Atzeni, Ceri, Paraboschi, Torlone

Basi di dati

McGraw-Hill, 1996-2002

Capitolo 3:

ALGEBRA E CALCOLO RELAZIONALE

07/06/2005

Linguaggi per basi di dati

- operazioni sullo schema
  - DDL: data definition language
- operazioni sui dati
  - DML: data manipulation language
    - interrogazione ("query")
    - aggiornamento

Linguaggi di interrogazione per basi di dati relazionali

- Dichiarativi
  - specificano le proprietà del risultato ("che cosa")
- Procedurali
  - specificano le modalità di generazione del risultato ("come")

Linguaggi di interrogazione

- Algebra relazionale: procedurale
- Calcolo relazionale:
  dichiarativo (teorico)
  SQL (Structured Query Language):
  parzialmente dichiarativo (reale)
- QBE (Query by Example):
  dichiarativo (reale)

Algebra relazionale

- Insieme di operatori
  - su relazioni
  - che producono relazioni
  - e possono essere composti

Operatori dell'algebra relazionale

- unione, intersezione, differenza
- ridenominazione
- selezione
- proiezione
- join (join naturale, prodotto cartesiano, theta-join)

Operatori insiemistici

- le relazioni sono insiemi
- i risultati debbono essere relazioni
- è possibile applicare unione, intersezione, differenza solo a relazioni definite sugli stessi attributi

Unione

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Quadri

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati ∪ Quadri

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45
9297	Neri	33

Intersezione

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Quadri

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati ∩ Quadri

Matricola	Nome	Età
7432	Neri	54
9824	Verdi	45

Differenza

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Quadri

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati – Quadri

Matricola	Nome	Età
7274	Rossi	42

Un’unione sensata ma impossibile

Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

Paternità ∪ Maternità

??

Ridenominazione

- operatore monadico (con un argomento)
- "modifica lo schema" lasciando inalterata l'istanza dell'operando

Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$REN_{Genitore \leftarrow Padre}$ **(Paternità)**

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Paternità

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$REN_{Genitore \leftarrow Padre}$ **(Paternità)**

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Maternità

Madre	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

$REN_{Genitore \leftarrow Madre}$ **(Maternità)**

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

$REN_{Genitore \leftarrow Padre}$ **(Paternità)**

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

$REN_{Genitore \leftarrow Madre}$ **(Maternità)**

Genitore	Figlio
Eva	Abele
Eva	Set
Sara	Isacco

$REN_{Genitore \leftarrow Padre}$ **(Paternità)** $\cup$ $REN_{Genitore \leftarrow Madre}$ **(Maternità)**

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco
Eva	Abele
Eva	Set
Sara	Isacco

Impiegati

Cognome	Ufficio	Stipendio
Rossi	Roma	55
Neri	Milano	64

Operai

Cognome	Fabbrica	Salario
Bruni	Monza	45
Verdi	Latina	55

$REN_{Sede, Retribuzione \leftarrow Ufficio, Stipendio}$ **(Impiegati)** $\cup$

$REN_{Sede, Retribuzione \leftarrow Fabbrica, Salario}$ **(Operai)**

Cognome	Sede	Retribuzione
Rossi	Roma	55
Neri	Milano	64
Bruni	Monza	45
Verdi	Latina	55

Selezione

- operatore monadico
- produce un risultato che
  - ha lo stesso schema dell'operando
  - contiene un sottoinsieme delle ennuple dell'operando,
  - quelle che soddisfano una condizione

Impiegati

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
9553	Milano	Milano	44
5698	Neri	Napoli	64

- impiegati che
  - guadagnano più di 50
  - guadagnano più di 50 e lavorano a Milano
  - hanno lo stesso nome della filiale presso cui lavorano

Selezione, sintassi e semantica

- sintassi
  SEL *Condizione* (*Operando*)
  - *Condizione*: espressione booleana (come quelle dei vincoli di ennupla)
- semantica
  - il risultato contiene le ennuple dell'operando che soddisfano la condizione

- impiegati che guadagnano più di 50

$SEL_{Stipendio > 50}$ (Impiegati)

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
5698	Neri	Napoli	64

$SEL_{Stipendio > 50}$ (Impiegati)

- impiegati che guadagnano più di 50 e lavorano a Milano

$SEL_{Stipendio > 50 \text{ AND } Filiale = 'Milano'}$ (Impiegati)

Matricola	Cognome	Filiale	Stipendio
5998	Neri	Milano	64

$SEL_{Stipendio > 50 \text{ AND } Filiale = 'Milano'}$ (Impiegati)

- impiegati che hanno lo stesso nome della filiale presso cui lavorano

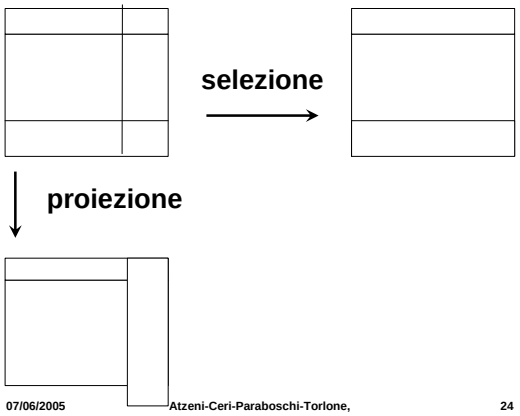
$SEL_{Cognome = Filiale}$(Impiegati)

Matricola	Cognome	Filiale	Stipendio
9553	Milano	Milano	44

$SEL_{Cognome = Filiale}$(Impiegati)

Selezione e proiezione

- operatori "ortogonali"
- selezione:
  - decomposizione orizzontale
- proiezione:
  - decomposizione verticale



Proiezione

- operatore monadico
- produce un risultato che
  - ha parte degli attributi dell'operando
  - contiene ennuple cui contribuiscono tutte le ennuple dell'operando

Impiegati

Matricola	Cognome	Filiale	Stipendio
7309	Neri	Napoli	55
5998	Neri	Milano	64
9553	Rossi	Roma	44
5698	Rossi	Roma	64

- per tutti gli impiegati:
  - matricola e cognome
  - cognome e filiale

Proiezione, sintassi e semantica

- sintassi

  PROJ $_{\text{ListaAttributi}}$ **(Operando)**
- semantica
  - il risultato contiene le ennuple ottenute da tutte le ennuple dell'operando ristrette agli attributi nella lista

- matricola e cognome di tutti gli impiegati

Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

PROJ $_{\text{Matricola, Cognome}}$ **(Impiegati)**

- cognome e filiale di tutti gli impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

PROJ $_{\text{Cognome, Filiale}}$ **(Impiegati)**

Cardinalità delle proiezioni

- una proiezione
  - contiene al più tante ennuple quante l'operando
  - può contenerne di meno
- se X è una superchiave di R, allora $\text{PROJ}_X(R)$ contiene esattamente tante ennuple quante R

Selezione e proiezione

- **Combinando selezione e proiezione, possiamo estrarre interessanti informazioni da una relazione**

- **matricola e cognome degli impiegati che guadagnano più di 50**

Matricola	Cognome		
7309	Rossi		
5998	Neri		
5698	Neri		

$$PROJ_{Matricola,Cognome}\ (SEL_{Stipendio > 50}\ (Impiegati))$$

- **Combinando selezione e proiezione, possiamo estrarre informazioni da una relazione**
- **non possiamo però correlare informazioni presenti in relazioni diverse, né informazioni in ennuple diverse di una stessa relazione**

Join

- **il join è l'operatore più interessante dell'algebra relazionale**
- **permette di correlare dati in relazioni diverse**

Prove scritte in un concorso pubblico

- **I compiti sono anonimi e ad ognuno è associata una busta chiusa con il nome del candidato**
- **Ciascun compito e la relativa busta vengono contrassegnati con uno stesso numero**

1	25
2	13
3	27
4	28

1	Mario Rossi
2	Nicola Russo
3	Mario Bianchi
4	Remo Neri

Mario Rossi	25
Nicola Russo	13
Mario Bianchi	27
Remo Neri	28

Numero	Voto
1	25
2	13
3	27
4	28

Numero	Candidato
1	Mario Rossi
2	Nicola Russo
3	Mario Bianchi
4	Remo Neri

Numero	Candidato	Voto
1	Mario Rossi	25
2	Nicola Russo	13
3	Mario Bianchi	27
4	Remo Neri	28

Join naturale

- operatore binario (generalizzabile)
- produce un risultato
  - sull'unione degli attributi degli operandi
  - con ennuple costruite ciascuna a partire da una ennupla di ognuno degli operandi

