Basi di dati

Linguaggi di interrogazione

Docente: Stefano Paraboschi
parabosc@elet.polimi.it

Classificazione

a linguaggi formali

- **Algebra relazionale**
- **Calcolo relazionale**
- **Programmazione logica**

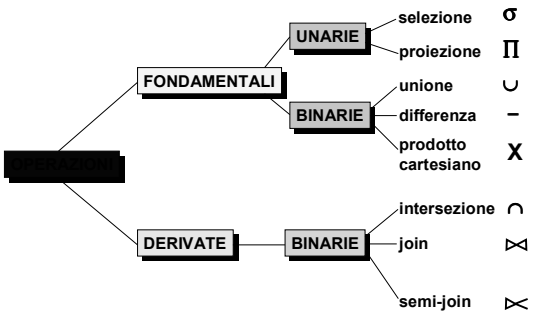
b linguaggi programmativi

- **SQL: Structured Query Language**
- **QBE: Query By Example**

Algebra relazionale

- **definita da Codd (70)**
- **molto utile per imparare a formulare query**
- **insieme minimo di 5 operazioni che danno l'intero potere espressivo del linguaggio**

Una visione d'insieme



Esempio : gestione degli esami universitari

studente

MATR	NOME	CITTA'	C-DIP
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

esame

MATR	COD-CORSO	DATA	VOTO
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

corso

COD-CORSO	TITOLO	DOCENTE
1	matematica	Barozzi
2	informatica	Meo

Selezione

$\sigma_{\text{Nome='Paola'}}$ **STUDENTE**

e' una tabella (priva di nome) con

- **schema :**
  lo stesso schema di STUDENTE
- **istanza :**
  le tuple di STUDENTE che soddisfano il predicato di selezione

Matr	Nome	Città	CDip
415	Paola	Torino	Inf

Sintassi del predicato di selezione

espressione booleana di predicati semplici

operazioni booleane :
- AND (P1 AND P2) (∧)
- OR (P1 OR P2) (∨)
- NOT (P1) (¬)

predicati semplici :
- TRUE, FALSE
- termine comparatore termine

comparatore :
- =, !=, <, <=, >, >=

termine :
- costante, attributo
- espressione aritmetica di costanti e attributi

7

Esempio di selezione

σ STUDENTE
(Città='Torino') OR
((Città='Roma')
AND NOT (CDip='Log'))

MATR	NOME	CITTA'	C-DIP
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

8

Proiezione

$\Pi_{Nome,CDip}$ STUDENTE

è una tabella (priva di nome) con
- schema :
  gli attributi Nome e CDip
- istanza :
  la restrizione delle tuple sugli attributi Nome e CDip

Nome	CDip
Carlo	Inf
Paola	Inf
Antonio	Log

9

Proiezione e duplicati

- nel modello formale la proiezione elimina i duplicati

Π_{CDip} STUDENTE

CDip
Inf
Log

- nel modello informale (e nei sistemi) la eliminazione dei duplicati va richiesta esplicitamente

10

Assegnamento

- serve per dare un nome al risultato di una espressione algebrica
- non fa parte delle operazioni algebriche

INFORMATICI = $\sigma_{CDIP='Inf'}$ STUDENTI

TORINESI = $\sigma_{Città='Torino'}$ STUDENTI

11

UNIONE E DIFFERENZA

- tabella1 ∪ tabella2

- tabella1 – tabella2

tabella1 E tabella2 DEVONO ESSERE COMPATIBILI
- STESSO GRADO
- DOMINI ORDINATAMENTE DELLO STESSO TIPO

12

Unione

INFORMATICI ∪ TORINESI

è una tabella (priva di nome) con

- **schema :**
  lo schema di INFORMATICI
- **istanza :**
  la unione delle tuple di INFORMATICI e TORINESI

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf

Per quanto riguarda le istanze, è commutativa

13

Differenza

INFORMATICI - TORINESI

è una tabella (priva di nome) con

- **schema :**
  lo schema di INFORMATICI
- **istanza :**
  la differenza delle tuple di INFORMATICI e TORINESI

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf

Attenzione: non è commutativa

14

Prodotto cartesiano

R × S

è una tabella (priva di nome) con

- **schema :**
  gli attributi di R e S
  (grado(RxS)= grado(R)+grado(S))
- **istanza :**
  tutte le possibili coppie di tuple di R e S
  (card(RxS)=card(R)*card(S))

15

Esempio

R1(A,B)

A	B
a	1
b	3

R2(C,D)

C	D
c	1
b	3
a	2

R1xR2 (A,B,C,D)

A	B	C	D
a	1	c	1
a	1	b	3
a	1	a	2
b	3	c	1
b	3	b	3
b	3	a	2

16

Intersezione

TABELLA1 ∩ TABELLA2

Come gli altri operatori insiemistici, si può fare se TABELLA1 e TABELLA2 sono compatibili

Derivabile tramite la seguente formula:

R ∩ S = R - (R - S)

17

Intersezione

INFORMATICI ∩ TORINESI

è una tabella (priva di nome) con

- **schema :**
  lo schema di INFORMATICI
- **istanza :**
  la intersezione delle tuple di INFORMATICI e TORINESI

