Lezione 15 - Decomposizioni che hanno un join senza perdita

Prof.ssa Maria De Marsico
demarsico@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Parte del materiale è tratto dal materiale preparato dalla prof.ssa Moscarini per il canale A-L

Cosa significa join senza perdita

Sapienza Università di Roma
Dipartimento di Informatica

Quando uno schema di relazione viene di solito decomposto:

- **quando non è in 3NF**
- per motivi di <u>efficienza degli accessi</u>
  - più piccola è la taglia delle tuple maggiore è il numero che riusciamo a caricare in memoria nella stessa operazione di lettura
  - se le informazioni della tupla non vengono utilizzate dallo stesso tipo di operazioni nella base di dati meglio decomporre lo schema
  - <u>Esempio</u>: lo schema Studente potrebbe essere decomposto separando le informazioni anagrafiche (CF, Nome, Cognome, DataNascita, LuogoNascita, ecc.) da quelle accademiche (Matricola, CorsoLaurea, AnnoCorso, ecc.)

Cosa significa join senza perdita

- Abbiamo visto che quando uno schema viene decomposto, **non basta** che i sottoschemi siano in 3NF

- Rivediamo i due esempi (uno più «astratto», l'altro più «concreto»)

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo lo schema $R=ABC$ con l'insieme di dipendenze funzionali $F=\{A\rightarrow B,\ C\rightarrow B\}$ (lo schema non è in 3NF per la presenza in F^+ delle dipendenze parziali $A\rightarrow B$ e $C\rightarrow B$, dato che la chiave è AC. Tale schema può essere decomposto in:

- $R1=AB$ con $\{A\rightarrow B\}$ e
- $R2=BC$ con $\{C\rightarrow B\}$.

- Tale decomposizione **pur preservando tutte le dipendenze in F^+** non è soddisfacente.

Cosa vogliamo ottenere – La 3NF non basta

- Consideriamo l'istanza **legale** di *R*

R

A	B	C
a1	b1	c1
a2	b1	c2

sono veri i due fatti *(a1,b1,c1)* e *(a2,b1,c2)* e **non altri**

- In base alla decomposizione data, questa istanza si decompone in

R1

A	B
a1	b1
a2	b1

R2

B	C
b1	c1
b1	c2

- E dovrebbe essere possibile ricostruirla **esattamente** tramite join … invece …

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
Università di Roma
Dipartimento di Informatica

- … e invece se si effettua il join delle due istanze legali risultanti dalla decomposizione si ottiene

R

A	B	C
a1	b1	c1
a2	b1	c2
a1	b1	c2
a2	b1	c1

tuple estranee alla realtà di interesse
quindi
perdita di informazione

- Occorre garantire che il join delle istanze risultanti dalla decomposizione non riveli <u>perdita di informazione</u>

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
Università di Roma
Dipartimento di Informatica

Consideriamo ora lo schema

R=(Matricola, Provincia, Comune) con l'insieme di dipendenze funzionali

F={Matricola→Provincia, Comune→Provincia} (lo schema non è in 3NF per la presenza in F^+ delle dipendenze parziali *Matricola→Provincia* e *Comune→Provincia*, dato che la chiave è (Matricola, Comune) (Comune non è determinato da nessun altro attributo!)

Tale schema può essere decomposto in:

R1=(Matricola, Provincia) con { *Matricola→Provincia*} e

R2=(Provincia, Comune) con { *Comune→Provincia*}.

Tale schema **pur preservando tutte le dipendenze in F^+** non è soddisfacente.

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
Università di Roma
Dipartimento di Informatica

Consideriamo l'i stanza **legale** di *R*

R

Matricola	Provincia	Comune
501	Roma	Tivoli
502	Roma	Mandela

sono veri i due fatti *(501,Roma,Tivoli)* e *(501, Roma,Mandela)* e **non altri**

In base alla decomposizione data, questa istanza si decompone in

R1

Matricola	Provincia
501	Roma
502	Roma

R2

Provincia	Comune
Roma	Tivoli
Roma	Mandela

E dovrebbe essere possibile ricostruirla **esattamente** tramite join … invece …

Cosa vogliamo ottenere – La 3NF non basta

- … e invece se si effettua il join delle due istanze legali risultanti dalla decomposizione si ottiene

