta da tutte le istanze della base di dati.
Vediamo alcune dipendenze funzionali valide sullo schema della tabella 2.1:
RISULTATO(MAT, INSEGN, COGN, NOME, DATA_N, VOTO, DATA_AP)

D_1: MAT $\rightarrow$ COGN NOME DATA_N
D_2: MAT INS $\rightarrow$ VOTO DATA_AP
D_3: COGN NOME$\rightarrow$ NOME

Possiamo osservare che:

- la dipendenza D_1 indica che gli attributi MAT, COGN, NOME e DATA_N hanno "vita comune" e che l'attributo MAT determina gli altri;

- la dipendenza D_2 indica che gli attributi MAT e INS determinano la data (DATA_AP) e il voto (VOTO) dell'esame;

- infine, la dipendenza D_3 viene detta *inutile* o *banale*, infatti è ovvio che se due tuple sono uguali sugli attributi COGN e NOME allora saranno uguali anche sul singolo attributo NOME.

Inoltre possiamo osservare come, a partire da alcune delle dipendenze funzionali riconosciute valide, se ne possano identificare delle altre. Diremo che quest'ultime sono logicamente implicate dalle prime, denotando tale relazione con il simbolo $\models$, dove perciò $F \models Y \rightarrow X$ significa che la dipendenza funzionale $Y \rightarrow X$ è logicamente implicata da F (dove F è un insieme di dipendenze funzionali). Per quanto appena detto segue che:

D_1,D_2 $\models$ MAT INS $\rightarrow$ COGN NOME DATA_N VOTO DATA_AP

ed ovviamente che :

D_1,D_2 $\models$ MAT INS $\rightarrow$ MAT INS COGN NOME DATA_N VOTO DATA_AP

quindi (MAT,INS) è *superchiave* per la tabella RISULTATO, in quanto determina tutti gli attributi della tabella.

Abbiamo notato quindi come sia possibile, partendo da un insieme di dipendenze funzionali, "derivare" da esso altre dipendenze funzionali per implicazione logica. Introduciamo ora un insieme di regole di inferenza che consentono di calcolare facilmente le dipendenze funzionali implicate da altre dipendenze funzionali. Tali regole si basano su un insieme di assiomi detti *assiomi di Armstrong*. Tale insieme di assiomi gode delle proprietà di *correttezza* e *completezza*. La correttezza permette di affermare che tutto ciò che viene derivato da un insieme F di dipendenze funzionali tramite assiomi di Armstrong è effettivamente derivabile da F (in altre parole non è possibile dedurre alcuna falsa dipendenza). La completezza permette invece di affermare che, dato un insieme F di dipendenze funzionali, tramite gli assiomi di Armstrong è possibile individuare l'insieme di tutte le dipendenze logicamente derivabili da F.

3.1 Assiomi di Armstrong

Dato un insieme F di dipendenze funzionali su un insieme di attributi X si definiscono i seguenti assiomi:

- **A1 (Dipendenze funzionali banali)**
  Dati $Y, Z \subseteq X$ con $Z \subseteq Y$, allora $F \models Y \rightarrow Z$

- **A2 (Assioma dell'aumento)**
  Se $Y \rightarrow Z \in F$ e $W \subseteq X$, allora $F \models YW \rightarrow ZW$

- **A3 (Assioma di transitività)**
  Se $Y \rightarrow Z \in F$ e $Z \rightarrow W \in F$ allora $F \models Y \rightarrow W$

3.2 Regole di inferenza

Le seguenti regole di inferenza permettono di derivare dipendenze funzionali a partire da altre e sono dimostrabili mediante gli assiomi appena introdotti in sezione 3.1.

- **Unione:**
  Se $Y \rightarrow Z \in F$ e $Y \rightarrow W \in F$, allora $F \models Y \rightarrow ZW$.
  Dimostrazione:
  $1 : Y \rightarrow Z$
  $2 : YW \rightarrow ZW \;\; Assioma\, 2\, su\, (1)$
  $3 : Y \rightarrow W$
  $4 : YY \rightarrow YW \;=\; Y \rightarrow YW \;\; Assioma\, 2\, su\, (3)$
  $5 : Y \rightarrow ZW \;\; Assioma\, 3\, su\, (3, 4)$

- **Pseudo transitività:**
  Se $Y \rightarrow Z \in F$ e $ZW \rightarrow T \in F$, allora $F \models YW \rightarrow T$.
  Dimostrazione:
  $1 : Y \rightarrow Z$
  $2 : YW \rightarrow ZW \;\; Assioma\, 2\, su\, (1)$
  $3 : ZW \rightarrow T$
  $4 : YW \rightarrow T \;\; Assioma\, 3\, su\, (2, 3)$

- **Decomposizione:**
  Se $Y \rightarrow Z \in F$ e $W \subset Z$, allora $F \models Y \rightarrow W$.
  Dimostrazione:
  $1 : Y \rightarrow Z \;\; con\, Z \;=\; W \cup T \;\; quindi \;\; Y \rightarrow WT$

$2: WT \rightarrow W$ *Assioma* 1 *su* (1)

$3: Y \rightarrow W$ *Assioma* 3 *su* $(1, 2)$

Gli assiomi di Armstrong e le regole di inferenza appena introdotte permettono di ragionare su schemi relazionali tramite le dipendenze funzionali che, direttamente o tramite inferenza, riconosciamo valide su di essi.

3.3 Proprietà degli insiemi di dipendenze funzionali

In questa sezione si presentano le principali proprietà degli insiemi di dipendenze funzionali. In particolare si introducono una serie di definizioni preliminari per giungere all'introduzione della copertura minima di un insieme di dipendenze funzionali. Tale copertura minima consente infatti di rappresentare in forma minimale tutta l'informazione contenuta in un insieme di dipendenze funzionali. Si illustrano inoltre alcune proprieà della copertura minima e due algoritmi, di cui uno grafico, per il suo calcolo.

Definizione 3.2 ***(Chiusura di un insieme di dipendenze funzionali)*** *Dato un insieme di dipendenze funzionali F si dice* ***chiusura di*** $\mathbf{F}$ *(F^+) l'insieme di tutte le dipendenze funzionali logicamente implicate da F attraverso gli assiomi di Armstrong:*

$$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$$

Definizione 3.3 ***(Equivalenza tra insiemi di dipendenze funzionali)*** *Dati due insiemi di dipendenze funzionali F e G si dice che F è equivalente a G ($F \equiv G$) se:*

$$F^+ = G^+.$$

Definizione 3.4 ***(Minimalità di un insieme di dipendenze funzionali)*** *Un insieme di dipendendenze funzionali F si dice* ***minimale*** *se:*

- $\forall\, X \rightarrow Y \in F: |Y| = 1$
- $\forall\, X \rightarrow Y \in F: F - \{X \rightarrow Y\} \not\equiv F$*: la dipendenza funzionale $X \rightarrow Y$ che non dovesse verificare la condizione si dice* dipendenza funzionale ridondante*, in quanto derivabile dalle altre.*

- $\forall X \to Y \in F : \forall Z \subset X : F - \{X \to Y\} \cup \{Z \to Y\} \not\equiv F$*: nel caso in cui* Z *non verifichi la condizione, ogni attributo di* $X \setminus Z$ *in si dice* attributo ridondante*, in quanto eliminabile dalla parte sinistra della dipendenza* $X \to Y$*.*

Definizione 3.5 ***(Copertura minima)***
Dato un insieme di dipendenze funzionali F *si dice che un insieme di dipendenze funzionali* G *è* ***copertura minima*** *di* F *se valgono le seguenti condizioni:*

- $F \equiv G$
- G *è minimale*

Teorema 3.1 ***(La copertura non è unica)***
La copertura minima di un insieme di dipendenze funzionali non è unica.

Dimostrazione.
Sia dato un insieme di dipendenze funzionali F tali che :

F = { CG → B, CG → D, BC → D, CD → B }

su un insieme di attributi X = {B,C,D,G}. Andiamo a calcolare la copertura minima di F.

1^ocaso :

CG → B ridondante, infatti:

1 : $CG \to D$

2 : $CD \to B$

3 : $CCG \to B$ Pseudo transitività su (1,2)

4 : $CG \to B$

otteniamo F' = { CG → D, BC → D, CD → B }

con F' ≡ F e minimale quindi F' è copertura minima di F.

2^ocaso :

CG → D ridondante, infatti:

1 : $CG \to B$

2 : $BC \to D$

3 : $CGC \to D$ Pseudo transitività su (1,2)

4 : $CG \to D$

otteniamo F" = { CG → B, BC → D, CD → B }

con F" $\equiv$ F e minimale quindi F" è un'altra copertura minima di F.

