



# **Basi di Dati**

## **Introduzione al**

### **Calcolo relazionale**

# Contenuti - Roadmap

Lab (progettazione)	Corso di Teoria	Lab (SQL)
<ul style="list-style-type: none"><li>• Metodologie e modello Entità Associazioni</li><li>• Progettazione concettuale e logica</li></ul>	<ul style="list-style-type: none"><li>• Modello Relazionale</li><li><li>• Algebra relazionale</li><li>• Ottimizzazione logica</li><li>• <b>Calcolo relazionale</b></li><li>• La normalizzazione</li><li>• Metodi di accesso e indici</li><li>• Gestione della concorrenza</li><li>• Gestione del ripristino</li></li></ul>	<ul style="list-style-type: none"><li>• Linguaggio SQL</li></ul>

# Base di dati «Ricoveri»

**pazienti**

<u>COD</u>	Cognome	Nome	Residenza	AnnoNascita
A102	Necchi	Luca	TO	1950
B372	Rossigni	Piero	NO	1940
B543	Missoni	Nadia	TO	1960
B444	Missoni	Luigi	VC	2000
S555	Rossetti	Gino	AT	2010

**reparti**

<u>COD</u>	Nome-Rep	Primario
A	Chirurgia	203
B	Pediatria	574
C	Medicina	530
L	Lab-Analisi	530
R	Radiologia	405

**ricoveri**

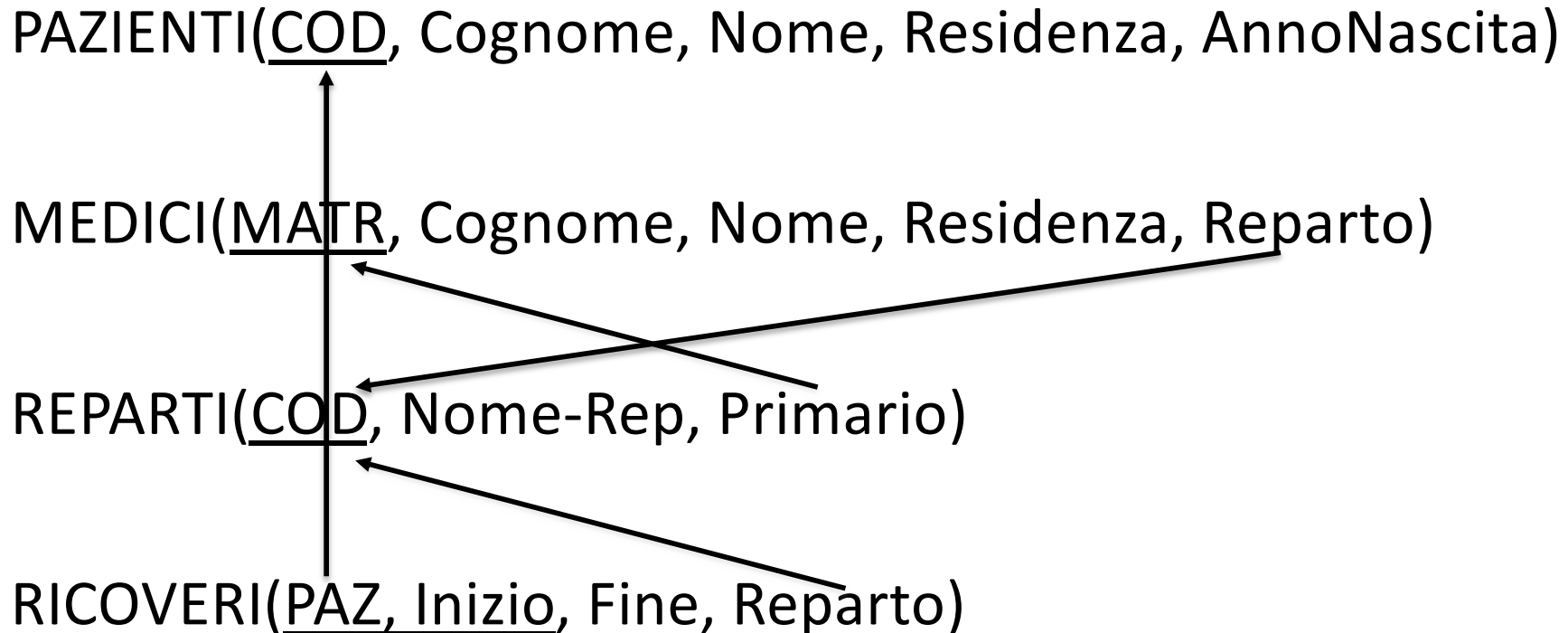
<u>PAZ</u>	Inizio	Fine	Reparto
A102	2/05/2014	9/05/2014	A
A102	2/12/2004	2/01/2005	A
S555	5/10/2014	3/12/2014	B
B444	1/12/2004	2/01/2005	B
S555	6/09/2015	1/11/2015	A

