Basi di Dati
La teoria della normalizzazione
-- seconda parte --

Corso B

Esempio

ESAMI(MATR, NomeS, IndirizzoS, CAPS, CodiceFiscaleS, DataNascitaS, Corso, Voto, DataEsame, CodProf, NomeProf, Qualifica, TipoUfficio)

Notazione abbreviata:

ESAMI(MATR, NS, IS, CAP, CF, DN, Co, Vo, DE, CP, NP, Q, TU)

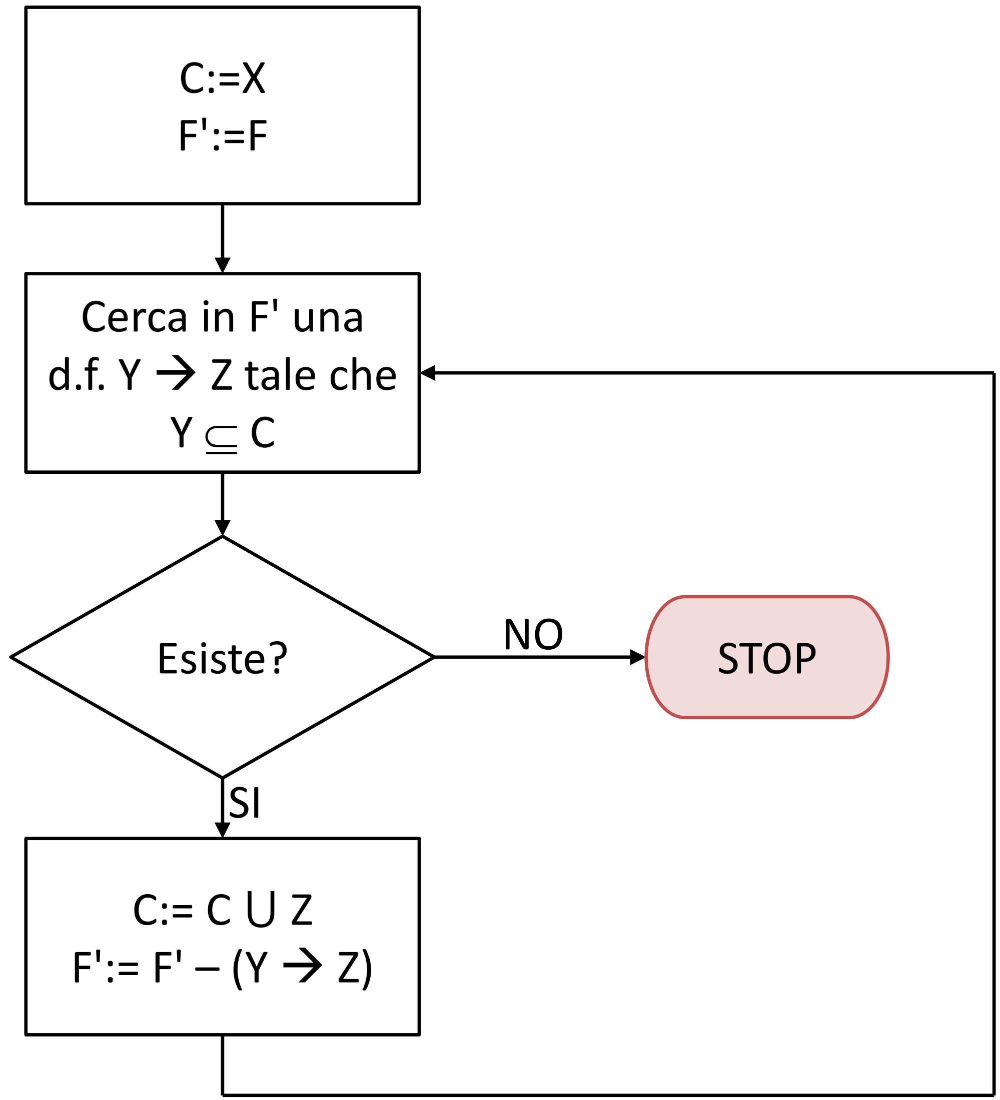
Esempi di vincoli

ESAMI(<u>MATR</u>, NS, IS, CAP, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q, TU)

Dipendenze funzionali:

- MATR → NS, IS, CF, DN
- CF → MATR
- IS → CAP
- MATR, Co → Vo, DE, CP
- CP → NP, Q
- Q → TU
- CP → Co

Algoritmo per il calcolo di X^+



Utilizzo dell'algoritmo

- Dati due insiemi di dipendenze funzionali *F* e *G*, posso usare l'algoritmo su ogni dipendenza funzionale g di *G* per testare se *g* è deducibile da *F*

- Faccio lo stesso per ogni dipendenza funzionale *f* di *F* per verificare che *f* sia deducibile da *G*

Proprietà dell'algoritmo

Dati due insiemi *F* e *G* di dipendenze funzionali equivalenti, dato un insieme *X*, allora la sua chiusura X^+ sarà la stessa sia che venga calcolata a partire da *F*, sia che venga calcolata a partire da *G*, ovvero $X^+_F = X^+_G$

Cioè:

$$X^+_G \subseteq X^+_F \text{ e } X^+_F \subseteq X^+_G$$

Proprietà (dimostrazione)

Dimostriamo $X^+_F \subseteq X^+_G$

- Prendo un attributo $A_i \in X^+_F$
- Questo significa che $F \vdash X \rightarrow A_i$
- F e G sono però equivalenti, quindi $X \rightarrow A_i \in G^+$
- Quindi $A_i \in X^+_G$

In maniera simile dimostro che $X^+_G \subseteq X^+_F$

Conseguenza della proprietà

Ogni volta che due progettisti lavorano con dipendenze funzionali equivalenti, fanno esattamente gli stessi ragionamenti sui vincoli della loro base di dati

Applicazione: ricerca chiavi

Ricordiamo la definizione di superchiave

L'insieme $K \subseteq A$ è una superchiave se:

$$\forall t_i, t_j \in r \; (t_i[K] = t_j[K]) \Rightarrow (t_i[A] = t_j[A])$$

La definizione è del tutto identica alla definizione di vincolo di dipendenza funzionale $K \rightarrow A$

Nuova definizione di superchiave

Data una relazione $r(A)$, un insieme di attributi $K \subseteq A$ è **superchiave** quando nella relazione r vale il vincolo di dipendenza funzionale $K \rightarrow A$

Ricerca chiavi

- Il progettista determina l'insieme di vincoli di dipendenza funzionale a partire dalla sua analisi della realtà
- Nella fase di ricerca delle chiavi candidate (e in particolare delle chiavi primarie), se il lavoro di analisi della realtà è fatto bene, il progettista può avvalersi dell'insieme delle dipendenze funzionali
- Non è detto che il vincolo di superchiave emerga **sintatticamente** dall'insieme di dipendenze

Ricerca delle chiavi

- Il progettista, dopo aver analizzato la realtà, fa delle congetture e le verifica
- Ad esempio, il progettista può congetturare che K sia chiave della relazione
- Si calcola la chiusura K^+ di K e si verifica che $K^+ = A$

Esempio

Congettura: MATR è superchiave?

- La chiusura di MATR è {MATR,NS,IS,CAP,CF,DN}
- La congettura è quindi sbagliata

Congettura: {MATR,Co,NS} è superchiave?

Verifichiamolo!

Esempio

Applicazione dell'algoritmo con X={MATR,Co,NS}

- Il sottoinsieme {MATR} ci dà {MATR,NS,IS,CF,DN}, quindi

  C={MATR,NS,IS,CF,DN,Co}

- {IS} ci dà CAP, quindi

  C={MATR,NS,IS,CAP,CF,DN,Co}

- {MATR,Co} ci dà {Vo,DE,CP}, quindi

  C={MATR,NS,IS,CAP,CF,DN,Co,Vo,DE,CP}

- {CP} ci dà {NP,Q} quindi

  C={MATR,NS,IS,CAP,CF,DN,Co,Vo,DE,CP,NP,Q}

- {Q} ci dà {TU}, quindi

  C={MATR,NS,IS,CAP,CF,DN,Co,Vo,DE,CP,NP,Q,TU}

Esempio

Abbiamo quindi che

$K=\{MATR,Co,NS\}$

$K^+=\{MATR,NS,IS,CAP,CF,DN,Co,Vo,DE,CP,NP,Q,TU\}=A$

cioè: $K \rightarrow A$

Quindi $K=\{MATR,Co,NS\}$ è **superchiave**

Chiave candidata

La chiave è una superchiave minimale, bisogna quindi ancora verificare questa proprietà

- Provo la chiusura {NS,Co}, ma la proprietà di superchiave non è verificata (**esercizio**)
- Provo la chiusura {MATR,Co} e si verifica che è superchiave (**esercizio**)

Problema

Un bravo progettista, scoperta una chiave candidata non si ferma ma si chiede se non ce ne siano anche altre

Nel nostro esempio avremmo:

- {MATR,Co}
- {CF,Co}

ma anche

- {MATR,CP}
- {CF,CP}

Problema

Verifica di *K={MATR,CP}*

- MATR ci dà {NS,IS,CAP,CF,DN}
- CP ci dà {NP,Q,**Co**}

Quindi ho {MATR,Co}, cioè ricado nell'esempio precedente

Quindi anche in questo caso $K^{+}=A$
(nella relazione ESAMI ho quattro proposte diverse di identificazione delle tuple)

Proprietà

Se due progettisti identificano due insiemi di dipendenze funzionali F e G equivalenti, i due progettisti arriveranno alle **stesse identiche** chiavi candidati

Esempio

Immaginiamo ora una relazione conto corrente CC così composta:

CC(Titolare, NumeroConto, NumeroAgenzia, CittaAgenzia, DirettoreAgenzia, SaldoCC, DataMovimentazione)

Abbreviando:

CC(T,NCC,NA,CA,DA,SCC,DM)

Esempio

Cerchiamo ora delle dipendenze funzionali significative

- NCC → NA
- NCC → SCC,DM
- NA,CA → DA (il direttore è l'unico direttore dell'agenzia)
- DA → CA (un direttore può dirigere solo agenzie della stessa città)

Esempio

Notiamo che l'attributo *Titolare* non appare mai nelle dipendenze funzionali

Posso infatti presumere che una persona possa essere titolare di più conti correnti (non ho quindi il vincolo Titolare → NCC)

Viceversa, uno stesso conto corrente può essere intestato a più persone (quindi non ho il vincolo NCC→Titolare)

Esempio

Il progettista deve usare il procedimento descritto precedentemente per la ricerca delle chiavi candidate

Titolare non appare mai nelle dipendenze funzionali, di conseguenza, *Titolare* **deve far parte** delle chiavi candidate!