□

Vediamo ora gli algoritmi che consentono il calcolo della copertura minima di un insieme di dipendenze funzionali. Ovviamente lo scopo di tali algoritmi è quello di eliminare dall'insieme le dipendenze funzionali ridondanti o gli attributi ridondanti. Per la specifica dell'algoritmo risulta utile definire la chiusura di un insieme di attributi.

Definizione 3.6 ***(Chiusura di un insieme di attributi)***
Sia dato un insieme di dipendenze funzionali F su un insieme di attributi X. Sia inoltre $Y \subseteq X$. ***Si chiama chiusura di*** $\mathbf{Y}$ *rispetto a F il seguente insieme di attributi:*

$$Y^+ = \{A \mid Y \rightarrow A \in F^+\} \text{ ed } A \text{ è un singolo attributo}$$

Algoritmo 3.1 ***(Algoritmo per il calcolo della chiusura di un insieme di attributi X)***

$X^+ = X$
repeat
  $oldX^+ = X^+$
  foreach $Y \rightarrow Z \in F$ **do**
    if $Y \subset X^+$ **then** $X^+ = X^+ \cup Z$
  enddo
until $oldX^+ = X^+$

Algoritmo 3.2 ***(Algoritmo per il calcolo della copertura minima di un insieme di dipendenze funzionali F)***

input: F output: G

1. $G = F$
2. *eliminare da G le dipendenze funzionali banali*
3. *sostituire ogni dipendenza funzionale di G del tipo $Y \rightarrow A_1 \dots A_n$ con le dipendenze funzionali $Y \rightarrow A_1, \dots, Y \rightarrow A_n$*

4. *per ogni dipendenza funzionale* $Y \rightarrow A \in G$*:*

   per ogni attributo $B \in Y$ *calcolare* $(Y - \{B\})^+$ *rispetto a* G

   se $A \in (Y - \{B\})^+$ *allora* $G = (G \setminus \{Y \rightarrow A\}) \cup \{Y \setminus \{B\} \rightarrow A\}$

5. *per ogni dipendenza funzionale* $Y \rightarrow A \in G$:

   calcolare Y^+ *rispetto a* $G \setminus \{Y \rightarrow A\}$ *se* $A \in Y^+$ *allora*
   $G = G \setminus \{Y \rightarrow A\}$

Attraverso il concetto di chiusura di un insieme di attributi è possibile ridefinire anche il vincolo di superchiave e di chiave candidata.

Definizione 3.7 ***(Superchiave)***
Sia $R(X)$ *lo schema di una relazione su* X *e sia* F *un insieme di dipendenze funzionali su* X*. Si dice che* $Y \subseteq X$ *è* ***superchiave*** *per* $R(X)$ *se:*

$Y \rightarrow X \ \in F^+$ *(oppure* $Y^+ = X$*)*

Definizione 3.8 ***(Chiave candidata)***
Sia $R(X)$ *lo schema di una relazione su* X *e sia* F *un insieme di dipendenze funzionali su* X*. Si dice che* $Y \subseteq X$ *è* ***chiave candidata*** *per* $R(X)$ *se valgono le seguenti condizioni:*

- $Y \rightarrow X \in F^+$ *(oppure* $Y^+ = X$*)*
- $\forall W \subset Y : W \rightarrow X \notin F^+$

3.3.1 Rappresentazione grafica di dipendenze funzionali

Le dipendenze funzionali possono essere rappresentate in forma grafica seguendo il seguente approccio:

- un attributo nella parte sinistra: $A \rightarrow B$ viene rappresentato come in Figura 3.1.
- due o più attributi nella parte sinistra: $AB \rightarrow C$ viene rappresentato come in Figura 3.2.

Figura 3.1: Rappresentazione grafica di A → B

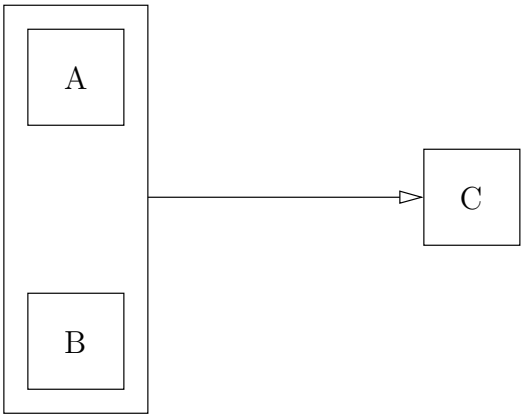

Figura 3.2: Rappresentazione grafica di AB → C

Sulla rappresentazione grafica di un insieme di dipendenze funzionali possono essere applicate delle regole di trasformazione che consentono di calcolare la copertura minima dell'insieme di dipendenze funzionali rappresentato graficamente. Tali regole mirano ovviamente all'eliminazione degli attributi ridondanti e delle dipendenze funzionali ridondanti. Si dimostra infatti che, a partire dalle dipendenze funzionali mantenute nello schema, le dipendenze funzionali eliminate dalle regole possono essere logicamente derivate tramite regole ed assiomi.

Definizione 3.9 ***(Regole di trasformazione per uno schema grafico di dipendenze funzionali)***

- *R1: la regola R1 viene mostrata in Figura 3.3.*
  Dimostrazione:
  a) da $XA \rightarrow B$ ***e*** $X \rightarrow A$ ***deriva*** $X \rightarrow B$
  $1: X \rightarrow A$
  $2: XA \rightarrow B$
  $3: XX \rightarrow B$ *Pseudotransitività su* (1) *e* (2)
  $4: X \rightarrow B$ *poichè* $X \cup X = X$
  b) da $X \rightarrow B$ ***e*** $X \rightarrow A$ ***deriva*** $XA \rightarrow B$
  $1: X \rightarrow B$
  $2: XA \rightarrow BA$ *Assioma* 2 *su* (1)
  $3: XA \rightarrow B$ *Decomposizione su* (2)

- *R2: la regola R2 viene mostrata in Figura 3.4.*
  Dimostrazione:
  $1: A \rightarrow B$
  $2: XA \rightarrow BX$ *Assioma* 2 *su* (1)
  $3: XA \rightarrow B$ *Decomposizione su* (2)

- *R3: la regola R3 viene mostrata in Figura 3.5.*
  Dimostrazione:
  $1: X \rightarrow A$
  $2: A \rightarrow B$
  $3: X \rightarrow B$ *Assioma* 3 *su* (2)

- *R4: la regola R4 viene mostrata in Figura 3.6 (versione 1) e 3.7 (versione 2).*
  Dimostrazione (versione 1):
  $1: XY \rightarrow B$
  $2: YB \rightarrow C$
  $3: XY \rightarrow C$ *Pseudo transitività su* $(1, 2)$
  Dimostrazione (versione 2):
  $1: X \rightarrow B$

