Calcolo relazionale

- Fa riferimento ad una famiglia di linguaggi dichiarativi, basati sul calcolo dei predicati del primo ordine
- Specifica le proprietà del risultato piuttosto che indicare la procedura per calcolarlo
- Ne esistono versioni con potenzialità espressive diverse
  - calcolo relazionale su domini
  - calcolo su tuple con dichiarazioni di range
    (è la base di molti costrutti del linguaggio SQL)

Calcolo su domini

- Le espressioni del calcolo sui domini hanno la forma

$$\{ A_1: x_1, \ldots, A_k: x_k \mid f(x_1,..,x_k) \}$$

  - $A_1,...,A_k$ sono attributi distinti (anche non nella base di dati)
  - $x_1,..,x_k$, sono variabili a valori nei corrispondenti domini
  - $f(x_1,..,x_k)$ è una formula logica (vale vero o falso)

  - $A_1: x_1, \ldots, A_k: x_k$ è detta Target list in quanto definisce la struttura del risultato
  - Il risultato è una relazione su $A_1,...,A_k$ che contiene le tuple i cui valori sostituiti a $x_1,..,x_k$ rendono vera la formula f

La formula f

- Formule atomiche
  - $R(A_1:x_1,\ldots,A_p:x_p)$ dove $R(A_1,..,A_p)$ è uno schema di relazione e $x_1,..,x_p$ sono variabili $\rightarrow$ è vera per valori di $x_1,..,x_p$ che formano una tupla di R
  - $x\vartheta y$ o $x\vartheta c$ con x,y variabili, c costante e ϑ operatore di confronto $(=,\geq,\leq,\neq,<,>)$
- Operatori logici
  - Se f_1 e f_2 sono formule allora sono formule
    - $f_1 \vee f_2$ (or)
    - $f_1 \wedge f_2$ (and)
    - $\neg f_1$ (not)

Quantificatori

- Quantificatore esistenziale
  - Data una formula f e una variabile x si definisce la formula

    $$\exists x\ (f)$$

  - La formula è vera se esiste almeno un valore a che sostituito alla variabile x rende vera f
- Quantificatore universale
  - Data una formula f e una variabile x si definisce la formula

    $$\forall x\ (f)$$

  - La formula è vera se per ogni possibile valore per la variabile x, la formula f è vera

Relazione fra ∀ e ∃

- Valgono le leggi di De Morgan

$$\forall x \neg P \equiv \neg \exists x \; P$$
$$\neg \forall x \; P \equiv \exists x \neg P$$
$$\forall x \; P \equiv \neg \exists x \neg P$$
$$\exists x \; P \equiv \neg \forall x \neg P$$

Ad esempio.....

Per ogni impiegato c non è capo
$\forall$ m $\neg$ Supervisione(Impiegato: m, Capo: c)

Non esiste nessun impiegato che ha c come capo
$\neg \exists$ m Supervisione(Impiegato: m, Capo: c)

- Quindi... uno solo dei due operatori è sufficiente...

Esempio [1]

- Si considera il seguente schema

Impiegati(Matricola,Nome,età,Stipendio)

Supervisione(Capo,Impiegato)

Q: *Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 1.700*

Target list

{Matricola: m, Nome: n, Età: e, Stipendio: s |

Impiegati(Matricola: m,Nome: n, Età: e, Stipendio: s) $\wedge$ (s>1700)}

condizione

.....in algebra relazionale $\sigma_{Stipendio>1700}$(Impiegati)

Esempio [2]

Q: *Trovare matricola, nome, età di tutti gli impiegati*

.....in algebra relazionale $\pi_{Matricola,Nome,Età}(Impiegati)$

Target list

{Matricola: m, Nome: n, Età: e |

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

condizione

... e anche...

Target list

{Matricola: m, Nome: n, Età: e |

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

condizione

Esempio [3]

Q: *Trovare la matricola dei capi degli impiegati che guadagnano più di 1.700*

.....in algebra relazionale

$\pi_{Capo}(Supervisione \bowtie_{Impiegato=Matricola}(\sigma_{Stipendio>1700}(Impiegati)))$

Target list

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

∧ Supervisione(Impiegato: m, Capo: c) ∧ s>1700 }

condizione

La variabile m comune alle due condizioni atomiche realizza la correlazione fra le tuple prevista dal join

Esempio [4]

Q: *Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 1700*

Target list

{NomeCapo: nc, StipCapo: sc |

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

s>1700 ∧

Supervisione(Impiegato: m, Capo: c) ∧

Impiegati(Matricola: c, Nome: nc, Età: ec, Stipendio: sc)}

Le variabili m e c realizzano due join

Esempio [5]

Q: *Trovare gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di entrambi*

{Matricola: m, Nome: n, Stipendio: s,

MatrCapo: c, NomeCapo: nc, StipCapo: sc |

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

Supervisione(Impiegato: m, Capo: c) ∧

Impiegati(Matricola: c, Nome: nc, Età: ec, Stipendio: sc) ∧

s>sc }

Variabili riferite a tuple diverse

Esempio [6]