Posso quindi considerare $K=\{T,NCC\}$

N.B. (importante)

1. Gli attributi coinvolti dalle dipendenze funzionali **non necessariamente** coprono tutti gli attributi della relazione

2. Gli attributi della relazione non coinvolti dalle dipendenze funzionali **devono sempre** far parte delle chiavi candidate congetturate

Normalizzazione

- Abbiamo introdotto i vincoli delle dipendenze funzionali per caratterizzare le molecole informative come prerequisito per la normalizzazione

- La normalizzazione consisterà nel partire da relazioni (come ESAMI) e dividerle in schemi di relazione che minimizzino le anomalie

- Tralasceremo la teoria delle dipendenze funzionali ci concentreremo sul problema della scomposizione

Esempio

Consideriamo una versione "semplificata" della relazione ESAMI, chiamata *S*

MATR	NomeS	Voto	Corso	CodC	Titolare
31	Piero	27	Base Dati	01	Pensa
31	Piero	26	Algoritmi	02	Damiani
37	Giovanni	25	Base Dati	03	Demo
…	…	…	…	…	…

Immaginiamo di sostituire la relazione con due relazioni scomponendola

Esempio

- Relazione S_1

MATR	NomeS	Voto	Corso
31	Piero	27	Base Dati
31	Piero	26	Algoritmi
37	Giovanni	25	Base Dati
…	…	…	…

- Relazione S_2

Corso	CodC	Titolare
Base Dati	01	Pensa
Algoritmi	02	Damiani
Base Dati	03	Demo
…	…	…

(per ottenere le tuple, è sufficiente proiettare sullo schema di S_1 e di S_2)

Problema

La base dati che ottengo dopo la scomposizione è equivalente dal punto di vista informativo alla base dati originale?

Problema

Per esempio la prima tupla di S ci dice che lo studente Piero ha dato l'esame di Basi di Dati con Pensa

In S1, posso usare Corso come ponte per aggiungere le informazioni mancanti (ad esempio il nome del profossere)

Posso quindi "ricomporre" le informazioni utilizzando il **natural-join**

Problema

Calcolo quindi $S_1 \bowtie S_2$

MATR	NomeS	Voto	Corso
31	Piero	27	Base Dati
31	Piero	26	Algoritmi
37	Giovanni	25	Base Dati
…	…	…	…

$\bowtie$

Corso	CodC	Titolare
Base Dati	01	Pensa
Algoritmi	02	Damiani
Base Dati	03	Demo
…	…	…

MATR	NomeS	Voto	Corso	CodC	Titolare
31	Piero	27	Base Dati	01	Pensa
31	Piero	27	Base Dati	03	Demo
31	Piero	26	Algoritmi	02	Damiani
37	Giovanni	25	Base Dati	01	Pensa
37	Giovanni	25	Base Dati	03	Demo

Problema

Confrontiamo le due relazioni

MATR	NomeS	Voto	Corso	CodC	Titolare
31	Piero	27	Base Dati	01	Pensa
31	Piero	26	Algoritmi	02	Damiani
37	Giovanni	25	Base Dati	03	Demo
…	…	…	…	…	…

MATR	NomeS	Voto	Corso	CodC	Titolare
31	Piero	27	Base Dati	01	Pensa
31	Piero	27	Base Dati	03	Demo
31	Piero	26	Algoritmi	02	Damiani
37	Giovanni	25	Base Dati	01	Pensa
37	Giovanni	25	Base Dati	03	Demo

Ho due nuove tuple che prima non esistevano

Tuple spurie

- Se ho una base di dati composta da *S* e la sostituisco con le due relazioni S_1 ed S_2, il natural join **perde** informazioni (pur avendo tuple in più), perché introduce le cosiddette **tuple spurie**

- Le tuple spurie sono indistinguibili da quelle originarie (non so più decidere se Piero ha dato l'esame con Demo o Pensa)

Decomposizione con join con perdita di informazione

Considero uno schema di relazione *R(A)*, considero due sottoinsiemi di attributi $A_1 \subseteq A$ e $A_2 \subseteq A$, con $A_1 \cup A_2 = A$

Considero l'istanza di relazione r(A)

Decompongo lo schema di partenza nelle due relazioni:

- $r_1(A_1) = \Pi_{A_1}(r(A))$
- $r_2(A_2) = \Pi_{A_2}(r(A))$

In generale (lemma):

$$r(A) \subseteq r_1(A_1) \bowtie r_2(A_2)$$

Tuple spurie

La differenza insiemistica

$$r_1(A_1) \bowtie r_2(A_2) - r(A)$$

Consiste nell'insieme delle tuple spurie di $r_1(A_1) \bowtie r_2(A_2)$

Decomposizione con join senza perdita di informazioni

<u>Definizione</u>

Data uno schema di relazione $R(A)$, dati due sottoinsiemi di attributi $A_1 \subseteq A$ e $A_2 \subseteq A$, con $A_1 \cup A_2 = A$ $\{R_1(A_1), R_2(A_2)\}$ è una decomposizione senza perdita di informazione se **per ogni** istanza $r(A)$ di $R(A)$ vale

$$r(A) = r_1(A_1) \bowtie r_2(A_2)$$

dove

- $r_1(A_1) = \Pi_{A_1}(r(A))$
- $r_2(A_2) = \Pi_{A_2}(r(A))$

Esempio

Immaginiamo che il codice del corso identifichi il concetto di corso

Consideriamo quindi un nuovo sottoinsieme per S_1 con CodC al posto di Corso:

MATR	**NomeS**	**Voto**	**CodC**
31	Piero	27	01
31	Piero	26	02
37	Giovanni	25	03
…	…	…	…

Esempio

Calcolo quindi nuovamente $S_1 \bowtie S_2$

MATR	NomeS	Voto	CodC
31	Piero	27	01
31	Piero	26	02
37	Giovanni	25	03
…	…	…	…

$\bowtie$

Corso	CodC	Titolare
Base Dati	01	Pensa
Algoritmi	02	Damiani
Base Dati	03	Demo
…	…	…

MATR	NomeS	Voto	Corso	CodC	Titolare
31	Piero	27	Base Dati	01	Pensa
31	Piero	26	Algoritmi	02	Damiani
37	Giovanni	25	Base Dati	03	Demo

Questa volta la relazione originale è ricostruita senza perdita di informazioni (senza tuple spurie)

Teorema

Sia R(A) uno schema con dipendenze funzionali F, decomposto in $\{R_1(A_1), R_2(A_2)\}$ dove $A_1 \cup A_2 = A$

La decomposizione di ogni istanza **corretta**(*) r(A) di R(A) è **senza perdita di informazione** se e solo se (condizione necessaria e sufficiente)

$$F \vdash A_1 \cap A_2 \rightarrow A_1 \vee F \vdash A_1 \cap A_2 \rightarrow A_2$$

* r(A) soddisfa le dipendenze funzionali F

Teorema (riformulazione)

Sia R(A) uno schema con dipendenze funzionali F, decomposto in $\{R_1(A_1), R_2(A_2)\}$ dove $A_1 \cup A_2 = A$

La decomposizione di ogni istanza **corretta**(*) r(A) di R(A) è **senza perdita di informazione** se e solo se (condizione necessaria e sufficiente)

$$(A_1 \cap A_2 \text{ superchiave di } A_1) \vee (A_1 \cap A_2 \text{ superchiave di } A_2)$$

* r(A) soddisfa le dipendenze funzionali F

Esempio

$\{MATR, NomeS, Voto, CodC\} \cap \{CodC, Corso, Titolare\} = \{CodC\}$

Ma abbiamo detto che CodC è superchiave di S_2, cioè CodC $\rightarrow$ Corso, Titolare

Quindi la decomposizione è senza perdita di informazione

Dimostrazione

Dimostreremo solo la condizione di sufficienza

Ipotesi: $F \vdash A_1 \cap A_2 \rightarrow A_1 \vee F \vdash A_1 \cap A_2 \rightarrow A_2$

Tesi: $r(A) = r_1(A_1) \bowtie r_2(A_2)$ ovvero

- $r(A) \subseteq r_1(A_1) \bowtie r_2(A_2)$
- $r_1(A_1) \bowtie r_2(A_2) \subseteq r(A)$

Il lemma garantisce che $r(A) \subseteq r_1(A_1) \bowtie r_2(A_2)$, quindi la prima inclusione è provata

Dimostriamo quindi che $r_1(A_1) \bowtie r_2(A_2) \subseteq r(A)$

Dimostrazione

Ipotesi: $F \vdash A_1 \cap A_2 \rightarrow A_1 \vee F \vdash A_1 \cap A_2 \rightarrow A_2$

Tesi: $r_1(A_1) \bowtie r_2(A_2) \subseteq r(A)$

Preso un elemento qualsiasi dell'insieme a sinistra devo dimostrare che è incluso nell'insieme di destra

Introdurremo una rappresentazione grafica per facilitare la dimostrazione

Dimostrazione

Consideriamo $R_1(A_1)$, comprendente anche gli attributi $A_1 \cap A_2$

	A1 – A2	**A1 ∩ A2**
t_1	- - -	///

Consideriamo $R_2(A_2)$, comprendente anche gli attributi $A_1 \cap A_2$

	A1 ∩ A2	**A2 – A1**
t_2	///	+ + +

Dimostrazione

Costruiamo un elemento generico del natural-join $r_1(A_1) \bowtie r_2(A_2)$

	A1 – A2	A1 ∩ A2	A2 – A1
t	- - -	///	+ + +

Dobbiamo verificare che la tupla *t* sia contenuta in *r(A)*

Dimostrazione

Partizioniamo gli attributi di R(A) in tre gruppi: attributi solo di A_1, attributi solo di A_2, attributi in comune

Dobbiamo ricordare che

- $r_1(A_1) = \Pi_{A_1}(r(A))$
- $r_2(A_2) = \Pi_{A_2}(r(A))$

t_1 sarà un frammento di una tupla t'_1 di $r(A)$

	A1 – A2	A1 ∩ A2	A2 – A1
t'_1	- - -	///	???