Matr	Nome	Città	CDip
415	Paola	Torino	Inf

18

Join

STUDENTE $\triangleright\triangleleft_{\text{STUDENTE.Matr=ESAME.Matr}}$ ESAME

è equivalente alla seguente espressione (operatore derivato):

$\sigma_{\text{STUDENTE.Matr=ESAME.Matr}}$ STUDENTE $\times$ ESAME

attributi omonimi sono resi non ambigui usando la notazione "puntata":
ESAME.Matr, STUDENTE.Matr

19

Join

STUDENTE $\triangleright\triangleleft_{\text{STUDENTE.Matr=ESAME.Matr}}$ ESAME

produce una tabella (priva di nome) con

- **schema :**
  la concatenazione degli schemi di STUDENTE e ESAME
- **istanza:**
  le tuple ottenute concatenando quelle tuple di STUDENTE e di ESAME che soddisfano il predicato

STUDENTE. Matr	Nome	Città	CDip	ESAME. Matr	Cod-Corso	Data	Voto
123	Carlo	Bologna	Inf	123	1	7-9-97	30
123	Carlo	Bologna	Inf	123	2	8-1-98	28
702	Antonio	Roma	Log	702	2	7-9-97	20

20

Sintassi del predicato di join

espressione congiuntiva di predicati semplici:

ATTR1 comp ATTR2

ove ATTR1 appartiene a TAB1
ATTR2 appartiene a TAB2
comp: =, !=, <, <=, >, >=

EQUI-JOIN :
soli confronti di uguaglianza

21

Join naturale

equi-join di tutti gli attributi omonimi (si omette il predicato, si elimina la colonna ripetuta)

STUDENTE $\triangleright\triangleleft$ ESAME

Matr	Nome	Città	CDip	Cod-Corso	Data	Voto
123	Carlo	Bologna	Inf	1	7-9-97	30
123	Carlo	Bologna	Inf	2	8-1-98	28
702	Antonio	Roma	Log	2	7-9-97	20

22

Join naturale di tre tabelle

STUDENTE $\triangleright\triangleleft$ ESAME $\triangleright\triangleleft$ CORSO

Matr	Nome	Città	CDip	Cod-Corso	Data	Voto	Titolo	Docente
123	Carlo	Bologna	Inf	1	7-9-97	30	matem	barozzi
123	Carlo	Bologna	Inf	2	8-1-98	28	infor	meo
702	Antonio	Roma	Log	3	7-9-97	20	infor	meo

23

Semi-join

STUDENTE $\triangleright\!<_{\text{STUDENTE.Matr=ESAME.Matr}}$ ESAME

è equivalente alla seguente espressione (operatore derivato):

$\Pi_{\text{STUDENTE.*}}$ (STUDENTE $\triangleright\triangleleft_{\text{STUDENTE.Matr=ESAME.Matr}}$ ESAME)

Produce come risultato un sottoinsieme di STUDENTE

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
702	Antonio	Roma	Log

24

Semi-join naturale

STUDENTE ⋉ ESAME

è equivalente alla seguente espressione:

$\Pi_{\text{STUDENTE.*}}$ (STUDENTE ⋈ ESAME)

E' equivalente alla query precedente

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
702	Antonio	Roma	Log

25

Espressioni algebriche

- **Concatenazione di più operazioni algebriche**
- **Esprimono interrogazioni in modo formale**
- **Consentono di estrarre informazioni dai dati**

26

Selezione e proiezione

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

- **quali studenti sono iscritti al diploma di informatica?**

$\Pi_{\text{Nome}}\ \sigma_{\text{CDip='Inf'}}$ **STUDENTE**

NOME
Carlo
Paola

27

Selezione e proiezione

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

- **quali studenti di Logistica non sono di Milano?**

$\Pi_{\text{Nome}}\ \sigma_{\text{CDip='Log'} \wedge \text{Città != 'Milano'}}$ **STUDENTE**

NOME
Antonio

28

Esempio : gestione degli esami universitari

STUDENTE

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

ESAME

Matr	Cod-Corso	Data	Voto
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

CORSO

Cod-Corso	Titolo	Docente
1	matematica	Barozzi
2	informatica	Meo

29

Ricordiamo anche che...

STUDENTE ⋉ ESAME ⋉ CORSO

Matr	Nome	Città	CDip	Cod-Corso	Data	Voto	Titolo	Docente
123	Carlo	Bologna	Inf	1	7-9-97	30	matem	barozzi
123	Carlo	Bologna	Inf	2	8-1-98	28	infor	meo
702	Antonio	Roma	Log	3	7-9-97	20	infor	meo

30

Selezione, proiezione e join

- quali studenti hanno preso 30 in matematica?

Matr	Nome	Città	CDip	Cod-Corso	Data	Voto	Titolo	Docente
123	Carlo	Bologna	Inf	1	7-9-97	30	matem	barozzi
123	Carlo	Bologna	Inf	2	8-1-98	28	infor	meo
702	Antonio	Roma	Log	3	7-9-97	20	infor	meo

31

Selezione, proiezione e join

- quali studenti hanno preso 30 in matematica?

$\Pi_{\text{Nome}}\ \sigma_{\text{Voto=30} \wedge \text{Titolo='matematica'}}$
(STUDENTE $\triangleright\triangleleft$ ESAME $\triangleright\triangleleft$ CORSO)

Nome
Carlo

32

Equivalenza di espressioni

- quali studenti hanno preso 30 in matematica?

