Teoria della Progettazione delle Basi di Dati Relazionali

Angelo Montanari

Dipartimento di Matematica e Informatica
Università di Udine

Introduzione

Proprietà richieste (forme normali)

Algoritmi (normalizzazione)

Concetto base: **dipendenza dei dati** (vincolo sull'insieme delle possibili istanze di un dato schema di relazione)

Dato uno schema di relazione R, se un attributo A (risp., un insieme di attributi X) determina un attributo B (risp., un insieme di attributi Y) diremo che vi è una **dipendenza funzionale** di B da A (risp., di Y da X) o che A (risp., X) **determina funzionalmente** B (risp., Y)

Si noti che la proiezione di R su A, B (risp., X, Y) definisce una funzione $f : A \to B$ (risp., $X \to Y$).

Ridondanze e anomalie

Un esempio.

INFO_FORN(NOME_FORN, IND_FORN, NOME_PROD, PREZZO)

vs.

FORNITORI(NOME_FORN, IND_FORN)

FORNISCE(NOME_FORN, NOME_PROD, PREZZO)

Problemi con la prima soluzione (la *relazione universale*):

- RIDONDANZA
- ANOMALIE DI AGGIORNAMENTO, INSERIMENTO e CANCELLAZIONE

Osservazione: legame tra dipendenze e ridondanza

Le dipendenze funzionali - 1

Definizione. Una **dipendenza funzionale** tra un insieme di attributi X e un insieme di attributi Y di uno schema di relazione $R(T)$, con $X, Y \subseteq T$, denotata $X \rightarrow Y$, è un vincolo di integrità sulle istanze della relazione

Definizione. Un'istanza $r \in R(T)$ **soddisfa** $X \rightarrow Y$ (equivalentemente, $X \rightarrow Y$ **vale** in r) se per ogni coppia di tuple $t_1, t_2 \in r$, se $t_1[X] = t_2[X]$, allora $t_1[Y] = t_2[Y]$

Osservazione: in ogni istanza *valida* r di $R(T)$, $\Pi_{XY}(r)$ è una funzione con dominio X e codominio Y

Definizione. Dato un insieme F di dipendenze funzionali su $R(T)$, si dice che r è un'**istanza valida** di $R(T)$, **rispetto a** F, se soddisfa tutte le dipendenze funzionali $X \rightarrow Y \in F$

Le dipendenze funzionali - 2

Significato delle dipendenze funzionali e loro legame con la nozione di chiave

Caso 1. R rappresenti un tipo di entità

Caso 2. R rappresenti una relazione molti a uno

Validità universale delle dipendenze funzionali (per ogni istanza valida)

Istanze che soddisfano le dipendenze funzionali (istanze valide) e istanze che violano le dipendenze funzionali

Dipendenze derivate (semantica)

Ragionamento sulle dipendenze funzionali (un esempio, transitività)

Implicazione logica tra dipendenze funzionali (nozione semantica)

Definizione. Dati una relazione $R(T)$ e un insieme di dipendenze funzionali F (abbreviato $R\langle T, F\rangle$), diciamo che F **implica logicamente** $X \to Y$ ($F \models X \to Y$) se ogni istanza valida r dello schema $R\langle T, F\rangle$ soddisfa $X \to Y$

Definizione. La **chiusura** di un **insieme di dipendenze funzionali** F (denotata F^+) è l'insieme delle dipendenze funzionali logicamente implicate da F:

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

Dipendenze funzionali e chiavi

La nozione di **chiave** può essere riespressa in termini di dipendenze funzionali

$X \subseteq T$ è chiave di $R\langle T, F\rangle$ se e solo se

1. $X \rightarrow A_1 A_2 \ldots A_n$, con $T = \{A_1, A_2, \ldots, A_n\}$, è in F^+ (la dipendenza di tutti gli attributi da X appartiene a F o è logicamente implicata da F)
2. per nessun sottoinsieme proprio $Y \subset X$, $Y \rightarrow A_1 A_2 \ldots A_n$ appartiene a F^+

Un calcolo delle dipendenze funzionali

Gli assiomi di Armstrong

- **Riflessività** (dipendenze banali) Se $Y \subseteq X \subseteq T$, allora $X \to Y$ (in particolare, $X \to X$)

  Osservazione: tali dipendenze valgono per ogni relazione (dipendono da T, non da F)

- **Aumento** Se $X \to Y$ e $Z \subseteq T$, allora $XZ \to YZ$ (dove XZ abbrevia $X \cup Z$)

- **Transitività** Se $X \to Y$ e $Y \to Z$, allora $X \to Z$

Teorema. Gli assiomi di Armstrong sono **corretti** (se f è derivabile da F mediante gli assiomi di Armstrong, allora f è implicato logicamente da F, abbreviato $F \vdash f$ implica $F \models f$)

Dipendenze derivate (sintassi)

Definizione. Una **derivazione** di una dipendenza funzionale f da F (denotato $F \vdash f$) è una sequenza finita $f_1, \ldots, f_m$ di dipendenze funzionali, dove $f_m = f$ e ogni dipendenza f_i appartiene a F o è ottenuta da dipendenze in $\{f_1, \ldots, f_{i-1}\}$ usando una regola di inferenza valida (assiomi di Armstrong o regole di inferenza da essi ottenute)

Si noti che ogni sottosequenza iniziale $f_1, \ldots, f_k$ di $f_1, \ldots, f_m$, con $k \leq m$, è a sua volta una derivazione, ossia $F \vdash f_k$ per ogni $k = 1, \ldots, m$

Esempio. Dati $R(A, B, C, D)$ e $F = \{A \to C, B \to D\}$, dimostrare che AB è una superchiave per R, ossia che $F \models AB \to ABCD$ (sfruttiamo la correttezza degli assiomi di Armstrong)