Dimostrazione

Partizioniamo gli attributi di R(A) in tre gruppi: attributi solo di A_1, attributi solo di A_2, attributi in comune

Dobbiamo ricordare che

- $r_1(A_1) = \Pi_{A_1}(r(A))$
- $r_2(A_2) = \Pi_{A_2}(r(A))$

t_2 sarà un frammento di una tupla t'_2 di $r(A)$

	A1 – A2	**A1 ∩ A2**	**A2 – A1**
t'_1	- - -	///	**???**
t'_2	**???**	///	+ + +

Dimostrazione

Ricordiamo l'ipotesi: $F \vdash A_1 \cap A_2 \rightarrow A_2$

La relazione r(A) di partenza è una relazione corretta per ipotesi, quindi il vincolo $A_1 \cap A_2 \rightarrow A_2$ dev'essere verificato in r(A)

	A1 – A2	A1 ∩ A2	A2 – A1
t'_1	- - -	///	???
t'_2	???	///	+ + +

Questo vuol dire che se $t'_1[A_1 \cap A_2] = t'_2[A_1 \cap A_2]$, allora $t'_1[A_2] = t'_2[A_2]$, di conseguenza ??? = + + +

Dimostrazione

Ricordiamo l'ipotesi: $F \vdash A_1 \cap A_2 \rightarrow A_2$

La relazione r(A) di partenza è una relazione corretta per ipotesi, quindi il vincolo $A_1 \cap A_2 \rightarrow A_2$ dev'essere verificato in r(A)

	A1 – A2	A1 ∩ A2	A2 – A1
t'_1	- - -	///	+ + +
t'_2	???	///	+ + +

Questo vuol dire che se $t'_1[A_1 \cap A_2] = t'_2[A_1 \cap A_2]$, allora $t'_1[A_2] = t'_2[A_2]$, di conseguenza ??? = + + +

Dimostrazione

Confrontiamo con $r_1(A_1) \ltimes r_2(A_2)$:

	A1 – A2	A1 ⋂ A2	A2 – A1
t	- - -	///	+ + +

Ho verificato che l'elemento *t* di $r_1(A_1) \ltimes r_2(A_2)$ è contenuto in $r(A)$

	A1 – A2	A1 ⋂ A2	A2 – A1
t'_1	- - -	///	+ + +
t'_2	???	///	+ + +

[CVD]

Restrizione

- Consideriamo una relazione R(A) abbinata alle sue dipendenze funzionali F
- Immaginiamo di decomporre la relazione in $R_1(A_1)$ *e* $R_2(A_2)$
- A questo punto potremmo voler riscrivere le dipendenze funzionali F in modo che ad ogni relazione sia associato il suo insieme di dipendenze funzionali ($R_1(A_1)$, *con* F_1 *e* $R_2(A_2)$ *con* F_2) che discendano dalla relazione di partenza
- F_1 è la **restrizione** di F in R_1, F_2 restrizione di F in R_2

Restrizione (definizione)

Data una decomposizione $R_i(A_i)$ di $R(A)$, l'insieme F_i è definito come

$$F_i = \{ X \rightarrow Y \mid (F \vdash X \rightarrow Y) \wedge (X,Y \subseteq A_i) \}$$

Perché non prendere semplicemente le dipendenze F e trasferirle in base agli attributi interessati?

A cosa serve questa definizione?

Caso particolare 1

- Consideriamo una relazione $R(\ldots,A,B,C,\ldots)$ con l'insieme di d.f. $F = \{A \rightarrow B, B \rightarrow C\}$

- Consideriamo ora una decomposizione in $R_1(\ldots,A,B,\ldots)$ e $R_2(\ldots,B,C,\ldots)$

- Viene naturale immaginare quindi: $F_1 = \{A \rightarrow B\}$ e $F_2 = \{B \rightarrow C\}$

Caso particolare 2

- Consideriamo una relazione $R(\ldots,A,B,C,\ldots)$ con l'insieme di d.f. $F = \{A \rightarrow B, B \rightarrow C\}$
- Consideriamo ora una decomposizione in $R_1(\ldots,A,C,\ldots)$ e $R_2(\ldots,B,C,\ldots)$
- Si riesce subito a costruire $F_2 = \{B \rightarrow C\}$
- Ma per $R_1(\ldots,A,C,\ldots)$?

Caso particolare 2

- Consideriamo una relazione $R(\ldots,A,B,C,\ldots)$ con l'insieme di d.f. $F = \{A \rightarrow B, B \rightarrow C\}$

- Consideriamo ora una decomposizione in $R_1(\ldots,A,C,\ldots)$ e $R_2(\ldots,B,C,\ldots)$

- Si riesce subito a costruire $F_2 = \{B \rightarrow C\}$ ma per $R_1(\ldots,A,C,\ldots)$?
- Apparentemente non ci sono d.f. in R_1

Caso particolare 2

Apparentemente non ci sono d.f. in R_1

E' un errore perché da $F = \{A \rightarrow B, B \rightarrow C\}$ è deducibile $A \rightarrow C$

Quindi $F_1 = \{A \rightarrow C\}$ come conseguenza logica di F

Decomposizione che mantiene le dipendenze

Data una relazione $R(A)$ con dipendenze funzionali F, decomposta in $R_1(A_1)$ con la restrizione F_1 e $R_2(A_2)$ con la restrizione F_2, la decomposizione $\{R_1, R_2\}$ mantiene le dipendenze quando $F_1 \cup F_2 \vdash F$

Caso particolare 2

- Consideriamo nuovamente la relazione $R(\ldots,A,B,C,\ldots)$ con l'insieme di d.f. $F = \{A \rightarrow B, B \rightarrow C\}$

- Consideriamo la decomposizione in $R_1(\ldots,A,C,\ldots)$ e $R_2(\ldots,B,C,\ldots)$ *con le restrizioni* $F_1 = \{A \rightarrow C\}$ e $F_2 = \{B \rightarrow C\}$

- Non riesco in nessun modo a dedurre $A \rightarrow B$ da $F_1 \cup F_2$

Vincoli locali di dipendenza

- Quando abbiamo presentato la definizione di decomposizione che mantiene le dipendenze, avremmo dovuto specificare: decomposizione che mantiene la **località** delle dipendenze

- Le dipendenze sono vincoli, se definiti su una sola relazione sono locali

- La verifica dei vincoli locali è più efficiente di quella dei vincoli globali

Mantenimento delle dipendenze

Il mantenimento della località delle dipendenze è una proprietà auspicabile in una decomposizione

- Nel caso particolare 1, nella decomposizione $\{R_1,R_2\}$ il vincolo $A \rightarrow B$ è un vincolo locale di R_1 e $B \rightarrow C$ è un vincolo locale di R_2, quindi $\{R_1,R_2\}$ è una decomposizione che mantiene la località di tutti i vincoli
- Nel caso particolare 2, il vincolo $A \rightarrow B$ **non è perso**, ma per farlo rispettare, il progettista deve determinare un vincolo globale (meno efficiente)

Conclusione

- l calcolo delle restrizioni richiede una complessità esponenziale (devo calcolare F^+)

- La verifica ex post della deducibilità di F da $F_1 \cup F_2$ ha complessità polinomiale (basta applicare l'algoritmo di chiusura)

Le forme normali

Le forme normali sono delle "ricette" di buona progettazione di basi di dati volte alla minimizzazione delle anomalie

Le forme normali

Esistono diverse forme normali:

- I^a forma normale (1NF) già spiegata: relazioni i cui valori sono associati a domini semplici
- II^a forma nomale (2NF): restrizione della 1NF (non la introdurremo, ha solo valore storico)
- III^a forma normale (3NF): restrizione della 2NF
- Boyce-Codd Normal Form (BCNF): restrizione della 3NF
- IV^a forma normale (4NF)
- Project-Join Normal Form (5NF o PJ/NF)
- Domain/Key Normal Form (DKNF)
- …

Le forme normali

- La 2NF, 3NF e BCNF sono forme normali basate sulle dipendenze funzionali
- La 4NF è stata introdotta su una classe di vincoli di cui non abbiamo parlato, i vincoli di dipendenza funzionale multi-valore
- La PJ/NF si basa su vincoli di decomposizione con join senza perdita
- La DKNF si basa su un ragionamento su chiavi e domini

Le forme normali

Tratteremo solo

- III^a forma normale
- Boyce-Codd Normal Form

Forma normale BCNF

Data una relazione *R(A)* in 1NF abbinata a delle dipendenze funzionali *F*, la relazione è in **BCNF** se, per ogni $X \rightarrow Y \in F$ si verifica una delle seguenti condizioni:

1. $Y \subseteq X$ ($X \rightarrow Y$ è una dipendenza **riflessiva**)
2. X è **superchiave** di R

Esempio

La relazione ESAMI non è in BCNF, in quanto esistono diverse dipendenze funzionali che non rispettano né la condizione 1 né la condizione 2

Consideriamo invece un sottoinsieme dello schema

STUDENTE(MATR,NS,CF,DN) con le d.f.

F = {MATR → CF,NS,DN; CF → MATR}

Analizzando la relazione troviamo, come chiave, sia MATR che CF

Le dipendenze funzionali sono tutte del tipo 2, quindi STUDENTE è una relazione in BCNF

Intuizione

- Una relazione è ben presentata quando contiene un unico concetto (molecola informativa) ancorché identificato con due chiavi diverse

- Esempio, la relazione STUDENTE coglie l'unico concetto di studente, ancorché identificato sia da MATR che da CF

Esempio

Consideriamo ora

CC(Titolare,NConto,NAgenzia,CittaAgenzia,Saldo)

$F = \{NConto \rightarrow NAgenzia, CittaAgenzia, Saldo\}$

- Limitandosi ad *F*, *CC* non è in BCNF
- Se però togliamo *Titolare*, *CC* è in BCNF

Esempio

Consideriamo ora un nuovo esempio:

CC(<u>NConto</u>,NAgenzia,CittaAgenzia,Saldo)

$F_{CC} = \{NConto \rightarrow NAgenzia, CittaAgenzia, Saldo\}$

TITOLARI(<u>Titolare,NConto</u>)

$F_{TITOLARI} = \{ \emptyset \}$

(tralasciamo le riflessive, dette anche **banali**)

Si può dire che *TITOLARI* è in BCNF?

Esempio

Consideriamo ora un nuovo esempio:

CC(<u>NConto</u>,NAgenzia,CittaAgenzia,Saldo)

$F_{CC} = \{NConto \rightarrow NAgenzia, CittaAgenzia, Saldo\}$

TITOLARI(<u>Titolare,NConto</u>)

$F_{TITOLARI} = \{ \varnothing \}$

(tralasciamo le riflessive, dette anche **banali**)

Si può dire che *TITOLARI* è in BCNF?