STUDENTE

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

ESAME

Matr	Cod-Corso	Data	Voto
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

CORSO

Cod-Corso	Titolo	Docente
1	matematica	Barozzi
2	informatica	Meo

33

Equivalenza di espressioni

- quali studenti hanno preso 30 in matematica?

Π_{Nome}
(STUDENTE $\triangleright\triangleleft$
($\sigma_{\text{Voto=30}}$ ESAME) $\triangleright\triangleleft$
($\sigma_{\text{Titolo='matematica'}}$ CORSO))

- equivalente a:

$\Pi_{\text{Nome}}\ \sigma_{\text{Voto=30} \wedge \text{Titolo='matematica'}}$
(STUDENTE $\triangleright\triangleleft$ ESAME $\triangleright\triangleleft$ CORSO)

34

Selezione, proiezione e join

- quali professori hanno esaminato Antonio?

Matr	Nome	Città	CDip	Cod-Corso	Data	Voto	Titolo	Docente
123	Carlo	Bologna	Inf	1	7-9-97	30	matem	barozzi
123	Carlo	Bologna	Inf	2	8-1-98	28	infor	meo
702	Antonio	Roma	Log	3	7-9-97	20	infor	meo

35

Selezione, proiezione e join

- quali professori hanno esaminato Antonio?

$\Pi_{\text{Docente}}\ \sigma_{\text{Nome = 'Antonio'}}$
(STUDENTE $\triangleright\triangleleft$ ESAME $\triangleright\triangleleft$ CORSO)

Docente
Meo

36

Equivalenza di espressioni

STUDENTE

Matr	Nome	Città	CDip
123	Carlo	Bologna	Inf
415	Paola	Torino	Inf
702	Antonio	Roma	Log

ESAME

Matr	Cod-Corso	Data	Voto
123	1	7-9-97	30
123	2	8-1-98	28
702	2	7-9-97	20

CORSO

Cod-Corso	Titolo	Docente
1	matematica	Barozzi
2	informatica	Meo

Equivalenza di espressioni

- quali professori hanno esaminato Antonio?

$\Pi_{Docente}$ (CORSO $\triangleright\triangleleft$ (ESAME $\triangleright\triangleleft$ $\sigma_{Nome = 'Antonio'}$ STUDENTE))

- equivalente a:

$\Pi_{Docente}\ \sigma_{Nome = 'Antonio'}$ (STUDENTE $\triangleright\triangleleft$ ESAME $\triangleright\triangleleft$ CORSO)

Espressioni con unione e differenza

- estrarre la matricola degli studenti romani oppure degli studenti che hanno sostenuto un esame il giorno 8-1-01

$(\Pi_{Matr}\ \sigma_{Città='Roma'}$ STUDENTE)
$\cup$
$(\Pi_{Matr}\ \sigma_{Data=8\text{-}1\text{-}01}$ ESAME)

Matr
702

$\cup$

Matr
123

$\longrightarrow$

Matr
702
123

Espressioni con unione e differenza

- estrarre la matricola degli studenti che hanno preso almeno un voto superiore a 28 e non sono mai scesi sotto il 25

$(\Pi_{Matr}\ \sigma_{Voto>=28}$ ESAME)
-
$(\Pi_{Matr}\ \sigma_{Voto<25}$ ESAME)

Matr
123

-

Matr
702

$\longrightarrow$

Matr
123

Espressioni complesse

- estrarre il nome degli studenti che non hanno mai preso meno di 28

Π_{Nome} STUDENTE $\triangleright\triangleleft$
(Π_{Matr} ESAME
-
$\Pi_{Matr}\ \sigma_{Voto<28}$ ESAME)

- spiegazione: prima trovo le matricole di tutti gli studenti meno le matricole di coloro che hanno preso meno di 28, poi trovo i loro nomi.

Espressioni complesse

- estrarre il nome degli studenti che hanno sostenuto "informatica" e "matematica" lo stesso giorno

Π_{Nome} STUDENTE $\triangleright\triangleleft$
((ESAME $\triangleright\!<_{Titolo='informatica'}$ CORSO)
$\triangleright\!<_{Matr = Matr \wedge Data = Data}$
(ESAME $\triangleright\!<\ \sigma_{Titolo='matematica'}$ CORSO))

- spiegazione: prima trovo le matricole di coloro che hanno dato i due esami nello stesso giorno, poi trovo i loro nomi.

Espressioni complesse

• estrarre l'ultimo esame di ciascuno studente

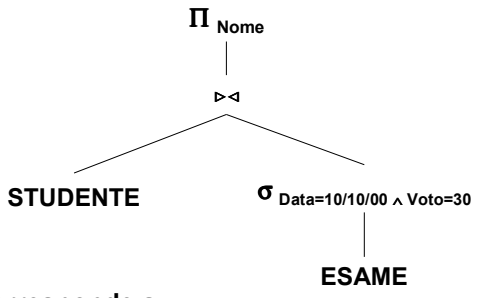
ESAME
-
(ESAME $\rtimes_{Matr = Matr \wedge Data < Data}$ ESAME)

• spiegazione: prima trovo gli esami che non sono ultimi, cioè che sono seguiti da qualche esame a pari studente in data superiore, poi sottraggo da tutti gli esami questi "esami non ultimi" e trovo gli ultimi esami di ciascuno studente

