

Basi di Dati

prof. Letizia Tanca

**Linguaggi di query
basati sulla
programmazione logica**

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):
 - niente simboli di funzione
 - modello di valutazione non procedurale (non SLD Resolution)

Un programma Datalog è un insieme di regole

- Ogni regola è composta da una testa (head o LHS) e da un corpo (body o RHS)

$$p \text{ :- } p_1, p_2, \dots, p_n$$

- Ogni p è chiamato *letterale*, ed è una *istanza* di un predicato così composto:
 - nome
 - lista di argomenti tra parentesi:
 - **costanti**
 - **variabili**
 - **simbolo “don’t care” (_) (non possono apparire nella testa)**
- **Safety:** le variabili del LHS devono apparire nel RHS

Alcune regole

- *Dipendente* (X, Y) :- *Persona*(X), *Assunto*(X, Y, D),
Ditta(Y), $D < \text{“9.11.2005”}$.
- *Parte_di*(X, Z) :- *Prodotto*(X), *Prodotto*(Y), *Prodotto*(Z),
Parte_di(X, Y), *Parte_di*(Y, Z).
- *Ospite*(X, Y) :- *Persona*(X), *Luogo_di_Soggiorno*(Y),
Dorme(X, Y).

Una regola può essere formata dalla sola testa:
Parte_di(a, b). Mi dice che a è parte del prodotto b
Queste particolari regole si chiamano *fatti*

Base di dati d'esempio per Datalog

GENITORI

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

Regole Datalog e BD

- Ciascuna tupla corrisponde a un **fatto di base** (o *letterale ground*). Ad esempio:

Genitori("Carlo","Antonio").

rappresenta la prima tupla della relazione
GENITORI

- Esempio di regola:

Padre(X,Y):- Persona(X,_, 'M'), Genitori(X,Y).

Unificazione

- **Interpretazione** della regola:

Padre(X,Y) :- Persona(X,_, 'M'), Genitori(X,Y).

- *LHS è vero se RHS è vero*
- *RHS è vero se, per ogni letterale di RHS, tutte le sue variabili sono unificabili, ovvero sostituibili, con valori costanti che rendono vero il letterale*

- **Persona(X,_, 'M')**: le unificazioni possibili per *X* sono:

{“Carlo”, “Antonio”, “Gianni”, “Andrea”}

- **Genitori(X, Y)**: le unificazioni possibili per la coppia *X, Y* sono: { (“Carlo”, “Antonio”), etc ...} **(l'intero contenuto di GENITORI)**

Interpretazione delle regole Datalog

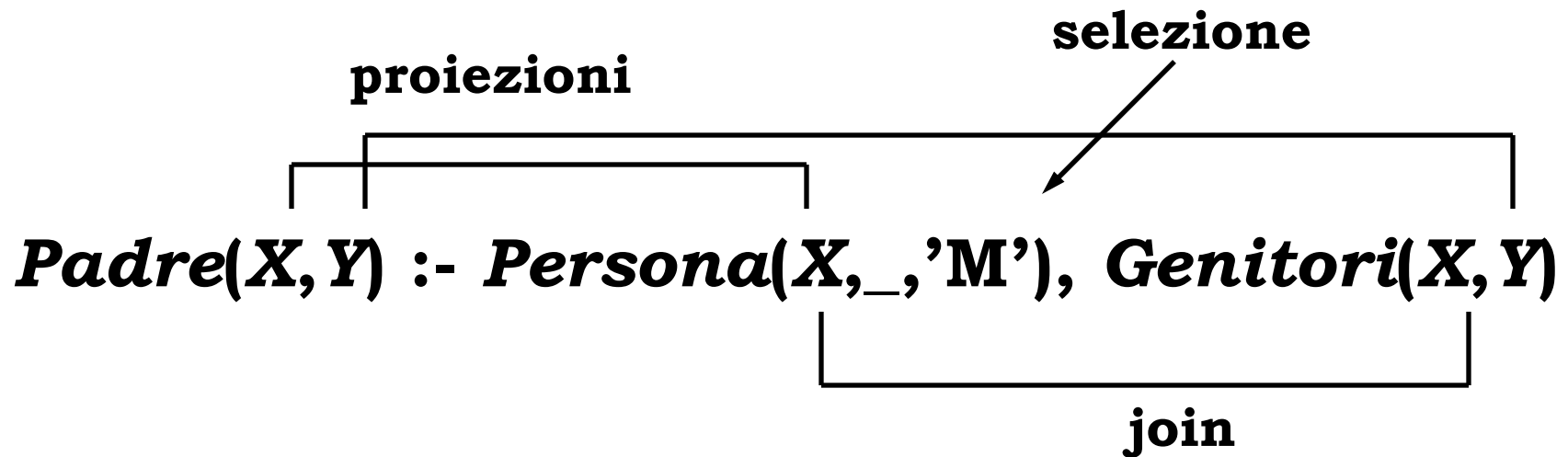
- Le sostituzioni per X che vanno bene sono