2 : $YB \rightarrow C$

3 : $XY \rightarrow C$ *Pseudo transitività su* $(1, 2)$

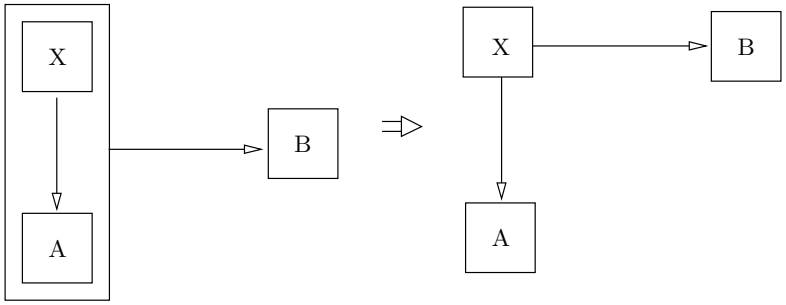

Figura 3.3: Regola 1

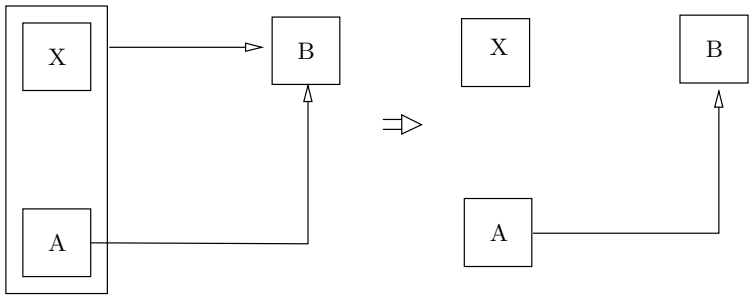

Figura 3.4: Regola 2

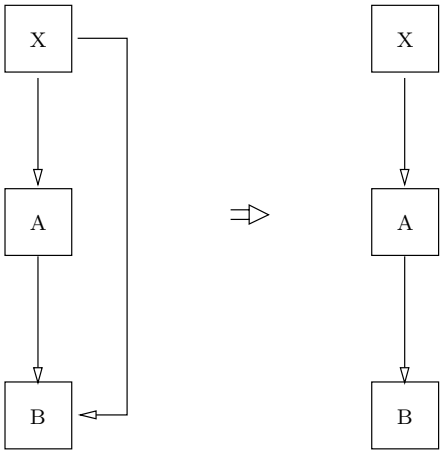

Figura 3.5: Regola 3

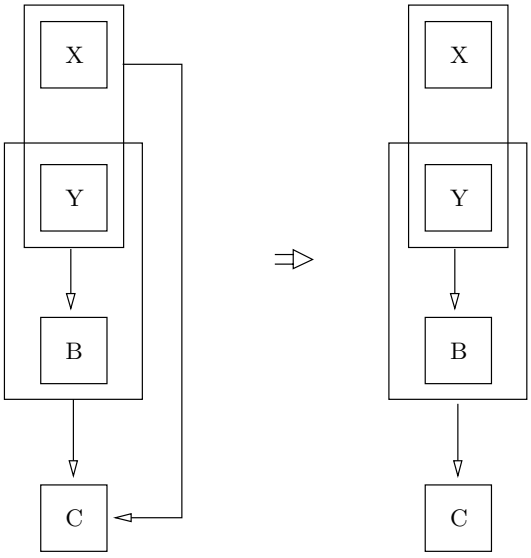

Figura 3.6: Regola 4 (versione 1)

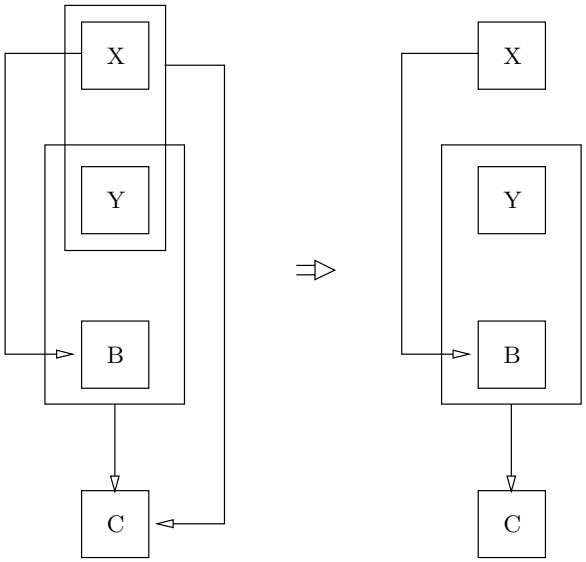

Figura 3.7: Regola 4 (versione 2)

Algoritmo 3.3 ***(Algoritmo grafico per il calcolo della copertura minima)***

0: eseguire i primi due punti dell'algoritmo non grafico (algoritmo 3.2)
1: rappresentare graficamente le dipendenze funzionali di F
2: applicare le regole di trasformazione appena introdotte partendo da R1 (se possibile)

Esempio 3.1 *Trovare la copertura minima sul seguente insieme di dipendenze funzionali :*

$F = \{ A \rightarrow B, AC \rightarrow D, AB \rightarrow D \}$

definito sul seguente insieme di attributi:

$X = \{ A, B, C, D \}$

Vediamo una possibile rappresentazione delle dipendenze funzionali in Figura 3.8.

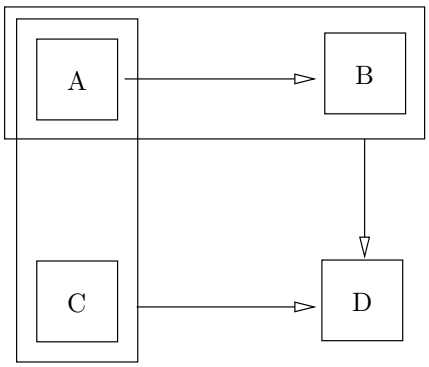

Figura 3.8: Rappresentazione dipendenze funzionali dell'esempio 3.1.

Applichiamo le regole di trasformazione allo schema precendente come fatto nelle Figure 3.9 e 3.10

Il nuovo insieme di dipendenze funzionali sarà quindi:

$F' = \{ A \rightarrow B, A \rightarrow D \}$

Andiamo a calcolare l'insieme delle chiavi candidate, proviamo con gli attributi C ed A:

$\{C, A\}^+ = \{A, B, C, D\}$

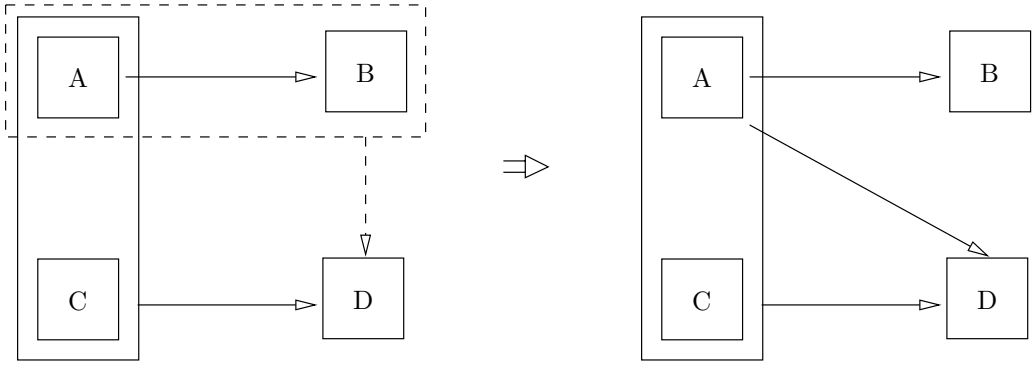

Figura 3.9: Applicazione R1.

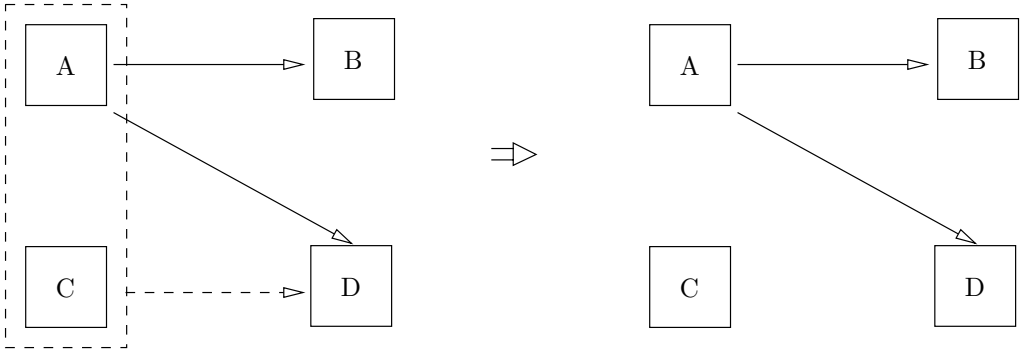

Figura 3.10: Applicazione R2.

visto che con l'insieme $\{A, C\}$ riusciamo ad ottenere tutti gli attributi esso rappresenta una chiave candidata.

Esempio 3.2 *Ci basiamo su un nuovo insieme di dipendenze funzionali :*

$F = \{\ B \rightarrow A,\ AC \rightarrow D,\ AB \rightarrow D\ \}$

Vediamo una possibile rappresentazione delle dipendenze funzionali in Figura 3.11. Applichiamo le regole ed otteniamo un nuovo schema come mostrato in Figura 3.12.

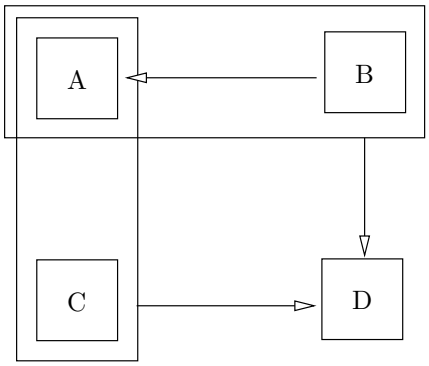

Figura 3.11: Rappresentazione dipendenze funzionali dell'esempio 3.2.

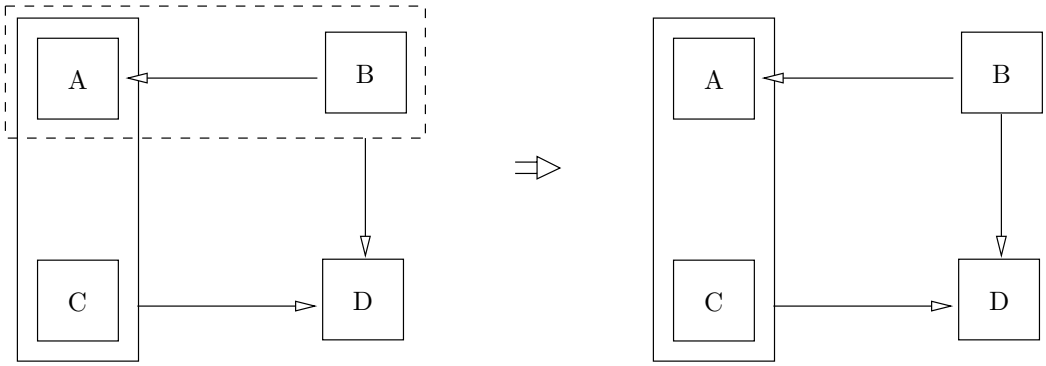

Figura 3.12: Applicazione R1.

