Lezione XX - Decomposizioni

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

Premessa

- Abbiamo visto che uno schema di relazione in 3NF ha delle buone proprietà che lo rendono preferibile ad uno che non è in 3NF
- Più che preferibile, dovremmo dire che è essenziale ottenere schemi in 3NF, visti i problemi che altrimenti possono presentarsi
- Lo schema di una base di dati è dato **dall'insieme** degli schemi delle relazioni che lo compongono
- Un obbiettivo da tenere presente quando si progetta una base di dati è quello di produrre uno schema in cui **ogni schema di relazione che lo compone** sia in 3NF.
- La proprietà deve essere posseduta **individualmente** da ogni schema, e viene **verificata individualmente** per ogni schema
- Nella fase di **progettazione concettuale** si individuano i concetti che devono essere rappresentati nella base di dati. **1 concetto = 1 relazione**

Decomposizione: casi

- Se il lavoro di individuazione è fatto **accuratamente** lo schema relazionale che può essere derivato in modo automatico con **opportune regole**, è in 3NF.

- Se tuttavia dopo tale processo ci trovassimo a produrre uno schema che **non è in 3NF** dovremmo procedere ad una fase di **decomposizione** di tale schema in maniera analoga a quella esaminata nell'esempio sui dati di un'Università.

- Il mancato rispetto della 3NF è **l'unico motivo per decomporre**? **No**, la decomposizione può essere decisa **per motivi di opportunità**: più piccole sono le tuple, più ne vengono trasferite in memoria durante una operazione di lettura (blocco di memoria).

- Ovviamente se tutte le informazioni **sono utilizzate sempre insieme**, la decomposizione non ha senso **in questo caso** (dovremmo sempre utilizzare il join per riottenere i dati completi, appesantendo sia il traffico dalla memoria che la computazione)

- **Cambia qualcosa tra i due casi nel modo di procedere per la decomposizione e nelle proprietà che vogliamo ottenere? No, i procedimenti sono identici.**

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Uno schema che non è in 3NF può essere **decomposto** in **più modi** in un insieme di schemi in 3NF. Ad esempio lo schema $R=ABC$ con l'insieme di dipendenze funzionali $F=\{A\rightarrow B,\ B\rightarrow C\}$ non è in 3NF per la presenza in F^+ della dipendenza transitiva $B\rightarrow C$, dato che la chiave è evidentemente A.
- R può essere decomposto in:
  - $R1=AB$ con $\{A\rightarrow B\}$ e
  - $R2=BC$ con $\{B\rightarrow C\}$

  - **Oppure**

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

- **Entrambe** le decomposizioni contengono schemi che **sono** in 3NF, tuttavia la seconda soluzione **non è soddisfacente**.

Cosa succede alle dipendenze?

- Ricordiamo che mentre la proiezione di una istanza su un sottoschema è piuttosto intuitiva (si prendono solo le colonne degli attributi che compaiono nel sottoschema), la proiezione di un insieme di dipendenze ha una formulazione leggermente più complessa. Dato F e un sottoschema R_i la proiezione è definita come

$$\pi_{Ri}(F) = \{X \rightarrow Y / X \rightarrow Y \in F^+ \wedge XY \subseteq R_i\}$$

- In pratica prendiamo le dipendenze in F^+ per includere anche quelle che, una volta decomposta la parte destra, possono essere soddisfatte all'interno dello stesso schema, e quelle che vengono ottenute per transitività che hanno la stessa proprietà.
  - Esempio: schema ABCD, sottoschemi ABC e ABD, dipendenza AB→CD, C→D, D→A
  - AB→CD non entrerebbe in nessuna delle due proiezioni, ma AB→C fa parte della proiezione rispetto ad ABC e AB→D dell'altra
  - C→A fa parte della proiezione su ABC, anche se C→D non rientra in nessuna delle due

Cosa succede alle dipendenze?