Join, sintassi e semantica

- $R_1(X_1)$, $R_2(X_2)$
- R_1 JOIN R_2 è una relazione su X_1X_2

$$\{ t \text{ su } X_1X_2 \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
A	Mori
B	Bruni

Impiegato	Reparto	Capo
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni

- ogni ennupla contribuisce al risultato:
  - join completo

Un join non completo

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Un join vuoto

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
D	Mori
C	Bruni

Impiegato	Reparto	Capo

Un join completo, con n x m ennuple

Impiegato	Reparto
Rossi	B
Neri	B

Reparto	Capo
B	Mori
B	Bruni

Impiegato	Reparto	Capo
Rossi	B	Mori
Rossi	B	Bruni
Neri	B	Mori
Neri	B	Bruni

Cardinalità del join

- Il join di R_1 e R_2 contiene un numero di ennuple compreso fra zero e il prodotto di $|R_1|$ e $|R_2|$
- se il join coinvolge una chiave di R_2, allora il numero di ennuple è compreso fra zero e $|R_1|$
- se il join coinvolge una chiave di R_2 e un vincolo di integrità referenziale, allora il numero di ennuple è pari a $|R_1|$

Cardinalità del join, 2

- $R_1(A,B)$, $R_2(B,C)$
- in generale

$$0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1| \times |R_2|$$

- se B è chiave in R_2

$$0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1|$$

- se B è chiave in R_2 ed esiste vincolo di integrità referenziale fra B (in R_1) e R_2:

$$|R_1 \text{ JOIN } R_2| = |R_1|$$

Join, una difficoltà

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

- alcune ennuple non contribuiscono al risultato: vengono "tagliate fuori"

Join esterno

- Il join esterno estende, con valori nulli, le ennuple che verrebbero tagliate fuori da un join (interno)
- esiste in tre versioni:
  - sinistro, destro, completo

Join esterno

- sinistro: mantiene tutte le ennuple del primo operando, estendendole con valori nulli, se necessario
- destro: ... del secondo operando ...
- completo: … di entrambi gli operandi ...

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $JOIN_{LEFT}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $JOIN_{RIGHT}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $JOIN_{FULL}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL
NULL	C	Bruni

Join e proiezioni

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Impiegato	Reparto
Neri	B
Bianchi	B

Reparto	Capo
B	Mori

Proiezioni e join

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

Impiegato	Reparto
Neri	B
Bianchi	B
Verdi	A

Reparto	Capo
B	Mori
B	Bruni
A	Bini

Impiegato	Reparto	Capo
Neri	B	Mori
Neri	B	Bruni
Bianchi	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

Join e proiezioni

- $R_1(X_1), R_2(X_2)$

$$PROJ_{X_1}(R_1 \text{ JOIN } R_2) \subseteq R_1$$

- $R(X)$, $X = X_1 \cup X_2$

$$(PROJ_{X_1}(R)) \text{ JOIN } (PROJ_{X_2}(R)) \supseteq R$$

Prodotto cartesiano

- un join naturale su relazioni senza attributi in comune
- contiene sempre un numero di ennuple pari al prodotto delle cardinalità degli operandi (le ennuple sono tutte combinabili)

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Codice	Capo
A	Mori
B	Bruni

Impiegati JOIN Reparti

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

- Il prodotto cartesiano, in pratica, ha senso (quasi) solo se seguito da selezione:

$$SEL_{Condizione}\ (R_1\ JOIN\ R_2)$$

- L'operazione viene chiamata theta-join e indicata con

$$R_1\ JOIN_{Condizione}\ R_2$$

Perché "theta-join"?

- La condizione C è spesso una congiunzione (AND) di atomi di confronto $A_1 \vartheta A_2$ dove ϑ è uno degli operatori di confronto (=, >, <, …)

Equi-join

- Se l'operatore di confronto nel theta-join è sempre l'uguaglianza (=) allora si parla di equi-join

Nota: ci interessa davvero l'equi-join, non il theta-join più generale

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Codice	Capo
A	Mori
B	Bruni

Impiegati $JOIN_{Reparto=Codice}$ Reparti

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B	B	Bruni

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
A	Mori
B	Bruni

Impiegati JOIN Reparti

Join naturale ed equi-join

Impiegati

Impiegato	Reparto

Reparti

Reparto	Capo

Impiegati JOIN Reparti

$$\text{PROJ}_{\text{Impiegato,Reparto,Capo}} (\text{SEL}_{\text{Reparto=Codice}} (\text{Impiegati JOIN REN}_{\text{Codice} \leftarrow \text{Reparto}} (\text{Reparti})))$$

Esempi

Impiegati

Matricola	Nome	Età	Stipendio
7309	Rossi	34	45
5998	Bianchi	37	38
9553	Neri	42	35
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

Supervisione

Impiegato	Capo
7309	5698
5998	5698
9553	4076
5698	4076
4076	8123

- **Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni**

$$\text{SEL}_{\text{Stipendio>40}}(\text{Impiegati})$$

Matricola	Nome	Età	Stipendio
7309	Rossi	34	45
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

$$\text{SEL}_{\text{Stipendio>40}}(\text{Impiegati})$$

- **Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni**

$$\text{PROJ}_{\text{Matricola, Nome, Età}} (\text{SEL}_{\text{Stipendio>40}}(\text{Impiegati}))$$

Matricola	Nome	Età	
7309	Rossi	34	
5698	Bruni	43	
4076	Mori	45	
8123	Lupi	46	

$$PROJ_{Matricola,\ Nome,\ Età}\ (\ SEL_{Stipendio>40}(Impiegati)\ )$$

- **Trovare le matricole dei capi degli impiegati che guadagnano più di 40 milioni**

$$PROJ_{Capo}\ (Supervisione\ JOIN_{Impiegato=Matricola}\ (SEL_{Stipendio>40}(Impiegati)))$$

- **Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 milioni**

$$PROJ_{Nome,Stipendio}\ (\ Impiegati\ JOIN_{Matricola=Capo}\ PROJ_{Capo}(Supervisione\ JOIN_{Impiegato=Matricola}\ (SEL_{Stipendio>40}(Impiegati))))$$

- **Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo**

$$PROJ_{Matr,Nome,Stip,MatrC,NomeC,StipC}\ (SEL_{Stipendio>StipC}(\ REN_{MatrC,NomeC,StipC,EtàC\ \leftarrow\ Matr,Nome,Stip,Età}(Impiegati)\ JOIN_{MatrC=Capo}\ (Supervisione\ JOIN_{Impiegato=Matricola}\ Impiegati)))$$

- **Trovare le matricole dei capi i cui impiegati guadagnano tutti più di 40 milioni**

$$PROJ_{Capo}\ (Supervisione)\ -\ PROJ_{Capo}\ (Supervisione\ JOIN_{Impiegato=Matricola}\ (SEL_{Stipendio\ \leq\ 40}(Impiegati)))$$

Equivalenza di espressioni

- **Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l'istanza attuale della base di dati**
- **L'equivalenza è importante in pratica perché i DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno "costose"**

Un'equivalenza importante

- **Push selections (se A è attributo di R_2)**

$\text{SEL}_{A=10}(R_1 \text{ JOIN } R_2) = R_1 \text{ JOIN } \text{SEL}_{A=10}(R_2)$

- **Riduce in modo significativo la dimensione del risultato intermedio (e quindi il costo dell'operazione)**

Nota

- **In questo corso, ci preoccupiamo poco dell'efficienza:**
  - **l'obiettivo è di scrivere interrogazioni corrette e leggibili**
- **Motivazione:**
  - **I DBMS si preoccupano di scegliere le strategie realizzative efficienti**

Selezione con valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

$\text{SEL}_{\text{Età} > 40}$ **(Impiegati)**

- **la condizione atomica è vera solo per valori non nulli**

Un risultato non desiderabile

$\text{SEL}_{\text{Età}>30}$ **(Persone)** $\cup$ $\text{SEL}_{\text{Età}\leq 30}$ **(Persone)** $\neq$ **Persone**

- **Perché? Perché le selezioni vengono valutate separatamente!**
- **Ma anche**

$\text{SEL}_{\text{Età}>30 \vee \text{Età}\leq 30}$ **(Persone)** $\neq$ **Persone**

- **Perché? Perché anche le condizioni atomiche vengono valutate separatamente!**

Selezione con valori nulli: soluzione

$\text{SEL}_{\text{Età} > 40}$ **(Impiegati)**

- **la condizione atomica è vera solo per valori non nulli**
- **per riferirsi ai valori nulli esistono forme apposite di condizioni:**