Regole derivate

Alcune regole derivate di particolare interesse:

- **Unione**

  Se $X \to A_1, \ldots, X \to A_n$, allora $X \to A_1 \ldots A_n$

  (si consideri, ad esempio, il caso $n = 2$)

- **Pseudo-Transitività**

  Se $X \to Y$ e $YW \to Z$, allora $XW \to Z$

- **Decomposizione**

  Se $X \to Y$ e $Z \subseteq Y$, allora $X \to Z$

Chiusura di un insieme di attributi

Definizione. Dato uno schema $R\langle T, F\rangle$ e un insieme di attributi $X \subseteq T$, la **chiusura** di X **rispetto** a F (denotata X_F^+ o, se non vi sono ambiguità, semplicemente X^+) è l'insieme di attributi $\{A \in T \mid F \vdash X \to A\}$

Legame tra derivazione di dipendenze funzionali e chiusura di insiemi di attributi

Teorema (fondamentale). $F \vdash X \to Y$ se e solo se $Y \subseteq X^+$

Teorema. Gli assiomi di Armstrong sono **completi** ($F \models f$ implica $F \vdash f$)

Determinazione delle chiusure - 1

Calcolare F^+ è **troppo costoso** (già le dipendenze banali...)

Un primo algoritmo (*chiusura lenta*) per il calcolo della chiusura di un insieme di attributi X rispetto ad F (X^+)

algoritmo CHIUSURA LENTA

input $R\langle T, F\rangle$, $X \subseteq T$

output X^+

begin

$X^+ := X$;

while X^+ è cambiato **do**

for each $W \rightarrow V \in F$ **with** $W \subseteq X^+$ **and** $V \not\subseteq X^+$ **do**

$X^+ := X^+ \cup V$

end

Determinazione delle chiusure - 2

Esempio. Sia dato uno schema $R\langle T, F\rangle$, con $T = \{A, B, C, D, E, G, H, I\}$ e $F = \{ADG \to GI, ACH \to ADG, BC \to AD, CE \to ACH\}$. Inoltre, sia X_j^+ il valore della variabile X^+ alla fine della j-esima iterazione del ciclo **while**

La chiusura di BCE viene calcolata nel seguente modo:

$X_0^+ (= X) = BCE$

$X_1^+ = BCE \cup AD \cup ACH = ABCDEH$

$X_2^+ = ABCDEH \cup ADG = ABCDEGH$

$X_3^+ = ABCDEGH \cup GI = ABCDEGHI (= T)$

L'algoritmo termina con $BCE^+ = T$ (BCE è superchiave)

Teorema. L'algoritmo *chiusura lenta* termina sempre ed è corretto e completo

Determinazione delle chiusure - 3

Un secondo algoritmo (*chiusura veloce*) per il calcolo della chiusura di un insieme di attributi X rispetto ad F (X^+)

algoritmo CHIUSURA VELOCE

input $R\langle T, F\rangle$, $X \subseteq T$

output X^+

begin

inizializzazione delle liste $num(f)$ e $L(A)$

for each $f = W \rightarrow V \in F$ **do**

$num(f) := |W|$;

for each $A \in W$ **do**

aggiungi f a $L(A)$;

inizializzazione delle variabili

$X^+ := X$;

$N_{ATT} := X$;

Determinazione delle chiusure - 4

while $N_{ATT} \neq \emptyset$ **do** # ciclo principale
scegli A da N_{ATT};
$N_{ATT} := N_{ATT} \setminus \{A\}$;
for each $f = W \rightarrow V \in L(A)$ **do**
$num(f) := num(f) - 1$;
if $num(f) = 0$ **then**
$\Delta := V \setminus X^+$;
$N_{ATT} := N_{ATT} \cup \Delta$;
$X^+ := X^+ \cup \Delta$

end

Osservazione. Mentre Δ in $X^+ := X^+ \cup \Delta$ può essere sostituito da V, è fondamentale non inserire in N_{ATT} attributi già presi in esame

Teorema. L'algoritmo *chiusura veloce* termina sempre ed è corretto e completo

Determinazione delle chiusure - 5

Esempio. Sia dato uno schema $R\langle T, F\rangle$, con $T = \{A, B, C, D, E\}$ e $F = \{f_1 = DB \rightarrow E, f_2 = B \rightarrow C, f_3 = A \rightarrow B\}$

Inizializzazione:

$num(f_1) = 2, num(f_2) = 1, num(f_3) = 1$

(nella tabella sottostante i valori correnti di $num(f_1), num(f_2)$ e $num(f_3)$ sono riportati nella quarta, quinta e sesta colonna, rispettivamente)

$L(A) = \{f_3\}, L(B) = \{f_1, f_2\}, L(C) = \emptyset, L(D) = \{f_1\}, L(E) = \emptyset$

X^+	N_{ATT}	A	$L(A)$	f_1	f_2	f_3	f	Δ	X^+	N_{ATT}
AD	AD	A	f_3	2	1	0	f_3	B	ADB	BD
ADB	BD	B	f_1, f_2	1	0	0	f_2	C	$ADBC$	CD
$ADBC$	CD	C	$\emptyset$	1	0	0	$\emptyset$	$\emptyset$	$ADBC$	D
$ADBC$	D	D	f_1	0	0	0	f_1	E	$ADBCE$	E
$ADBCE$	E	E	$\emptyset$	0	0	0	$\emptyset$	$\emptyset$	$ADBCE$	$\emptyset$

Insiemi di dipendenze equivalenti

Definizione. Due insiemi di dipendenze F e G sugli attributi T di una relazione R sono equivalenti ($F \equiv G$) se e solo se $F^+ = G^+$. Diciamo che F è una copertura di G, e viceversa G è una *copertura* di F, se $F \equiv G$

Dalla definizione segue immediatamente che $F \equiv G$ se e solo se $F^+ \subseteq G^+$ e $G^+ \subseteq F^+$

È possibile dimostrare che $F^+ \subseteq G^+$ se e solo se $F \subseteq G^+$ e, analogamente, che $G^+ \subseteq F^+$ se e solo se $G \subseteq F^+$

Coperture minimali (o canoniche) - 1

Insiemi di dipendenze minimali (***coperture minimali***/*canoniche*)

Definizione. Sia F un insieme di dipendenze funzionali.