- Obiettivo: dopo la decomposizione, TUTTE le dipendenze originarie devono poter essere soddisfatte senza effettuare controlli tra relazioni diverse, ma semplicemente soddisfacendo vincoli relativi ai singoli sottoschemi della decomposizione (cioè tramite istanze legali dei singoli sottoschemi)

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo due istanze **legali** degli schemi ottenuti

R1

A	B
a1	b1
a2	b1

R2

A	C
a1	c1
a2	c2

•L'istanza dello schema originario *R* che posso ricostruire da questa (l'unico modo è di ricostruirla facendo un join naturale!) è la seguente

R

A	B	C
a1	b1	c1
a2	b1	c2

•**MA** non è un'istanza legale di R, in quanto **non soddisfa** la dipendenza funzionale $B \rightarrow C$ (che sarà pure transitiva **ma va soddisfatta comunque!**)

•**Occorre preservare TUTTE le dipendenze in F^+, senza che sia necessario ricorrere ogni volta ad un join per assicurarsene**

Esempio

- Consideriamo lo schema *R=(Matricola, Comune, Provincia)* con l'insieme di dipendenze funzionali
- $F=\{Matricola \rightarrow Comune,\ Comune \rightarrow Provincia\}$
- Lo schema non è in 3NF per la presenza in F^+ della dipendenza transitiva $Comune \rightarrow Provincia$, dato che la chiave è evidentemente *Matricola (Provincia dipende transitivamente da Matricola)*.
- *R* può essere decomposto in:
- $R1=(Matricola, Comune)$ con $\{Matricola \rightarrow Comune\}$
- $R2=(Comune, Provincia)$ con $\{Comune \rightarrow Provincia\}$
  - **Oppure**
- $R1=(Matricola, Comune)$ con $\{Matricola \rightarrow Comune\}$
- $R2=(Matricola, Provincia)$ con $\{Matricola \rightarrow Provincia\}$

- Entrambi gli schemi **sono** in 3NF, tuttavia la seconda soluzione **non è soddisfacente**.

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo le istanze **legali** degli schemi ottenuti

R1

Matricola	Comune
501	Tivoli
502	Tivoli

R2

Matricola	Provincia
501	Roma
502	Rieti

- L'istanza dello schema originario *R* che posso ricostruire da questa (l'unico modo è di ricostruirla facendo un join naturale!) è la seguente

R

Matricola	Comune	Provincia
501	Tivoli	Roma
502	Tivoli	Rieti

- **MA** non è un'istanza legale di R, in quanto **non soddisfa** la dipendenza funzionale *Comune*→*Provincia* (che sarà pure transitiva **ma possiamo rivelare praticamente che va soddisfatta comunque!**)
- **E' evidente che c'è stato un errore di inserimento, ma non abbiamo potuto evitarlo usando i vincoli, e abbiamo potuto rilevarlo solo effettuando il join**

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo ora 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 $\{A\rightarrow B\}$ *nella proiezione di F su AB*
  - $R2=BC$ con $\{C\rightarrow B\}$ *nella proiezione di F su BC*

- Questa volta la decomposizione **preserva tutte le dipendenze in F^+ … ma** non è comunque soddisfacente.

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo l'i stanza **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 perdita di informazione

- ATTENZIONE: in questo caso la perdita di informazione non è rappresentata da tuple che mancano, ma da tuple che non dovrebbero esistere!!!

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo ora lo schema
  - $R=(Matricola, Progetto, Capo)$ con l'insieme di dipendenze funzionali
  - $F=\{Matricola \rightarrow Progetto, Capo \rightarrow Progetto\}$
- Il progetto ha più capi ma ogni capo ha un solo progetto, e un impiegato su un progetto dà conto ad un solo capo (ogni capo segue un gruppo)
- Lo schema non è in 3NF per la presenza in F^+ delle dipendenze parziali $Matricola \rightarrow Progetto$ e $Capo \rightarrow Progetto$, dato che la chiave è (Matricola, Capo)
- Tale schema può essere decomposto in:

- $R1=(Matricola, Progetto)$ che include nella proiezione la dipendenza $\{Matricola \rightarrow Progetto\}$
- $R2=(Progetto, Capo)$ con la dipendenza $\{Capo \rightarrow Progetto\}$.