**medici**

<u>MATR</u>	Cognome	Nome	Residenza	Reparto
203	Neri	Piero	AL	A
574	Bisi	Mario	MI	B
461	Bargio	Sergio	TO	B
530	Belli	Nicola	TO	C
405	Mizzi	Nicola	AT	R
501	Monti	Mario	VC	A

# Base di dati «Ricoveri»

- Schema relazionale con vincoli di integrità referenziale



# Base di Dati "Impiegati"

**impiegati**

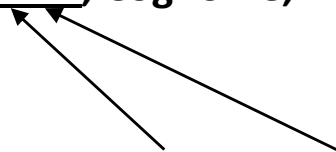
<u>MATR</u>	Cognome	Nome	Età	Stipendio
203	Neri	Piero	50	40
574	Bisi	Mario	60	60
461	Bargio	Sergio	30	61
530	Belli	Nicola	40	38
405	Mizzi	Nicola	55	60
501	Monti	Mario	25	35

**organigramma**


Capo	Impiegato
203	405
203	501
574	203
574	530
405	461

IMPIEGATI(MATR, Cognome, Nome, Età, Stipendio)

ORGANIGRAMMA(Capo, Impiegato)



# Altri linguaggi di interrogazione

- L'algebra relazionale è un linguaggio di tipo procedurale 
  - L'utente indica le operazioni da compiere per arrivare al risultato, il DBMS sceglie poi la strategia ottimale
- È possibile interrogare le basi di dati con un approccio dichiarativo fondato sulla logica: si specificano le proprietà del risultato anziché la procedura per generarlo
- Noi tratteremo solo il **calcolo relazionale su tuple con dichiarazione di range**, che è la base teorica di SQL
- Non tratteremo invece il **calcolo sui domini**

# Calcolo relazionale su tuple con dichiarazione di range

Si chiama...

- *calcolo su tuple* perché le variabili denotano **tuple** e
- *con dichiarazione di range* perché permette di specificare qual è il **range** di valori (cioè le relazioni) che le variabili possono assumere

# Calcolo relazionale su tuple con dichiarazione di range

Un'interrogazione è composta da tre parti:

$$\{ T \mid L \mid F \}$$

- *Target (T)*: specifica quali attributi compaiono nel risultato
- *Range list (L)*: specifica il dominio delle variabili libere (cioè non quantificate) in F
- *Formula (F)*: specifica una formula logica che il risultato deve soddisfare

**Il risultato di un'interrogazione è dato da:**

- **l'insieme dei valori degli attributi T**
- **presi dalle tuple nelle variabili in L**
- **che rispettano la formula in F**



# Esempio

Elencare il nome e il cognome dei pazienti residenti a Torino

*{ p.Nome, p.Cognome | p(Pazienti) | p.Residenza='TO' }*

## 2. Range list

La range list introduce variabili abbinate a relazioni di base con la seguente sintassi

*nome\_variabibile(Nome\_relazione\_di\_base)*

Ad es. in

*{ p.Nome, p.Cognome | p(Pazienti) | p.Residenza='TO' }*

*p(Pazienti)* significa che la variabile *p* assume valori nella relazione *Pazienti*, cioè è una (qualunque) tupla di *Pazienti*.

# 3. Formula

La formula è un predicato del primo ordine che vincola le variabili della range list. I predicati di base (atomi) sono del tipo:

- $x.A_i \varphi \text{ costante}$
- $x.A_i \varphi y.A_j$

dove  $x$  e  $y$  sono delle variabili,  $A_i$  e  $A_j$  sono degli attributi e  $\varphi$  è un operatore di confronto.