Sì, tutte le dipendenze sono del tipo 1

BCNF e anomalie

- La BCNF, rispetto alle dipendenze funzionali non presenta più anomalie di inserzione, cancellazione e di update

- La proprietà è valida localmente, ovvero all'interno della stessa relazione

- Globalmente possono ancora esserci delle anomalie

Normalizzazione in BCNF

La caratteristica della BCNF è quella di avere una procedura affiancata alla definizione che permette di arrivare a relazioni in BCNF partendo da qualsiasi relazione

Boyce e Codd hanno infatti proposto una procedura di normalizzazione che parte dall'intero schema di *SC* di una base di dati, definito come un insieme di relazioni abbinate alle dipendenze funzionali

$$SC = \{ (R_i(A_i), F_i) \}$$

Procedura di normalizzazione

1. Parto dallo schema di basi di dati $SC = \{(R_i(A_i),F_i)\}$
2. Cerco, all'interno di SC, una $(R_i(A_i),F_i)$ non in BCNF
   - *cioè, cerco in* F_i *almeno una d.f.* $X \rightarrow Y$ *non BCNF ovvero*

     $(Y \not\subset X) \wedge (X \text{ non superchiave di } R_i(A_i))$

3. Se non esiste una $X \rightarrow Y$ non BCNF mi fermo (*SC* è già in BCNF)
4. Se esiste una $X \rightarrow Y$ non BCNF occorre effettuare una trasformazione dello schema

Modifica dello schema

a. Tolgo la relazione non in BCNF da SC
   – SC: = SC – (R_i(A_i),F_i)
b. Aggiungo ad SC due nuove relazioni con le relative dipendenze funzionali
   – $SC := SC \cup (R_1(\boldsymbol{A_i - Y}), F_1)$
   – $SC := SC \cup (R_2(\boldsymbol{XY}), F_2)$

   dove F_1 è la restrizione di F_i in $R_1(A_i – Y)$ ed F_2 è la restrizione di F_i in $R_2(XY)$
c. Torno al passo 2 della procedura

Procedura completa

1. Parto dallo schema di basi di dati $SC = \{(R_i(A_i),F_i)\}$
2. Cerco, all'interno di SC, una $(R_i(A_i),F_i)$ non in BCNF
   – *cioè, cerco in* F_i *almeno una d.f.* $X \rightarrow Y$ *non BCNF ovvero*

     $(Y \not\subset X) \wedge (X \text{ non superchiave di } R_i(A_i))$

3. Se non esiste una $X \rightarrow Y$ non BCNF mi fermo (*SC* è già in BCNF)
4. Se esiste una $X \rightarrow Y$ non BCNF occorre effettuare una trasformazione dello schema
   a. Tolgo la relazione non in BCNF da SC
      - $SC: = SC - (R_i(A_i),F_i)$
   b. Aggiungo ad SC due nuove relazioni con le relative dipendenze funzionali
      - $SC: = SC \cup (R_1(\boldsymbol{A_i - Y}), F_1)$
      - $SC: = SC \cup (R_2(\boldsymbol{XY}), F_2)$

      dove F_1 è la restrizione di F_i in $R_1(A_i - Y)$ ed F_2 è la restrizione di F_i in $R_2(XY)$
   c. Torno al passo 2 della procedura

Proprietà della procedura

La procedura prevede una decomposizione effettuata con join senza perdita

- Considero $(R_i(A_i),F_i)$, in F_i esiste una dipendenza funzionale $X \rightarrow Y$ non in BCNF
- La decomposizione in $R_1(\mathbf{A_i} - \mathbf{Y})$ e $R_2(\mathbf{XY})$ è una decomposizione con join senza perdita, infatti
- $(A_i - Y) \cap XY = X$ (X sono attributi anche di A_i)
- $X \rightarrow Y$ è una dipendenza di F_i quindi $(A_i - Y) \cap XY \rightarrow Y$
- Applico l'espansione con $W=X$ e ottengo $(A_i - Y) \cap XY \rightarrow XY$
- Ma XY è esattamente lo schema di R_2, quindi $(A_i - Y) \cap XY$ è superchiave di R_2

[CVD]

Proprietà della procedura

Grazie al teorema, possiamo concludere che il passo 4 della procedura attua una decomposizione con join senza perdita, quindi il natural-join su R_1 ed R_2 ricostruisce $R_i(A_i)$ senza perdita di informazione (senza tuple spurie)

Esempio

STUDENTI(MATR,NS,IS,CAP,PROV)

F = {MATR → NS,IS; IS → CAP; IS → PROV}

- Bisogna per prima cosa aver chiare le chiavi
- In questo caso l'unica chiave è MATR
- IS → CAP e IS → PROV non sono in BCNF
- Quale scelgo? Dal punto di vista della procedura posso sceglierne una qualsiasi
- Scelgo, ad esempio, IS → PROV

Tecnica di applicazione manuale

- Dal punto di vista manuale la R_1 è la relazione di partenza senza l'attributo *Y*, cancellando anche la dipendenza funzionale presa in esame
  - R_1(MATR,NS,IS,CAP)
  - F_1 = {MATR → NS,IS; IS → CAP}
- Ora posso scrivere la R_2
  - R_2(IS,PROV)
  - F_2={IS → PROV}
- Ora si può iniziare da capo
  - La relazione R_2 è già in BCNF
  - La relazione R_1 non lo è (IS non è chiave)

Tecnica di applicazione manuale

- Considero quindi la dipendenza funzionale IS → CAP
- Riscrivo R_1
  - R_1(MATR,NS,IS)
  - F_1={MATR → NS,IS}
- Scrivo R3
  - R_3(IS,CAP)
  - F_3 = {IS → CAP}

- Ora abbiamo tre relazioni, R_1(MATR,NS,IS), R_2(IS,PROV) ed R_3(IS,CAP) tutte e tre in BCNF

Verifica finale

- Consideriamo le tre restrizioni delle dipendenze funzionali *F* (F_1, F_2 ed F_3) e ne facciamo l'unione: dobbiamo ritrovare F

- In questo caso siamo fortunati, abbiamo cioè una decomposizione con il mantenimento della località di tutte le dipendenze funzionali

Altro esempio

STUDENTI(MATR,NS,IS,CAP,PROV)

F = {MATR $\rightarrow$ NS,IS; IS $\rightarrow$ CAP; **CAP $\rightarrow$ PROV**}

Abbiamo due dipendenze funzionali non in BCNF

- Scelgo IS $\rightarrow$ CAP
- Cancello CAP dalla relazione di partenza
  - R_1(MATR,NS,IS,PROV)
  - F_1 = {MATR $\rightarrow$ NS,IS}
- Costruisco R2
  - R_2(IS,CAP)
  - F_2 = {IS $\rightarrow$ CAP}

Altro esempio

R_1(MATR,NS,IS,PROV)

F_1 = {MATR $\rightarrow$ NS,IS}

Ci chiediamo ora, F_1 è una restrizione valida di *F*?

No, perché nelle dipendenze originali c'era la dipendenza funzionale dedotta: MATR$\rightarrow$ PROV (dedotta da F per transitività)

La restrizione funziona se all'interno di F_1 ci sono tutte le dipendenze date o **derivate** che fanno parte di R_1

La restrizione corretta è quindi

$$F_1 = \{MATR \rightarrow NS,IS; MATR \rightarrow PROV\}$$

Altro esempio

Nella relazione R_1, constatiamo che in F_1 le dipendenze sono tutte in BCNF

Ho quindi le due relazioni in BCNF

R_1(MATR,NS,IS,PROV) con F_1 = {MATR $\rightarrow$ NS,IS; MATR$\rightarrow$PROV}

R_2(IS,CAP) con F_2 = {IS $\rightarrow$ CAP}

Nelle restrizioni trovo solo dipendenza funzionale X $\rightarrow$ Y con X chiave

Verifica finale

- Consideriamo le due restrizioni delle dipendenze funzionali *F* (F_1, F_2) e ne facciamo l'unione

- Questa volta non ritrovo più l'insieme *F*, infatti:

$$F' = \{MATR \rightarrow NS,IS;\ MATR \rightarrow PROV;\ IS \rightarrow CAP\}$$

- Manca cioè CAP → PROV, che non è deducibile da F'
- Dal punto di vista della definizione la dipendenza è persa, ma dal punto di vista della progettazione, CAP → PROV diventa un vincolo globale (quindi costoso)

Esempio

Riprendiamo l'esempio

STUDENTI(MATR,NS,IS,CAP,PROV)

F = {MATR $\rightarrow$ NS,IS; IS $\rightarrow$ CAP; CAP $\rightarrow$ PROV}

Proviamo ad applicare il processo di normalizzazione partendo da CAP $\rightarrow$ PROV

R_1(MATR,NS,IS,CAP)

F_1 = {MATR $\rightarrow$ NS,IS; IS $\rightarrow$ CAP}

R_2(CAP,PROV)

Fy = {CAP $\rightarrow$ PROV}

Esempio

R_2 è in BCNF, considero quindi

R_1(MATR,NS,IS,CAP)

F_1 = {MATR → NS,IS; IS → CAP}

Ho solo IS → CAP che non è in BCNF, quindi posso trasformare

R_1(MATR,NS,IS)

F_1 = {MATR → NS,IS}

R_3(IS,CAP)