IS NULL

IS NOT NULL

- **si potrebbe usare (ma non serve) una "logica a tre valori" (vero, falso, sconosciuto)**

- **Quindi:**

$\text{SEL}_{\text{Età}>30}$ **(Persone)** $\cup$ $\text{SEL}_{\text{Età}\leq 30}$ **(Persone)** $\cup$ $\text{SEL}_{\text{Età IS NULL}}$ **(Persone)**

=

$\text{SEL}_{\text{Età}>30 \vee \text{Età}\leq 30 \vee \text{Età IS NULL}}$ **(Persone)**

=

Persone

Impiegati

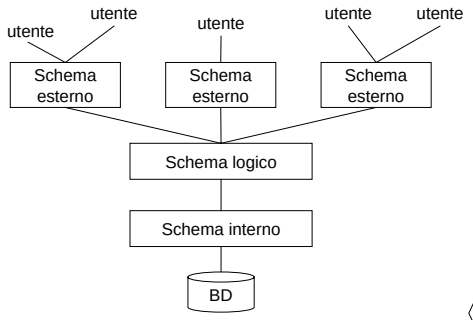
Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

SEL $_{\text{(Età > 40) OR (Età IS NULL)}}$ **(Impiegati)**

Viste (relazioni derivate)

- Rappresentazioni diverse per gli stessi dati (schema esterno)
- Relazioni derivate:
  - relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni)
- Relazioni di base: contenuto autonomo
- Le relazioni derivate possono essere definite su altre derivate, ma …

Architettura standard (ANSI/SPARC) a tre livelli per DBMS



Viste, esempio

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Direzione

Reparto	Capo
A	Mori
B	Bruni

- una vista:

Supervisione =

PROJ $_{\text{Impiegato, Capo}}$ **(Afferenza JOIN Direzione)**

Viste virtuali e materializzate

- Due tipi di relazioni derivate:
  - viste materializzate
  - relazioni virtuali (o viste)

Viste materializzate

- relazioni derivate memorizzate nella base di dati
  - vantaggi:
    - immediatamente disponibili per le interrogazioni
  - svantaggi:
    - ridondanti
    - appesantiscono gli aggiornamenti
    - sono raramente supportate dai DBMS

Viste virtuali

- relazioni virtuali (o viste):
  - sono supportate dai DBMS (tutti)
  - una interrogazione su una vista viene eseguita "ricalcolando" la vista (o quasi)

Interrogazioni sulle viste

- Sono eseguite sostituendo alla vista la sua definizione:

$SEL_{Capo='Leoni'}$ (Supervisione)

viene eseguita come

$SEL_{Capo='Leoni'}$($PROJ_{Impiegato,\ Capo}$ (Afferenza JOIN Direzione))

Viste, motivazioni

- Schema esterno: ogni utente vede solo
  - ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto
  - ciò che e' autorizzato a vedere (autorizzazioni)
- Strumento di programmazione:
  - si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute
- Utilizzo di programmi esistenti su schemi ristrutturati

Invece:

- L'utilizzo di viste non influisce sull'efficienza delle interrogazioni

Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi
- Senza vista:

$PROJ_{Impiegato}$ (Afferenza JOIN Direzione) JOIN $REN_{ImpR,RepR \leftarrow Imp,Reparto}$ ($SEL_{Impiegato='Rossi'}$ (Afferenza JOIN Direzione))

- Con la vista:

$PROJ_{Impiegato}$ (Supervisione) JOIN $REN_{ImpR,RepR \leftarrow Imp,Reparto}$ ($SEL_{Impiegato='Rossi'}$ (Supervisione))

Viste e aggiornamenti, attenzione

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Verdi	A

Direzione

Reparto	Capo
A	Mori
B	Bruni
C	Bruni

Supervisione

Impiegato	Capo
Rossi	Mori
Neri	Bruni
Verdi	Mori

- Vogliamo inserire, nella vista, il fatto che Lupi ha come capo Bruni; oppure che Belli ha come capo Falchi; come facciamo?

Viste e aggiornamenti

- "Aggiornare una vista":
  - modificare le relazioni di base in modo che la vista, "ricalcolata" rispecchi l'aggiornamento
- L'aggiornamento sulle relazioni di base corrispondente a quello specificato sulla vista deve essere univoco
- In generale però non è univoco!
- Ben pochi aggionamenti sono ammissibili sulle viste

Una convenzione e notazione alternativa per i join

- Nota: è sostanzialmente l'approccio usato in SQL
- Ignoriamo il join naturale (cioè non consideriamo implicitamente condizioni su attributi con nomi uguali)
- Per "riconoscere" attributi con lo stesso nome gli premettiamo il nome della relazione
- Usiamo "assegnazioni" (viste) per ridenominare le relazioni (e gli attributi solo quando serve per l'unione)

- Trovare gli impiegati che guadagnano più del proprio capo, mostrando matricola, nome e stipendio dell'impiegato e del capo

$$\text{PROJ}_{\text{Matr,Nome,Stip,MatrC,NomeC,StipC}} (\text{SEL}_{\text{Stipendio>StipC}} (\text{REN}_{\text{MatrC,NomeC,StipC,EtàC} \leftarrow \text{Matr,Nome,Stip,Età}}(\text{Impiegati}) \text{ JOIN}_{\text{MatrC=Capo}} (\text{Supervisione JOIN}_{\text{Impiegato=Matricola}} \text{Impiegati})))$$

$$\text{PROJ}_{\text{Matr,Nome,Stip,MatrC,NomeC,StipC}} (\text{SEL}_{\text{Stip>StipC}} (\text{REN}_{\text{MatrC,NomeC,StipC,EtàC} \leftarrow \text{Matr,Nome,Stip,Età}}(\text{Imp}) \text{ JOIN}_{\text{MatrC=Capo}} (\text{Sup JOIN}_{\text{Imp=Matr}} \text{Imp})))$$

$$\text{Capi} := \text{Imp}$$

$$\text{PROJ}_{\text{Imp.Matr, Imp.Nome, Imp.Stip,Capi.Matr,Capi.Nome, Capi.Stip}} (\text{SEL}_{\text{Imp.Stip>Capi.Stip}} (\text{Capi JOIN}_{\text{Capi.Matr=Capo}} (\text{Sup JOIN}_{\text{Imp=Imp.Matr}} \text{Imp})))$$

Calcolo relazionale

- Una famiglia di linguaggi dichiarativi, basati sul calcolo dei predicati del primo ordine
- Diverse versioni:
  - calcolo relazionale su domini
  - calcolo su ennuple con dichiarazioni di range

Calcolo su domini, sintassi e semantica

- Le espressioni hanno la forma:
  $\{ A_1: x_1, \ldots, A_k: x_k \mid f \}$
  - f e' una formula (con connettivi booleani e quantificatori)
  - $A_1: x_1, \ldots, A_k: x_k$ "target list":
    - $A_1, \ldots, A_k$ attributi distinti (anche non nella base di dati)
    - $x_1, \ldots, x_k$ variabili distinte
- Semantica: il risultato e' una relazione su $A_1, \ldots, A_k$ che contiene ennuple di valori per $x_1, \ldots, x_k$ che rendono vera la formula f

Commenti

- Differenze rispetto al calcolo dei predicati (per chi lo conosce):
  - simboli di predicato
    - relazioni nella base di dati
    - predicati "standard" predefiniti (=, >, ...)
  - non ci sono "simboli di funzione"
  - interessano (quasi) solo "formule aperte"
  - utilizziamo notazione non posizionale

Base di dati per gli esempi

Impiegati(Matricola,Nome, Età, Stipendio)
Supervisione(Capo, Impiegato)

Esempio 0a

- **Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni**

$SEL_{Stipendio>40}(Impiegati)$

{ Matricola: m, Nome: n, Età: e, Stipendio: s | Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 }

Esempio 0b

- **Trovare matricola, nome ed età di tutti gli impiegati**

$PROJ_{Matricola, Nome, Età}(Impiegati)$

{ Matricola: m, Nome: n, Età: e | ∃s (Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)}

{ Matricola: m, Nome: n, Età: e | Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)}

Esempio 1

- **Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni**

$PROJ_{Matricola, Nome, Età}(SEL_{Stipendio>40}(Impiegati))$

{ Matricola: m, Nome: n, Età: e | Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 }

Esempio 2

- **Trovare le matricole dei capi degli impiegati che guadagnano più di 40 milioni**

$PROJ_{Capo}$ (Supervisione $JOIN_{Impiegato=Matricola}$ $(SEL_{Stipendio>40}(Impiegati))$)

{ Capo: c | Supervisione(Capo:c,Impiegato:m) ∧ Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 }