Il predicato è costruito con i soliti operatori:

- $\wedge$  (AND),  $\vee$  (OR),  $\neg$  (NOT),  $\Rightarrow$  (implicazione)

In più abbiamo i quantificatori universale e esistenziale che associano un range alle variabili non libere:

- $\forall$  (per ogni),  $\exists$  (esiste)

Ad es. in  $\{ p.Nome, p.Cognome \mid p(Pazienti) \mid p.Residenza='TO' \}$

$p.Residenza='TO'$  significa che, data una tupla  $p$  di *Pazienti*, perché questa faccia parte del risultato, l'attributo *Residenza* di  $p$  deve valere 'TO'.

# 1. Target list

La target list è l'elenco delle informazioni che voglio in uscita

Sintassi possibili:

- *variabile.Attributo1, variabile.Attributo2...*
- *variabile.(Attributo1,Attributo2)*
- *variabile.\** (restituisce tutti gli attributi)
- *Nome: variabile.Attributo*

**Le variabili usate nella target list devono essere dichiarate nella Range list.**

Ad es. in  $\{ p.Nome, p.Cognome \mid p(Pazienti) \mid p.Residenza='TO' \}$   
*p.Nome, p.Cognome* significa che nel risultato compariranno nome e cognome dei pazienti.

# Esempio di join

Elencare il nome, il cognome e la data del ricovero dei pazienti ricoverati nel reparto A

2. Devo usare due variabili:

- $L: p(Pazienti), r(Ricoveri)$

3. Mi serve il join tra pazienti e ricoveri...

- $F: p.COD=r.PAZ \dots$

3. ... e ho bisogno di aggiungere la condizione sui reparti

- $F: p.COD=r.PAZ \wedge r.Reparto='A'$

1. Estraggo cognome e nome del paziente e data del ricovero

- $T: p.Cognome, p.Nome, r.Inizio$

# Esempio

Elencare il nome, il cognome e la data del ricovero dei pazienti ricoverati nel reparto A

$$\{p.Cognome, p.Nome, r.Inizio \mid p(Pazienti), r(Ricoveri) \mid p.COD=r.PAZ \wedge r.Reparto='A'\}$$

# Variante con quantificazione esistenziale

Supponiamo che nell'interrogazione siano richieste solo informazioni sul paziente e nessuna informazione sul ricovero.

$$\{p.Cognome, p.Nome \mid p(Pazienti), r(Ricoveri) \mid p.COD=r.PAZ \wedge r.Reparto='A'\}$$

Posso cancellare dalla target list il riferimento a ricovero e riformulare l'interrogazione **utilizzando la quantificazione esistenziale** (e quindi eliminarla dalla range list):

$$\{p.Cognome, p.Nome \mid p(Pazienti) \mid \exists r(Ricoveri)(p.COD=r.PAZ \wedge r.Reparto='A')\}$$

# Sintassi della quantificazione universale ed esistenziale

- $\exists$  *variabile(Relazione)(formula)*
- $\forall$  *variabile(Relazione)(formula)*

**N.B.:** la formula è un predicato del primo ordine che può contenere sia *variabili libere che quantificate (vincolate)*. Tuttavia tutte le variabili libere presenti nella formula devono essere dichiarate nella range list.

**La target list, quindi, utilizza solo variabili libere.**



# Esempio

Elencare il nome e il cognome dei medici che curano il paziente A102

**L:**  $m(\text{Medici})$

**T:**  $m.\text{Cognome}, m.\text{Nome}$

**F:**  $\exists r(\text{Ricoveri})(r.\text{Reparto}=m.\text{Reparto} \wedge r.\text{PAZ}='A102')$

$\{m.\text{Cognome}, m.\text{Nome} \mid m(\text{Medici}) \mid$   
 $\exists r(\text{RICOVERI})(r.\text{Reparto}=m.\text{Reparto} \wedge r.\text{PAZ}='A102'))\}$

# Esempio

Elencare il nome e il cognome dei medici che curano il paziente Piero Rossini

**L:**  $m(\text{Medici})$

**T:**  $m.\text{Cognome}, m.\text{Nome}$

**F:**  $\exists p(\text{Pazienti})(\exists r(\text{Ricoveri})(p.\text{COD}=r.\text{PAZ} \wedge r.\text{Reparto}=m.\text{Reparto} \wedge p.\text{Nome}='Piero' \wedge p.\text{Cognome}='Rossini'))$

$$\{m.\text{Cognome}, m.\text{Nome} \mid m(\text{Medici}) \mid \exists p(\text{Pazienti})(\exists r(\text{Ricoveri})(p.\text{COD}=r.\text{PAZ} \wedge r.\text{Reparto}=m.\text{Reparto} \wedge p.\text{Nome}='Piero' \wedge p.\text{Cognome}='Rossini'))\}$$