Rappresentazione delle espressioni tramite alberi

• Ogni espressione dell'algebra relazionale può essere rappresentata in modo grafico da un albero, che esplicita l'ordine di valutazione degli operatori
• Ogni operatore corrisponde a un nodo
  • operatori unari con un ramo in ingresso e uno in uscita
  • operatori binari con due rami in ingresso e uno in uscita

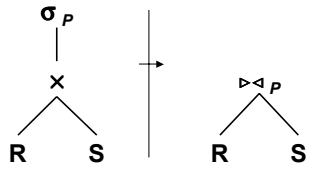
Esempio di albero



Corresponde a:
Π_{Nome} STUDENTE $\bowtie \sigma_{Data=10/10/00 \wedge Voto=30}$ ESAME

Trasformazioni di equivalenza per espressioni algebriche

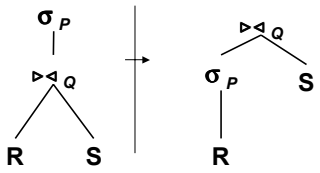
1) Eliminazione dei prodotti cartesiani



Vale se *p* è una congiunzione di predicati del tipo ATTR *comp* ATTR

Trasformazioni di equivalenza per espressioni algebriche

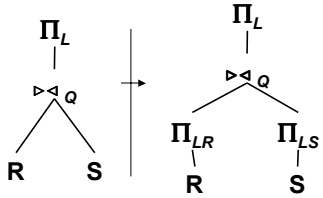
2) Push della selezione rispetto al join



Vale se *p* è un predicato che si applica ai soli attributi di R

Trasformazioni di equivalenza per espressioni algebriche

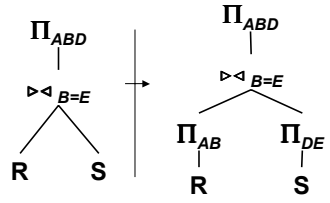
3) Push della proiezione rispetto al join



JR e JS sono gli attributi DI R e S necessari a valutare *Q*
LR = *L* - schema(S) + *JR*
LS = *L* - schema(R) + *JS*

Esempio

3) Push della proiezione rispetto al join

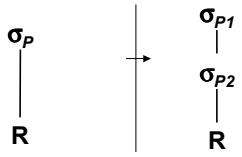


R(A,B,C)
S(D,E,F,G)

L=ABD
JR=B
JS=E
LR=AB
LS=DE

Trasformazioni di equivalenza per espressioni algebriche

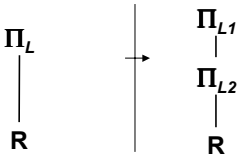
4) Idempotenza della selezione



$P = P1 \wedge P2$

Trasformazioni di equivalenza per espressioni algebriche

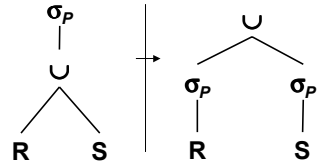
5) Idempotenza della proiezione



$L1 = L$
$L2 \supseteq L$

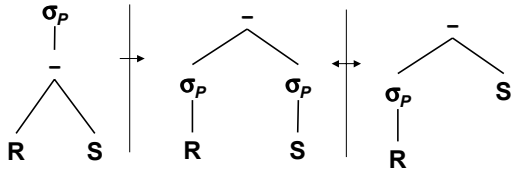
Trasformazioni di equivalenza per espressioni algebriche

6) Push della selezione rispetto all'unione



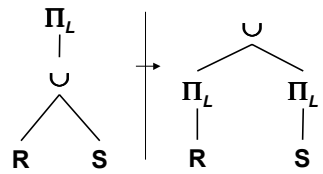
Trasformazioni di equivalenza per espressioni algebriche

7) Push della selezione rispetto alla differenza



Trasformazioni di equivalenza per espressioni algebriche

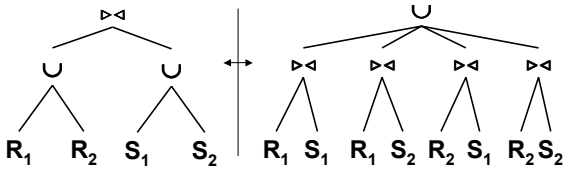
8) Push della proiezione rispetto all'unione



***Attenzione*: non vale il push della proiezione rispetto alla differenza e all'intersezione**

Trasformazioni di equivalenza per espressioni algebriche

9) Commutazione di join e unione



Formule utili

$R \bowtie R = R$
$R \cup R = R$
$R - R = \emptyset$

$R \bowtie \sigma_P R = \sigma_P R$
$R \cup \sigma_P R = R$
$R - \sigma_P R = \sigma_{\neg P} R$

$\sigma_{P1} R \bowtie \sigma_{P2} R = \sigma_{P1 \wedge P2} R$
$\sigma_{P1} R \cup \sigma_{P2} R = \sigma_{P1 \vee P2} R$
$\sigma_{P1} R - \sigma_{P2} R = \sigma_{P1 \wedge \neg P2} R$

$\sigma_P \emptyset = \emptyset$
$\Pi_L \emptyset = \emptyset$

$R \cup \emptyset = R$
$R - \emptyset = R$
$\emptyset - R = \emptyset$
$R \cap \emptyset = \emptyset$