F_3 = {IS → CAP}

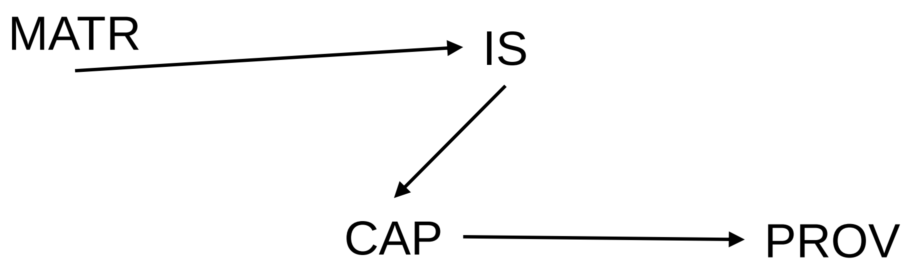
Verifica finale

- Le tre relazioni sono in BCNF
- Calcolo l'unione di F_1, F_2 ed F_3 e, in questo caso riottengo l'insieme F iniziale

  $$F' = \{MATR \rightarrow NS,IS;\ IS \rightarrow CAP;\ CAP \rightarrow PROV\}$$

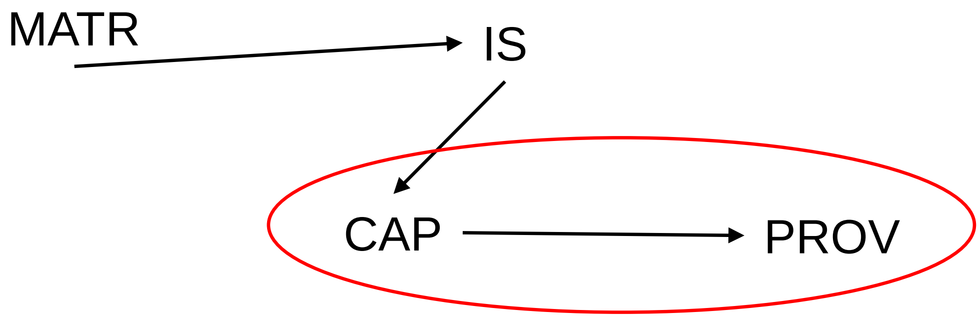
Raccomandazione

- Nella trasformazione in BCNF occorre effettuare delle scelte che portino a basi di dati che mantengano la località delle dipendenze funzionali
- Possiamo costruire il grafo delle dipendenze:



Euristica

Posso iniziare a decomporre la relazione a partire dalla "periferia" (o dalle foglie, se fosse un albero)



Terminazione

- La procedura garantisce la terminazione innanzitutto perché lavora su insiemi finiti di dipendenze funzionali
- Ci sono condizioni per cui si entra in loop senza fine?
- La terminazione è garantita dal passo di decomposizione

Terminazione

- Consideriamo la relazione $R_i(A_i)$ con la dipendenza funzionale non BCNF $X \rightarrow Y$ e le due relazioni risultanti dalla decomposizione: $R_1(A_i - Y)$ e $R_2(XY)$
- Consideriamo il grado (cardinalità dello schema) delle relazioni delle tre relazioni:
  - $|A_i - Y| < |A_i|$ per costruzione
  - Posso sostenere che anche $|XY| < |A_i|$ (strettamente) ?
- Partiamo dall'ipotesi che $|XY| = |A_i|$, ma dato che $XY \subseteq A_i$, questo implica che $XY = A_i$
- Abbiamo però $X \rightarrow Y$ (d.f. data) e applicando l'espansione abbiamo $X \rightarrow XY$, cioè $X \rightarrow A_i$
- Allora X è una superchiave (contraddizione con l'ipotesi)

Terminazione

- Il grado di R_1 e di R_2 è strettamente minore del grado di R_i
- Alla fine mi trovo con una relazione $R_n(\{A,B\})$ con A e B due attributi
- Le dipendenze funzionali saranno per forza della forma $A \rightarrow B$ o $B \rightarrow A$ quindi, per espansione $A \rightarrow AB$ o $B \rightarrow AB$ (con A o B superchiave)

- Quindi $R_n(\{A,B\})$ è in BCNF

Complessità di calcolo

- La complessità teorica dell'algoritmo è esponenziale
- La ricerca della dipendenza funzionale (passo 2) è polinomiale (per verificare che un insieme è superchiave basta applicare l'algoritmo di chiusura che è polinomiale)
- Il problema è il calcolo delle restrizioni F_1 ed F_2
- Il calcolo delle restrizioni, dal punto di vista algoritmico, è esponenziale: richiede il calcolo di F^+ per estrarre tutte le dipendenze in $A_i – Y$ e XY

Proprietà (ricapitolazione)

1. Il processo di normalizzazione procede con decomposizione con join senza perdita
2. La procedura termina in un numero finito di passi e termina correttamente
3. La complessità teorica dell'algoritmo è esponenziale
4. Lo schema finale dipende dalle scelte delle dipendenze funzionali da processare
5. Lo schema finale può mantenere la località delle dipendenze funzionali
6. In alcuni casi la normalizzazione genera necessariamente schemi con perdita della località delle dipendenze funzionali

Esempio su ESAMI

ESAMI(<u>MATR</u>, NS, IS, CAP, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q, TU)