- 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	Progetto	Capo
501	30	E1
502	30	E2

sono veri i due fatti *(501,30,E1)* e *(501,30,E2)* e non altri

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

R1

Matricola	Progetto
501	30
502	30

R2

Progetto	Capo
30	E1
30	E2

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

Matricola	Voto	CodiceEsame
501	30	E1
502	30	E2
501	30	E2
502	30	E1

tuple estranee alla realtà di interesse
quindi
perdita di informazione

- Perché «**perdita di informazione**» ?
- **Perché anche effettuando il join tra le istanze per ricostruire quella non decomposta, non sappiamo più quali sono le informazioni corrette senza effettuare un controllo incrociato!**

Cosa succede ai dati?

- Obiettivo: dopo la decomposizione, dobbiamo poter essere sicuri che verranno inserite solo informazioni corrette senza dover ricorrere a controlli incrociati

Domanda

- Soddisfare una delle due richieste soddisfa automaticamente anche l'altra?

Cosa vogliamo ottenere – La 3NF non basta

- Consideriamo ora lo schema $R=ABC$ con l'insieme di dipendenze funzionali $F=\{\varnothing\}$ ovviamente in 3NF.
- Per motivi pratici vogliamo decomporlo

- $R1=AB$ con $\{\varnothing\}$ e
- $R2=BC$ con $\{\varnothing\}$.

- Tale schema **pur preservando tutte le dipendenze in F^+ (che sono quelle banali)** non è soddisfacente.

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

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

•I

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

•n 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 perdita di informazione … ma chi potrebbe identificarla senza fare un confronto con le istanze della decomposizione?

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Consideriamo ora lo schema
- *R=(Cliente, Prodotto, Data)* con l'insieme di dipendenze funzionali
- $F=\{\varnothing\}$
- Semplicemente teniamo conto di quando un cliente ha ordinato un certo prodotto
- (lo schema è banalmente in 3NF ma vogliamo decomporlo in:

- *R1= (Cliente, Prodotto)* con $\{\varnothing\}$ e
- *R2= (Prodotto, Data)* con $\{\varnothing\}$.

- 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

Cliente	Prodotto	Data
501	30	Data1
502	30	Data2

sono veri i due fatti *(501,30,Data1)* e *(501,30,Data2)* e non altri

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

R1

Cliente	Prodotto
501	30
502	30

R2

Prodotto	Data
30	Data1
30	Data2

- 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

Cliente	Prodotto	Data
501	30	Data1
502	30	Data2
501	30	Data2
502	30	Data1

tuple estranee alla realtà di interesse
quindi
perdita di informazione

Cosa vogliamo ottenere – La 3NF non basta

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

•In conclusione, quando si decompone uno schema per ottenerne uno in 3NF occorre tenere presenti altri due requisiti dello schema decomposto:

- deve **preservare le dipendenze funzionali** che valgono su ogni istanza legale dello **schema originario**

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

Due casi

- Possiamo considerare due casi

- abbiamo una decomposizione **già data**, e dobbiamo **verificare che soddisfi** le due condizioni di preservare le dipendenze e i dati

- abbiamo uno schema e vogliamo **calcolare** una decomposizione **che soddisfi** le due condizioni di preservare le dipendenze e i dati (oltre ovviamente ad ottenere sottoschemi in 3NF)

Verifica conservazione dipendenze

- Per **verificare** che vengano **preservate le dipendenze dobbiamo verificare l'equivalenza dell'insieme originario e di quello ottenuto dalle sue proiezioni**

- **Sfruttiamo la relazione tra chiusura di un insieme di attributi rispetto ad un insieme di dipendenze e la chiusura dell'insieme di dipendenze stesso**
  - $X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq (X)^+_F$
- Applichiamo l'algoritmo che calcola, per ogni dipendenza in F originario, **la chiusura del determinante rispetto alla decomposizione di F, e verifica che contenga il dipendente.**
- **L'algoritmo calcola la chiusura rispetto alla decomposizione (unione delle proiezioni) utilizzando la chiusura rispetto all'insieme originario**
- Lo facciamo **solo per le dipendenze che non sono già preservate in base alla definizione delle proiezioni** (parte destra e sinistra contenute in uno dei sottoschemi)