Esercizio 3.1 *Costruire la copertura minima e le chiavi candidate partendo dal seguente insieme di dipendenze funzionali*

$F = \{ABF \rightarrow C,\ C \rightarrow A,\ CD \rightarrow B,\ D \rightarrow E,\ D \rightarrow G,\ CG \rightarrow B,\ CD \rightarrow E,\ A \rightarrow F\}$

Soluzione: Applichiamo le regole di trasformazione alla rappresentazione grafica delle dipendenze funzionali (Figura 3.13), possiamo vederne l'effetto in Figura 3.14.

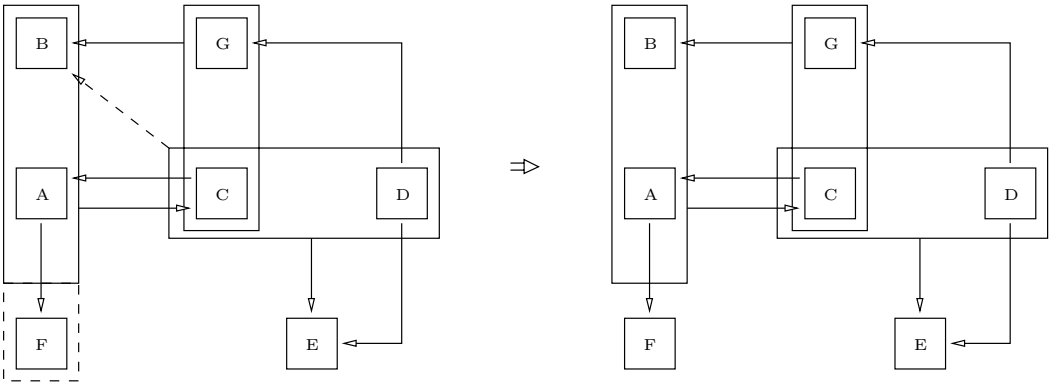

Figura 3.13: Applicazione R1,R4

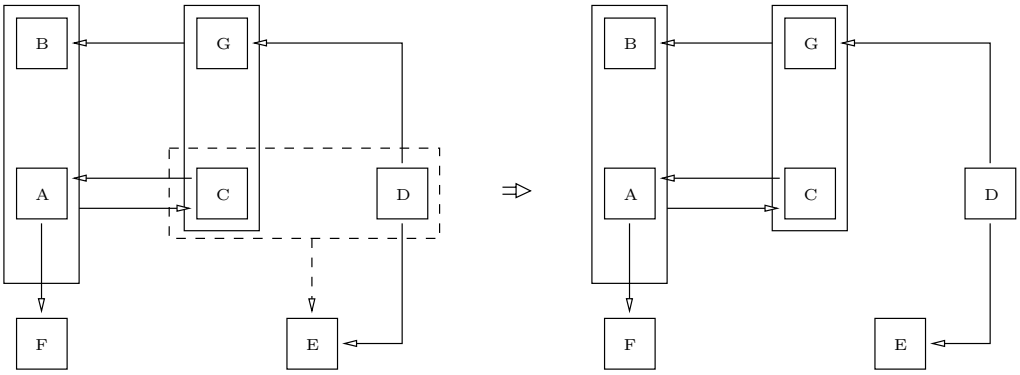

Figura 3.14: Applicazione R2

Chiavi candidate:

$\{D, C\} \Rightarrow \{D, C\}^+ = \{ D, E, G, B, C, A, F\}$

$\{D, A, B\} \Rightarrow \{D, A, B\}^+ = \{ D, G, B, E, A, F, C\}$

Capitolo 4

Normalizzazione

E‘ possibile eliminare la ridondanza inutile presente in uno schema relazionale sottoponendo lo schema stesso ad un processo chiamato *processo di normalizzazione*. Quest'ultimo permette di decomporre lo schema relazionale in diverse tabelle *normalizzate*, ovvero in tabelle che soddisfano determinate *forme normali*.

4.1 Forma normale di "Boyce and Codd"

La forma normale di "Boyce and Codd" o BCNF è, tra tutte le forme normali, la più restrittiva. Una relazione riconosciuta essere in BCNF non contiene difatti alcuna ridondanza inutile. Per tale ragione, il test su BCNF può aiutare a stabilire se una relazione deve subire un processo di normalizzazione.

Definizione 4.1 ***(Forma normale di "Boyce and Codd")*** *Una relazione $R(X)$ è in forma normale di "Boyce and Codd" (BCNF) rispetto ad un insieme di dipendenze funzionali F se, per ogni dipendenza funzionale non banale $Y \rightarrow Z$ che vale su X è soddisfatta la seguente condizione:*

$$Y \text{ è superchiave per } R(X) \ : \ Y \rightarrow X \ \in \ F^{+} \ (\text{oppure } Y^{+} = X)$$

Estendiamo la definizione 4.1 di BCNF valida per una relazione ad uno schema relazionale: si dice che uno schema relazionale $R_1(X_1), \ldots, R_n(X_n)$ è in BCNF se ogni $R_i(X_i)$ con $i \in \{1, \ldots, n\}$ è riconosciuta essere in BCNF. Dicendo quindi che uno schema relazionale è in BCNF affermiamo che tale schema risulta essere

corretto. In caso invece che lo schema relazionale fallisca il test BCNF si comprende come su di esso sia necessario un processo di normalizzazione. Il concetto è riassunto in Figura 4.1.

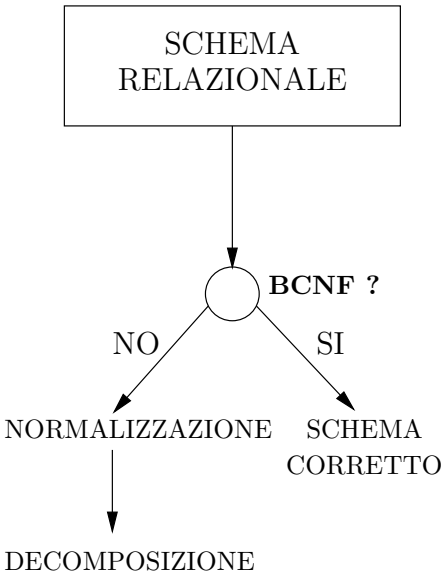

Figura 4.1: Applicazione BCNF

Osservazione: Partendo da uno schema relazionale che non sia prodotto dalla traduzione di uno schema concettuale (progettazione logica), non saremo quasi mai in presenza di uno schema in BCNF.

Algoritmo 4.1 ***(Test per verificare se uno schema relazionale è in BCNF)***

input: $\{R_1(X_1), \ldots, R_n(X_n)\}$, F *insieme di dipendenze funzionali*
output: $TRUE/FALSE$
per ogni $R_i(X_i)$ *dello schema:*
    per ogni $Y \subseteq X_i$
        calcolare Y^+
        se $X_i \subseteq Y^+$ *oppure* $(Y^+ \cap (X_i - Y) = \emptyset)$ *allora continua*
        altrimenti ritorna FALSE
ritorna TRUE

Vediamo un esempio di applicazione dell'algoritmo appena introdotto.

Esempio 4.1 *Sia X un insieme di attributi tali che :*

$X = \{A,\ B,\ C,\ D,\ E\}$

e sia F un insieme di dipendenze funzionali definito nel seguente modo:

$F = \{A \rightarrow B,\ B \rightarrow C,\ B \rightarrow D\}$

abbiamo a disposizione il seguente schema:

$R_1(A,\ C,\ E)$ $\quad CC(R_1) = \{A,\ E\}$

$R_2(B,\ D)$ $\quad CC(R_2) = \{B\}$

Andiamo a vedere se lo schema è in BCNF.