$R \times \emptyset = \emptyset$
$R \bowtie \emptyset = \emptyset$

Ottimizzazione algebrica

- **Tra tutte le rappresentazioni equivalenti, conviene scegliere quella meno costosa da eseguire**
- **Criterio informale: minimizzare la dimensione dei risultati intermedi**
- **Tecnica:**
  - **utilizzare dove possibile le trasformazioni di push (2, 3, 6, 7, 8);**
  - **usare le trasformazioni di idempotenza (4, 5) per generare nuove selezioni e proiezioni**

Esempio di ottimizzazione

Si ha lo schema:

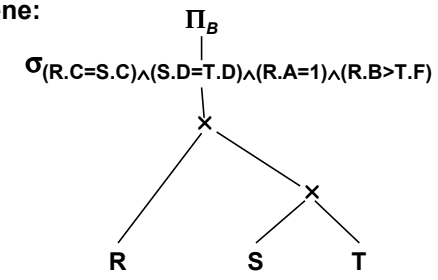
R(A,B,C)
S(C,D,E)
T(D,E,F,G)

Si deve ottimizzare l'espressione algebrica:

$\Pi_B \sigma_{(R.C=S.C) \wedge (S.D=T.D) \wedge (R.A=1) \wedge (R.B>T.F)} R \times S \times T$

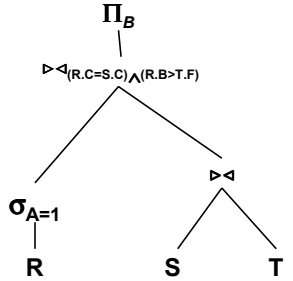
Esempio di ottimizzazione

- **Con la rappresentazione ad albero, si ottiene:**



Esempio di ottimizzazione

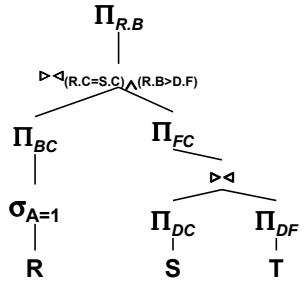
- **Applicando le trasformazioni, si ottiene:**



Omettiamo (S.D=T.D)

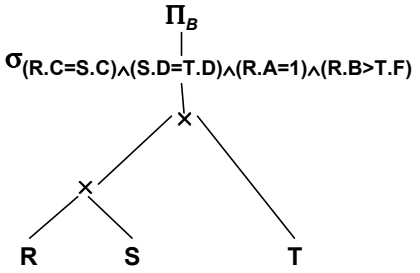
Esempio di ottimizzazione

• **Aggiungendo le proiezioni, si ottiene:**



Esercizio

• **Ripetere con:**



Esercizi di ottimizzazione

1) Si ha lo schema: R(A,B)
S(A,B)

Ottimizzare l'espressione algebrica:

$\sigma_{(S.A=R.A)\wedge(R.A>2)\wedge(S.A=1)}\ R \times S$

Il risultato è una relazione vuota, perchè il predicato è una CONTRADDIZIONE!!!!

Esercizi di ottimizzazione

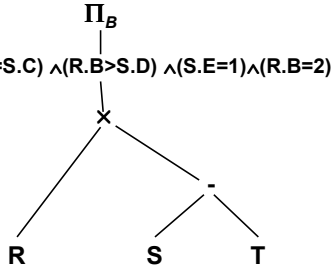
2) Si ha lo schema: R(A,B,C)
S(C,D,E)
T(C,D,E)

Ottimizzare l'espressione algebrica:

$\Pi_{AD}\sigma_{(R.C=S.C)\wedge(S.E=1)\wedge(R.B=2)\wedge(R.B>S.D)}\ (R \times (S - T))$

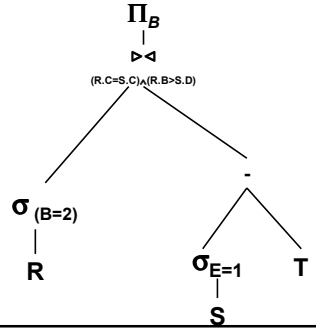
Esempio di ottimizzazione

• **Con la rappresentazione ad albero, si ottiene:**



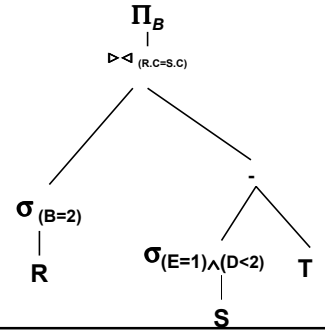
Esempio di ottimizzazione

• **Applicando le trasformazioni, si ottiene:**



Esempio di ottimizzazione

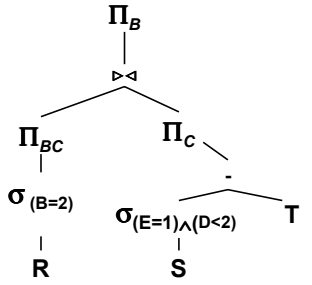
• Ragionando sui predicati, si ottiene:



67

Esempio di ottimizzazione

• Aggiungendo le proiezioni, si ottiene:



68

Ricerca della minima espressione contenitore (contain-view)