- La verifica **ha successo** se **per ogni dipendenza dell'insieme originario** il dipendente è contenuto nella chiusura del determinante rispetto all'insieme ottenuto dalle proiezioni

- Le istanze ottenute con il join di quelle sui sottoschemi **sono legali**

Verifica conservazione dati

- Per **verificare** che vengano **preservati i dati** dobbiamo verificare la possibilità **di ricostruire esattamente ogni istanza legale dal join naturale delle sue proiezioni rispetto ai sottoschemi della decomposizione**

- Sfruttiamo una istanza dello schema originario **costruita appositamente, che rendiamo prima di tutto legale** facendo in modo che vengano soddisfatte tutte le dipendenze

- L'algoritmo che si utilizza ha come passi **la costruzione dell'istanza** e il progressivo **soddisfacimento di tutte le dipendenze** in modo che l'istanza diventi **una istanza legale**

- **Il risultato della verifica dipende dal fatto che l'istanza legale contenga una specifica tupla**

Verifica

- Come abbiamo avuto modo di constatare, **verificare che vengano preservate le dipendenze non implica necessariamente che vengano preservati i dati** (join senza perdita)

- **Cosa accade se verifico che la decomposizione preserva i dati?**
  - **Vengono preservati i dati → OGNI istanza legale può essere ricostruita**
  - **Le istanze legali soddisfano TUTTE le dipendenze**
  - **... Ma il join potrebbe produrre anche istanze non legali, che verrebbero rilevate solo analizzando l'istanza non decomposta**

Esempio

- Consideriamo di nuovo lo schema *R=(Matricola, Comune, Provincia)* con l'insieme di dipendenze funzionali
- $F=\{Matricola \rightarrow Comune,\ Comune \rightarrow Provincia\}$
- e la decomposizione
  - *R1=(Matricola, Comune)* con $\{Matricola \rightarrow Comune\}$
  - *R2=(Matricola, Provincia)* con $\{Matricola \rightarrow Provincia\}$
- La decomposizione **ha un join senza perdita**. Possiamo verificarlo velocemente con l'algoritmo che abbiamo a disposizione
- A= Matricola, B = Comune, C = Provincia F= {A→B, B→C} R1=AB, R2=AC

	A	B	C
AB	a	a	b13
AC	a	b22->a	a

- A→B porta alla trasformazione b22->a
- Possiamo fermarci qui perché abbiamo ottenuto una tupla con tutte 'a'.
- Tuttavia sappiamo già che la dipendenza B→C **non viene preservata**

Calcolo di una decomposizione

- In questo secondo abbiamo unicamente lo schema (in 3NF oppure no, che è il caso più frequente) e l'insieme di dipendenze originari

- Utilizziamo un algoritmo di decomposizione che
  - garantisce di produrre **sottoschemi in 3NF**
  - garantisce che vengano **preservate le dipendenze**

- Con un **passaggio in più** (aggiunta di un sottoschema con una delle chiavi, **se nessuna chiave è già contenuta** in uno dei sottoschemi della decomposizione) otteniamo anche **la conservazione di dati** … perché la presenza della chiave impedisce in un eventuale join di inserire dati estranei

- **Possono esistere più decomposizioni con le stesse proprietà, non necessariamente fornite dall'algoritmo**

Casi intercambiabili?

- I due casi (verifica e calcolo di una decomposizione) **non sono intercambiabili !!!**

•**Errore frequente:** applicare l'algoritmo di **decomposizione** e **confrontare** il risultato **con la decomposizione di cui bisogna verificare** la conservazione delle dipendenze o dei dati

- Come abbiamo detto, l'algoritmo di decomposizione e l'aggiunta forniscono **una (o più possibili)** decomposizioni che godono di certe proprietà (contemporaneamente!), **non tutte**!

- Potremmo avere una decomposizione che soddisfa **una** delle proprietà di conservazione, **pur non essendo fornita** dall'algoritmo di decomposizione