R

Matricola	Provincia	Comune
501	Roma	Tivoli
502	Roma	Mandela
501	Roma	Mandela
502	Roma	Tivoli

tuple estranee alla realtà di interesse
quindi
perdita di informazione

Cosa vogliamo ottenere – La 3NF non basta

In conclusione, quando si decompone uno schema occorre tenere presente il seguente requisito dello schema decomposto:

- deve permettere di **ricostruire mediante join naturale** ogni **istanza legale dello schema originario** (senza aggiunta di informazione estranea)

Definizione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Se si decompone uno schema di relazione *R* si vuole che la **decomposizione** $\{R_1, R_2, \ldots, R_k\}$ ottenuta sia tale che **ogni istanza legale *r* di *R*** sia **ricostruibile** mediante **join naturale ($\bowtie$) da un istanza legale $\{r_1, r_2, \ldots, r_k\}$ dello schema decomposto $\{R_1, R_2, \ldots, R_k\}$.** Poiché per **ricostruire una tupla *t* di r è necessario che $t[R_i] \in r_i$, $i=1,\ldots,k$, si deve avere $r_i = \pi_{R_i}(r)$, $i=1,\ldots,k$.**

Definizione Sia *R* uno schema di relazione. Una decomposizione $\rho = \{R_1, R_2, \ldots, R_k\}$ di *R ha un join senza perdita* se **per ogni istanza legale *r*** di *R* si ha $r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ldots \bowtie \pi_{Rk}(r)$.

Problema

- Anche in questo caso partiamo da una decomposizione data, e cerchiamo un modo per verificare che goda della proprietà desiderata.

- Anche in questo caso facciamo alcune premesse.

Teorema

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Teorema Sia R uno schema di relazione e $\rho=\{R_1, R_2,\ldots, R_k\}$ una decomposizione di R. Per ogni istanza legale r di R, indicato con $m_\rho(r) =\pi_{R1}(r)\bowtie \pi_{R2}(r) \bowtie\ldots\bowtie\pi_{Rk}(r)$, si ha:

a) $r\subseteq m_\rho(r)$

b) $\pi_{Ri}(m_\rho(r))=\pi_{Ri}(r)$

c) $m_\rho(m_\rho(r))= m_\rho(r)$.

Dim.

Prova di a). Sia ***t* una tupla di** r. Per ogni i, $i=1,\ldots,k$, $t[R_i]\in\pi_{Ri}(r)$ e quindi $t\in m_\rho(r)$.

- una **qualsiasi**, quindi vale per **ogni** tupla di r
- Il **sottoinsieme** dei suoi valori su attributi di R_i sarà sicuramente in una tupla della proiezione $\pi_{Ri}(r)$
- quindi **OGNI** sottoinsieme dei suoi valori **corrispondente** ad un elemento della decomposizione dello schema **comparirà in una tupla di un elemento** della decomposizione dell'istanza
- poiché anche la proiezione è un **insieme** di tuple al più i duplicati corrisponderanno ad una **unica tupla nella proiezione**
- **Quindi ogni tupla potrà essere ricostruita tramite il join naturale …**
- **… anzi il join naturale <u>potrebbe</u> ottenere tuple che <u>non erano</u> in r ricomponendo <u>pezzi di tuple diverse</u>!**

Digressione

SAPIENZA
Università di Roma
Dipartimento di Informatica

Consideriamo la solita istanza **legale** di $R=ABC$ con l'insieme di dipendenze funzionali $F=\{A\rightarrow B,\ C\rightarrow B\}$ (no 3NF – dipendenze parziali)

R / r

A	B	C
a1	b1	c1
a2	b1	c1
a3	b1	c2

sono veri i fatti *(a1,b1,c1)* e *(a2,b1,c1)* e (a3, b1, c2) e **non altri**

Esempio: A= Matricola
B= Provincia
C= Comune

In base ad una delle possibili decomposizioni dello schema , questa istanza si decompone in

R1 / $\pi_{R1}(r)$

A	B
a1	b1
a2	b1
a3	b1

R2 / $\pi_{R2}(r)$

B	C
b1	c1
b1	c2

La prima è ottenuta proiettando l'istanza originale su AB
La seconda è ottenuta proiettando l'istanza originale su BC (notare l'eliminazione del duplicato)
Forse questa eliminazione ci farà perdere tuple originali? **NO**