• Date n query, è un'espressione che consente di estrarre i risultati delle *n* query tramite selezioni e proiezioni su di essa

• Esempio: R(A,B,C) S(B,C,D,E,F) T(D,G)

1) $\Pi_{R.C,G}$

$\sigma_{(A=1)\wedge(R.B=S.B)\wedge(S.D=T.D)\wedge(E=1)\wedge(G>3)}$

$R \times S \times T$

2) $\Pi_{R.C,F}$

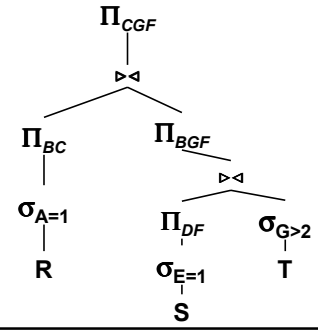
$\sigma_{(R.C>G)\wedge(A=1)\wedge(E=1)\wedge(F>3)\wedge(G>2)\wedge(S.D=T.D)\wedge(R.B=S.B)}$

$R \times S \times T$

69

Estrazione della contain-view

• La massima sub espressione comune (MSE):



70

Calcolo delle query tramite la MSE

1: Π_{CG} | $\sigma_{G>3}$ | MSE

2: Π_{CF} | $\sigma_{C>G \wedge F>3}$ | MSE

71

Calcolo relazionale

- Una famiglia di linguaggi formali
- Due tipi principali
  - Calcolo delle tuple (TRC, Tuple Relational Calculus)
  - Calcolo dei domini (DRC, Domain Relational Calculus)
- TRC in due versioni:
  - Con tuple ristrette sul range
  - Con tuple arbitrarie
- Guarderemo il TRC con tuple arbitrarie (anche se non è descritto nel libro!)

72

TRC è dichiarativo

- Esprime cosa si vuole nel risultato ma non come ottenerlo
- E' diverso dall'algebra, che è procedurale
- La dichiaratività è una caratteristica tipica dei linguaggi relazionali

73

Definizione formale del TRC

- **Forma standard: { *t* | *p(t)* }**
- ***p(t)* è una *formula*, costruita tramite *atomi***
- **Atomi**
  - ***t* ∈ R**
  - ***t1* [A1] *comp t2* [A2]**
  - ***t* [A] *comp k***
- ***comp* e' un comparatore: =, <>, >, >=, <, <=**
- ***k* è una costante**
- ***t[A]* è una restrizione sull'attributo A della tupla *t***

74

Definizione formale del TRC

- **Regole di costruzione delle formule**
  - **un atomo è una formula**
  - **se *p* è una formula, lo sono anche ¬*p* e *(p)***
  - **se *p1* e *p2* sono formule, lo sono anche *p1* ∧ *p2*, *p1* ∨ *p2*, *p1* ⇒ *p2***
  - **se *p* è una formula in cui *s* è una variabile, lo sono anche ∃ *s* ∈ R *(p(s))*, ∀ *s* ∈ R *(p(s))***

75

Proprietà del TRC

- **Legge di De Morgan**
  ***p1* ∧ *p2* ≡ ¬ (¬ *p1* ∨ ¬ *p2*)**
- **Corrispondenze tra quantificatori**
  ∀ *t* ∈ R *(p(t))* ≡ ¬ ∃ *t* ∈ R *(¬ p(t))*
- ***Definizione di implicazione***
  p1* ⇒ *p2* ≡ ¬ *p1* ∨ *p2

76

Forme Normali

- ***Dalle tre leggi segue che è possibile scrivere tutte le possibili espressioni senza implicazione e con:***
  - ***Un solo quantificatore***
  - ***Un solo operatore binario***
- ***La forma normale più usata (simile ad SQL) usa quantificatore esistenziale e congiunzione***

77

Esempi di TRC

- **Nome degli studenti che hanno preso 30 in "matematica"**

{ *t* | ∃ *t1* ∈ STUDENTE,
∃ *t2* ∈ ESAME,
∃ *t3* ∈ CORSO
(*t*[Nome]=*t1*[Nome] ∧
***t1*[Matr]=*t2*[Matr] ∧**
***t2*[CodCorso]=*t3*[CodCorso] ∧**
***t2*[Voto]=30 ∧**
***t3*[Titolo]='matematica') }**

78

Esempi di TRC

• Matricole degli studenti che hanno sostenuto "matematica" ma non "basi di dati"

$\{ t \mid \exists t1 \in \text{ESAME}, \exists t2 \in \text{CORSO}$
$(t[\text{Matr}]=t1[\text{Matr}] \wedge$
$t1[\text{CodCorso}]=t2[\text{CodCorso}] \wedge$
$t2[\text{Titolo}]=\text{'matematica'}) \wedge$
$\neg (\exists t3 \in \text{ESAME}, \exists t4 \in \text{CORSO}$
$(t[\text{Matr}]=t3[\text{Matr}] \wedge$
$t3[\text{CodCorso}]=t4[\text{CodCorso}] \wedge$
$t4[\text{Titolo}]=\text{'basi di dati'}))) \}$

Correttezza

- Si devono evitare formule *unsafe*:
  $\{ t \mid t \notin R \}$ dà un risultato infinito
- Si considerano corrette solo formule *indipendenti dal dominio*
  - la soluzione non dipende dal dominio degli attributi, ma solo dall'istanza del DB