Esempio 3

- **Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 milioni**

$PROJ_{NomeC,StipC}$
$(REN_{MatrC,NomeC,StipC,EtàC←Matr,Nome,Stip,Età}(Impiegati)$
$JOIN_{MatrC=Capo}$
(Supervisione $JOIN_{Impiegato=Matricola}$
$(SEL_{Stipendio>40}(Impiegati))))$

{ NomeC: nc, StipC: sc | Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 ∧ Supervisione(Capo:c,Impiegato:m) ∧ Impiegati(Matricola:c, Nome:nc, Età:ec, Stipendio: sc) }

Esempio 4

- Trovare gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di ciascuno di essi e del capo

$$\text{PROJ}_{\text{Matr,Nome,Stip,MatrC,NomeC,StipC}}(\text{SEL}_{\text{Stipendio>StipC}}(\text{REN}_{\text{MatrC,NomeC,StipC,EtàC} \leftarrow \text{Matr,Nome,Stip,Età}}(\text{Impiegati}) \text{ JOIN}_{\text{MatrC=Capo}} (\text{Supervisione JOIN}_{\text{Impiegato=Matricola}} (\text{Impiegati}))))$$

$$\{ \text{Matr: m, Nome: n, Stip: s, MatrC: c, NomeC: nc, StipC: sc} \mid \text{Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c,Impiegato:m)} \wedge \text{Impiegati(Matricola: c, Nome: nc, Età: ec, Stipendio: sc)} \wedge s > sc\}$$

Esempio 5

- Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 milioni.

$$\text{PROJ}_{\text{Matricola,Nome}}(\text{Impiegati JOIN}_{\text{Matricola=Capo}} (\text{PROJ}_{\text{Capo}}(\text{Supervisione}) - \text{PROJ}_{\text{Capo}}(\text{Supervisione JOIN}_{\text{Impiegato=Matricola}} (\text{SEL}_{\text{Stipendio} \leq 40}(\text{Impiegati})))))$$

$$\{\text{Matricola: c, Nome: n} \mid \text{Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c, Impiegato:m)} \wedge \neg \exists m'(\exists n'(\exists e'(\exists s'(\text{Impiegati(Matr: m', Nome: n', Età: e', Stip: s')} \wedge \text{Supervisione(Capo:c, Impiegato:m')} \wedge s' \leq 40)))))\}$$

Quantificatori esistenziali o universali?

- Sono intercambiabili, per le leggi di De Morgan:

$$\{\text{Matricola: c, Nome: n} \mid \text{Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c, Impiegato:m)} \wedge \neg \exists m'(\exists n'(\exists e'(\exists s'(\text{Impiegati(Matr: m', Nome: n', Età: e', Stip: s')} \wedge \text{Supervisione(Capo:c, Impiegato:m')} \wedge s' \leq 40)))))\}$$

$$\{\text{Matricola: c, Nome: n} \mid \text{Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c, Impiegato:m)} \wedge \forall m'(\forall n'(\forall e'(\forall s'(\neg(\text{Impiegati(Matr:m', Nome:n', Età:e', Stip:s')} \wedge \text{Supervisione(Capo:c, Impiegato:m')}) \vee s' > 40)))))\}$$

Calcolo su domini, discussione

- Pregi:
  - dichiarativitá
- Difetti:
  - "verbosità": tante variabili!
  - espressioni senza senso:

$$\{ \text{A: x} \mid \neg \text{R(A: x)} \}$$

$$\{ \text{A: x, B: y} \mid \text{R(A: x)} \}$$

$$\{ \text{A: x, B: y} \mid \text{R(A: x)} \wedge y=y \}$$

queste espressioni sono "dipendenti dal dominio" e vorremmo evitarle;
nell'algebra espressioni come queste non sono formulabili: l'algebra è indipendente dal dominio

Calcolo e algebra

- Calcolo e algebra sono "equivalenti"
  - per ogni espressione del calcolo relazionale che sia indipendente dal dominio esiste un'espressione dell'algebra relazionale equivalente a essa
  - per ogni espressione dell'algebra relazionale esiste un'espressione del calcolo relazionale equivalente a essa (e di conseguenza indipendente dal dominio)

Calcolo su ennuple con dichiarazioni di range

- Per superare le limitazioni del calcolo su domini:
  - dobbiamo "ridurre" le variabili; un buon modo: una variabile per ciascuna ennupla
  - far si' che i valori provengano dalla base di dati
- Il calcolo su ennuple con dichiarazioni di range risponde ad entrambe le esigenze

Calcolo su ennuple con dichiarazioni di range, sintassi

- Le espressioni hanno la forma:
  { *TargetList* | *RangeList* | *Formula* }
  - *RangeList* elenca le variabili libere della *Formula* ognuna con il relativo campo di variabilità (una relazione)
  - *TargetList* ha elementi del tipo Y: x.Z (oppure x.Z o anche x.*)
  - *Formula* ha:
    - atomi di confronto x.A ϑ c, x.A ϑ y.B
    - connettivi
    - quantificatori che associano un range alle variabili

      $\exists$x(R)(...) $\forall$x(R)(...)

Esempio 0a

- Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni

$SEL_{Stipendio>40}$(Impiegati)

{ Matricola: m, Nome: n, Età: e, Stipendio: s |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)
$\wedge$ s > 40 }

{ i.* | i(Impiegati) | i.Stipendio > 40 }

Esempio 0b

- Trovare matricola, nome ed età di tutti gli impiegati

$PROJ_{Matricola, Nome, Età}$(Impiegati)

{ Matricola: m, Nome: n, Età: e |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)}

{ i.(Matricola,Nome,Età) | i(Impiegati) | }

Esempio 1

- Trovare matricola, nome ed età degli impiegati che guadagnano più di 40 milioni

$PROJ_{Matricola, Nome, Età}(SEL_{Stipendio>40}$(Impiegati))

{ Matricola: m, Nome: n, Età: e |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)
$\wedge$ s > 40 }

{ i.(Matricola,Nome,Età) | i(Impiegati) | i.Stipendio > 40 }

Esempio 2

- Trovare le matricole dei capi degli impiegati che guadagnano più di 40 milioni

{ Capo: c | Supervisione(Capo:c,Impiegato:m) $\wedge$
Impiegati(Matricola: m, Nome: n, Età: e,
Stipendio: s) $\wedge$ s > 40 }

{ s.Capo | i(Impiegati) , s(Supervisione) |
i.Matricola=s.Impiegato $\wedge$ i.Stipendio > 40 }

Esempio 3

- Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 40 milioni

{ NomeC: nc, StipC: sc |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) $\wedge$
s > 40 $\wedge$
Supervisione(Capo:c,Impiegato:m) $\wedge$
Impiegati(Matricola:c, Nome:nc, Età:ec, Stipendio:sc) }

{ NomeC,StipC: i'.(Nome,Stip) |
i'(Impiegati), s(Supervisione), i(Impiegati) |
i'.Matricola=s.Capo $\wedge$ i.Matricola=s.Impiegato $\wedge$
i.Stipendio > 40 }

Esempio 4

- Trovare gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di ciascuno di essi e del capo

$$\{ \text{Matr: m, Nome: n, Stip: s, NomeC: nc, StipC: sc} \mid \text{Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c,Impiegato:m)} \wedge \text{Impiegati(Matricola: c, Nome: nc, Età: ec, Stipendio: sc)} \wedge s > sc\}$$

$$\{ \text{i.(Nome,Matr,Stip), NomeC,MatrC,StipC: i'.(Nome,Matr,Stip)} \mid \text{i'(Impiegati), s(Supervisione), i(Impiegati)} \mid \text{i'.Matricola=s.Capo} \wedge \text{i.Matricola=s.Impiegato} \wedge \text{i.Stipendio} > \text{i'.Stipendio} \}$$

Esempio 5

- Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 milioni.

$$\{\text{Matricola: c, Nome: n} \mid \text{Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s)} \wedge \text{Supervisione(Capo:c, Impiegato:m)} \wedge \neg \exists m'(\exists n'(\exists e'(\exists s'(\text{Impiegati(Matr: m', Nome: n', Età: e', Stip: s')} \wedge \text{Supervisione(Capo:c, Impiegato:m')} \wedge s' \leq 40\}$$

$$\{ \text{i.(Matricola, Nome)} \mid \text{s(Supervisione), i(Impiegati)} \mid \text{i.Matricola=s.Capo} \wedge \neg(\exists i'(\text{Impiegati})(\exists s'(\text{Supervisione}) (\text{s.Capo=s'.Capo} \wedge \text{s'.Impiegato=i'.Matricola} \wedge \text{i'.Stipendio} \leq 40)))\}$$

Attenzione!