Q: *Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 1700*

{Matricola: c, Nome: n |

Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s) ∧ — specifica tutti i capi

¬∃m'(∃n'(∃e'(∃s'(Impiegati(Matricola: m', Nome: n', Età: e',Stipendio: s') ∧

Supervisione(Impiegato: m', Capo: c) ∧ s'≤1700)))) }

Non esiste un Impiegato con capo c e stipendio inferiore a 1700

Esempio [7]

Q: *Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 1700*

{Matricola: c, Nome: n |

Impiegati(Matricola: c, Nome: n, Età: e, Stipendio: s) ∧ — specifica tutti i capi

∀m'(∀ n'(∀ e'(∀ s' (¬(Impiegati(Matricola: m', Nome: n', Età: e',Stipendio: s')

∧ Supervisione(Impiegato: m', Capo: c)) ∨ s'>1700)))) }

O c non ha impiegati dipendenti o l'impiegato guadagna più di 1700

Si ottiene dall'espressione precedente applicando le leggi di De Morgan

¬ ∃m (A ∧B ∧C) = ∀m ¬ (A ∧B ∧C) = ∀m (¬ (A ∧ B) ∨ ¬ C)

Pregi e difetti

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

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

queste espressioni sono dipendenti dal dominio: le risposte possibili dipendono dal dominio delle variabili. Se il dominio è infinito, la risposta è infinita....

Calcolo e algebra

- L'algebra relazionale è indipendente dal dominio
- Si considerano solo le espressioni del calcolo relazionale che sono indipendenti dal dominio
- 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)
- Dimostrazione costruttiva...

Calcolo su tuple con dichiarazioni di range

- Per superare le limitazioni del calcolo su domini
  - Riduzione delle variabili → le variabili rappresentano tuple
  - Dipendenza dal dominio → i valori provengono solo dalla base di dati
- Le espressioni del calcolo su tuple con dichiarazioni di range hanno la forma

  { target list | range list | f }

  - La target list è la lista degli obiettivi dell'interrogazione
  - La range list elenca le variabili libere della formula f
  - f è una formula con valore vero o falso

Target list

- La target list è composta da elementi del tipo
  - Y: *x*.Z
    - x è una variabile che rappresenta una tupla il cui range è una relazione definita su un insieme di attributi X
    - $Z \subseteq X$
    - Y è una lista di attributi e |Y|=|Z|
  - *x*.Z
    abbreviazione per Z: *x*.Z
  - *x*.*
    abbreviazione per X: *x*.X (si prelevano tutti gli attributi della tupla *x*)

Target list: esempio

- *i* è una variabile associata ad una tupla della relazione Impiegati(Matricola,Nome,Età,Stipendio)
  - *i*.*
    definisce come risultato una tupla che contiene tutti gli attributi di Impiegati, ovvero {Matricola,Nome,Età,Stipendio}
  - *i*.(Nome,Età)
    rappresenta una tupla definita sulla relazione con i due attributi {Nome,Età}
  - NomeCapo,StipCapo: *i*.(Nome,Stipendio)
    definisce una tupla con i due attributi {NomeCapo,StipCapo}

Range list

- Elenca le variabili libere della formula f con i relativi campi di variabilità (relazione di appartenenza R)

$$x(R)$$

Ad esempio...

- *i* (Impiegati)
  i può assumere i valori delle tuple contenute nella relazione Impiegati
- *i* (Impiegati), *s* (Supervisione)
- *i* (Impiegati), *s* (Supervisione), *i1* (Impiegati)

La formula f

- La formula f è costruita utilizzando
  - atomi
    - x.A ϑ c - confronto del valore dell'attributo A della tupla x con la costante c
    - $x_1.A_1 \ \vartheta \ x_2.A_2$ - confronto fra il valore dell'attributo A_1 della tupla x_1 e quello dell'attributo A_2 della tupla x_2
  - connettivi logici ($\wedge, \vee, \neg$)
  - quantificatori che associano un range alle variabili
    - $\exists$ x(R) (f)
      vero se esiste nella relazione R una tupla x che soddisfa f
    - $\forall$x(R) (f)
      vero se tutte le tuple x in R soddisfano f

Esempio [1]

Q: *Trovare matricola, nome, età e stipendio degli impiegati che guadagnano più di 1.700*

{ i.* | i(Impiegati) | i.Stipendio > 1700 }

Target list | Range list | condizione

Q: *Trovare matricola, nome, età di tutti gli impiegati*

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

Target list | Range list | condizione (vuota)

Esempio [2]

Q: *Trovare matricola dei capi degli impiegati che guadagnano più di 1.700*

Target list

Range list

{ s.Capo | i(Impiegati), s(Supervisione) |

i.Matricola = s.Impiegato ∧ i.Stipendio>1700 }

condizione di join

condizione di selezione

Esempio [3]

Q: *Trovare nome e stipendio dei capi degli impiegati che guadagnano più di 1700*