AR è esprimibile tramite TRC

E' sufficiente mostrare che si possono realizzare i cinque operatori fondamentali:

• Selezione, $\sigma_{A=1} R$:
$\{ t \mid \exists t \in R (t[A]=1) \}$
• Proiezione, $\Pi_{AC} R$:
$\{ t \mid \exists t1 \in R (t[A,C]=t1[A,C]) \}$

AR è esprimibile tramite TRC

- Prodotto cartesiano, $R(A,B,C) \times S(D,E,F)$:
$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t[A,B,C]=t1[A,B,C] \wedge$
$t[D,E,F]=t2[D,E,F]) \}$

Esempio di join, $R(A,C) \bowtie_{A=B} S(B,D)$:
$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t[A,C] = t1[A,C] \wedge$
$t[B,D] = t2[B,D] \wedge$
$t[A] = t[B]) \}$

AR è esprimibile tramite TRC

• Unione, $R \cup S$:
$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t = t1 \vee t = t2) \}$
• Differenza, R – S:
$\{ t \mid \exists t \in R (t \notin S) \}$

Anche TRC è esprimibile tramite AR

• La prova è più complicata
• Si devono escludere espressioni unsafe e dipendenti dal dominio
• Sotto queste ipotesi TRC e AR hanno lo stesso potere espressivo

Linguaggi di query basati sulla programmazione logica

• La programmazione logica è un paradigma di programmazione basato su regole
• Linguaggio di riferimento: Prolog (1970)
• Datalog: Prolog per basi di dati (1984)
• Differenze principali rispetto a Prolog (per chi lo conosce):
  • mancano i simboli di funzione
  • modello di valutazione pienamente dichiarativo

85

Base di dati d'esempio per Datalog

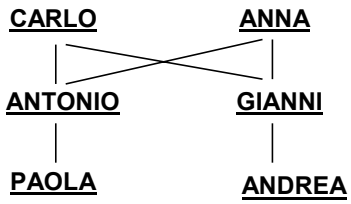
GENITORE

Genitore	Figlio
Carlo	Antonio
Carlo	Gianni
Anna	Antonio
Anna	Gianni
Gianni	Andrea
Antonio	Paola

PERSONA

Nome	Età	Sesso
Carlo	65	M
Antonio	40	M
Anna	60	F
Gianni	43	M
Andrea	22	M
Paola	20	F

86



87

Regole Datalog

• Ogni regola è composta da una testa (head o LHS) e da un corpo (body o RHS)
  ES: P :- P1, P2, … Pn

• Ogni P rappresenta un predicato (chiamato letterale), così composto:
  • nome
  • argomenti:
    • costanti
    • variabili
    • simbolo "don't care" (non nella testa)
  ES: persona(X,_,"M")

88

Regole Datalog

Hanno l'interpretazione che la testa è vera se il corpo è vero

• Esempi di regole

Padre(*X*,*Y*) :- *Persona*(*X*,_,'M'), *Genitore*(*X*,*Y*)
Madre(*X*,*Y*) :- *Persona*(*X*,_,'F'), *Genitore*(*X*,*Y*)

•Le variabili del LHS devono apparire nel RHS

89

Fatti Datalog

• Ciascuna tupla corrisponde a un fatto (letterale *ground*)
• Ad esempio:

Genitore("Carlo","Antonio").

90

Query in Datalog

- Esempio di query (detta *goal*)
  ?- *Padre*("Carlo",*X*)
- Valutazione: si cerca una regola che abbia per testa il predicato *Padre* e si cercano valori per la variabile *X (unificazione)*
- Si ottiene *X* = {"Antonio" "Gianni"}
- Un goal senza variabili restituisce *True* o *False*
  - ?- *Padre*("Carlo","Antonio") ⇒ *True*
  - ?- *Padre*("Carlo","Andrea") ⇒ *False*

91

Altre regole

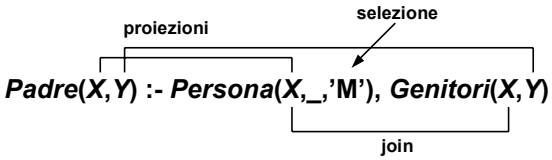
Nonno(*X*,*Z*) :- *Padre*(*X*,*Y*), *Genitore*(*Y*,*Z*)

Fratello(*X*,*Y*) :- *Genitore*(*Z*,*X*), *Genitore*(*Z*,*Y*), $X \neq Y$

Zio(*X*,*Y*) :- *Persona*(*X*,_,'M'), *Fratello*(*X*,*Z*),
Genitore(*Z*,*Y*)

92

Corrispondenza tra Datalog e l'algebra relazionale



- Espressione corrispondente in algebra relazionale:

PADRE = $\Pi_{1,5}\ \sigma_{3 = \text{'M'}}$ PERSONA $\triangleright\!\triangleleft_{1 = 1}$ GENITORI

93

Database estensionale e intensionale

- Database estensionale (EDB)
  - insieme delle tabelle presenti nel DB
- Database intensionale (IDB)
  - insieme dei predicati che sono a sinistra in una regola
  - è la conoscenza dedotta a partire da EDB
- Normalmente si impone EDB ∩ IDB = ∅

94

Negazione in Datalog

- Alcuni letterali del corpo possono essere negati
  es: Padre(*X*,*Y*) :- *Genitore*(*X*,*Y*), not Madre(X,Y)
- L'uso della negazione in Datalog aumenta il potere espressivo del linguaggio
- Anche l'uso della negazione richiede cautela:
  q(*X*) :- ¬ *p*(*X*)
  p(0).
  ?- *q*(*X*) produce un risultato infinito!