$\{ \text{“Carlo”}, \text{“Gianni”}, \text{“Antonio”} \}$

- Per le variabili X e Y si ottengono le seguenti coppie di sostituzioni

$\{ (\text{“Carlo”}, \text{“Antonio”}), (\text{“Carlo”}, \text{“Gianni”}), (\text{“Gianni”}, \text{“Andrea”}), (\text{“Antonio”}, \text{“Paola”}) \}$

Corrispondenza tra Datalog e l'algebra relazionale



- Espressione corrispondente in algebra relazionale:

$$\text{PADRE} = \Pi_{1,5} \sigma_{3 = 'M'} (\text{PERSONA} \bowtie_{1=1} \text{GENITORI})$$

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 $EDB \cap IDB = \emptyset$

Query in Datalog

• Le query sulla base di dati estensionale si esprimono come **“goal”** :

?- genitori(“Anna”,*X*)

• Valutazione: si cerca una tupla della relazione GENITORI e una sostituzione che unifichi con la variabile *X*

• Si ottiene $X = \{\text{“Antonio”}, \text{“Gianni”}\}$

NOTA: Un goal senza variabili restituisce *True* o *False*

• ?- genitori (“Anna”, “Antonio”) \Rightarrow *True*

• ?- genitori (“Anna”, “Andrea”) \Rightarrow *False*

Query in Datalog (II)

- Anche le query sull'intera base di dati (EDB + IDB) si esprimono come “**goal**”:

?- Padre(“Carlo”,*X*)

- Valutazione: si cerca una regola che definisca *Padre* e una sostituzione che unifichi con la variabile *X*
- Si ottiene $X = \{\text{“Antonio”}, \text{“Gianni”}\}$

NOTA: anche qui un goal senza variabili restituisce *True* o *False*

- ?- Padre(“Carlo”, “Antonio”) \Rightarrow *True*
- ?- Padre(“Carlo”, “Andrea”) \Rightarrow *False*

Query in Datalog (III)

- Le query sull'intera base di dati (EDB + IDB) si esprimono come “*goal*”:

?- Padre(“Carlo”,X)

questa query esprime una semplice selezione

- per esprimere una query complessa, occorre scrivere prima delle regole che definiscano un predicato intensionale complesso e poi un goal su quel predicato.
Esempio: **trovare tutti i fratelli di Carlo:**

Fratello(X,Y) :- Genitori(Z,X), Genitori(Z,Y), X≠Y. → definisco il concetto di “fratello”

?- Fratello(“Carlo”,X). → esprimo il goal su questo concetto

Altre regole sulla stessa BD

- $Madre(X, Y) :- Persona(X, _, 'F'), Genitori(X, Y).$
- $Nonno(X, Z) :- Genitori(X, Y), Genitori(Y, Z).$
- $Zio(X, Y) :- Persona(X, _, 'M'), Fratello(X, Z), Genitori(Z, Y).$
- $Fratello(X, Y) :- Genitori(Z, X), Genitori(Z, Y), X \neq Y.$