Dovrebbe essere possibile ricostruire l'istanza di partenza **esattamente** tramite join … invece …

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- … e invece se si effettua il join delle due istanze legali risultanti dalla decomposizione si ottiene

R

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

$$m_\rho(r) = \pi_{R1}(r) \bowtie \pi_{R2}(r)$$

tuple estranee alla realtà di interesse
quindi
perdita di informazione (ma non di tuple!)

$$r \subset m_\rho(r)$$

- Occorre garantire che il join delle istanze risultanti dalla decomposizione non riveli <u>perdita di informazione</u>

Teorema

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Insiemi di tuple
Procediamo provando la **doppia inclusione**

Prova di b). $\pi_{Ri}(m_\rho(r))=\pi_{Ri}(r)$

Per **a) si ha $r \subseteq m_\rho(r)$ e, quindi, $\pi_{Ri}(r) \subseteq \pi_{Ri}(m_\rho(r))$.**

E' sufficiente, pertanto, mostrare che $\pi_{Ri}(r) \supseteq \pi_{Ri}(m_\rho(r))$.

Banalmente, per ogni tupla $t \in m_\rho(r)$ e per ogni i, $i=1,\ldots,k$, deve esistere una tupla $t' \in r$ tale $t[R_i]=t'[R_i]$. **Se tale tupla non esistesse, non troveremmo i valori di t su R_i ($t[R_i]$) nella proiezione $\pi_{Ri}(r)$ e di conseguenza non li avremmo nel join naturale**

Digressione

SAPIENZA
Università di Roma
Dipartimento di Informatica

R

r

A	B	C
a1	b1	c1
a2	b1	c1
a3	b1	c2

R1

$\pi_{R1}(r)$

A	B
a1	b1
a2	b1
a3	b1

R2

$\pi_{R2}(r)$

B	C
b1	c1
b1	c2

$$m_\rho(r) = \pi_{R1}(r) \bowtie \pi_{R2}(r)$$

R

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

$\pi_{R1}(m_\rho(r))$

R

A	B
a1	b1
a1	**b1**
a2	b1
a2	**b1**
a3	**b1**
a3	b1

$\pi_{R2}(m_\rho(r))$

R

B	C
b1	c1
b1	**c2**
b1	c1
b1	**c2**
b1	**c1**
b1	c2

duplicati = vengono eliminati dalla proiezione

Teorema

Prova di c). $m_\rho(m_\rho(r))=m_\rho(r)$.

Per b) si ha $\pi_{Ri}(m_\rho(r))=\pi_{Ri}(r)$.

Pertanto , applicando la definizione dell'operatore m_ρ (join delle proiezioni) avremo

$$m_\rho(m_\rho(r))=\pi_{R1}(m_\rho(r)) \rhd\lhd \pi_{R2}(m_\rho(r)) \rhd\lhd \dots \rhd\lhd \pi_{Rk}(m_\rho(r)) =$$

$$\pi_{R1}(r) \rhd\lhd \pi_{R2}(r) \rhd\lhd \dots \rhd\lhd \pi_{Rk}(r)=m_\rho(r).$$

Digressione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

$m_\rho(r) = \pi_{R1}(r) \bowtie \pi_{R2}(r)$

A	B	C
a1	b1	c1
a1	**b1**	**c2**
a2	b1	c1
a2	**b1**	**c2**
a3	**b1**	**c1**
a3	b1	c2

$\pi_{R1}(m_\rho(r)$

R

A	B
a1	b1
a2	b1
a3	b1

$\pi_{R2}(m_\rho(r)$

R

B	C
b1	c1
b1	**c2**

$m_\rho(m_\rho(r)) = \pi_{R1}(m_\rho(r)) \bowtie \pi_{R2}(m_\rho(r))$

A	B	C
a1	b1	c1
a1	**b1**	**c2**
a2	b1	c1
a2	**b1**	**c2**
a3	**b1**	**c1**
a3	b1	c2

Che si fa?

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Abbiamo uno schema di relazione ***R***, un insieme di dipendenze funzionali *F* e una decomposizione ρ.