F = { MATR → NS, IS, CF, DN

CF → MATR

IS → CAP

MATR, Co → Vo, DE, CP

CP → NP, Q

Q → TU

(*)~~CP → Co~~ }

con le chiavi {MATR,Co} e {CF,Co}

* in questo esempio non uso l'ultima dipendenza funzionale

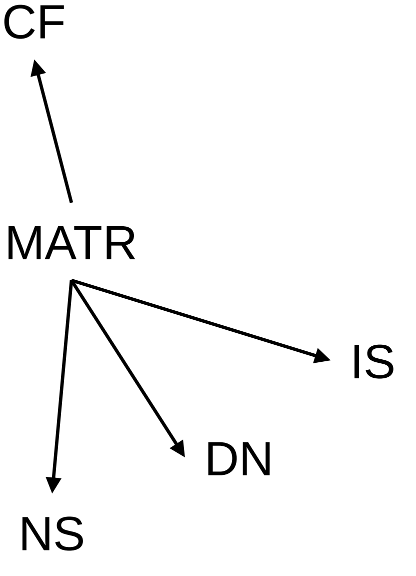
Esempio su ESAMI

Costruisco il grafo delle dipendenze

Esempio su ESAMI

Costruisco il grafo delle dipendenze

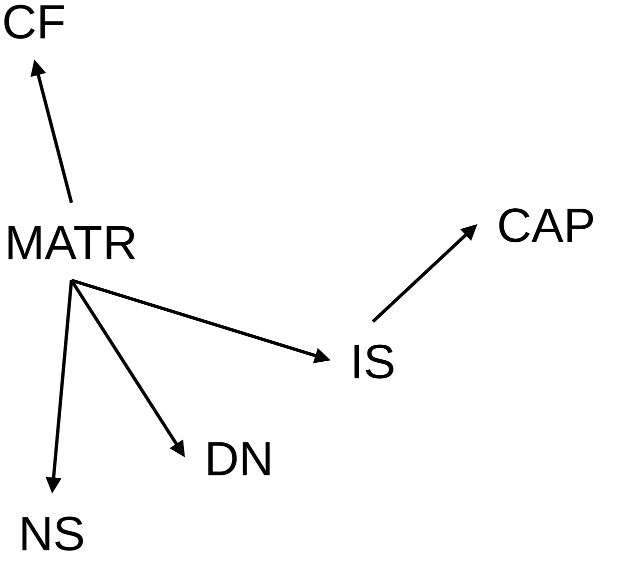
MATR→ NS,IS,CF,DN



Esempio su ESAMI

Costruisco il grafo delle dipendenze

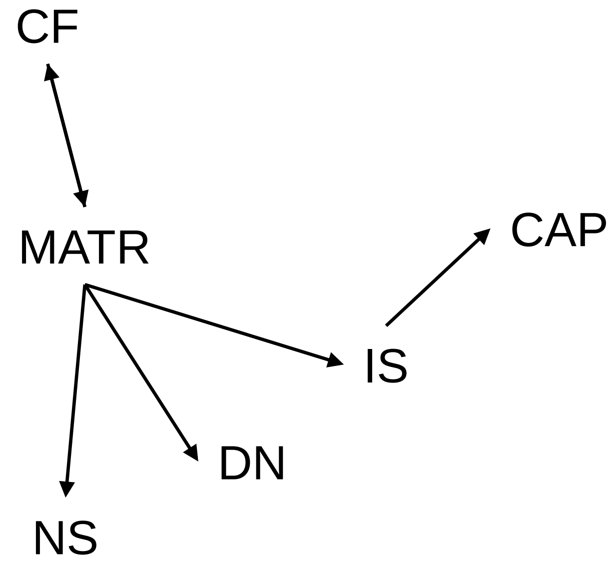
IS → CAP



Esempio su ESAMI

Costruisco il grafo delle dipendenze

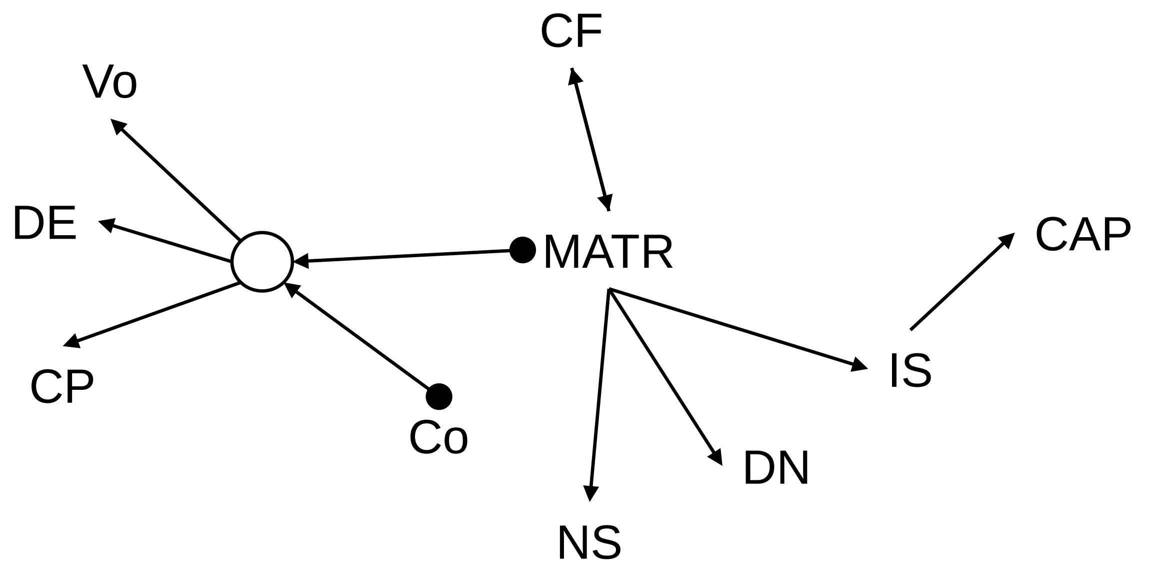
CF → MATR



Esempio su ESAMI

Costruisco il grafo delle dipendenze

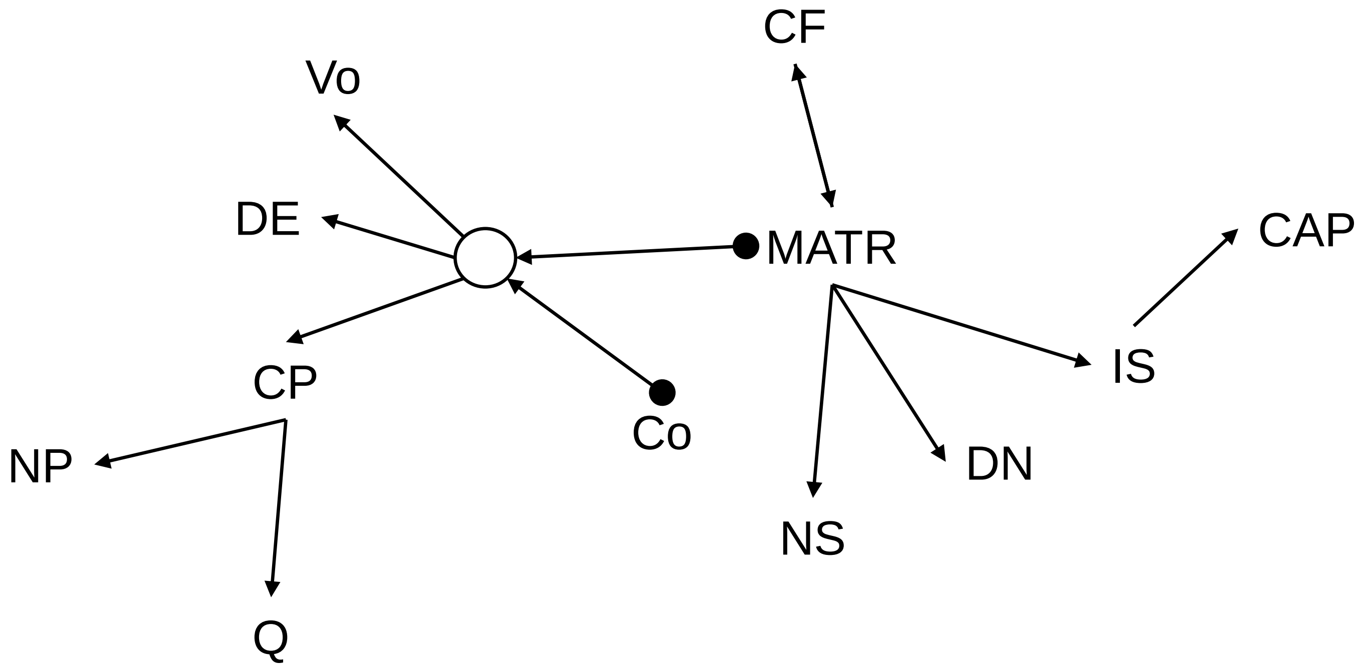
MATR,Co → Vo,DE,CP



Esempio su ESAMI

Costruisco il grafo delle dipendenze

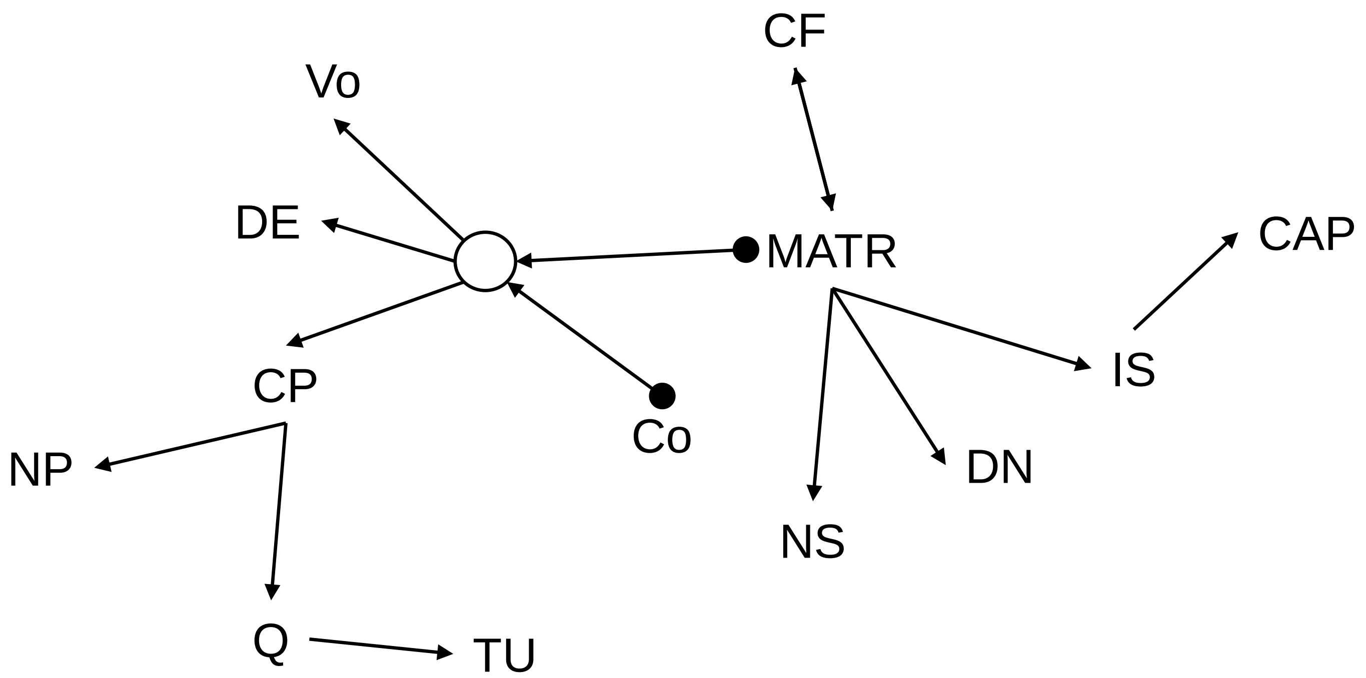
CP → NP,Q



Esempio su ESAMI

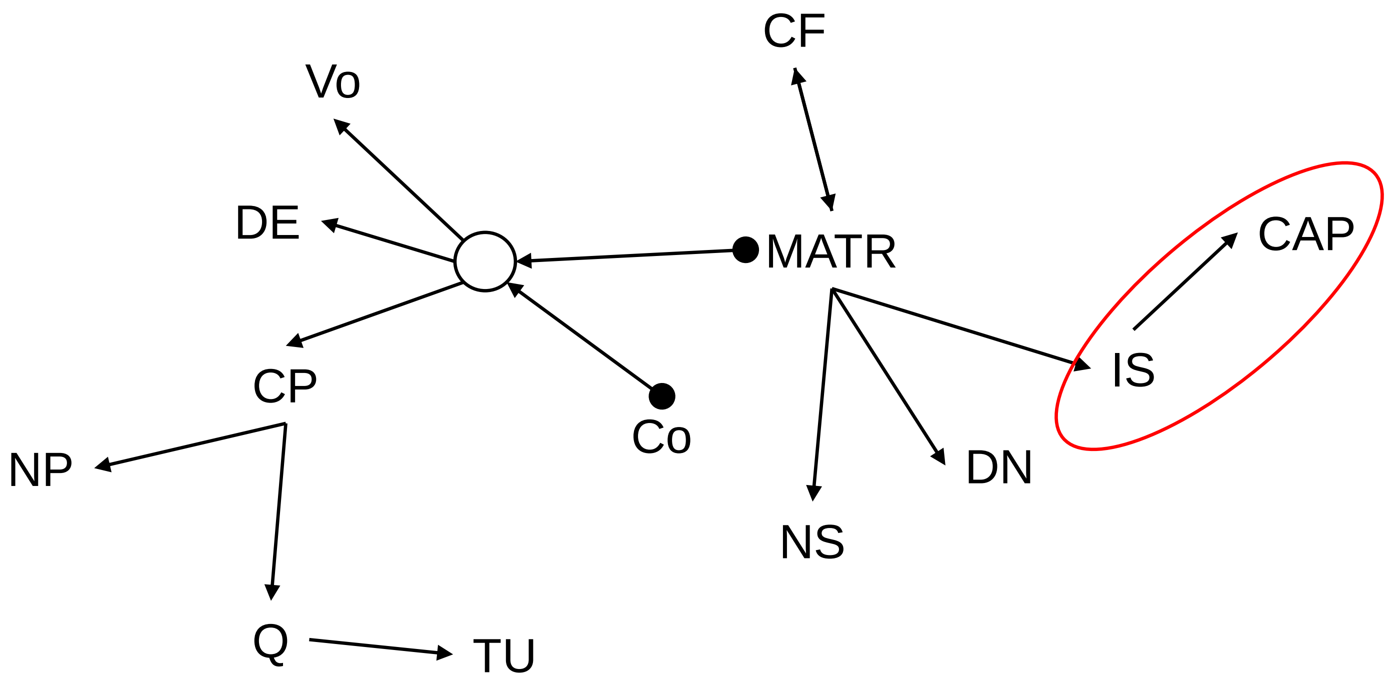
Costruisco il grafo delle dipendenze

Q → TU



Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 4

F
MATR → NS, IS, CF, DN
CF → MATR
*IS → CAP
MATR, Co → Vo, DE, CP
CP → NP, Q
Q → TU

Uso IS → CAP

Cancello CAP dalla relazione iniziale

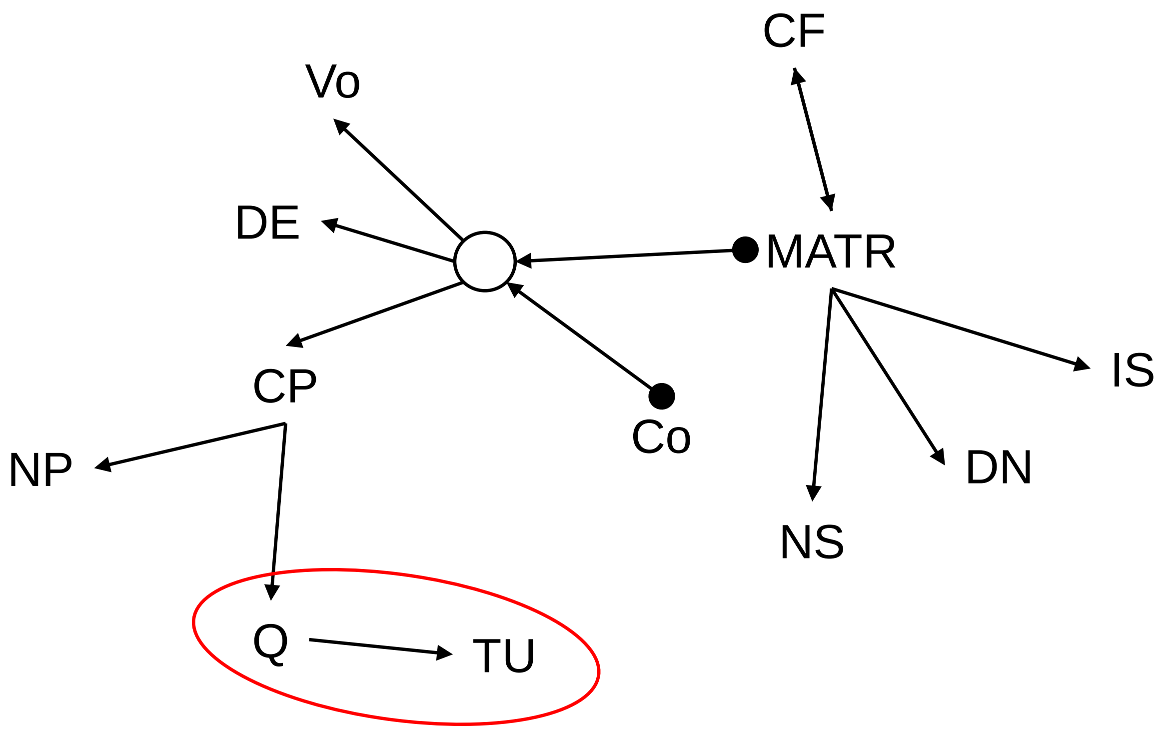
- ESAMI(<u>MATR</u>, NS, IS, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q, TU)
- F_1 = F – {IS → CAP}

Costruisco INDIRIZZI

- INDIRIZZI(<u>IS</u>,CAP)
- F_2 = {IS → CAP}

Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 4

F1
MATR → NS, IS, CF, DN
CF → MATR