1. Data $X \to Y \in F$, X contiene un *attributo estraneo* (o ridondante) A se e solo se $(F \setminus \{X \to Y\}) \cup \{X \setminus A \to Y\} \equiv F$, ossia se e solo se $X \setminus A \to Y \in F^+$
2. $X \to Y \in F$ è una *dipendenza ridondante* se e solo se $F \setminus \{X \to Y\} \equiv F$, ossia se e solo se $X \to Y \in (F \setminus \{X \to Y\})^+$

Coperture minimali (o canoniche) - 2

3. F è una *copertura canonica* se e solo se:
   a. ogni parte destra di una dipendenza ha un unico attributo
   b. le dipendenze non contengono attributi estranei
   c. non vi sono dipendenze ridondanti

Nota bene: l'ordine delle operazioni a, b e c è fondamentale

Teorema. Per ogni insieme di dipendenze F, esiste una **copertura canonica** (non necessariamente unica)

Calcolo della copertura minimale: l'algoritmo - 1

Algoritmo polinomiale per il calcolo di una copertura minimale/canonica (si assuma che ogni dipendenza funzionale abbia un unico attributo nella parte destra)

algoritmo CALCOLO DELLA COPERTURA CANONICA

input Insieme di dipendenze funzionali F

output G copertura canonica di F

begin

$G := F$;

for each $X \to Y \in G$ **with** $|X| > 1$ **do**

begin

$Z := X$;

for each $A \in X$ **do if** $Y \subseteq (Z \setminus \{A\})^+_F$ **then** $Z := Z \setminus \{A\}$;

$G := (G \setminus \{X \to Y\}) \cup \{Z \to Y\}$

end;

Calcolo della copertura minimale: l'algoritmo - 2

for each $X \to Y \in G$ **do**
if $Y \subseteq X^+_{G \setminus \{X \to Y\}}$ **then** $G := G \setminus \{X \to Y\}$
end

Osservazione. Un insieme di dipendenze funzionali può avere più coperture canoniche

Esempio. Per l'insieme $F = \{AB \to C, A \to B, B \to A\}$, sia $F = \{B \to C, A \to B, B \to A\}$ che $F = \{A \to C, A \to B, B \to A\}$ sono coperture canoniche

Decomposizione di schemi - 1

Definizione. Dato uno schema $R\langle T, F\rangle$, $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ è una **decomposizione** di R se e solo se $\cup_i T_i = T$

Non tutte le decomposizioni **conservano** i dati.

Esempio. Si consideri una relazione con schema $R(P, T, C)$, con associato l'insieme $F = \{T \rightarrow C, C \rightarrow P\}$ di dipendenze funzionali, che memorizza informazioni relative ai proprietari (P) di case (C) dotate di uno o più telefoni (T). Sia data la seguente istanza di R

P	T	C
$p1$	$t1$	$c1$
$p1$	$t2$	$c2$
$p1$	$t3$	$c2$

Decomposizione di schemi - 2

Esempio (continua). Supponiamo di decomporre lo schema nelle due relazioni $R_1(P, T)$ e $R_2(P, C)$. Le corrispondenti istanze sono $r_1 = \Pi_{P,T}(R)$

P	T
$p1$	$t1$
$p1$	$t2$
$p1$	$t3$

e $r_2 = \Pi_{P,C}(R)$

P	C
$p1$	$c1$
$p1$	$c2$

Decomposizione di schemi - 3

Esempio (continua). Per ricostruire l'istanza iniziale occorre fondere le due istanze della decomposizione attraverso un natural join. Il risultato che si ottiene non coincide, però, con l'istanza data: ha più tuple (e viola le dipendenze)

P	T	C
$p1$	$t1$	$c1$
$p1$	$t1$	$c2$
$p1$	$t2$	$c1$
$p1$	$t2$	$c2$
$p1$	$t3$	$c1$
$p1$	$t3$	$c2$

Decomposizioni lossless join

Definizione. Una decomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ di $R\langle T, F\rangle$ *preserva i dati* (è **lossless join**) se e solo se per ogni istanza $r \in R$ che soddisfa F, $r = \Pi_{T_1}(r) \bowtie \ldots \bowtie \Pi_{T_k}(r)$

In generale, vale solo il seguente teorema:

Teorema. Sia $m_\rho(r) = \Pi_{T_1}(r) \bowtie \ldots \bowtie \Pi_{T_k}(r)$. Data una decomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ qualunque di $R\langle T, F\rangle$, per ogni istanza $r \in R$ si ha che $r \subseteq m_\rho(r)$

Alcune (altre) proprietà interessanti (ad esempio, $r \subseteq s$ implica $m_\rho(r) \subseteq m_\rho(s)$)

Un punto di vista diverso: le *tuple isolate* (tuple che possono essere rappresentate solo attraverso alcune delle relazioni componenti)

Controllo dei lossless join: un algoritmo - 1

algoritmo CONTROLLO DEI LOSSLESS JOIN

input Uno schema relazionale $R\langle T, F\rangle$, con $T = \{A_1, \ldots, A_n\}$,
e una scomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$

output ρ è/non è una scomposizione lossless join

metodo **inizializzazione**

costruire una tabella con n colonne e k righe, dove la colonna j corrisponde all'attributo A_j e la riga i allo schema R_i;
porre il simbolo a_j nella posizione (i, j) se $A_j \in T_i$,
altrimenti porre b_{ij};

elaborazione

si considerino ordinatamente e ripetutamente le dipendenze $X \to Y \in F$ fino a quando esse non provocano più variazioni nella tabella