- Come facciamo a **verificare** che la decomposizione data ha un join senza perdita?

- L'algoritmo che mostriamo permette di effettuare la verifica in tempo polinomiale.

- **F ci serve perché** … ci serviremo di nuovo di una particolare **istanza legale** di R

Algoritmo di verifica

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Algoritmo – verifica che una decomposizione abbia del join senza perdita

Input uno schema di relazione ***R***, un insieme ***F*** di dipendenze funzionali su R, una decomposizione $\rho=\{R_1, R_2,\ldots, R_k\}$ di R;

Output decide se ρ ha un join senza perdita;

begin

Costruisci una tabella ***r*** nel modo seguente:

una colonna per ogni attributo di R e una riga per ogni elemento della decomposizione (sottoschema)

r ha $|R|$ colonne e $|\rho|$ righe

all'incrocio dell'*i*-esima riga e della *j*-esima colonna metti

il simbolo a_j **se l'attributo** $A_j \in R_i$

l'attributo A_j **fa parte del sottoschema** R_i

il simbolo b_{ij} altrimenti

Indice i = elemento della decomposizione = riga
Indice j = attributo = colonna

repeat

for every $X \rightarrow Y \in F$

do if ci sono due tuple t_1 e t_2 in r tali che $t_1[X]=t_2[X]$ e $t_1[Y]\neq t_2[Y]$

then for every attribute A_j in Y **do if** $t_1[A_j]='a_j'$ **then** $t_2[A_j]:=t_1[A_j]$

else $t_1[A_j]:=t_2[A_j]$

gestiamo correttamente anche il caso in cui $t_2[A_j]='a_j'$

until *r* ha una riga con tutte *'a'* **or** *r* non è cambiato;

if *r* ha una riga con tutte *'a'* **then** ρ ha un join senza perdita

else ρ non ha un join senza perdita

end

anche in questo caso l'algoritmo termina <u>sempre</u>! occorre poi verificare se in r c'è la tupla che cerchiamo

Osservazioni

- Possiamo considerare gli $\mathbf{a}_j$ come valori particolari appartenenti al dominio dell'attributo $\mathbf{A}_j$
- Possiamo considerare i $\mathbf{b}_{ij}$ come valori particolari appartenenti al dominio dell'attributo $\mathbf{A}_j$
- Possiamo considerare **tutti i valori** $\mathbf{a}_j$ uguali tra di loro
- Il valore $\mathbf{b}_{ij}$ è diverso da $\mathbf{a}_j$ e da un altro valore $\mathbf{b}_{kj}$ anche se appartengono tutti allo stesso dominio (quello dell'attributo $\mathbf{A}_j$)
- Come **conseguenza**, la nostra r **iniziale** è una particolare **istanza** dello schema R

Osservazioni

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Per i nostri scopi, **trasformiamo** questa istanza iniziale in una **istanza legale**
- Per fare questo, dobbiamo fare in modo che **tutte** le dipendenze in F siano **soddisfatte**
- **Allora, ogni volta che troviamo due tuple uguali sugli attributi a sinistra, facciamo in modo che siano uguali anche su quelli a destra (attributo per attributo!)**
- **Nel fare questo, diamo la precedenza alle a (non diventano MAI b)**

Osservazioni

Sapienza Università di Roma
Dipartimento di Informatica

- Se due tuple sono **uguali sulla parte sinistra** ma sono **diverse in un attributo della parte a destra** della dipendenza, e **una delle due** ha una ***a*** come valore di quell'attributo, facciamo **diventare *a* anche** il valore dell'attributo nell'altra tupla
- Se due tuple sono **uguali sulla parte sinistra** ma sono **diverse in un attributo della parte a destra** della dipendenza, e **nessuna delle due** ha una ***a*** come valore di quell'attributo, facciamo diventare uguali le due b (ad esempio, se abbiamo $\mathbf{b_{ij}}$ e $\mathbf{b_{kj}}$ **facciamo diventare entrambi i valori $\mathbf{b_{ij}}$ oppure $\mathbf{b_{kj}}$**)
- **In pratica due valori saranno uguali se sono entrambi *a* <u>oppure</u> se hanno una *b* <u>con lo stesso pedice</u>**
- L'algoritmo di ferma quando **TUTTE** le coppie di tuple soddisferanno la condizione che **<u>se</u>** gli attributi delle parti sinistre delle dipendenze sono uguali, **<u>lo sono anche</u>** gli attributi delle parti destre
- Ma questo significa che **alla fine <u>r è stata trasformata in un'istanza legale di R</u>**