~~IS → CAP~~
MATR, Co → Vo, DE, CP
CP → NP, Q
*Q → TU

Uso Q → TU

Cancello TU dalla relazione iniziale

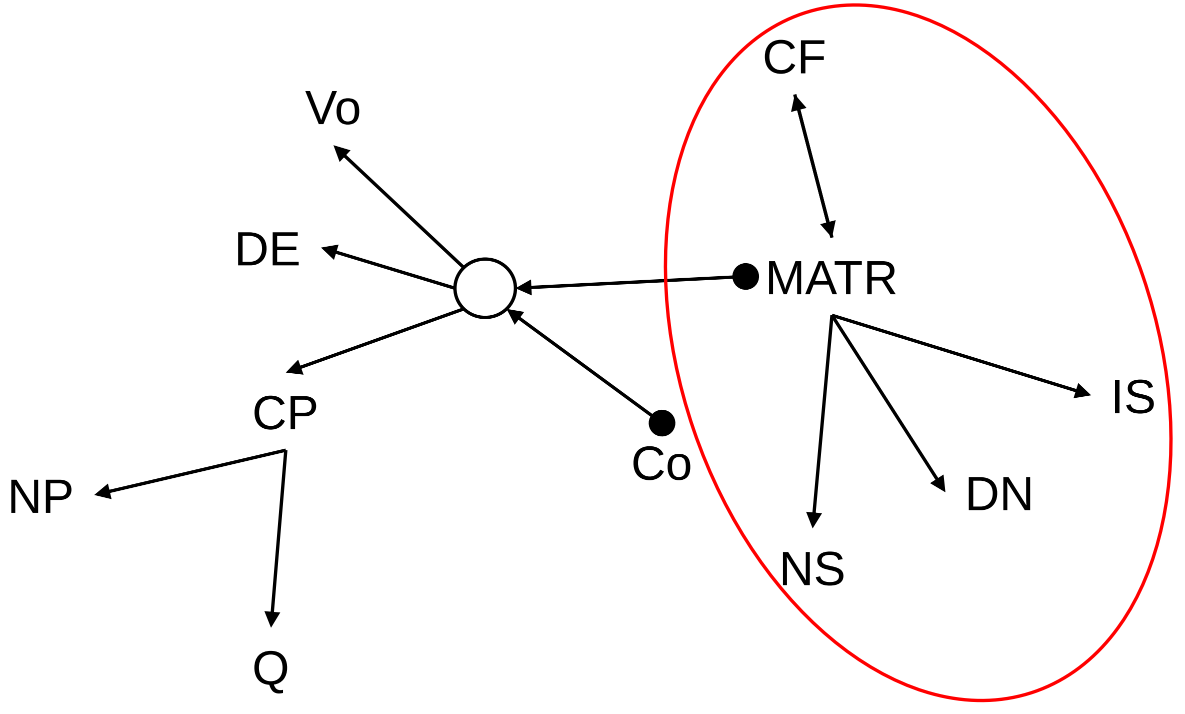
- ESAMI(<u>MATR</u>, NS, IS, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q)
- $F_1 = F_1 - \{Q \rightarrow TU\}$

Costruisco QUALIFICHE

- QUALIFICHE(<u>Q</u>,TU)
- $F_3 = \{Q \rightarrow TU\}$

Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 4

F1
*MATR → NS, IS, CF, DN
CF → MATR
~~IS → CAP~~
MATR, Co → Vo, DE, CP
CP → NP, Q
~~Q → TU~~

Uso MATR → NS,IS,CF,DN

Cancello NS,IS,CF,DN da ESAMI

- ESAMI(<u>MATR</u>, <u>Co</u>, Vo, DE, CP, NP, Q)
- $F_1 = F_1 - \{MATR \rightarrow NS,IS,CF,DN\}$

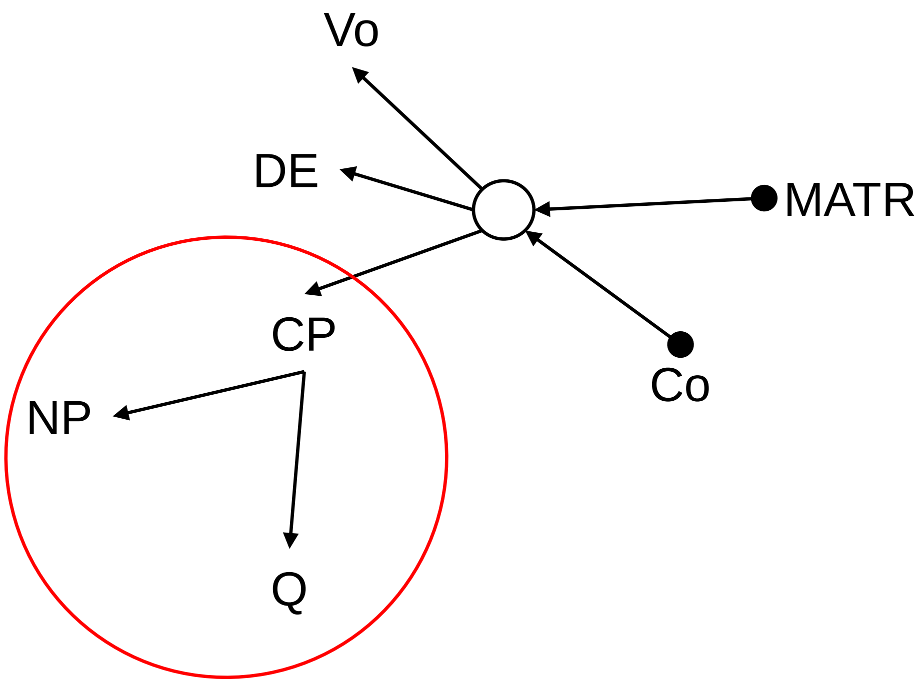
Costruisco STUDENTI

- STUDENTI(<u>MATR</u>,NS,IS,CF,DN)
- $F_4 = \{MATR \rightarrow NS,IS,CF,DN;$ **CF → MATR**$\}$

Nella restrizione ritrovo CF → MATR!

Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 4

F1
~~MATR → NS, IS, CF, DN~~
~~CF → MATR~~
~~IS → CAP~~
MATR, Co → Vo, DE, CP
*CP → NP, Q
~~Q → TU~~

Uso CP → NP,Q

Cancello NP,Q da ESAMI

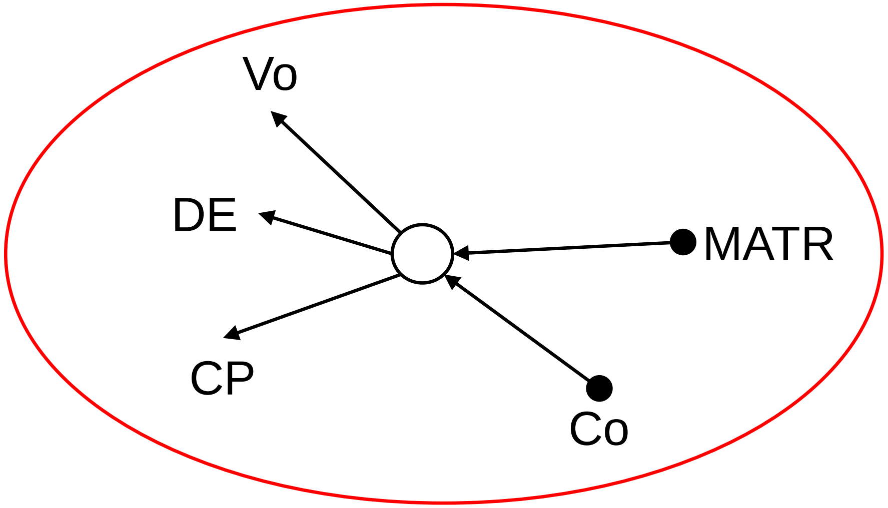
- ESAMI(<u>MATR</u>, <u>Co</u>, Vo, DE, CP)
- $F_1 = F_1 - \{CP \rightarrow NP,Q\}$

Costruisco DOCENTI

- DOCENTI(<u>CP</u>,NP,Q)
- $F_5 = \{CP \rightarrow NP,Q\}$

Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 3

F1
~~MATR → NS, IS, CF, DN~~
~~CF → MATR~~
~~IS → CAP~~
MATR, Co → Vo, DE, CP
~~CP → NP, Q~~
~~Q → TU~~

Rimane solo

ESAMI(<u>MATR</u>, <u>Co</u>, Vo, DE, CP)