Controllo dei lossless join: un algoritmo - 2

per ogni dipendenza $X \rightarrow Y$ e ogni coppia di righe, se le due righe coincidono su X, allora renderle uguali su Y
e propagare le variazioni
(dati due simboli da uguagliare, se uno dei due è a_j, sostituire l'altro con a_j; se sono b_{ij} e b_{lj}, sostituire b_{lj} con b_{ij}, o viceversa, a discrezione;
quando due simboli sono uguagliati, uguagliare tutte le occorrenze di tali simboli nella tabella)

risultato

se al termine dell'elaborazione, esiste almeno una riga con tutte a (riga $a_1, \ldots, a_n$), la decomposizione è lossless join, altrimenti non lo è

Un esempio - 1

Esempio. Siano dati lo schema $R\langle T, F\rangle$, con $T = \{A, B, C, D, E\}$ e $F = \{A \to C, DE \to C, B \to C, CE \to A, C \to D\}$, e la decomposizione $\rho = \{R_1(\{A, D\}), R_2(\{A, B\}), R_3(\{B, E\}), R_4(\{C, D, E\}), R_5(\{A, E\})\}$

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{23}	b_{24}	b_{25}
b_{31}	a_2	b_{33}	b_{34}	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	b_{53}	b_{54}	a_5

Un esempio - 2

Esempio (continua). Con l'applicazione delle dipendenze $A \to C$, $DE \to C$ (non produce effetti) e $B \to C$, si ottiene la tabella:

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{13}	b_{24}	b_{25}
b_{31}	a_2	b_{13}	b_{34}	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	b_{13}	b_{54}	a_5

Un esempio - 3

Esempio (continua). Con l'applicazione delle dipendenze $CE \to A$ e $C \to D$, si ottiene la tabella:

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{13}	a_4	b_{25}
a_1	a_2	b_{13}	a_4	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	b_{13}	a_4	a_5

Un esempio - 4

Esempio (continua). All'iterazione successiva, grazie agli effetti delle dipendenze $DE \to C, B \to C$ e $CE \to A$, si ottiene la tabella:

A	B	C	D	E
a_1	b_{12}	a_3	a_4	b_{15}
a_1	a_2	a_3	a_4	b_{25}
a_1	a_2	a_3	a_4	a_5
a_1	b_{42}	a_3	a_4	a_5
a_1	b_{52}	a_3	a_4	a_5

L'applicazione delle dipendenze funzionali non produce più effetti.
C'è una riga con tutte a: decomposizione lossless join

Risultati - 1

Teorema. L'algoritmo proposto termina ed è corretto e completo.

La terminazione è immediata (è sufficiente osservare come ogni applicazione di una dipendenza che produce effetti riduca il numero di valori diversi presenti nella tabella)

È, inoltre, facile mostrare che se la tabella finale non contiene una riga di tutte a, allora la decomposizione non è lossless join (la tabella finale stessa costituisce un controesempio: essa può essere vista come un'istanza r della relazione, che soddisfa tutte le dipendenze, tale che $(a_1, \ldots, a_n) \notin r$ e $(a_1, \ldots, a_n) \in m_\rho(r)$; l'ultima appartenenza segue dalla definizione di decomposizione e dalle modalità di inizializzazione ed elaborazione della tabella)

Risultati - 2

Osservazione. L'esecuzione dell'algoritmo può essere terminata non appena viene prodotta una riga con tutte a (una volta generata, una tale riga non può essere più rimosssa)

Un caso particolare:

Teorema. Sia $\rho = \{R_1(T_1), R_2(T_2)\}$ una decomposizione di $R\langle T, F\rangle$. ρ è una decomposizione lossless join se e solo se $T_1 \cap T_2 \rightarrow T_1 \in F^+$ oppure $T_1 \cap T_2 \rightarrow T_2 \in F^+$ (o entrambe)

Decomposizioni che preservano le dipendenze - 1

(Contro)esempio. Consideriamo nuovamente la relazione $R(P, T, C)$, con associato l'insieme di dipendenze $F = \{T \to C, C \to P\}$, che memorizza informazioni relative ai proprietari (P) di case (C) dotate di uno o più telefoni (T), e la relativa istanza

P	T	C
$p1$	$t1$	$c1$
$p1$	$t2$	$c2$
$p1$	$t3$	$c2$

Sia data la decomposizione $\rho = \{R_1(\{P, T\}), R_2(\{T, C\})\}$

Decomposizioni che preservano le dipendenze - 2

Controesempio (continua). Le corrispondenti istanze sono
$r_1 = \Pi_{P,T}(R)$

P	T
$p1$	$t1$
$p1$	$t2$
$p1$	$t3$

e $r_2 = \Pi_{T,C}(R)$

T	C
$t1$	$c1$
$t2$	$c2$
$t3$	$c2$

Decomposizioni che preservano le dipendenze - 3

Controesempio (continua). La decomposizione preserva i dati perché $PT \cap TC (= T) \rightarrow TC$ è derivabile da F (aumento di $T \rightarrow C$ con T) e, quindi, per la correttezza degli assiomi di Armstrong, appartiene a F^+

La decomposizione non conserva, però, la dipendenza $C \rightarrow P$, dove $C \in R_2$ e $P \in R_1$.

Conseguenze della **perdita di dipendenze**: inserimenti inconsistenti

Un esempio di inserimento inconsistente

Esempio. Si supponga di voler inserire nella base di dati il fatto che la casa c_2 ha un nuovo telefono t_4 intestato alla persona p_2.