- Il calcolo su ennuple con dichiarazioni di range non permette di esprimere alcune interrogazioni importanti, in particolare le unioni:

$$R_1(AB) \cup R_2(AB)$$

- Quale potrebbe essere il range per una variabile? Oppure due variabili?

- Nota: intersezione e differenza sono esprimibili
- Per questa ragione SQL (che è basato su questo calcolo) prevede un operatore esplicito di unione, ma non tutte le versioni prevedono intersezione e differenza

Calcolo e algebra relazionale: limiti

- Calcolo e algebra sono sostanzialmente equivalenti: l'insieme di interrogazioni con essi esprimibili è quindi significativo; il concetto è robusto
- Ci sono però interrogazioni interessanti non esprimibili:
  - calcolo di valori derivati: possiamo solo estrarre valori, non calcolarne di nuovi; calcoli di interesse:
    - a livello di ennupla o di singolo valore (conversioni somme, differenze, etc.)
    - su insiemi di ennuple (somme, medie, etc.)

    le estensioni sono ragionevoli, le vedremo in SQL
  - interrogazioni inerentemente ricorsive, come la chiusura transitiva

Chiusura transitiva

Supervisione(<u>Impiegato,</u> Capo)

- Per ogni impiegato, trovare tutti i superiori (cioè il capo, il capo del capo, e cosi' via)

Impiegato	Capo
Rossi	Lupi
Neri	Bruni
Lupi	Falchi

Impiegato	Superiore
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Rossi	Falchi

Chiusura transitiva, come si fa?

- Nell'esempio, basterebbe il join della relazione con se stessa, previa opportuna ridenominazione
- Ma:

Impiegato	Capo
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Falchi	Leoni

Impiegato	Superiore
Rossi	Lupi
Neri	Bruni
Lupi	Falchi
Falchi	Leoni
Rossi	Falchi
Lupi	Leoni
Rossi	Leoni

Chiusura transitiva, impossibile!

- Non esiste in algebra e calcolo relazionale la possibilità di esprimere l'interrogazione che, per ogni relazione binaria, ne calcoli la chiusura transitiva
- Per ciascuna relazione, è possibile calcolare la chiusura transitiva, ma con un'espressione ogni volta diversa:
  - quanti join servono?
  - non c'è limite!

Datalog

- Un linguaggio di programmazione logica per basi di dati derivato dal Prolog
- Utilizza predicati di due tipi:
  - estensionali: relazioni della base di dati
  - intensionali: corrispondono alle viste
- Il linguaggio è basato su regole utilizzate per "definire" i predicati estensionali

Datalog, sintassi

- Regole:

  testa ← *corpo*

  - *testa* è un predicato atomico (intensionale)
  - *corpo* è una lista (congiunzione) di predicati atomici
- Le interrogazioni sono specificate per mezzo di predicati atomici (convenzionalmente preceduti da "?")

Esempio -1

- Trovare matricola, nome, età e stipendio degli impiegati che hanno 30 anni

{ Matricola: m, Nome: n, Età: e, Stipendio: s |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)
∧ s = 30 }

? Impiegati(Matricola: m, Nome: n, Età: 30, Stipendio: s)

Esempio 0a

- Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 40 milioni

{ Matricola: m, Nome: n, Età: e, Stipendio: s |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 }

- Serve un predicato intensionale

ImpRicchi(Matricola: m, Nome: n, Età: e, Stipendio: s) ←
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) , s >40

? ImpRicchi(Matricola: m, Nome: n, Età: e, Stipendio: s)

Esempio 0b

- Trovare matricola, nome ed età di tutti gli impiegati

$PROJ_{Matricola, Nome, Età}$(Impiegati)

{ Matricola: m, Nome: n, Età: e |
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)}

InfoPubbliche(Matricola: m, Nome: n, Età: e)
← Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s)

? InfoPubbliche(Matricola: m, Nome: n, Età: e)

Esempio 2

- Trovare le matricole dei capi degli impiegati che guadagnano più di 40 milioni

{ Capo: c | Supervisione(Capo:c,Impiegato:m) ∧
Impiegati(Matricola: m, Nome: n, Età: e, Stipendio: s) ∧ s > 40 }

CapiDeiRicchi (Capo:c) ←
ImpRicchi(Matricola: m, Nome: n, Età: e, Stipendio: s),
Supervisione (Capo:c,Impiegato:m)

? CapiDeiRicchi (Capo:c)

Esempio 5

- Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 40 milioni.
- serve la negazione

CapiDiNonRicchi (Capo:c) ←
Supervisione (Capo:c,Impiegato:m),
Impiegati (Matricola: m, Nome: n, Età: e, Stipendio: s) ,
s ≤ 40
CapiSoloDiRicchi (Matricola: c, Nome: n) ←
Impiegati (Matricola: c, Nome: n, Età: e, Stipendio: s) ,
Supervisione (Capo:c,Impiegato:m),
not CapiDiNonRicchi (Capo:c)

? CapiSoloDiRicchi (Matricola: c, Nome: n)

Esempio 6

- Per ogni impiegato, trovare tutti i superiori.
- Serve la ricorsione

Superiore (Impiegato: i, SuperCapo: c) ←
Supervisione (Impiegato: i, Capo: c)

Superiore (Impiegato: i, SuperCapo: c) ←
Supervisione (Impiegato: i, Capo: c'),
Superiore (Impiegato: c', SuperCapo: c)

Datalog, semantica

- La definizione della semantica delle regole ricorsive è delicata (in particolare con la negazione)
- Potere espressivo:
  - Datalog non ricorsivo senza negazione è equivalente al calcolo senza negazione e senza quantificatore universale
  - Datalog non ricorsivo con negazione è equivalente al calcolo e all'algebra
  - Datalog ricorsivo senza negazione e calcolo sono incomparabili
  - Datalog ricorsivo con negazione è più espressivo di calcolo e algebra

Algebra relazionale
Calcolo relazionale
Datalog

Paolo Atzeni
06/06/2005

(con materiale di M. Lenzerini)

Algebra relazionale

Algebra relazionale

Costitutita da un insieme di operatori
- definiti su relazioni
- che producono relazioni
- e possono essere composti

Operatori dell'algebra relazionale:
- unione, intersezione, differenza
- ridenominazione
- selezione
- proiezione
- join (join naturale, prodotto cartesiano, theta-join)

Potenza dell'algebra relazionale

La potenza di un linguaggio di query si può studiare da due angolazioni diverse:

– Potere computazionale: Data una istanza B di basi di dati, cosa può essere calcolato a partire da B mediante il linguaggio?

– Potere espressivo: Quali funzioni sulle istanze di basi di dati si possono esprimere con le espressioni del linguaggio ?

Vogliamo caratterizzare la potenza dell'algebra relazionale come linguaggio di query. Analizziamo quindi queste due domande nel caso dell'algebra relazionale.

Iniziamo dalla prima angolazione, introducendo la nozione di automorfismo.

Automorfismi per istanze di basi di dati

Il dominio attivo di una istanza di una base di dati è l'insieme dei valori che vi compaiono. Se D è il dominio di B, il dominio attivo D_B è ovviamente un sottoinsieme di D.

Definizione: Sia **B** una istanza di base di dati, e sia D il suo dominio. Una funzione parziale

$$h: D \rightarrow D$$

è un automorfismo per **B** se

1. è una permutazione del dominio attivo D_B
2. estendendo h a tuple, relazioni e istanze, otteniamo una funzione che è l'identità su **B**

Definizione: Sia **B** una istanza di base di dati, sia D_B il suo dominio attivo, e sia f una funzione del tipo f: $D_B \rightarrow D_B$. Una relazione r si dice invariante rispetto a f se f(r)=r, dove f(r) denota l'estensione di f alla relazione r.

Automorfismo: esempio 1

Successione

Predecessore	Successore
Enrico I	Enrico II
Enrico II	Stefano
Stefano	Giovanni

Due valori sono indistinguibili in una istanza di base di dati B se la funzione che
– li scambia
– è l'identità sugli altri valori
è un automorfismo per B.

L'unico automorfismo per l'istanza di base di dati mostrata sopra è la funzione identità, ciò significa che per ogni coppia di valori v,w che compaiono in essa, v e w sono distinguibili tra loro.

Automorfismo: esempio 2

EsitoEsami

Studente	Voto
a	25
b	18
a	18
b	25

Quanti automorfismi ci sono per questa istanza di basi di dati ?

I valori 25 e 18 sono distinguibili in questa istanza?