- $R_1(A,\ C,\ E)$:

  $\{A\}^+ = \{A,\ B\}$

  $= \{A,\ B,\ C,\ D\}$

  - $X_i \subseteq Y^+$ *?*

    $\{A,\ C,\ E\} \nsubseteq \{A,\ B,\ C,\ D\}$

  - $Y^+ \cap (X_i - Y) = \emptyset$ *?*

    $\{A,\ B,\ C,\ D\} \cap \{\ \{A,\ C,\ E\} - \{A\}\ \} = \{C\} \neq \emptyset$

  R_1 *non è in BCNF*

- $R_2(B,\ D)$

  $\{B\}^+ = \{B,\ C,\ D\}$

  - $X_i \subseteq Y^+$ *?*

    $\{B,\ D\} \subseteq \{B,\ C,\ D\}\ \surd$

  $\{D\}^+ = \{D\}$

  - $X_i \subseteq Y^+$ *?*

    $\{B,\ D\} \nsubseteq \{D\}$

  - $Y^+ \cap (X_i - Y) = \emptyset$ *?*

    $\{D\} \cap \{\ \{B,\ D\} - \{D\}\ \} = \emptyset\ \surd$

  R_2 *è in BCNF*

4.2 Decomposizioni di schemi relazionali

La decomposizione di uno schema relazionale $R(X)$ è l'insieme di relazioni:
$\{R_1(X_1), R_2(X_2), \ldots, R_n(X_n)\}$ tale che $X = X_1 \cup X_2 \cup \cdots \cup X_n$
Si noti come non venga richiesto che gli X_i siano tra di loro disgiunti.
Una motivazione che potrebbe portare ad una decomposizione di uno schema relazionale è che tale operazione potrebbe permetterci di portare le nuove relazioni in BCNF (qualora quella originaria non lo sia). Notiamo però come non tutte le decomposizioni siano desiderabili, ovvero che non tutte possano effettivamente portare ad un miglioramento. Individuiamo perciò alcune proprietà essenziali che debbono essere soddisfatte da una "buona" decomposizione.

Definizione 4.2 ***(Decomposizione senza perdita)***
Data una relazione $R(X)$ e un insieme di dipendenze funzionali F, si dice che la decomposizione $R_1(X_1), \ldots, R_n(X_n)$ è senza perdita se, per ogni istanza r di $R(X)$ che soddisfa le dipendenze funzionali di F vale che:

$$\Pi_{X_1}(r) \bowtie \ldots \bowtie \Pi_{X_n}(r) = r$$

Tale proprietà richiede quindi che la relazione di partenza sia ricomponibile tramite il calcolo di un join naturale sulle relazioni ottenute mediante la decomposizione. Notiamo inoltre che le operazioni di join devono restituire esattamente le tuple della relazione di partenza, ovvero che esse non devono fornirne nè di meno nè di più. Sebbene ottenere più tuple rispetto a quelle di partenza potrebbe sembrare un aumento dell'informazione, tale risultato è invece da ritenersi una perdita di informazione.

Osservazione: Una decomposizione $R_1(X_1), R_2(X_2)$ di una relazione $R(X)$ è senza perdita rispetto ad un insieme di dipendenze funzionali F se e solo se vale una delle seguenti condizioni:

- $X_0 \rightarrow X_1 \in F^+$
- $X_0 \rightarrow X_2 \in F^+$

con $X_0 = X_1 \cap X_2$

Esempio 4.2 *Una possibile decomposizione della tabella seguente :*

RISULTATO(MAT, INS, COGN, NOME, DATA_N, VOTO, DATA_APP)

sulla quale sono definite le seguenti dipendenze funzionali :

MAT $\rightarrow$ *COGN NOME DATA_N*

MAT INS $\rightarrow$ *VOTO DATA_APP*

potrebbe essere :

R_1*(MAT, COGN, NOME, DATA_N)*

R_2*(INS, MAT, VOTO, DATA_APP)*

Possiamo notare che la decomposizione R_1, R_2 *è senza perdita rispetto a F, prendiamo infatti* X_0 = *MAT e osserviamo che :*

MAT $\rightarrow$ *COGN NOME DATA_N* $\in F$

Definizione 4.3 ***(Decomposizione che conserva le dipendenze funzionali)***

Data una relazione $R(X)$ *e un insieme di dipendenze funzionali* F*, si dice che la decomposizione* $R_1(X_1), \ldots, R_n(X_n)$ *conserva le dipendenze funzionali di* F *se vale la seguente condizione:*

$$(\Pi_{X_1}(F) \cup \cdots \cup \Pi_{X_n}(F))^+ = F^+$$

dove $\Pi_{X_i}(F) = \{Y \rightarrow Z \mid Y \rightarrow Z \in F \wedge Y \cup Z \subseteq X_i\}$

La ragione per cui è desiderabile che una decomposizione preservi le dipendenze funzionali sta nel fatto che quest'ultime possono essere viste come vincoli di integrità per la relazione decomposta. Suddividendo una relazione $R(X)$ in una decomposizione $R_1(X_1), \ldots, R_n(X_n)$ che non conserva le dipendenze funzionali, può accadere che tale decomposizione sia comunque senza perdita. L'operazione di join tra le varie X_i potrebbe quindi portare alla costruzione di una relazione che non soddisfa F e che violi perciò i vincoli di integrità. Per evitare questa situazione sarebbe necessario, a fronte di ogni aggiornamento di un R_i, effettuare un'operazione di join per verificare che i nuovi valori non violino i vincoli di integrità che F rappresenta. Se invece la decomposizione conserva le dipendenze funzionali la verifica della soddisfazione dei vincoli d'integrità espressi nelle dipendenze stesse può essere eseguita sulle singole relazioni senza richiedere join aggiuntivi.

Esempio 4.3 *Vediamo un esempio di decomposizione relativa allo schema dell'esempio 4.2 che non conserva le dipendenze funzionali:*

R_1*(MAT, COGN, NOME)*

R_2*(DATA_N, DATA_APP)*

R_3*(INS, MAT)*

R_4*(INS, VOTO)*

infatti possiamo notare che :

$$\{ \prod_{X_1}(F) \cup \prod_{X_2}(F) \cup \prod_{X_3}(F) \cup \prod_{X_4}(F)\}$$
$$= \quad \{ \emptyset \cup \emptyset \cup \emptyset \cup \emptyset\} \neq F^+$$

Teorema 4.1 *Data una relazione R(X) e un insieme di dipendenze funzionali F dove R(X) non è in BCNF non è sempre possibile generare una decomposizione di R(X) tali che:*

- *sia in BCNF*
- *sia senza perdita*
- *conservi le dipendenze funzionali*

Dimostrazione.

Per dimostrare quanto detto basterà fornire un controesempio. Prendiamo uno schema R definito nel seguente modo:

R(VIA, CITTA, CAP)

con le seguenti dipendenze funzionali :

CAP → CITTA

VIA CITTA → CAP

e chiavi candidate:

CC: {VIA, CITTA}, {CAP, VIA}.

Possiamo facilmente osservare che non è in BCNF visto che CAP non è superchiave. Il resto della dimostrazione viene lasciato come esercizio, basterà infatti verificare che tutte le possibili decomposizioni non soddifano ad almeno uno dei punti elencati nel teorema. □

4.3 Terza forma normale ed altre forme normali

Definizione 4.4 ***(Terza forma normale)***
Data una relazione $R(X)$ e un insieme di dipendenze funzionali F si dice che $R(X)$ è in terza forma normale se, per ogni dipendenza funzionale non banale $Y \rightarrow Z$ di F che vale su X, è soddisfatta una delle seguenti condizioni:

- *Y è superchiave di $R(X)$;*
- *Ogni attributo di Z è parte di una chiave candidata di $R(X)$.*

La terza forma normale modella la situazione in cui ammettiamo la presenza di un po' di ridondanza al fine di non perdere qualche dipendenza funzionale a fronte di una ipotetica decomposizione dello schema relazionale.

Esempio 4.4 *Vediamo un esempio di schema in terza forma normale :*
R(VIA,CITTA,CAP)
CC: {VIA,CITTA}, {CAP,VIA}
Prendiamo le dipendenze funzionali e vediamo se esse soddisfano alle condizioni elencate nella definizione 4.4:
VIA CITTA → CAP: OK in quanto {VIA,CITTA} è superchiave $\surd$
CAP → CITTA: OK in quanto CITTA $\subseteq$ *{CITTA,VIA}* $\surd$

Definizione 4.5 ***(Seconda forma normale)***
Data una relazione $R(X)$ e un insieme di dipendenze funzionali F si dice che $R(X)$ è in seconda forma normale se, per ogni dipendenza funzionale $Y \rightarrow Z$ di F che vale su X, è soddisfatta una delle seguenti condizioni:

- *Ogni attributo di Z è parte di una chiave candidata di $R(X)$;*
- *Y non è un sottoinsieme proprio di una chiave candidata di $R(X)$.*

La seconda forma normale ammette che ci siano delle dipendenze interne alla tabella, non ammettendo però che tali dipendenze siano stabilite a partire da un sottoinsieme proprio di una chiave candidata.
Notiamo inoltre che se una relazione è in forma normale $BCNF$ allora essa è anche in terza forma normale. L'essere in terza forma normale per una relazione implica inoltre che tale relazione sia anche in seconda forma normale. Questa

affermazione è vera in quanto, secondo le Definizioni 4.1 e 4.4 rispettivamente di $BCNF$ e di terza forma normale, se ogni dipendenza funzionale soddisfa la condizione apposta da $BCNF$ allora soddisfa anche la prima delle due condizioni della terza forma normale essendo esse la medesima. Inoltre, per quanto visto nelle Definizioni 4.4 e 4.5, se una dipendenza funzionale $Y \rightarrow Z$ soddisfa la prima condizione della terza forma normale, ovvero che Y sia superchiave (Definizione 3.7) dello schema, è verificato che Y non possa essere un sottoinsieme proprio di una chiave candidata (Definizione 3.8) e quindi è soddisfatta la seconda clausula della seconda forma normale. Se invece $Y \rightarrow Z$ soddisfa la seconda condizione della terza forma normale essa soddisfa la prima condizione della seconda forma normale in quanto esse coincidono.