Se applicato alla relazione R, un tale inserimento verrebbe proibito perché violerebbe la dipendenza $C \rightarrow P$ (la casa c_2 è già associata alla persona p_1 e il telefono va intestato all'unico proprietario della casa)

Se applicato alle relazioni R_1 ed R_2, l'inserimento va a buon fine, perché non viola le dipendenze associate ai due schemi (la dipendenza $C \rightarrow P$ è stata persa per effetto della decomposizione)

Decomposizione alternativa che **preserva dati e dipendenze**:
$\rho = \{R_1(\{T, C\}), R_2(\{C, P\})\}$

Proiezione di insiemi di dipendenze

Definizione. Dati $R\langle T, F\rangle$ e $T_i \subseteq T$, la proiezione di F su T_i, denotata $\Pi_{T_i}(F)$, è l'insieme di dipendenze

$$\{X \to Y \in F^+ \mid X, Y \subseteq T_i\}$$

Nota Bene: la definizione utilizza l'insieme F^+ non l'insieme F

Esempio. Si consideri la relazione $R\langle T, F\rangle$, con $T = \{A, B, C\}$ e $F = \{A \to B, B \to C, C \to A\}$. Si ha che $\Pi_{AB}(F) = \{A \to B, B \to A\}$ (più le dipendenze banali e dipendenze immediatamente derivabili, quale, ad esempio, $A \to AB$) e $\Pi_{AC}(F) = \{A \to C, C \to A\}$ (più le dipendenze banali e dipendenze immediatamente derivabili).

Un algoritmo per il calcolo delle proiezioni

Un algoritmo banale (di complessità esponenziale) per il calcolo delle proiezioni di un insieme di dipendenze

algoritmo CALCOLO DELLE PROIEZIONI

input $R\langle T, F\rangle$ e $T_i \subseteq T$

output una copertura della proiezione di F su T_i

begin

for each $Y \subseteq T_i$ **do**

begin

$Z := Y_F^+$

restituisci $Y \rightarrow (Z \cap T_i)$

end

end

Conservazione delle dipendenze: formalizzazione

Definizione. Una decomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ di una relazione $R\langle T, F\rangle$ **preserva le dipendenze** se e solo se $\cup_{i=1}^{k} \Pi_{T_i}(F) \equiv F$

Esempio. Si consideri lo schema di relazione $R\langle T, F\rangle$, dove $T = \{A, B, C\}$ e $F = \{A \to B, B \to C, C \to A\}$. Dato che $AC \not\subseteq AB$ and $AC \not\subseteq BC$, uno potrebbe essere indotto a pensare che la decomposizione $\rho = \{R_1(\{A, B\}), R_2(\{B, C\})\}$ non preservi le dipendenze (la dipendenza $C \to A$ non viene proiettata né su R_1 né su R_2).

Così non è: da $\{A \to B, B \to A\} \subseteq \Pi_{AB}(F)$ e $\{B \to C, C \to B\} \subseteq \Pi_{BC}(F)$, segue $\Pi_{AB}(F) \cup \Pi_{BC}(F) \vdash C \to A$

Proiezioni e calcolo della chiusura

Un algoritmo (di complessità polinomiale) per il calcolo della chiusura di un insieme di attributi rispetto all'unione delle proiezioni di un insieme di dipendenze

algoritmo CALCOLO DELLA CHIUSURA

input $R\langle T, F\rangle$, $X \subseteq T$ e $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$

output X_G^+, dove $G = \cup_{i=1}^{k} \Pi_{T_i}(F)$

begin

$X_G^+ := X$;

while (X_G^+ è cambiato) **do**

for each $i := 1$ **to** k **do**

$$X_G^+ = X_G^+ \cup ((X_G^+ \cap T_i)_F^+ \cap T_i)$$

end

Nota bene: l'insieme G non viene esplicitamente determinato

Controllo della conservazione delle dipendenze

Un algoritmo (di complessità polinomiale) per stabilire se una data scomposizione preserva o meno le dipendenze associate ad una data relazione

algoritmo CONTROLLO DELLA CONSERVAZIONE DELLE DIPENDENZE

input una decomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ di $R\langle T, F\rangle$

output ρ è/non è una decomposizione che preserva le dipendenze

begin

for each $X \rightarrow Y \in F$ **do**

if $Y \not\subseteq X_G^+$ **then** termina con NO

termina con SI

end

dove X_G^+ è determinato con l'algoritmo per il calcolo della chiusura

Un esempio - 1

Esempio. Sia data lo schema di relazione $R\langle T, F\rangle$, dove $T = \{A, B, C, D\}$ e $F = \{A \to B, B \to C, C \to D, D \to A\}$. Data la circolarità delle dipendenze, è facile vedere come ogni attributo determini tutti gli altri.

Controlliamo se la decomposizione $\rho = \{R_1(\{A, B\}), R_2(\{B, C\}), R_3(\{C, D\})\}$ preserva o meno le dipendenze utilizzando l'algoritmo proposto.

Le prime tre dipendenze sono banalmente preservate.
Concentriamo la nostra attenzione sulla quarta, verificando se $A \in D_G^+$, dove $G = \Pi_{AB}(F) \cup \Pi_{BC}(F) \cup \Pi_{CD}(F)$

Un esempio - 2

Iniziamo il calcolo di D_G^+, ponendo $D_G^+ = \{D\}$. Il corpo del ciclo eseguito per R_1 non modifica D_G^+ perché $\{D\} \cup ((\{D\} \cap \{A, B\})_F^+ \cap \{A, B\}) = \{D\}$. Lo stesso accade per R_2 perché $\{D\} \cup ((\{D\} \cap \{B, C\})_F^+ \cap \{B, C\}) = \{D\}$. Per R_3 si ha:

$$\begin{aligned} D_G^+ &= \{D\} \cup ((\{D\} \cap \{C, D\})_F^+ \cap \{C, D\}) \\ &= \{D\} \cup ((\{D\})_F^+ \cap \{C, D\}) \\ &= \{D\} \cup (\{A, B, C, D\} \cap \{C, D\}) \\ &= \{C, D\} \end{aligned}$$

Analogamente, nel passo successivo, eseguendo il corpo del ciclo per R_2 su $D_G^+ = \{C, D\}$ produce $D_G^+ = \{B, C, D\}$. Infine, nel terzo passo si ottiene $D_G^+ = \{A, B, C, D\}$, da cui risulta vero che $A \in D_G^+$.