7

Invarianza: esercizi

1. Relativamente all'esercizio 2, dare un esempio di relazione r ed un esempio di automorfismo h tale che r non è invariante rispetto ad h
2. Ricordiamo che la chiusura transitiva r* di una relazione binaria r è la relazione che si ottiene aggiungendo fino a che possibile la coppia <a,b> se esiste un valore c tale che le coppie <a,c> e <c,b> sono in r o sono state aggiunte precedentemente.

   Sia S uno schema di basi di dati con uno schema di relazione binaria R(A,B). Sia B una istanza di S dove r è l'istanza della relazione R. Sia h un automorfismo su B.

   Dire se r* è invariante rispetto ad h.

8

Potere computazionale dell'algebra relazionale

Data una istanza di basi di dati, vogliamo caratterizzare le relazioni che possono essere calcolate mediante l'algebra relazionale.

Sia C un sottoinsieme del dominio attivo D_B di una istanza **B** di basi di dati.

- Un automorfismo h per B si dice C-fixed se h è l'identità su C
- Si dice che una espressione dell'algebra menziona C se tutte le costanti che compaiono in essa appartengono a C

Teorema Sia **B** una istanza di base di dati, e C un sottoinsieme del suo dominio attivo. Una relazione r può essere ottenuta come risultato di una espressione dell'agebra relazionale su **B** che menziona C se e solo se r è invariante rispetto ad ogni C-fixed automorfismo per **B**.

9

Potere espressivo dell'algebra relazionale

Ricordiamo che il potere espressivo di un linguaggio di query mira a caratterizzare quali funzioni sulle istanze di basi di dati si possono esprimere con le espressioni del linguaggio.

Per studiare il potere espressivo dell'algebra relazionale, dobbiamo quindi per prima cosa capire in che modo possiamo interpretare una query come una funzione.

10

Query come funzione

Assumiamo di disporre di un insieme U infinito numerabile di attributi, che possono assumere valori di un insieme qualunque D (dominio per le base di dati). Indichiamo con RU l'insieme di tutte le relazioni finite su valori del dominio D con attributi in U.

Dato uno schema S di basi di dati, ed un dominio D, indichiamo con I(S) l'insieme di tutte le possibili istanze di S formate con valori di D.

Una query Q su S è una funzione ricorsiva parziale

$$Q : I(S) \rightarrow RU$$

cioè che associa ad ogni istanza di S una relazione su D.

11

Espressività dei linguaggi

Un linguaggio di query L è semplicemente un insieme di espressioni, quelle formata in accordo con le leggi di L, ognuna delle quali definisce una query.

Diciamo che una espressione E di L rappresenta la query Q (ovvero che rappresenta la funzione Q) se Q è la funzione definita da E.

Diciamo che un linguaggio di query L può esprimere una funzione F, o semplicemente esprime F, oppure che F è esprimibile in L, se esiste almeno una espressione E di L che rappresenta la funzione F.

12

Esempio di query esprimibile in algebra

Sia S uno schema di basi di dati con uno schema di relazione binaria R(A,B). Ricordiamo che una relazione binaria può essere vista come un grafo.

Sia SIM la query (ovvero, la funzione) su S che, per ogni istanza T in I(S), restituisce la relazione corrispondente al grafo ottenuto aggiungendo l'arco <a,b> per ogni arco <b,a> in r, dove r è l'istanza di relazione in T corrispondente a R.

Teorema. La query SIM è esprimibile in algebra relazionale.

13

Esempio di query non esprimibile in algebra

Ricordiamo che la chiusura transitiva r* di una relazione binaria r è la relazione che si ottiene aggiungendo fino a che possibile la coppia <a,b> se esiste un valore c tale che le coppie <a,c> e <c,b> sono in r o sono state aggiunte precedentemente.

Sia S uno schema di basi di dati con uno schema di relazione binaria R(A,B). Sia TRANS la query (ovvero, la funzione) su S che, per ogni istanza T in I(S), restituisce la chiusura transitiva di r, dove r è l'istanza di relazione in T corrispondente a R.

Teorema. La query TRANS non è esprimibile in algebra relazionale.

14

Confronto tra potere computazionale e potere espressivo

Non si deve confondere il potere computazionale ed il potere espressivo di un linguaggio.

Sia **B** una istanza di base di dati. La chiusura transitiva di una relazione r in **B** è invariante rispetto ad ogni automorfismo per **B**. Ne segue che, data una istanza di basi di dati con una relazione r, esiste una espressione dell'algebra relazionale che, quando viene valutata su **B**, fornisce come risultato la chiusura transitiva di r. Quindi il potere computazionale dell'algebra è sufficiente a calcolare la chiusura transitiva di una relazione.

Al contrario, abbiamo appena visto che la query TRANS non è esprimibile in algebra relazionale, ovvero che non esiste alcuna espressione E dell'algebra relazionale tale che, per ogni istanza di base di dati con la relazione r, E(r) fornisce come risultato la chiusura transitiva di r. Quindi, il potere espressivo dell'algebra relazionale non è sufficiente a definire una query che, per ogni istanza, calcola la chiusura transitiva di una relazione in quella istanza.

15

Calcolo relazionale

16

Che cosa è il calcolo relazionale

Il calcolo relazionale è un linguaggio di interrogazione basato sulla logica del primo ordine.

Due versioni del calcolo relazionale:
- Calcolo relazionale sul dominio
- Calcolo relazionale sulle tuple

Un linguaggio di query L1 si dice almeno espressivo quanto un altro linguaggio di query L2 se per ogni espressione di L2 esiste una espressione di L1 che rappresenta le stessa query. Due linguaggi L1 e L2 sono ugualmente espressivi, o equivalenti se L1 è almeno espressivo quanto L2 e L2 è almeno espressivo quanto L1.

Le due versioni del calcolo relazionale sono equivalenti. Noi analizzeremo solo il calcolo relazionale sul dominio.

17

Calcolo relazionale: le basi

Idea base: si usa il linguaggio della logica del primo ordine per esprimere le tuple che si vogliono ottenere come risultato.

Sia U un insieme infinito numerabile di attributi, che possono assumere valori di un insieme qualunque D (dominio per le base di dati), e sia COST un insieme di simboli di costanti che contiene un simbolo di costante per ogni valore di D.

Sia S uno schema di base di dati. Diciamo che una base di dati B su S e D è una istanza di S che contiene come dati le costanti di COST (che, come abbiamo visto prima, sono interpretabili come valori di D).

Sia L_S il linguaggio della logica del primo ordine i cui simboli di costante sono quelli in COST, ed i cui simboli di predicato sono quelli di S più tutti quelli che possono comparire in una condizione booleana dell'algebra relazionale (ad esempio: <,>,…), ed i cui simboli di funzione sono assenti.

18

Calcolo relazionale: sintassi

Una espressione del calcolo relazionale (sul dominio) ha la forma:

$$\{ A_1 : x_1,\ldots,A_k : x_k \mid F \}$$

dove:

- $k > 0$
- $A_1,\ldots,A_k$ sono attributi,
- $x_1,\ldots,x_k$ sono variabili,
- F è il corpo dell'espressione, cioè una formula del linguaggio L_S del primo ordine in cui le uniche variabili libere sono $x_1,\ldots,x_k$
- $A_1 : x_1,\ldots, A_k : x_k$ viene chiamata la target list della espressione, e si può anche scrivere semplicemente come $x_1,\ldots,x_k$ (intendendo che ogni valore nella terget list è identificato dalla posizione, e non dall'attributo)

Se R è una relazione con schema $R(A_1,\ldots,A_h)$, in F, un atomo con simbolo di predicato R ha la forma

$$R(x_1,\ldots,x_h)$$

che si può anche scrivere come $R(A_1 : x_1,\ldots,A_h : x_h)$.

19

Calcolo relazionale: semantica

Le espressioni del calcolo relazionale si valutano su particolari interpretazioni. Nel seguito, faremo sempre riferimento ad uno schema S di basi di dati.

Una interpretazione corretta per il calcolo relazionale è una coppia <D,B>, dove D è un dominio, e B è una base di dati su S e D.

Il valore della espressione del calcolo relazionale

$$\{ A_1 : x_1,\ldots, A_k : x_k \mid F \}$$

in una interpretazione M = <D,B> è l'insieme delle tuple $<c_1,\ldots,c_k>$ di costanti in COST tale che

$$<M,v> \models F$$

dove v è l'assegnazione che associa ad ogni variabile x_i nella target list la corrispondente costante c_i nella tupla $<c_1,\ldots,c_k>$.

20

Calcolo relazionale: semantica

Quando è evidente qual è il dominio D su cui è definita la base di dati B, la specifica di D si può omettere e scrivere direttamente $<B,v> \models F$ invece di $< <D,B>, v > \models F$.