Esempio 4.5 *Vediamo un esempio di schema in seconda forma normale e non in terza forma normale:*

R(A,B,C)

CC(R): $\{A\}$

dipendenze funzionali sullo schema

$A \rightarrow B$

$B \rightarrow C$

Lo schema R è in BCNF ?

No, perchè B non è superchiave.

Lo schema R è in terza forma normale ?

No, perchè $C \notin \{A\}$

Lo schema R è in seconda forma normale ?

Sì, in quanto:

$A \rightarrow B$: $\{A\}$ *è superchiave per R* $\surd$

$B \rightarrow C$: $\{B\} \not\subset \{A\}$ $\surd$

Esempio 4.6 *Dato il seguente insieme di dipendenze funzionali :*

$F = \{A \rightarrow B, E \rightarrow F, F \rightarrow E, I \rightarrow G, C \rightarrow D,\ GH \rightarrow I, D \rightarrow L\}$

e la seguente decomposizione :

D: $\{R_1(A,B,C), R_2(E,F), R_3(G,H,I), R_4(D,C,L)\}$

calcolare per ogni R_i *l'insieme delle chiavi candidate e la forma normale in cui si trova. Possiamo riassumere la situazione corrente in Figura 4.2.*

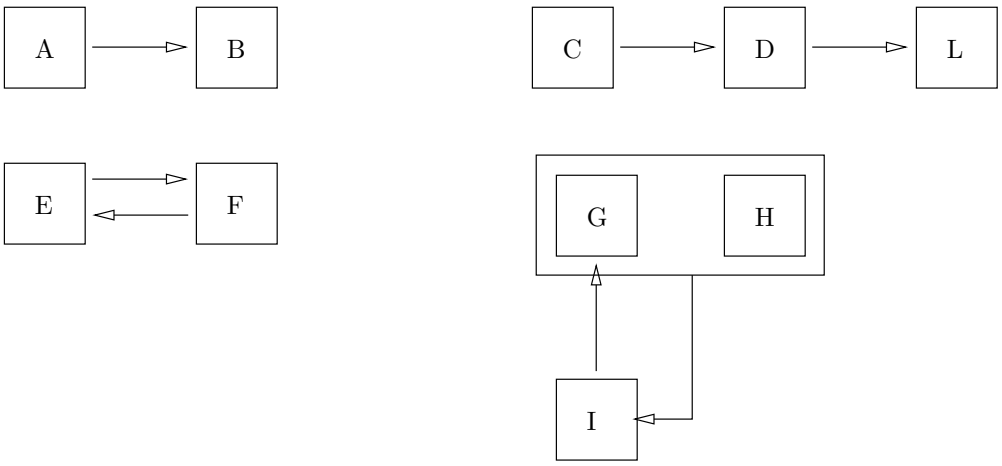

Figura 4.2: Rappresentazione grafica delle dipendenze funzionali dell'esempio 4.6

1. *Andiamo ora a calcolare le chiavi candidate per ciascuno schema:*
   CC(R_1): $\{A,C\}$
   CC(R_2): $\{E\},\{F\}$
   CC(R_3): $\{I,H\},\{G,H\}$
   CC(R_4): $\{C\}$

2. *Vediamo in che forma normale sono i vari schemi:*
   - *R_1 : $(A \rightarrow B)$*

     Non è BCNF: A non è superchiave

     Non è in terza forma normale: A non è superchiave e $B \notin \{A,C\}$

     Non è in seconda forma normale: $B \notin \{A,C\}$ e $\{A\} \subset \{A,C\}$

   - *R_2 : $(E \rightarrow F, F \rightarrow E)$*

     È in BCNF: sia E che F sono superchiavi

   - *R_3 : $(F \rightarrow G, GH \rightarrow I)$*

     Non è in BCNF: F non è superchiave

     È in terza forma normale: $G \subseteq \{G,H\}$ e $\{G,H\}$ è superchiave

   - *R_4 : $(C \rightarrow D, D \rightarrow L)$*

Non è in BCNF: D non è superchiave

Non è in terza forma normale: $L \notin \{C\}$

È in seconda forma normale: $\{C\} \not\subset \{C\}$ *e* $\{D\} \not\subset \{C\}$

4.4 Forme Normali e Test di Verifica

Obiettivo di questa parte è quello di introdurre alcuni algoritmi per la verifica in quale forma normale si trova il nostro schema. Possiamo avere due tipi di verifiche :

1. su un insieme di attributi X non decomposto
2. su una decomposizione X_1, X_2,..., X_n di X

Andiamo a vedere quali sono le regole da seguire per ogni forma normale:

- **BCNF** :
  - verifica su X
    - $\forall\ Y \rightarrow Z \in$ F verificare che $Y^+ = X$
  - verifica su X_1, X_2,..., X_n
    - $\forall\ X_i : \forall\ Y \subseteq X_i : (X_i \subseteq Y^+) \vee (Y^+ \cap (X_i \setminus Y) = \emptyset)$
- **Terza forma normale** :
  - verifica su X:
    - calcolare le chiavi candidate su X : K_1,K_2,...K_n
    - $\forall\ Y \rightarrow Z \in F$ verificare che:
      $(Y^+ = X) \vee (Z \subseteq \bigcup_{i=1}^{m} K_i)$
  - verifica su X_1, X_2,..., X_n:
    - $\forall\ X_i$: calcolare le chiavi candidate di X_i : $K_{i,1}$,$K_{i,2}$,...K_{i,n_i} e
      $\forall\ Y \rightarrow Z \in F : (Y \cup Z \subseteq X_i) \Rightarrow (X_i \subseteq Y^+) \vee (Z \subseteq \bigcup_{j=1}^{n_i} K_{i,j})$
- **Seconda forma normale** :
  - verifica su X:

- calcolare le chiavi candidate su X : K_1,K_2,...,K_n
- $\forall\ Y \rightarrow Z \in F$ verificare che:

$$\forall i \in \{1, ..., n\} : Y \not\subset K_i$$

- verifica su X_1, X_2,..., X_n:
  - $\forall\ X_i$: calcolare le chiavi candidate di X_i : $K_{i,1}$,$K_{i,2}$,...K_{i,n_i} e

$$\forall\ Y \rightarrow Z \in F \text{ verificare che:}$$

$$(Y \cup Z) \subseteq X_i \Rightarrow (\forall j \in \{1, ..., n_i\} : Y \not\subset K_{i,j}) \vee (Z \subseteq \bigcup_{j=1}^{n_i} K_{i,j})$$

Algoritmo 4.2 ***(Algoritmo per il calcolo di una decomposizione in terza forma normale che conserva le dipendenze funzionali ed è senza perdita)***

input: *insieme di attributi X, insieme di dipendenze funzionali F (MINIMALE)*
output: *decomposizione X_1, X_2,..., X_n di X*

- *calcolare le chiavi candidate di X rispetto a F*
- *i = 1*

  for each $Y \rightarrow Z \in F$ *do*
  $X_i = Y \cup Z$;
  $i = i + 1$;
  enddo
  if(nessuno degli X_i *generati contiene una chiave candidata di X)*
  X_i = *una chiave candidata di X;*
  else
  i = i - 1;
  endif
  return (X_1,X_2,...,X_i);

Esempio 4.7 *Supponiamo d'avere a disposizione il seguente insieme di attributi:*
$X = \{A,B,C,D,E,F\}$
e il seguente insieme di dipendenze funzionali:
$F = \{B \rightarrow C, B \rightarrow A, D \rightarrow E, D \rightarrow A\}$
calcolare una decomposizione che sia in terza forma normale.