Conservazione di dati e dipendenze

Come confermato dagli esempi precedenti, conservazione dei dati e conservazione delle dipendenze sono due proprietà **indipendenti**:

- vi possono essere decomposizioni lossless join che non preservano le dipendenze
- vi possono essere decomposizioni che preservano le dipendenze, ma non sono lossless join

Forme normali - 1

Normalizzazione dei dati: processo attraverso il quale schemi di relazione insoddisfacenti, ossia schemi che presentano ridondanze e possono, quindi, causare anomalie di aggiornamento, sono decomposti in schemi di relazione più piccoli che possiedono le proprietà volute

Forme normali: strumento per la valutazione della qualità di schemi di relazione individuali. Permettono di stabilire se uno schema si trova o meno nella forma (normale) voluta. In caso di violazione, lo schema può essere decomposto in modo che gli schemi delle relazioni componenti rispettino la forma (normale) voluta

Forme normali - 2

Prima forma normale (1NF), basata sulla definizione di relazione

Seconda forma normale (2NF), **terza** forma normale (3NF) e forma normale di **Boyce-Codd** (BCNF), tutte basate sulla nozione di dipendenza funzionale

Quarta forma normale, basata sulla nozione di dipendenza multivalore

Quinta forma normale, basata sulla nozione di join-dipendenza

Forme normali - 3

Osservazione fondamentale

in generale non è sufficiente garantire che il processo di decomposizione generi relazioni nella forma normale voluta; occorre anche garantire che:

1. la decomposizione *conservi i dati* (sia lossless join)
2. la decomposizione *preservi le dipendenze*

Definizione. Dato uno schema di relazione $R\langle T, F\rangle$, un attributo $A \in T$ si dice **primo** se e solo se appartiene ad almeno una chiave; altrimenti, si dice **non primo**

Il problema di stabilire se un attributo è primo o meno è NP-completo

Prima forma normale - 1

Eliminazione degli attributi multivalore

Relazione non in 1NF:

DIPARTIMENTO	DNUMERO	DNOME	MANAGER	DSEDE

con dipendenze:

$DNUMERO \rightarrow DNOME$, $DNUMERO \rightarrow MANAGER$, $DNUMERO \rightarrow DSEDE$(?)

Istanza di relazione:

DIPARTIMENTO	DNUMERO	DNOME	MANAGER	DSEDE
	5	Ricerca	RSS..	$\{Roma, Bologna\}$
	4	Sviluppo	VRD..	$Venezia$
	1	Direzione	BNC..	$Bologna$

Prima forma normale - 2

Relazione in 1NF (con ridondanza):

DIPARTIMENTO	DNUMERO	DNOME	MANAGER	DSEDE
	5	Ricerca	RSS..	*Roma*
	5	Ricerca	RSS..	*Bologna*
	4	Sviluppo	VRD..	*Venezia*
	1	Direzione	BNC..	*Bologna*

Prima forma normale - 3

Eliminazione degli attributi strutturati

Relazione IMP_PROG (Impiegato_Progetto) con attributi CF (codice fiscale), INOME (nome impiegato) e PROG (progetto a cui l'impiegato dedica il maggior numero di ore, con indicazione del numero di ore) non in 1NF:

IMP_PROG	CF	INOME	PROG(PNUMERO,ORE)

con dipendenze:

$CF \rightarrow INOME, CF \rightarrow PROG(?)$

L'attributo strutturato PROG può essere visto come una relazione annidata

Prima forma normale - 4

Istanza di relazione:

IMP_PROG	CF	INOME	PROG(PNUMERO,ORE)
	RSS..	Rossi Antonio	(1,18)
	VRD..	Verdi Filippo	(3,36)
	BNC..	Bianchi Livio	(3,12)

Relazione in 1NF (senza ridondanze):

IMP_PROG	CF	INOME	PNUMERO	ORE
	RSS..	Rossi Antonio	1	18
	VRD..	Verdi Filippo	3	36
	BNC..	Bianchi Livio	3	12

Prima forma normale - 5

Eliminazione degli attributi strutturati e multivalore

Relazione IMP_PROG (Impiegato_Progetto) con una relazione annidata PROG multivalore non in 1NF:

IMP_PROG	CF	INOME	PROG(PNUMERO,ORE)

con dipendenze:

$CF \rightarrow INOME, CF \rightarrow PROG(?)$

Prima forma normale - 6

Istanza di relazione:

IMP_PROG	CF	INOME	PROG(PNUMERO,ORE)
	RSS..	Rossi Antonio	{(1,18), (2,18)}
	VRD..	Verdi Filippo	{(3,36)}
	BNC..	Bianchi Livio	{(2,12),(3,12),(5,12)}

Relazione IMP_PROG in 1NF

IMP_PROG	CF	INOME	PNUMERO	ORE

Seconda forma normale - 1

La nozione di *dipendenza funzionale parziale* e *totale*

Una dipendenza funzionale $X \rightarrow Y$ è una dipendenza funzionale **totale** se la rimozione di un qualsiasi attributo A da X rende la dipendenza non più valida $(\forall A(X \setminus \{A\} \not\rightarrow Y))$