Potere espressivo di Datalog

- Datalog senza negazione permette di rappresentare gli operatori $\{\sigma, \Pi, \times, \cup\}$
- $\{\sigma, \Pi, \times\}$ già visti
- Per l'unione si usano più regole con la stessa testa; $P = R \cup S$:

$$P(X, Y) :- R(X, Y).$$

$$P(X, Y) :- S(X, Y).$$

- Per la differenza serve il *not*; $P = R - S$:

$$P(X, Y) :- R(X, Y), \neg S(X, Y)$$

- La negazione VA TENUTA SOTTO CONTROLLO

Esempio di schema di base di dati

PRESIDENTI (NOME-P, DATA-N, DATA-M, PARTITO, STATO, NOME-M)

CONGRESSI (# CONGRESSO, %S-REP, %C-REP, %S-DEM, %C-DEM)

AMMINISTRAZIONI (# AMMIN, DATA-IN, VICE-PRES, DATA-N-VP, NOME-P, DATA-N-P)

ELEZIONI (ANNO, VOTI-PRES, NOME-P, DATA-N-PRES, NOME-PERD, DATA-N-PERD, VOTI-PERD)

STATI (STATO, POPOLAZ, # AMMIN.)

PRESID-CONGR (NOME-P, DATA-N, # CONGR)

Interrogazioni

- ***Trovare l'anno di nascita del presidente J.F. Kennedy***
- ***Trovare gli anni in cui è stato eletto un presidente repubblicano proveniente dall'Illinois***
- ***Trovare i perdenti delle elezioni vinte da qualche presidente di nome Roosevelt***
- ***Trovare i nomi delle mogli dei presidenti provenienti dalla California eletti dopo il 1960***
- ***Trovare le persone che sono state presidenti OPPURE vicepresidenti in amministrazioni inaugurate dopo il 1880***
- ***Trovare le persone che sono state presidenti E ANCHE vicepresidenti in qualche amministrazione inaugurata dopo il 1880***
- ***Trovare le persone che sono state presidenti MA MAI vicepresidenti in amministrazioni inaugurate dopo il 1880***

Negazione in Datalog

- Alcuni letterali del corpo possono essere negati
- L'uso della negazione in Datalog aumenta il potere espressivo del linguaggio
- L'uso della negazione richiede cautela:

$q(X) :- \neg p(X).$

$p(0).$

.....e tutti gli altri sono “non P” ?

?- $q(X)$ produce un risultato infinito!

→ Ancora una volta si rischiano interrogazioni

UNSAFE

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 (immediata) presenta il letterale della testa all'interno del corpo della regola:

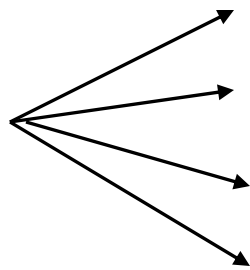
Antenato(X, Y) :- Genitore(X, Y).

Antenato(X, Y) :- Antenato(X, Z), Genitore(Z, Y).

Valutazione delle query ricorsive

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

**Nuovi elementi
rispetto a
GENITORE**



Basi di Dati

ANTENATO

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

Meccanismo di valutazione

Viene eseguito il seguente processo iterativo:
ad ogni iterazione *applico le regole e unisco il risultato al risultato precedente*

$$\mathbf{ANTENATO}^0 \Leftarrow \emptyset$$

$$\mathbf{ANTENATO}^1 \Leftarrow \mathbf{GENITORE}$$

$$\mathbf{ANTENATO}^2 \Leftarrow [(\Pi_{1,4} \mathbf{ANTENATO}^1 \triangleright \triangleleft_{2=1} \mathbf{GENITORE}) \cup \mathbf{GENITORE}] \cup \mathbf{ANTENATO}^1$$

$$\mathbf{ANTENATO}^3 \Leftarrow [(\Pi_{1,4} \mathbf{ANTENATO}^2 \triangleright \triangleleft_{2=1} \mathbf{GENITORE}) \cup \mathbf{GENITORE}] \cup \mathbf{ANTENATO}^2$$