Soluzione: *La chiave candidata è data dall'insieme* $\{B,D,F\}$.
Andiamo a calcolare per ogni dipendenza funzionale la decomposizione ad essa correlata:

- $B \rightarrow C \Rightarrow R_1(\underline{B},C)$
- $B \rightarrow A \Rightarrow R_2(\underline{B},A)$
- $D \rightarrow E \Rightarrow R_3(\underline{D},E)$
- $D \rightarrow A \Rightarrow R_4(\underline{D},A)$

Possiamo notare che nessuna delle R_i *generate contiene* $\{B,D,F\}$ *generiamo quindi una nuova relazione nel seguente modo:*

$R_5(B,D,F)$

Abbiamo ottenuto quindi una decomposizione in terza forma normale che conserva le dipendenze funzionali ed è senza perdita rispetto all'insieme di attributi X.

Algoritmo 4.3 ***(Algoritmo che calcola una decomposizione in terza forma normale che conserva le dipendenze funzionali e senza perdita (VERSIONE GRAFICA))***

input: *rappresentazione grafica di un insieme di attributi X e di un insieme di dipendenze funzionali F (minimale)*
output: *decomposizione* X_1, X_2,..., X_n *di X*

1. *calcolare le chiavi canditate di X rispetto a F*
2. *togliere i nodi da cui non partono e a cui non arrivano archi e che non sono contenuti in altri nodi*
3. *calcoliamo l'insieme dei nodi da processore N:*

   $N = \{\ n \mid n$ *non ha archi entranti e non è contenuto in altri altri nodi*$\}$

   se $N = \emptyset$ *allora* $N = \{n \mid n$ *è uno dei nodi con più archi uscenti*$\}$

4. $\forall\ n \in N$ *generare una relazione* $R(X_i)$ *dove :*

$X_i = n \cup \{A \mid A$ *è un attributo raggiungibile da n con percorsi lunghi 1*$\}$

5. *Elimino gli archi percorsi*
6. *Elimino nodi isolati*
7. *Se ci sono ancora nodi : vai al punto 2*
8. *Se nessun X_i contiene una chiave candidata aggiungere una relazione che ne contiene una.*

Vediamo un applicazione dell'algoritmo appena introdotto nell'esempio 4.8.

Esempio 4.8 *Utilizziamo l'algoritmo grafico per risolvere l'esercizio dell'esempio 4.7, ricordiamo che l'unica chiave candidata è rappresentata dall'insieme {B,D,F}. La situazione può essere riassunta come in Figura 4.3.*

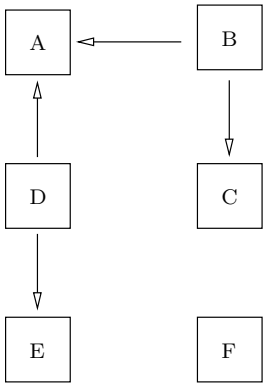

Figura 4.3: Esempio 4.8

Soluzione : *Rimuoviamo i nodi da cui non partono ne arrivano archi (togliamo quindi solamente F):*
Calcoliamo il primo insieme N:

$N_1 = \{B,D\}$

partiamo dai nodi B e D e percorriamo tutti gli archi uscenti costruendo le varie relazioni, successivamente rimuoviamo tutti gli archi percorsi, come si può notare in Figura 4.4. Otteniamo quindi le seguenti relazioni:

$R_1(B,A,C)$

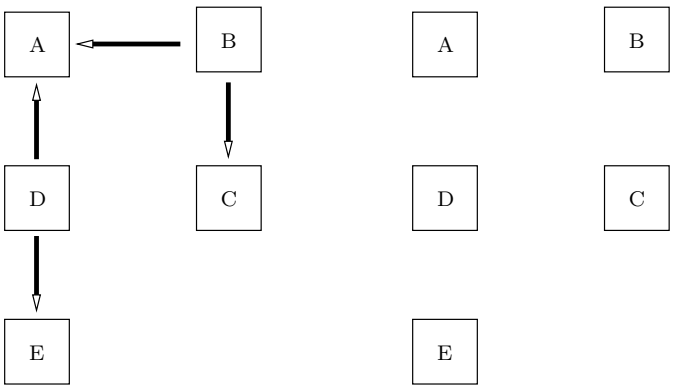

Figura 4.4: Applicazione algoritmo grafico esempio 4.8

$R_1(D,A,E)$

Visto che non ci sono più nodi su cui proseguire passiamo al passo 7 dell'algoritmo e visto che nessun R_i contiene $\{B,D,F\}$ aggiungiamo la relazione $R_3(B,D,F)$ ottenendo quindi una decomposizione in terza forma normale senza perdita e che conserva le dipendenze funzionali.

Esempio 4.9 *Supponiamo d'avere a disposizione il seguente insieme di attributi:*

$X = \{A,B,D,E,F,G,H,I,L,J,K\}$

e il seguente insieme di dipendenze funzionali:

$F = \{AB \rightarrow D,\ AB \rightarrow E, D \rightarrow E,\ B \rightarrow G,\ GH \rightarrow I,\ G \rightarrow H,\ GH \rightarrow L,\ IL \rightarrow J,\ I \rightarrow J,\ A \rightarrow K,\ D \rightarrow F\}$.

Calcolare la copertura minima le chiavi candidate e una decomposizione in terza forma normale attraverso il metodo grafico.

Soluzione : *L' insieme delle chiavi candidate è dato dall'insieme $\{A,B\}$.*

Possiamo riassumere la situazione in Figura 4.5. Applichiamo all'insieme di dipendenze funzionali alcune regole per semplificare lo svolgimento dell'esercizio. Vediamo ora quali sono i passi principali di applicazione dell'algoritmo:

- $N_1 = \{(A,B)\}$
  percorriamo tutti gli archi di lunghezza 1 partendo da (A,B), ovvero solamente l'arco che collega (A,B) a D otteniamo quindi una relazione definita nel seguene modo:

  $R_1(\underline{A,B},D)$.

possiamo vedere la situazione riassunta nella Figura 4.6 (a).

- $N_2 = \{A,B,D\}$
  percorrendo gli archi di lunghezza 1 otteniamo:

  $R_2(\underline{A},K)$

  $R_3(\underline{B},G)$

  $R_4(\underline{D},E,F)$

  Figura 4.6 (b).

- $N_3 = \{G\}$
  percorrendo gli archi di lunghezza 1 otteniamo:

  $R_5(\underline{G},H,I,L)$

  Figura 4.7 (a).

- $N_4 = \{I\}$
  percorrendo gli archi di lunghezza 1 otteniamo:

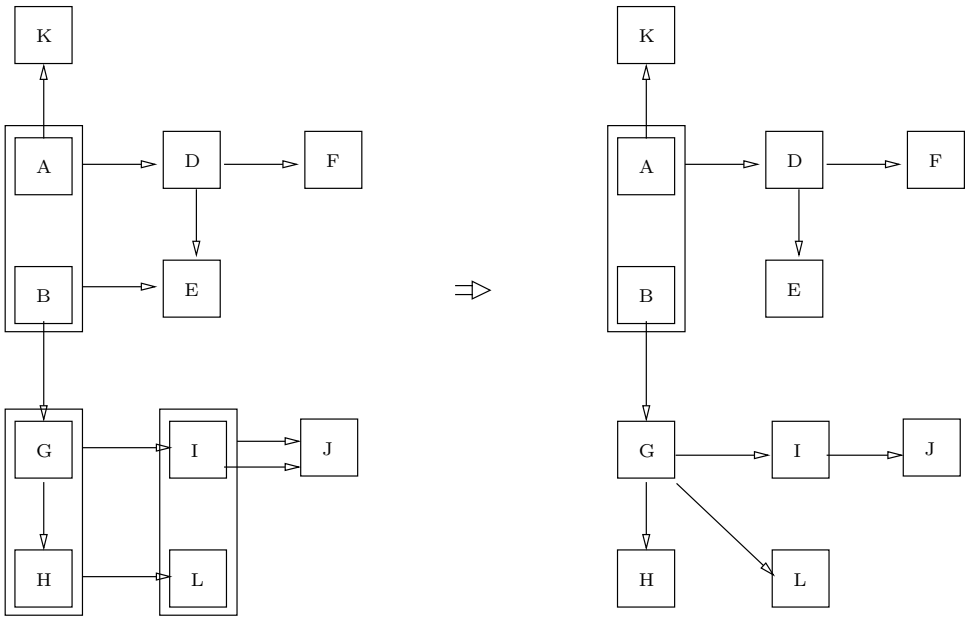

Figura 4.5: Semplificazione con regole

R_6 (I,L)

Figura 4.7 (b).