Una dipendenza funzionale $X \rightarrow Y$ è una dipendenza funzionale **parziale** se qualche attributo di X può essere rimosso senza pregiudicare la validità della dipendenza $(\exists A(X \setminus \{A\} \rightarrow Y))$

Definizione. Uno schema di relazione $R\langle T, F\rangle$ è in **seconda forma normale** (2NF) se ogni attributo non primo $A \in T$ dipende totalmente da ogni chiave di R

Seconda forma normale - 2

Sia dato lo schema di relazione (non in 2NF):

IMP_PROG	CF	PNUMERO	ORE	INOME	PNOME	PSEDE

con dipendenze:

$CF\ PNUMERO \rightarrow ORE$

$CF \rightarrow INOME$

$PNUMERO \rightarrow PNOME\ PSEDE$

Seconda forma normale - 3

Decomposizione in 2NF:

IP1	CF	PNUMERO	ORE

con dipendenze:

$CF\ PNUMERO \rightarrow ORE$

IP2	CF	INOME

con dipendenze:

$CF \rightarrow INOME$

IP3	PNUMERO	PNOME	PSEDE

con dipendenze:

$PNUMERO \rightarrow PNOME\ PSEDE$

Terza forma normale - 1

La nozione di *dipendenza funzionale transitiva* e *non transitiva*

Una dipendenza funzionale $X \to Y$ è una dipendenza funzionale **transitiva** se esiste un insieme di attributi Z che non è contenuto in alcuna chiave ed è tale che valgono sia $X \to Z$ che $Z \to Y$

Definizione. Uno schema di relazione $R\langle T, F\rangle$ è in **terza forma normale** (3NF) se ogni attributo non primo $A \in T$ dipende totalmente da ogni chiave di R (R è in 2NF) e dipende in modo non transitivo da ogni chiave di R

Definizione (alternativa). Uno schema di relazione $R\langle T, F\rangle$ è in **terza forma normale** (3NF) se e solo se, per ogni dipendenza funzionale non banale $X \to A \in F^+$, X è superchiave o A è primo

Terza forma normale - 2

Sia dato lo schema di relazione (non in 3NF):

IMP_DIP	CF	INOME	DNASCITA	IND	DNUMERO	DNOME	MANAGER

con dipendenze:

$CF \rightarrow INOME\ DNASCITA\ IND\ DNUMERO$

$DNUMERO \rightarrow DNOME\ MANAGER$

Terza forma normale - 3

Decomposizione in 3NF:

ID1	CF	INOME	DNASCITA	IND	DNUMERO

con dipendenze:

$CF \rightarrow INOME\ DNASCITA\ IND\ DNUMERO$

IP2	DNUMERO	DNOME	MANAGER

con dipendenze:

$DNUMERO \rightarrow DNOME\ MANAGER$

Un esempio di normalizzazione - 1

Sia dato lo schema di relazione (non in 2NF):

APPEZZAMENTO	PROP_ID	CONTEA	APP_ID	AREA	VALORE	ALIQUOTA

con dipendenze:

$PROP\_ID \rightarrow CONTEA\ APP\_ID\ AREA\ VALORE\ ALIQUOTA$

$CONTEA\ APP\_ID \rightarrow PROP\_ID\ AREA\ VALORE\ ALIQUOTA$

$CONTEA \rightarrow ALIQUOTA$

$AREA \rightarrow VALORE$

Un esempio di normalizzazione - 2

Decomposizione in 2NF:

APPEZZAMENTO1	PROP_ID	CONTEA	APP_ID	AREA	VALORE

con dipendenze:

$PROP\_ID \rightarrow CONTEA\ APP\_ID\ AREA\ VALORE$

$CONTEA\ APP\_ID \rightarrow PROP\_ID\ AREA\ VALORE$

$AREA \rightarrow VALORE$

APPEZZAMENTO2	CONTEA	ALIQUOTA

con dipendenze:

$CONTEA \rightarrow ALIQUOTA$

Un esempio di normalizzazione - 3

Decomposizione in 3NF:

APPEZZAMENTO11	PROP_ID	CONTEA	APP_ID	AREA

con dipendenze:

$PROP\_ID \rightarrow CONTEA\ APP\_ID\ AREA$

$CONTEA\ APP\_ID \rightarrow PROP\_ID\ AREA$

APPEZZAMENTO12	AREA	VALORE

con dipendenze:

$AREA \rightarrow VALORE$

Controllo della 3NF - 1

Come verificare se uno schema di relazione è in 3NF senza generare F^+?

Valgono il seguente teorema e l'immediato corollario

Teorema. Uno schema di relazione $R\langle T, F\rangle$ è in 3NF se e solo se, per ogni dipendenza funzionale $X \to A_1 \,..\, A_n \in F$ e per ogni $i \in \{1, .., n\}$, $A_i \in X$ oppure X è una superchiave oppure A_i è primo

Corollario. Uno schema di relazione $R\langle T, F\rangle$, con F copertura canonica, è in 3NF se e solo se, per ogni dipendenza funzionale elementare $X \to A \in F$, X è una superchiave oppure A è primo

Controllo della 3NF - 2

E' comunque necessario determinare l'insieme di attributi primi..