95

Potere espressivo di Datalog

- Datalog senza negazione permette di rappresentare gli operatori {σ, Π, ×, ∪}
- {σ, Π, ×} già visti
- Per l'<u>unione</u> si usano più regole con la stessa testa; P = R ∪ S:
  P(*X*,*Y*) :- *R*(*X*,*Y*)
  P(*X*,*Y*) :- *S*(*X*,*Y*)
- Per la <u>differenza</u> serve il *not*; P = R – S:
  P(*X*,*Y*) :- *R*(*X*,*Y*), ¬ *S*(*X*,*Y*)

96

Query ricorsive

• Datalog con negazione ha quindi un potere espressivo almeno pari all'algebra relazionale
• In effetti ha un potere superiore, perché permette l'espressione di query ricorsive
• Una query ricorsiva presenta il letterale della testa all'interno del corpo della regola:

Antenato(*X*,*Y*) :- *Genitore*(*X*,*Y*) ← start
Antenato(*X*,*Y*) :- *Antenato*(*X*,*Z*), *Genitore*(*Z*,*Y*)
← ricorsione

Meccanismo di valutazione

• Si può immaginare che venga seguito il seguente processo iterativo:

$ANTENATO^0 \Leftarrow GENITORI$
$ANTENATO^1 \Leftarrow (\Pi_{1,4} ANTENATO^0 \rhd\lhd_{2=3} GENITORI) \cup ANTENATO^0$
$ANTENATO^2 \Leftarrow (\Pi_{1,4} ANTENATO^1 \rhd\lhd_{2=3} GENITORI) \cup ANTENATO^1$
...

fino a che $ANTENATO^n$ risulta pari a $ANTENATO^{n-1}$ (punto fisso)

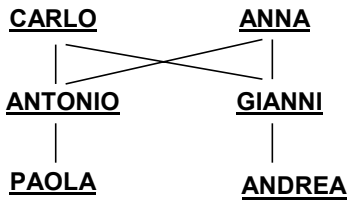
Valutazione delle query ricorsive

• Nella base dati d'esempio si ottiene il risultato illustrato a fianco

ANTENATO

Carlo	Antonio
Carlo	Gianni
Anna	Antonio
Anna	Gianni
Gianni	Andrea
Antonio	Paola
Carlo	Andrea
Carlo	Paola
Anna	Andrea
Anna	Paola

Nuovi elementi rispetto a GENITORE

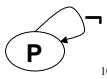


Predicati e funzioni

• Nella definizione delle regole si possono usare predicati speciali
  • operatori di confronto
    • = (unificazione), ≠, <, ≤, >, ≥
  • funzioni aritmetiche
    • +, –, *, +
• Bisogna fare molta attenzione:
  n(*X*) :- *n*(*X* – 1)
  n(0) .
  ?- *n*(*X*) produce un risultato infinito!

Regole corrette

•La negazione deve essere *safe*
  • tutte le variabili di un letterale negato devono comparire in un letterale positivo del corpo della regola
  S(*X*) :- R(X), ¬ *Q*(*Y*) non va bene
• La negazione deve essere *stratificata*
  • sinteticamente, non ci devono essere cicli di dipendenza tra letterali negati
  P(*X*) :- *R*(*X*)
  P(*X*) :- *R*(*X*), ¬ *P*(*X*) non va bene



Riepilogo della terminologia Datalog

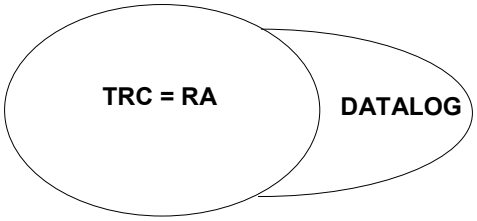
Modello relazionale standard	Datalog
Relazione	Predicato
Attributo	Argomento
Tupla	Fatto
Query	Regole
Interrogazione	Goal

103

Sintesi sui poteri espressivi

- TRC =RA, con TRC safe e non dipendente dal dominio
- Datalog più espressivo di TRC e RA (a causa della ricorsione), con espressioni Datalog safe, stratificate e che non producono infiniti simboli-

104



105

In che modo si differenziano i linguaggi formali?

- L'algebra è un linguaggio PROCEDURALE: si dice esattamente in che modo valutare le interrogazioni (e si può ottimizzare)
- TRC è un linguaggio DICHIARATIVO: si dice cosa si vuole ottenere ma non come ottenerlo.
- DATALOG è pure dichiarativo, ma la costruzione delle dipendenze tra le regole può essere fatta in modo procedurale.

106

Perchè vi abbiamo raccontato tutto ciò?

- SQL assomiglia al calcolo relazionale
- L'algebra assomiglia a quanto viene effettivamente valutato da un DBMS
- Datalog avvicina le basi di dati al mondo della logica e della deduzioni

107