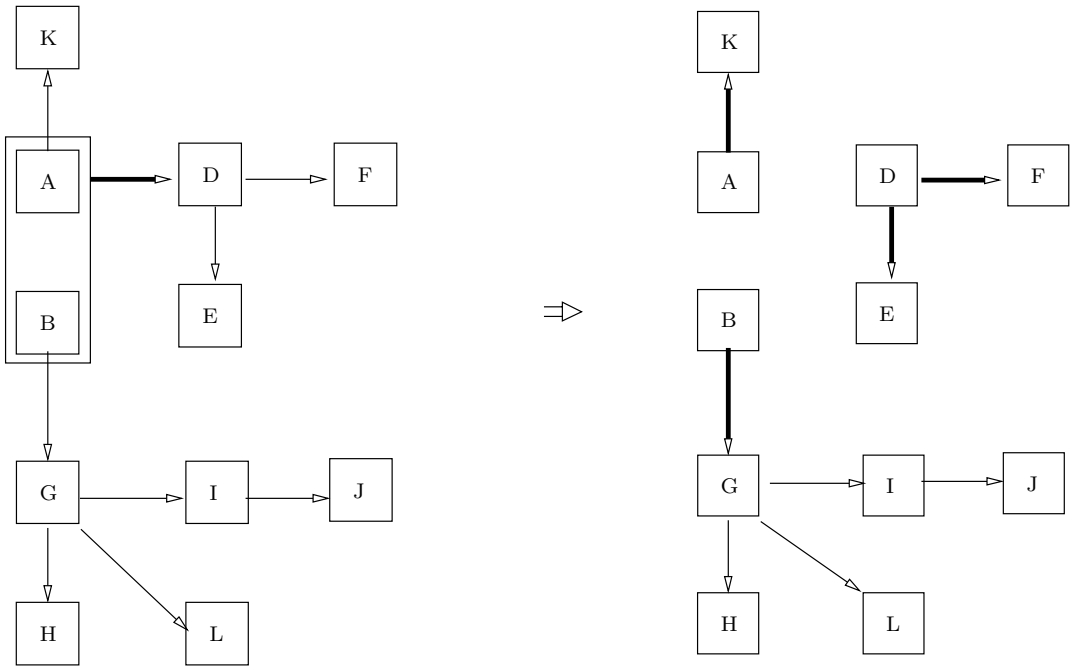

Figura 4.6: primo e secondo passo

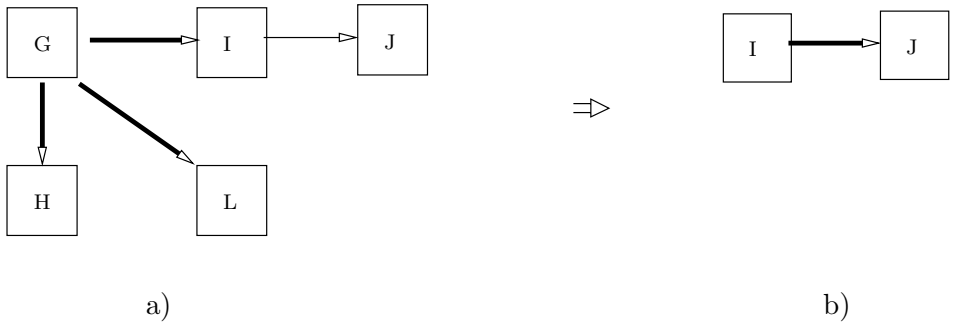

Figura 4.7: terzo e quarto passo

Abbiamo ottenuto le seguenti relazioni :

R_1 *(*$\underline{A,B}$*,D)*

R_2 *(*$\underline{A}$*,K)*

R_3 *(*$\underline{B}$*,G)*

R_4 *(*$\underline{D}$*,E,F)*

R_5 *(*$\underline{G}$*,H,I,L)*

R_6 *(*$\underline{I}$*,L)*

visto che X_1 *contiene* $\{A,B\}$ *non è necessario aggiungere una nuova relazione contenente la chiave candidata.*

4.5 Normalizzazione di schemi ER

Le dipendenze funzionali e le forme normali possono essere individuate anche su schemi ER. Vediamo un esempio:

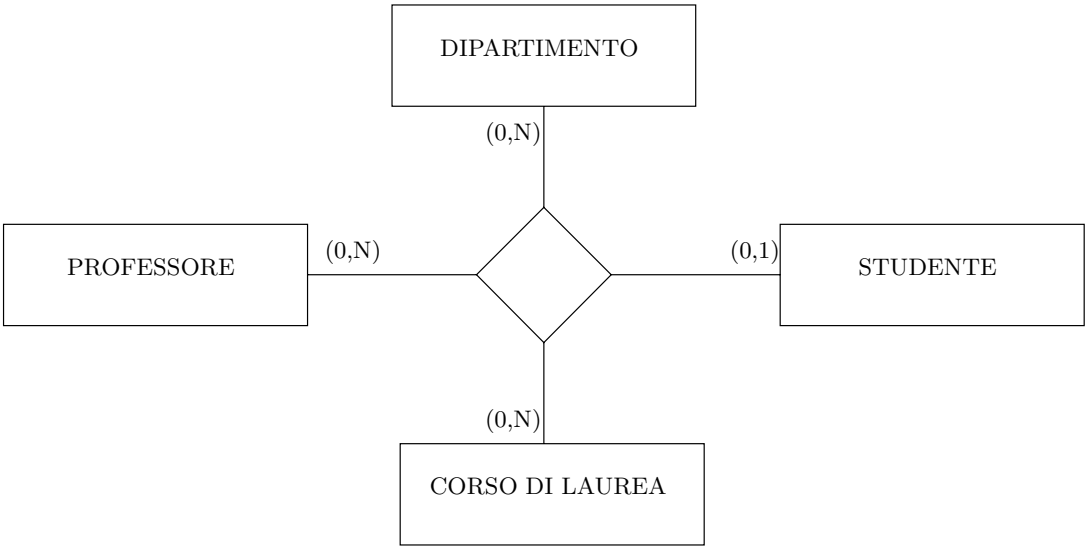

Figura 4.8: Schema ER con relazione quaternaria ridondante

Supponiamo poi che tutte le entità siano dotate di chiave primarie singole (formate da un singolo attributo). La relazione, chiamiamola X, sarà X={ STUDENTE, CORSO, PROFESSORE, DIPARTIMENTO }. Individuiamo inoltre le seguenti dipendenze funzionali:

$STUDENTE \rightarrow CORSO$

$STUDENTE \rightarrow PROFESSORE$

$PROFESSORE \rightarrow DIPARTIMENTO$

La chiave candidata di X sarà quindi $\{S\}$, in quanto $\{S\}^+ = X$. Verifichiamo quindi se la relazione X risulta essere in $BCNF$:

$S \rightarrow C$ $\surd$

$S \rightarrow P$ $\surd$

$P \rightarrow D$ NO!

La relazione X non passa quindi il test $BCNF$ e necessita di una decomposizione come mostrato in Figura 4.9.

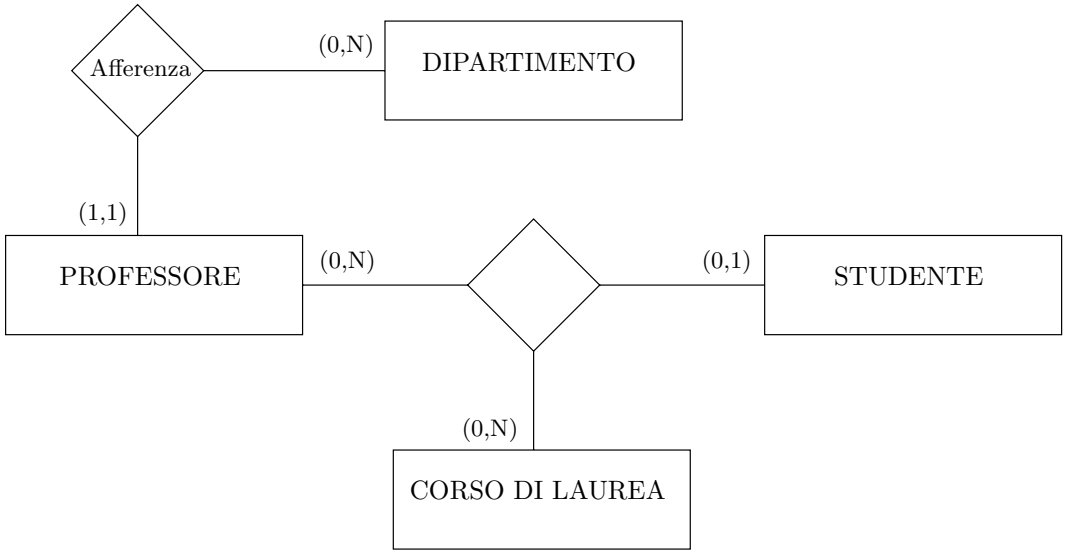

Figura 4.9: schema ER decomposto