Il risultato della valutazione di una espressione $\{ A_1 : x_1,\ldots, A_k : x_k \mid F \}$ rispetto ad una base di dati B su S e D è la relazione R tale che

- lo schema di R è l'insieme degli attributi $A_1,\ldots,A_k$ che compaiono nella target list
- l'istanza di R è l'insieme delle tuple che costituiscono il valore dell'espressione nell'interpretazione <D,B>.

Teorema Per ogni espressione del calcolo relazionale, c'è una espressione equivalente in cui la formula F in essa contenuta coinvolge solo negazione, un connettivo tra disgiunzione e congiunzione, ed un quantiifcatore tra esistenziale e universale.

21

Conjunctive query

Una particolare sottoclasse del calcolo relazionale è costituita dalle query congiuntive (conjunctive query).

Una conjunctive query è una espressione del calcolo relazionale (sul dominio) che ha la forma:

$$\{ x_1,\ldots,x_k \mid \exists\, y_1,\ldots,y_h \ (a_1 \wedge \ldots \wedge a_m) \}$$

Dove:

- $K > 0$, e $h >= 0$
- $a_1,\ldots,a_m$ sono atomi il cui predicato è un simbolo di relazione
- $x_1,\ldots,x_k,y_1,\ldots,y_h$ sono le uniche variabili che compaiono in $a_1,\ldots,a_m$

22

Calcolo relazionale: un problema

Consideriamo una relazione binaria Maternità con attributi Madre e Figlio, e consideriamo l'espressione

$$E = \{ \text{Nome} : x \mid \neg\, \exists y \ \text{Maternità}(\text{Madre}:x,\text{Figlio}:y) \}$$

Se il dominio D è infinito, qualunque sia la base di dati B sul dominio D, si ha che il valore dell'espressione E su <D,B> è infinito.

Un altro aspetto del problema è che se D' è un dominio diverso dal dominio D, il valore di E su D' e B può essere diverso rispetto al valore di E su D e B.

Un ulteriore aspetto del problema è che E non definisce una sola query (intesa come funzione ricorsiva parziale che per ogni base di dati B restituisce una relazione RU), ma definisce una query per ogni dominio della base di dati.

23

Calcolo relazionale: indipendenza dal dominio

Una espressione si dice indipendente dal dominio se essa rappresenta la stessa query (cioè la stessa funzione) indipendentemente dal dominio.

Un linguaggio si dice indipendente dal dominio se tutte le espressioni del linguaggio sono indipendenti dal dominio.

Teorema L'algebra relazionale è indipendente dal dominio, il linguaggio delle conjunctive query è independente dal dominio, mentre il calcolo relazionale è dipendente dal dominio.

Chiamiamo calcolo relazionale indipendente dal dominio l'insieme delle espressioni indipendenti dal dominio del calcolo relazionale.

Teorema Il seguente problema è indecidibile: data una espressione E del calcolo relazionale, dire se E è indipendente dal dominio, ovvero se è una espressione del calcolo relazionale indipendente dal dominio.

24

Calcolo relazionale con dichiarazioni di range

Sia E una espressione del calcolo relazionale e B una base di dati. Chiamiamo dominio attivo di E e B l'unione tra il dominio attivo di B e l'insieme dei valori corrispondenti alle costanti che compaiono in E.

Se riuscissimo ad imporre che le variabili presenti in una espressione E del calcolo relazionale assumano valori del solo domino attivo di E e B, allora otterremmo solo espressioni indipendenti dal dominio. In effetti questo si può ottenere, nel modo seguente.

Chiamiamo una componente di range

- o un insieme di costanti,
- oppure una espressione della forma R[A], dove R(X) è uno schema di relazione, ed A un attributo in X.

Una dichiarazione di range è un insieme finito di componenti di range.

Calcolo relazionale con dichiarazioni di range

Definiamo un nuovo linguaggio di query, che è una variante del calcolo relazionale sui domini, e che è chiamato calcolo relazionale con dichiarazioni di range.

Una espressione del calcolo relazionale con dichiarazioni di range è una espressione del calcolo relazionale in cui ad ogni variabile nella target list e ad ogni variabile quantificata nella formula F è associata una dichiarazione di range.

Sintassi:

- una query viene scritta come $\{ A_1 : x_1(G_1)\ldots, A_k : x_k(G_k) \mid F \}$
- le sottoformule quantificate in F vengono scritte $\exists x(G)\ F'$ e $\forall x(G)\ F'$

Semantica:

Nella valutazione, usiamo assegnazioni che rispettano le dichiarazioni di range. Ad esempio, alla variabile x in $\exists x(G)\ F'$ vengono associati solo valori che cadono nel range definito da G.

Calcolo relazionale con dichiarazioni di range

Esempio: Consideriamo lo schema di basi di dati costituito da Persona(Nome,Sesso,Età) e Discendenza(Genitore,Figlio). Vogliamo mostrare un esempio di query nel calcolo relazionale con dichiarazioni di range. Consideriamo una variante della query che avevamo visto in precedenza:

$$\{ \text{Nome} : x \mid \neg\, \exists y\ \text{Discendenza(Genitore:}x\text{,Figlio:}y) \}$$

Possiamo trasformare la query in questa query del calcolo relazionale con dichiarazioni di range:

$$\{ \text{Nome}:x(\text{Persona[Nome]}) \mid \neg\exists y(\text{Persona[Nome]})\ \text{Discendenza(Genitore:}x\text{,Figlio:}y) \}$$

Teorema Il calcolo relazionale con dichiarazioni di range è indipendente dal dominio.

Complessità del calcolo relazionale

Complessità delle conjunctive query:

- La complessità rispetto ai dati delle conjunctive query è polinomiale.
- Per quanto riguarda la complessità combinata, il problema di valutare se una tupla è contenuta nel risultato della valutazione di una conjunctive query è NP-completo.

Complessità del calcolo relazionale con dichiarazioni di range:

- La complessità rispetto ai dati del calcolo relazionale con dichiarazioni di range è polinomiale.
- Per quanto riguarda la complessità combinata, il problema di valutare se una tupla è contenuta nel risultato della valutazione di una espressione del calcolo relazionale con dichiarazioni di range è Pspazio-completo.

Espressività del calcolo relazionale con dichiarazioni di range

Teorema Il calcolo relazionale con dichiarazioni di range è equivalente al calcolo relazionale indipendente dal dominio.

Teorema Il calcolo relazionale con dichiarazioni di range è equivalente all'algebra relazionale.

Esercizio: Dimostrare il teorema che asserisce che il calcolo relazionale con dichiarazioni di range è equivalente all'algebra relazionale.

Suggerimento: Condurre la dimostrazione per induzione, in due parti:

- Si dimostra che per ogni espressione del calcolo relazionale con dichiarazioni di range c'è una equivalente espressione dell'algebra relazionale
- Si dimostra che per ogni espressione dell'algebra relazionale c'è una equivalente espressione espressione del calcolo relazionale che è indipendente dal dominio

Datalog

Datalog

Datalog è basato sulla logica del primo ordine senza simboli di funzione, come il calcolo relazionale. L'insieme delle costanti COST si chiama anche universo di Herbrand.

In Datalog, l'unità fondamentale è la clausola di Horn. Ricordiamo che una clausola è una disgiunzione di literal, e un literal o è un atomo, oppure è la negazione di un atomo. Tutte le variabili di una clausola si intendono quantificate universalmente. Una clausola di Horn è una clausola con al più un literal positivo.

Esempio (le maiuscole sono variabili, le minuscole sono costanti):
C = ($\neg$ p(X,a) $\vee$ q(Y,b)) che corrisponde a ($\forall$X $\forall$Y ($\neg$ p(X,a) $\vee$ q(Y,b))

Una clausola si scrive anche come l'insieme dei suoi literal. Ad esempio, C si può scrivere come {$\neg$ p(X,a) ,q(Y,b) }.

Una clausola di dice positiva se non ha literal negativi. Si dice unit se ha un literal, e si dice ground se vi compaiono costanti ma non variabili.

31

Datalog

Ci sono tre tipi di clausole di Horn:

- Fatti: clausole con un literal positivo e senza literal negativi. Ad esempio,
  { padre(mario, aldo) }
  è un fatto, che si scrive anche semplicemente come
  padre(mario,aldo).