Proposizione. Il problema di decidere se uno schema di relazione è in 3NF è NP-completo

Esempio. Consideriamo lo schema di relazione $R\langle\{PREFISSO, NUMERO, LOCALITA', NOMEABBONATO, VIA\}, \{PREFISSO\ NUMERO \rightarrow LOCALITA', PREFISSO\ NUMERO \rightarrow NOMEAABONATO, PREFISSO\ NUMERO \rightarrow VIA, LOCALITA' \rightarrow PREFISSO\}\rangle$ che descrive gli abbonati al telefono. Le chiavi della relazione sono $(PREFISSO\ NUMERO)$ e $(LOCALITA', NUMERO)$

Esiste, però, una **ridondanza**: ogni volta che si inserisce un nuovo numero telefonico di una certa località, occorre ripetere l'informazione sul prefisso

La forma normale di Boyce-Codd (BCNF)

Definizione. Uno schema di relazione $R\langle T, F\rangle$ è nella **forma normale di Boyce-Codd** (BCNF) se e solo se per ogni dipendenza funzionale non banale $X \to Y \in F^+$, X è una superchiave.

Come verificare se uno schema di relazione è in BCNF senza generare F^+?

Valgono il seguente teorema e l'immediato corollario

Teorema. Uno schema di relazione $R\langle T, F\rangle$ è in BCNF se e solo se, per ogni dipendenza funzionale non banale $X \to Y \in F$, X è una superchiave

Corollario. Uno schema di relazione $R\langle T, F\rangle$, con F copertura canonica, è in BCNF se e solo se, per ogni dipendenza funzionale elementare $X \to A \in F$, X è una superchiave

Ottimalità di BCNF

Teorema. In una relazione in BCNF, nessun valore può essere determinato a partire da altri utilizzando le dipendenze funzionali

Il teorema stabilisce l'**ottimalità** della BCNF rispetto ai vincoli imposti dalle dipendenze funzionali

Un secondo esempio di normalizzazione - 1

Sia dato lo schema di relazione (non in BCNF):

APPEZZAMENTO1A	PROP_ID	CONTEA	APP_ID	AREA

con dipendenze:

$PROP\_ID \rightarrow CONTEA\ APP\_ID\ AREA$

$CONTEA\ APP\_ID \rightarrow PROP\_ID\ AREA$

$AREA \rightarrow CONTEA$

Un secondo esempio di normalizzazione - 2

Decomposizione in BCNF:

APPEZZAMENTO1A1	PROP_ID	APP_ID	AREA

con dipendenze:

$PROP\_ID \rightarrow APP\_ID\ AREA$

APPEZZAMENTO1A2	AREA	CONTEA

con dipendenze:

$AREA \rightarrow CONTEA$

BCNF e conservazione delle dipendenze

Decomposizione in BCNF di uno schema di relazione e conservazione delle dipendenze non sono obiettivi che possano essere sempre raggiunti in modo congiunto

R	A	B	C

con dipendenze:

$A\ B \rightarrow C$

$C \rightarrow B$

Normalizzazione di schemi in BCNF - 1

Un algoritmo esponenziale che preserva i dati, ma non le dipendenze, è il seguente

algoritmo DECOMPOSIZIONE IN BCNF
input $R\langle T, F\rangle$ con F copertura canonica
output $\rho = \{R_1, \ldots, R_m\}$ decomposizione di R in BCNF che preserva i dati
begin
$\rho := \{R_1\langle T, F\rangle\}; n := 1;$
while esiste $R_i\langle T_i, F_i\rangle \in \rho$ non in BCNF a causa di $X \to A$ **do**
begin
$n := n + 1;$
$T' := X^+; F' := \Pi_{T'}(F_i); T'' := T_i - A; F'' := \Pi_{T''}(F_i);$
$\rho := (\rho \setminus R_i\langle T_i, F_i\rangle) \cup \{R_i\langle T', F'\rangle, R_n\langle T'', F''\rangle\}$
end;
end

Normalizzazione di schemi in BCNF - 2

La conservazione dei dati da parte dell'algoritmo precedente segue dal seguente teorema

Teorema. Sia $\rho = \{R_1, \ldots, R_m\}$ una decomposizione di $R\langle T, F\rangle$ che preserva i dati (rispetto a F) e sia $\sigma = \{S_1, S_2\}$ una decomposizione di R_1 che preserva i dati (rispetto a $\Pi_{T_1}(F)$). Allora la decomposizione $\rho' = \{S_1, S_2, R_2, \ldots, R_m\}$ preserva i dati (rispetto a F)

Osservazione. Esistono anche algoritmi polinomiali di decomposizione in BCNF, ma vengono poco utilizzati perché producono schemi poco naturali e fortemente decomposti

Decomposizioni di schemi in 3NF - 1

Un algoritmo polinomiale che preserva i dati e le dipendenze

algoritmo DECOMPOSIZIONE IN 3NF
input una relazione $R\langle T, F\rangle$
output una decomposizione $\rho = \{R_1, \ldots, R_m\}$ in 3NF
che preserva i dati e le dipendenze
begin
Passo 1. Trova una copertura canonica G di F e poni $\rho = \emptyset$
Passo 2. Sostituisci in G ogni insieme di dipendenze $X \to A_1$, $\ldots, X \to A_k$ con la dipendenza $X \to A_1, \ldots, A_k$
Passo 3. Per ogni dipendenza $X \to Y \in G$, inserisci uno schema con attributi XY in ρ (X è detta chiave sintetizzata)
Passo 4. Elimina da ρ ogni schema contenuto in un altro
Passo 5. Se nessuno schema in ρ ha come insieme di attributi una superchiave di R, aggiungi uno schema con attributi W, con W chiave di R, a ρ
end

Decomposizioni di schemi in 3NF - 2

Teorema. Dato uno schema $R\langle T, F\rangle$, l'algoritmo proposto produce una decomposizione $\rho = \{R_1, \ldots, R_m\}$ di R che preserva i dati e le dipendenze

Teorema. Dato uno schema $R\langle T, F\rangle$, l'algoritmo proposto produce una decomposizione $\rho = \{R_1, \ldots, R_m\}$ di R in schemi in 3NF (rispetto alle proiezioni di F su T_i, con $i = 1, \ldots, m$)