.....

fino a quando $\mathbf{ANTENATO}^n$ risulta pari a
 $\mathbf{ANTENATO}^{n-1}$ (**punto fisso**)

Predicati e funzioni

- Nella definizione delle regole si possono usare predicati speciali
 - operatori di confronto:
= (unificazione), \neq , $<$, \leq , $>$, \geq
 - funzioni aritmetiche:
 $+$, $-$, $*$, \div
- Bisogna fare molta attenzione:
 $n(X) :- n(X - 1).$
 $n(0).$
?- $n(X)$ produce un risultato infinito!

Regole corrette

Le regole devono essere *ben formate*:

- La negazione deve essere *safe*:

tutte le variabili di un letterale negato devono comparire anche in un letterale positivo del corpo della regola: es. **$S(X) :- \neg R(X)$** non va bene

- La negazione deve essere *stratificata*:

in sintesi, non ci devono essere cicli di dipendenza tra letterali negati. Ad es.:

$P(X) :- R(X), S(Y, X).$

$R(Y) :- T(X, Y), \neg P(Y)$ non va bene

Problemi con la negazione nei programmi ricorsivi

- Se adottiamo in modo ingenuo la procedura di valutazione iterativa descritta prima, possiamo incontrare problemi

Es:

Antenato(X, Y) :- Genitore(X, Y).

Antenato(X, Y) :- Antenato(X, Z), Genitore(Z, Y).

*Non-parente (X, Y) :- Persona(X, _, _), Persona (Y, _, _),
¬ Antenato(X, Y).*

- Al primo passo di iterazione, calcoliamo NONPARENTE sottraendo il valore corrente della relazione ANTENATO (GENITORE) dal prodotto cartesiano delle persone
- Al secondo passo di iterazione, aggiungiamo al valore precedente di NONPARENTE il nuovo valore computato, CHE SARA' PIU' PICCOLO, etc
- Il risultato finale sarà scorretto, perché conterrà tutte le persone che non hanno tra loro un rapporto genitore-figlio

Problemi con la negazione nei programmi ricorsivi

- La soluzione sta nello **STRATIFICARE** il programma

Strato 1:

Antenato(X, Y) :- Genitore(X, Y).

Antenato(X, Y) :- Antenato(X, Z), Genitore(Z, Y).

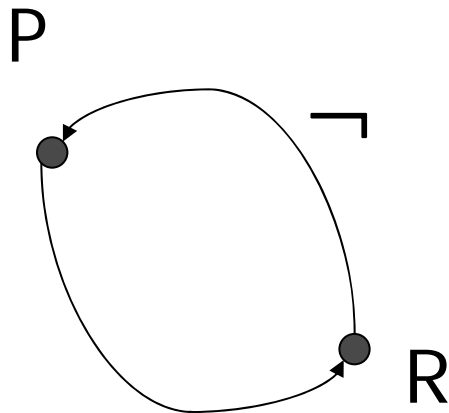
Strato 2:

*Non-parente (X, Y) :- Persona(X, _, _), Persona (Y, _, _),
¬ Antenato(X, Y).*

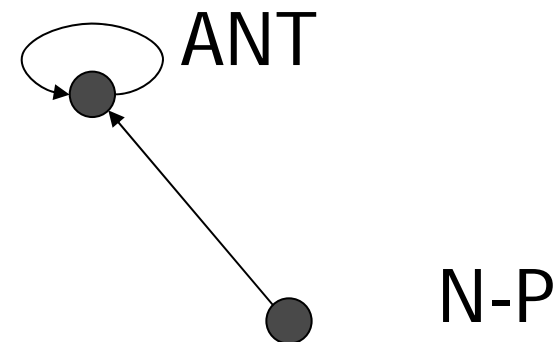
- Al livello dello strato 1, calcoliamo iterativamente la relazione **ANTENATO** come prima.
- Una volta raggiunto il punto fisso, calcoliamo la relazione **NONPARENTE** sottraendo il valore finale della relazione **ANTENATO** dal prodotto cartesiano delle persone