$F_1 = \{MATR,Co \rightarrow Vo,DE,CP\}$

{MATR,Co} è chiave quindi l'algoritmo **termina**

Risultato

ESAMI(<u>MATR</u>, <u>Co</u>, Vo, DE, CP) con F_1 = {MATR,Co → Vo,DE,CP}

INDIRIZZI(<u>IS</u>,CAP) con F_2 = {IS → CAP}

QUALIFICHE(<u>Q</u>,TU) con F_3 = {Q → TU}

STUDENTI(<u>MATR</u>,NS,IS,CF,DN) con
F_4 = {MATR → NS,IS,CF,DN; CF → MATR}

DOCENTI(<u>CP</u>,NP,Q) con F_5 = {CP → NP,Q}

tutte in BCNF

L'unione delle F_i dà l'insieme F di partenza, quindi la località delle dipendenze funzionali è mantenuta

Considerazione importante

- Dagli appunti fatti, sembra che la $R_2(XY)$ sia in BCNF fin dalla sua costruzione
- Consideriamo ad esempio una d.f. CP→NP,Q,TU e un'altra dipendenza funzionale Q → TU
- Quando arrivo a costruire una R_2(<u>CP</u>,NP,Q,TU), nella restrizione F_2 ho ancora Q → TU (Q non è chiave)

In generale, a rientrare nel ciclo non è solo R_1, ma anche R_2 che va quindi riesaminata

Osservazione

Immaginiamo di avere le seguenti dipendenze funzionali:

- MATR → NS,IS
- MATR → DN,CF

Il processo di normalizzazione avrebbe portato a due relazioni

- STUDENTE1(MATR,NS,IS)
- STUDENTE2(MATR,DN,CF)

Quando il progettista arriva allo stato finale del processo di normalizzazione e trova tavole **con esattamente la stessa chiave**, può accorparle

STUDENTE(MATR,NS,IS,DN,CF)

Località delle dipendenze

Esiste un intreccio di dipendenze funzionali che non permette il mantenimento della località delle dipendenze alla fine del processo di normalizzazione

Esempio

Usiamo una sottorelazione di ESAMI

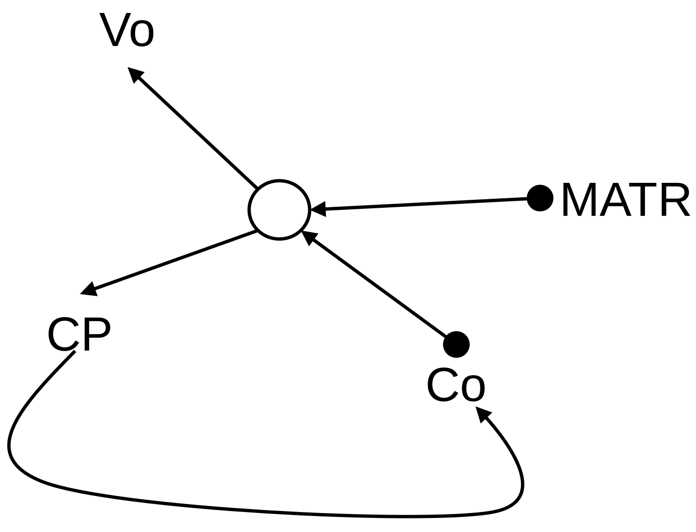
ESAMI(<u>MATR</u>, <u>Co</u>, Vo, CP)

F = {MATR, Co → Vo, CP; CP → Co} (il docente che firma l'esame è un titolare del corso)

con la chiave {MATR,Co}, ma anche {MATR,CP} diventa chiave: infatti la chiusura di {MATR,CP} è {MATR,Co,Vo,CP}

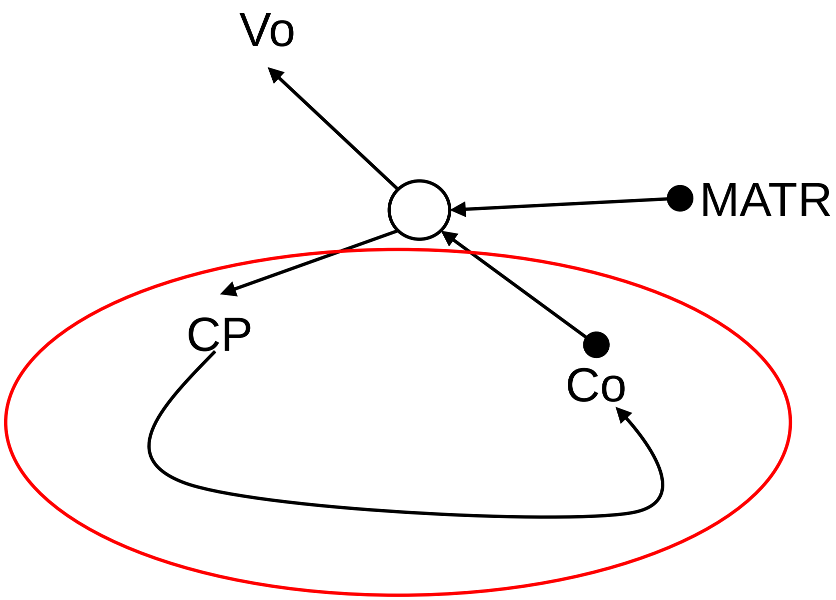
Passo 2

Costruisco il grafo delle dipendenze



Passo 2

Trovo X → Y con X non di tipo 1 o 2



Passo 4

Posso solo applicare la decomposizione utilizzando la dipendenza funzionale CP $\rightarrow$ Co (CP da solo non è chiave)

Elimino Co da ESAMI

R_1(MATR,Vo,CP)

Calcolo la restrizione:

Saremmo portati a dire che $F_1 = \{\emptyset\}$

Ma da F è deducibile MATR,CP $\rightarrow$ Vo ed esprimibile nello schema di R1, quindi F_1 = {MATR,CP $\rightarrow$ Vo}

Passo 4

Posso solo applicare la decomposizione utilizzando la dipendenza funzionale CP $\rightarrow$ Co (CP da solo non è chiave)

Elimino Co da ESAMI

R_1(MATR,Vo,CP) con F_1 = {MATR,CP $\rightarrow$ Vo}

Costruisco R_2 ed F_2:

R_2(CP,Co) con F_2 = {CP $\rightarrow$ Co}

Sono entrambe in BCNF e l'algoritmo termina

Verifica finale

F_1 = {MATR,CP → Vo} ed F_2 = {CP → Co} sono due restrizioni da cui non è possibile ricostruire tutte le dipendenze di partenza in F

Infatti MATR,Co→ Vo,CP non è più deducibile da $F_1 \cup F_2$

- Calcolo la chiusura di {MATR,Co} rispetto a $F_1 \cup F_2$:

  Parto da {MATR,Co}, ma non trovo nessuna X→Y con $X \subseteq$ {MATR,Co}

Vincolo globale?

Cosa succede se non tengo conto del vincolo globale MATR,Co $\rightarrow$ Vo,CP? Vediamo un esempio

MATR	CP	Vo
33	Pensa	25
33	Demo	30
…	…	…

CP	Co
Pensa	Basi di Dati
Demo	Basi di Dati
…	…

Le due relazioni rispettano i vincoli in F_1 ed F_2

Ma da questa istanza del BD risulta che lo stesso studente ha dato due volte Basi di dati!

Vincolo globale?

Cosa succede se non tengo conto del vincolo globale MATR,Co → Vo,CP?

MATR	CP	Vo
33	Pensa	25
33	Demo	30
…	…	…

CP	Co
Pensa	Basi di Dati
Demo	Basi di Dati
…	…

Il problema appare chiaro mettendo in join le due relazioni

MATR	Co	CP	Vo
33	**Basi di Dati**	Pensa	25
33	**Basi di Dati**	Demo	30
…		…	…

Vincolo globale?

Cosa succede se non tengo conto del vincolo globale MATR,Co → Vo,CP?

MATR	CP	Vo
33	Pensa	25
33	Demo	30
…	…	…

CP	Co
Pensa	Basi di Dati
Demo	Basi di Dati
…	…

Se avessimo considerato il vincolo MATR,Co → Vo,CP questa istanza non sarebbe stata corretta

Conclusione

La dipendenza persa dev'essere implementata come vincolo globale, altrimenti lo stato della base di dati può prendere una strada diversa da quella indicata dall'analisi della realtà

Osservazione

Anche se la normalizzazione BCNF risolve i problemi di anomalie, nella prassi ci possono essere dei passi di de-normalizzazione consapevole

Consideriamo le due relazioni

- STUDENTI(MATR,NS,IS,CF,DN)
- INDIRIZZI(IS,CAP)

Per risalire al CAP devo sempre effettuare un join

Invece, di fronte a queste situazioni, il progettista spesso procede ad una de-normalizzazione consapevole:

STUDENTI(MATR,NS,IS,CAP,CF,DN)

Reintroduco il rischio di anomalie, ma le reputo accettabili!

Osservazione

Altro esempio:

DOCENTI(CP,NP,Q)

QUALIFICHE(Q,TU)

Di solito il progettista de-normalizza le relazioni:

DOCENTI(CP,NP,Q,TU)

Il tutto però solo dopo aver analizzato il problema, aver eseguito una normalizzazione e ragionato sull'applicazione

Esempio

Consideriamo una dipendenza funzionale:

NConto → NAgenzia,CittaAgenzia,Saldo,DataMovim

Quindi, con la normalizzazione troviamo

CC(NConto,NAgenzia,CittaAgenzia,Saldo,DataMovim)

Le buone prassi di progettazione ci dicono che bisogna minimizzare il numero di relazioni per evitare i join, ma in alcuni casi può essere consigliata la frammentazione

Esempio

Consideriamo una dipendenza funzionale:

NConto → NAgenzia,CittaAgenzia,Saldo,DataMovim

Le buone prassi di progettazione ci dicono che bisogna minimizzare il numero di relazioni per evitare i join, ma in alcuni casi può essere consigliata la frammentazione

- ANAGRAFECC(NConto,NAgenzia,CittaAgenzia)
- STATOCC(NConto,Saldo,DataMovim)

Qui manteniamo la BCNF, ma splittiamo la relazione CC per velocizzare al massimo alcune applicazioni

Esempio

- ANAGRAFECC(NConto,NAgenzia,CittaAgenzia)
- STATOCC(NConto,Saldo,DataMovim)

Le operazioni di modifica dell'anagrafica sono piuttosto rare, mentre quelle di modifica del saldo sono molto frequenti

Splittando si velocizzano le operazioni di modifica del saldo, che ora agiscono su tuple più piccole