- Regola: clausola con un literal positivo ed almeno uno negativo. Ad esempio:
  {$\neg$ genitore(X), padre(X), $\neg$ uomo(X) }
  è una regola, che si scrive anche come
  padre(X) :- genitore(X), uomo(x).
  padre(X) si dice testa della regola, e genitore(X), uomo(x) si dice corpo della regola

- Goal: clausola con un literal negativo e senza literal positivi. Ad esempio
  {$\neg$ padre(mario,X) }
  è un goal, che si scrive anche come
  ?- padre(mario,X).

32

Datalog

L'insieme dei simboli di predicato è partizionato in due insiemi: EPred e IPred.

- EPred contiene i predicati che corrispondono alle relazioni della base di dati
- IPred contiene i predicati che corrispondono a relazioni che vengono definite nel programma Datalog.

La base di Herbrand HB è l'insieme di tutte le clausole ground unit positive che si possono formare con i predicati e le costanti. Denotiamo con EHB il sottoinsieme della base di Herbrand formato da atomi i cui predicati appartengono ad EPred, e con IHB il sottoinsieme della base di Herbrand formato da atomi i cui predicati appartengono ad IPred.

33

Programmi Datalog

In Datalog una query si esprime mediante un programma.

Un programma Datalog P è un insieme finito di clausole di Horn che
- contiene al massimo un goal
- è tale che, per ogni C in P che non è il goal, si ha che C è una clusola Datalog, ovvero:
  - o C appartiene a EHB, cioè è una clausola ground unit positiva il cui predicato appartiene a EPred,
  - oppure C è una regola che soddisfa queste condizioni:
    - il predicato che compare nella testa di C appartiene a IPred
    - tutte le variabili che compaiono nella testa compaiono anche nel corpo (safety condition)

34

Datalog: esempio di programma

Siano:

EPred = { par, lives }
IPred = { anc, person }

Questo è un programma Datalog:

anc(X,Y) :- par(X,Y).
anc(X,Y) :- par(X,Z), anc(Z,Y).
anc(adamo,X) :- person(X).
person(X) :- lives(X,Y).

35

Datalog: semantica

Sia C una clausola. Una sostituzione per C è un insieme finito della forma
{ X_1 / t_1, … ,X_n / t_n }
tale che le varie X_i sono variabili distinte di C, ogni t_i è un termine, e X_i è diverso da t_i per ogni i.

Se lambda è una sostituzione, e C è costituita dai literal L1,…,Ln, allora lambda(C) denota la clausola ottenuta da C sostituendo ad ogni Li il literal lambda(Li) ottenuto applicando a Li la sostituzione lambda. In particolare, lambda(Li) si ottiene sostituendo simultaneamente ogni variabile Xj che occorre in Li con il termine tj se e solo se Xj / tj è un elemento di lambda.

Se per una coppia di literal <L,M> esiste una sostituzione lambda tale che lambda(L)=lambda(M), allora lambda si dice unifier per L e M. Una sostituzione gamma si dice più generale di una sostituzione lambda se esiste una sostituzione theta tale che gamma · theta = lambda, dove gamma · theta indica la composizione di theta e gamma.

Il most general unifier (MGU) di L e M è un unifier lambda tale che non esiste alcun altro unifier di L e M che è più generale di lambda.

36

Datalog: semantica

Per stabilire la semantica di Datalog potremmo ricorrere semplicemente alla logica del primo ordine, in particolare alle nozioni di interpetazione, di modello, e di implicazione logica, in modo classico. In realtà, vogliamo interpretare ogni programma Datalog come una query, e quindi come funzione. È quindi necessario definire la semantica in modo adeguato a questo scopo.

Definizione La semantica di un programma P senza goal è data dalla funzione

$$M_P : 2^{EHB} \rightarrow 2^{IHB}$$

(dove 2^S denota l'insieme di tutti i sottoinsieme di S) tale che per ogni sottoinsieme V di EHB:

$$M_P(V) = \{ G \mid G \in IHB \wedge (P \cup V) \models G \}$$

Definizione La semantica di un programma P con goal H è data dalla funzione

$$M_{P,H} : 2^{EHB} \rightarrow 2^{HB}$$

tale che per ogni sottoinsieme V di EHB, si ha che

$$M_{P,H}(V) = \{ G \text{ istanza di } H \mid G \in HB \wedge (P \cup V) \models G \}$$

Datalog: interpretazioni di Herbrand

Nella definizione della semantica appena vista si fa riferimento alla condizione

$$(P \cup V) \models G$$

Herbrand e Skolem hanno indipendentemente dimostrato che, sulla base del fatto che P è un insieme di clausole, per verificare la condizione suddetta non è necessario considerare tutte le possibili interpretazioni di $P \cup V$, ma è sufficiente limitare l'attenzione alle interpretazioni di Herbrand.

Una interpretazione di Herbrand è una interpretazione che assegna

- ad ogni costante se stessa
- ad ogni atomo ground appartenente ad HB un valore di verità.

Una interpretazione di Herbrand si può vedere come un insieme di atomi ground, quelli ai quali si assegna il valore "vero". Un modello di Herbrand di un insieme di clausole è una interpretazione di Herbrand che soddisfa tutte le clausole dell'insieme.

Teorema (Herbrand e Skolem) Se P e V sono insiemi di clausole, allora $(P \cup V) \models G$ se e solo se ogni modello di Herbrand di $(P \cup V)$ è anche un modello di G.

Datalog: l'insieme cons

Il fatto che sia sufficiente limitare l'attenzione alle interpretazioni di Herbrand non assicura ancora che abbiamo a disposizione un metodo per calcolare la risposta ad una query Datalog.

Fortunatamente, vedremo che ci viene in aiuto un teorema fondamentale, per il quale abbiamo bisogno di questa nozione: se S è un insieme di clausole Datalog, indichiamo con cons(S) l'insieme di tutti i fatti ground che sono conseguenze logiche di S:

$$cons(S) = \{ F \mid F \in HB \wedge S \models F \}$$

Ovviamente, per un programma Datalog P, ed un insieme di fatti V, si ha che

$$M_P(V) = cons(P \cup V) \cap IHB$$

Quindi, se sappiamo come calcolare cons(S), per un insieme finito S, allora sappiamo "calcolare" la semantica di un programma Datalog.

Datalog: modello minimo

Sia S un insieme di clausole datalog (cioè regole e fatti).

Teorema Se S è un insieme di clausole Datalog (cioè regole e fatti), allora cons(S) è uguale alla intersezione di tutti i modelli di Herbrand di S, ed è esso stesso un modello di Herbrand.

cons(S) si chiama modello minimo di Herbrand di S (o semplicemente modello minimo di S).

Resta a questo punto di vedere come si calcola il modello minimo di un programma Datalog. Ci sono diversi algoritmi per questo calcolo, e noi illustreremo quello più semplice.

Datalog: algoritmo di valutazione

```
sottoinsiemeDiHB Compute(program P)    // P programma Datalog senza goal
{  old := ∅; new := P;
   while new <> old {
       old := new; passo := ∅;
       for each regola R della forma L0 :- L1,…,Ln in P
           for each tupla <F1,…,Fn> di fatti ground di old {
               for i := 0 to n Ki := Li;
               for i := 1 to n {
                   lambda := MGU(Ki,Fi);
                   if lambda = NIL
                   then { K0 = NIL; exit-for }
                   else for j := 0 to n Kj := lambda(Kj)
               }
               if K0 <> NIL then passo := passo ∪ {K0};
           } // passo contiene i fatti dedotti con un passo di regole a partire da old
       new := new ∪ passo;
   }
   return tutti i fatti in new;
}
```

Teorema Se P è un programma Datalog senza goal, allora Compute(P) termina restituendo il modello minimo di P.

Complessità di Datalog

Complessità di Datalog:

- La complessità rispetto ai dati di Datalog è polinomiale.
- Per quanto riguarda la complessità combinata, il problema di valutare se una tupla è contenuta nel risultato della valutazione di un programma Datalog è EXPTIME-completo.

Espressività di Datalog

Teorema Esiste un programma Datalog che, per ogni base di dati **B**, calcola la chiusura transitiva di r, dove r è una relazione binaria in **B**.

Sia RA_m l'algebra relazionale senza l'operatore di differenza.

Teorema Datalog è più espressivo di Ra_m (quindi delle conjunctive query).
La dimostrazione procede mostrando che:

- ogni espressione di Ra_m può essere trasformata in modo equivalente in Datalog,
- la chiusura transitiva di una relazione binaria si può esprimere in Datalog, ma non in algebra relazionale.

Teorema Esiste una query che si può esprimere nell'algebra relazionale ma non in Datalog. Quindi Datalog non è più espressivo dell'algebra relazionale e non è più espressivo del calcolo relazionale con dichiarazioni di range.

43