Teorema

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **Teorema** Sia R uno schema di relazione, F un insieme di dipendenze funzionali su R e $\rho=\{R_1, R_2, \ldots, R_k\}$ una decomposizione di R. L'Algoritmo di verifica decide correttamente se ρ ha un join senza perdita.
- **Dim.** Occorre dimostrare che:
- ρ ha un join senza perdita (**$m_\rho(r)= r$ <u>per ogni</u> r legale)**
  - **se e solo se**
- **quando l'algoritmo termina la tabella *r* ha una tupla con tutte *'a'*.**

Teorema: dimostrazione

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **Parte solo se.**

  •Supponiamo **per assurdo che ρ abbia un join senza** perdita ($\boldsymbol{m_\rho(r)= r}$) e che quando l'algoritmo termina la tabella *r* **non abbia** una tupla con tutte *'a'*. La tabella *r* può essere interpretata come **un'istanza legale di *R*** (basta sostituire ai simboli *'a'* e *'b'* valori presi dai domini dei corrispondenti attributi in modo tale che ad uno stesso simbolo venga sostituito lo stesso valore) in quanto l'algoritmo termina quando **non ci sono più violazioni delle dipendenze in *F*.** Poiché **nessun simbolo *'a'*** che compare nella **tabella costruita inizialmente viene <u>mai</u> modificato** dall'algoritmo, per ogni *i*, $i=1,\ldots,k$, $\boldsymbol{\pi_{R_i}(r)}$ contiene (<u>fin dall'inizio</u>!) una tupla con tutte *'a'* (quella ottenuta **proiettando l'istanza *r* sugli attributi di R_i** e precisamente nella riga corrispondente **al sottoschema R_i**) pertanto $\boldsymbol{m_\rho(r)}$ **contiene sicuramente una tupla con tutte *'a'*** e, quindi, $\boldsymbol{m_\rho(r)\neq r}$ **(contraddizione).**

Teorema: dimostrazione

- **Parte se**

- Per uno sketch della prova della parte “se” consultare il testo

J. D. Ullman, “Principles of database and knowledge-base systems”, vol. I, Computer Science Press, 1988.

- Una scansione delle pagine rilevanti verrà fornita insieme alle dispense del corso.

Corollario

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- **Corollario** Sia *R* uno schema di relazione, *F* un insieme di dipendenze funzionali su *R* e $\rho=\{R_1, R_2\}$ una decomposizione di *R*.

  •Se $R1 \cap R2 \to R1\text{-}R2$ o $R1 \cap R2 \to R2\text{-}R1$ allora ρ ha un join senza perdita.

Osservazione banale: $R_1= (R_1 - R_2) \cup (R_1 \cap R_2)$ e $R_2= (R_2 - R_1) \cup (R_1 \cap R_2)$

	$R_1 - R_2$	$R_1 \cap R_2$	$R_2\text{-}R_1$
R_1	$\underline{\boldsymbol{a}}$	$\underline{\boldsymbol{a}}$	$\underline{\boldsymbol{b}}_{2-1}$
R_2	$\underline{\boldsymbol{b}}_{1-2}$	$\underline{\boldsymbol{a}}$	$\underline{\boldsymbol{a}}$

Possiamo riordinare gli attributi in modo da avere un quadro più immediato della distribuzione dei valori nell'istanza di R

$\underline{\boldsymbol{a}}$, $\underline{\boldsymbol{b}}_{2-1}$, e $\underline{\boldsymbol{b}}_{1-2}$ sono vettori di valori del tipo a_j o b_{ij}
In base a $R1 \cap R2 \to R1\text{-}R2$ o $R1 \cap R2 \to R2\text{-}R1$ applicando l'algoritmo una delle due tuple (rispettivamente la seconda o la prima) diventerà di tutte *a*

Non è necessario che $R1 \cap R2 \to R1\text{-}R2$ o $R1 \cap R2 \to R2\text{-}R1$ siano in *F*, è sufficiente che siano in F^+ …