Problemi con la negazione nei programmi ricorsivi

- **Non tutti i programmi sono stratificabili**
- **Se il grafo delle dipendenze di un programma non presenta cicli allora il programma è stratificabile**



Non STRATIFICABILE



STRATIFICABILE

Usi di Datalog

Datalog costituisce anche un linguaggio per la definizione di viste e di vincoli

Esempi di vincoli:

incorrectdb1(X,Y) :- Madre(X,Z), Madre(Y,Z), X≠Y.

incorrectdb2(X) :- Antenato(X,X).

incorrectdb3() :-Genitori(X,Z),¬Persona(X,_,_).

Se includo le variabili nella testa posso ricavare gli oggetti che hanno causato la violazione

Riepilogo della terminologia Datalog

Relazione	→	<i>Predicato</i>
Attributo	→	<i>Argomento</i>
Tupla	→	<i>Fatto</i>
Vista	→	<i>Regola</i>
Interrogazione	→	<i>Goal</i>

Altri programmi logici ricorsivi

- **I cugini della stessa generazione sono i figli di cugini della stessa generazione**
sgc(X,X).
sgc(X,Y) :- genitore(Z,X), sgc(Z,W), genitore(W,Y).
- **Supervisori e subordinati**
supervisore(X,Y) :- sup_diretto(X,Y).
supervisore(X,Y) :- sup_diretto(X,Z), supervisore(Z,Y).

subordinato(X,Y) :- supervisore(Y,X).
incorrectdb1(X) :- supervisore(X,X).
incorrectdb2(X,Y,Z) :- sup_diretto(X,Y), sup_diretto(Z,Y), X ≠ Z.
- **Generazione della distinta base**
componente(X,Y):- parte_di(X,Y).
componente(X,Y):- parte_di(X,Z), componente(Z,Y).

Esercizi (A)

CLIENTE(CodiceFiscale, Cognome, Nome, DataNascita, Sesso, Nazionalità)

ALBERGO(PartitaIVA, NomeAlbergo, Località, NomeProprietario)

SERVIZI OFFERTI(PIVA-Albergo, NomeServizio)

PRENOTAZIONE(PIVA-Albergo, CF-Cliente, Datainizio, NCamera, Tariffagiornaliera, Datafine)

1. *Estrarre il nome e il cognome dei clienti che non hanno mai prenotato camere nell'albergo "Del Sole" di Torino.*
2. *Estrarre la partita IVA di ogni albergo il cui proprietario ha anche un albergo in una diversa località.*
3. *Estrarre i clienti che hanno prenotato un albergo di Torino per un periodo superiore a un mese oppure hanno prenotato un albergo che offre la sauna in una città qualsiasi.*

Esercizi (B)

LABORATORIO(Numero, NumeroPosti)

PRENOTAZIONE(NumeroLab, Data, OraInizio, OraFine, DocTitoloCorso, Esercitatore)

DOCENTE(NomeDocente, Dipartimento, Posizione, Materia, #prenot)

1. *Trovare i laboratori che sono stati prenotati da un docente ordinario di “Basi di dati”, ma che non hanno alcuna prenotazione relativa ad un docente del “Dipartimento di Ingegneria dei Sistemi”.*
2. *Trovare il nome e il dipartimento del docente che ha effettuato il maggior numero di prenotazioni*
3. *Trovare le informazioni dei laboratori che non sono mai stati prenotati né da un “professore ordinario” né da un esercitatore del dipartimento “DEI”*