Target list

{NomeCapo, StipCapo: c.(Nome,Stipendio) |

Range list

c(Impiegati), s(Supervisione), i(Impiegati) |

c.Matricola = s.Capo ∧ s.Impiegato=i.Matricola

∧ i.Stipendio > 1700}

condizioni di join

condizione di selezione

Le due variabili c e i accedono ai valori della stessa relazione in modo indipendente

Esempio [4]

Q: *Trovare gli impiegati che guadagnano più del rispettivo capo, mostrando matricola, nome e stipendio di entrambi*

Target list

{i.(Matricola,Nome,Stipendio),

MatrCapo,NomeCapo,StipCapo: c.(Matricola,Nome,Stipendio) |

c(Impiegati), s(Supervisione), i(Impiegati) |

c.Matricola = s.Capo ∧ s.Impiegato = i.Matricola ∧

i.Stipendio > c.Stipendio}

Range list

condizioni di join

condizione di selezione

Esempio [5]

Q: *Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 1700*

Target list

{c.(Matricola, Nome) |

Range list

c(Impiegati), s(Supervisione) |

c.Matricola = s.Capo ∧

join per trovare la tupla del capo in Impiegati

∀ i(Impiegati) (∀ s1(Supervisione)

(¬(s.Capo=s1.Capo ∧ s1.Impiegato=i.Matricola) ∨ i.Stipendio > 1700))}

i e s1 non sono variabili libere nella formula, ma variano all'interno delle relazioni corrispondenti

Esempio [6]

Q: *Trovare matricola e nome dei capi i cui impiegati guadagnano tutti più di 1700*

Target list

Range list

{c.(Matricola, Nome) |

c(Impiegati), s(Supervisione) |

c.Matricola = s.Capo ∧

join per trovare la tupla del capo in Impiegati

¬ (∃ i(Impiegati) (∃ s1(Supervisione)

(s.Capo=s1.Capo ∧ s1.Impiegato=i.Matricola ∧ i.Stipendio ≤ 1700))))}

condizioni di join rispettivamente per trovare i capi in Supervisione e accedere alla tupla con le informazioni di ogni impegato dipendente

Limitazioni: unione

- Il calcolo su tuple con dichiarazioni di range non permette di esprimere l'unione di relazioni
  - I risultati sono costruiti a partire dalle variabili libere
  - Ogni variabile ha come range una sola relazione

es.

$$R_1(A) \cup R_2(A)$$

  - Se si ha una sola variabile libera si fa riferimento ad una sola relazione
  - Se si hanno due variabili si può far riferimento alle due relazioni ma non si riesce a combinarle per produrre il risultato in modo corretto

Operazioni insiemistiche

- SQL che è basato sul calcolo su tuple con dichiarazione di range prevede un operatore esplicito per l'unione
- Gli operatori di intersezione e differenza sono esprimibili
  - Intersezione

$$\{ x_1.* \mid x_1(r_1) \mid \exists\, x_2(r_2)\ (x_1.A_1 = x_2.A_2 \wedge \ldots . x_1.A_k = x_2.A_k)\}$$

preleva le tuple di r_1 che hanno una tupla uguale in r_2

  - Differenza

$$\{ x_1.* \mid x_1(r_1) \mid \neg\exists\, x_2(r_2)\ (x_1.A_1 = x_2.A_2 \wedge \ldots . x_1.A_k = x_2.A_k)\}$$

preleva le tuple di r_1 che non hanno una tupla uguale in r_2

Altri limiti

- calcolo di valori derivati
  - possiamo solo estrarre valori, non calcolarne di nuovi
  - calcoli di interesse
    - a livello di tupla o di singolo valore (conversioni, somme, ecc.)
    - su insiemi di tuple (somme, medie, ecc.)
  - In SQL ci sono estensioni ad hoc
- interrogazioni inerentemente ricorsive, come la chiusura transitiva
  - Soluzione esterna a SQL.. con programma...

Chiusura transitiva

- Accesso ricorsivo alla stessa relazione per un numero non predeterminato di volte
  - Esempio

Supervisione(Impiegato, Capo)

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

Impiegato	Capo
Paolo Bianchi	Gaia Belli
Gaia Belli	Mario Mori
Mario Mei	Filippo Verdi

Impiegato	Superiore
Paolo Bianchi	Gaia Belli
Paolo Bianchi	Mario Mori
Gaia Belli	Mario Mori
Mario Mei	Filippo Verdi

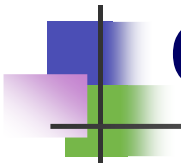
Soluzione col join?

- Se la gerarchia è a 2 livelli si può pensare a una soluzione con un join di Supervisione con se stessa

{Impiegato: i, Superiore: s |
Supervisione(Impiegato: i,Capo: s) ∨
(Supervisione(Impiegato: i,Capo: c) ∧
Supervisione(Impiegato: c, Capo: s)) }

join per trovare i capi dei capi

- Se si vuole estendere a gerarchie più profonde si devono aggiungere join a più termini



Conclusione

- 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!