# Esempio di intersezione

Elencare cognome e nome comuni a pazienti e medici

$$\{p.Cognome, p.Nome \mid p(Pazienti) \mid \\ \exists m(Medici)(m.Cognome=p.Cognome \wedge m.Nome=p.Nome)\}$$

o simmetricamente

$$\{m.Cognome, m.Nome \mid m(Medici) \mid \\ \exists p(Pazienti)(p.Cognome=m.Cognome \wedge p.Nome=m.Nome)\}$$

# Esempio con la negazione

Elencare i cognomi e nomi dei pazienti **non** comuni con i medici

È sufficiente anteporre il NOT alla quantificazione esistenziale

$$\{p.Cognome, p.Nome \mid p(Pazienti) \mid \\ \neg \exists m(\text{Medici})(m.Cognome=p.Cognome \wedge m.Nome=p.Nome)\}$$

# Esempio con la negazione

Elencare i cognomi e nomi dei pazienti **non** comuni con i medici

Alternativa applicando De Morgan:

$$\{p.Cognome, p.Nome \mid p(Pazienti) \mid \\ \forall m(Medici)(m.Cognome \neq p.Cognome \vee m.Nome \neq p.Nome)\}$$

# Esempio


Elencare i pazienti con almeno un ricovero in ogni reparto

$$\{p.* \mid p(\text{Pazienti}) \mid \forall r(\text{Reparti})(\exists r'(\text{Ricoveri})(r'.\text{Reparto}=r.\text{COD} \wedge r'.\text{PAZ}=p.\text{COD}))\}$$

**Spiegazione:** la variabile  $p$  assume, come valori, le tuple di *pazienti*. Se istanzio la variabile  $p$  su una ben precisa tupla, la devo restituire in output se è stato ricoverato **in ogni** reparto. Per ogni istanza di reparto in  $r$  **deve esistere** un ricovero di quel paziente nel reparto  $r$ .

# Esempio con doppia negazione

Neghiamo due volte la quantificazione universale


$$\{p.* \mid p(\text{Pazienti}) \mid \\ \neg\neg \forall r(\text{Reparti})(\exists r'(\text{Ricoveri})(r'.\text{Reparto}=r.\text{COD} \wedge r'.\text{PAZ}=p.\text{COD}))\}$$

$$\{p.* \mid p(\text{Pazienti}) \mid \\ \neg \exists r(\text{Reparti})(\neg \exists r'(\text{Ricoveri})(r'.\text{Reparto}=r.\text{COD} \wedge r'.\text{PAZ}=p.\text{COD}))\}$$

**In SQL l'interrogazione si esprime esattamente così!**

# Esempio con implicazione

Elencare i codici dei reparti in cui **ogni** medico che vi afferisce è residente a Torino.

Nota: non stiamo chiedendo che ogni medico della relazione Medici sia di Torino.

$$\{r.COD \mid r(Reparti) \mid \\ \forall m(\text{Medici})(m.Reparto=r.COD \Rightarrow m.Residenza='TO')\}$$

Quale sarebbe, invece, il risultato di

$$\{r.COD \mid r(Reparti) \mid \\ \forall m(\text{Medici})(m.Reparto=r.COD \wedge m.Residenza='TO')\}?$$

Stiamo chiedendo che, dato un reparto, tutte le tuple di Medici siano di medici che lavorano in quel reparto.



# Calcolo relazionale e SQL

Il calcolo relazionale con dichiarazione di range ha ispirato direttamente SQL

- $\{T | L | F\} \rightarrow \text{select... from... where...}$

La **target list** corrisponde alla **SELECT**

La **range list** corrisponde alla **FROM**

La **formula** corrisponde alla **WHERE**

# Esempio

Elencare i pazienti ricoverati due o più volte

$$\{p.* \mid p(\text{Pazienti}) \mid \\ \exists r'(\text{Ricoveri})(\exists r''(\text{Ricoveri})(p.COD=r'.PAZ \wedge p.COD=r''.PAZ \wedge \\ r'.Inizio \neq r''.Inizio))\}$$

# Esempio

Elencare i pazienti ricoverati una sola volta

È equivalente a:

Elencare i pazienti che sono stati ricoverati ma non...  
sono stati ricoverati due o più volte (quest'ultima è la  
query precedente)

$$\{p.* \mid p(\text{Pazienti}) \mid \\ \exists r(\text{Ricoveri}) (p.COD=r.PAZ \wedge \\ \neg \exists r'(\text{Ricoveri}) (\exists r''(\text{Ricoveri}) (p.COD=r'.PAZ \wedge p.COD=r''.PAZ \wedge \\ r'.Inizio \neq r''.Inizio)))\}$$

# Pattern di soluzione

Elencare i pazienti ricoverati una sola volta

Equivalente a: elencare i pazienti ricoverati (U) ma non ricoverati due o più volte (P)

**U** := pazienti ricoverati

**P** := pazienti ricoverati due o più volte

In algebra relazionale:  $R := U - P$

In calcolo relazionale:

$$\{p.* \mid p(Pazienti) \mid \\ formula\_U \wedge \neg formula\_P\}$$

# Esempio

Elencare i medici non primari

$$\{m.* \mid m(\text{Medici}) \mid \neg \exists r(\text{Reparti})(r.\text{Primario}=m.\text{MATR})\}$$

- In questo esempio, simile al precedente, pare manchi l'universo del discorso  $U$
- In realtà la parte  $U$  è definita implicitamente dalla variabile libera  $m$ : tutti i medici

# Prodotto cartesiano

Si rende semplicemente in questo modo

$$\{ x.^*, y.^* \mid x(R), y(S) \}$$

# Limitazioni

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

$$r_1 \cup r_2$$

- Infatti nella range list ogni variabile ha come dominio una sola relazione, mentre l'unione richiede che il risultato venga da una relazione o da un'altra
- Usando due variabili libere il risultato proviene da ognuna delle tue variabili (prodotto cartesiano)
- 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 relazionale sui domini

- Le variabili assumono come valori, anziché tuple, valori nei domini degli attributi
- Questo calcolo è universale, è cioè "completo" rispetto agli operatori dell'algebra relazionale
- Perché, allora, abbiamo introdotto il calcolo relazionale su tuple con dichiarazione di range?



# Motivazioni

1. Il calcolo relazionale con dichiarazione di range ha ispirato direttamente l'SQL
  - $\{T \mid L \mid F\} \rightarrow \text{select... from... where...}$
  - l'unione in SQL non è contenuta nel costrutto "select... from... where", ma è espressa al di fuori del costrutto
2. Trattare il calcolo relazionale sui domini in modo adeguato richiede lo sviluppo di un impianto logico che richiederebbe troppo tempo
  - formule safe

# Esempio

Riconsideriamo questo esempio

S= studenti, E = esami, P = Piano di studi

S

MATR	Nome
1	Rossi
2	Verdi
3	Bianchi

E

MATR	Corso
2	Programmazione
3	Algebra
2	Basi di dati
3	Programmazione
2	Algebra

P

<u>Corso</u>
Programmazione
Basi di dati
Algebra

Elencare gli studenti che hanno sostenuto tutti gli esami

# Esempio

Elencare gli studenti che hanno sostenuto tutti gli esami

$$\{ s.* \mid s(S) \mid \forall p(P)(\exists e(E)(e.MATR=s.MATR \wedge e.Corso=p.Corso)) \}$$

# Variante all'esempio

Variamo leggermente l'esempio

S= studenti, E = esami, O = Offerta formativa

S

<u>MATR</u>	Nome	Indirizzo
1	Rossi	Reti
2	Verdi	Sistemi
3	Bianchi	Reti

E

MATR	Corso	Indirizzo
2	Programmazione	Sistemi
3	Algebra	Sistemi
2	Basi di dati	Sistemi
3	Programmazione	Reti
2	Algebra	Sistemi

O

Corso	Indirizzo
Programmazione	Sistemi
Basi di dati	Sistemi
Programmazione	Reti
Basi di dati	Reti
Algebra	Sistemi

# Pattern di soluzione

Elencare gli studenti che hanno superato tutti gli esami del loro indirizzo

Devo introdurre l'implicazione  $\alpha \Rightarrow \beta$ , equivalente a  $(\neg\alpha \vee \beta)$

$$\{s.* \mid s(S) \mid \forall o(O)(o.Indirizzo=s.Indirizzo \Rightarrow \exists e(E)(e.MATR=s.MATR \wedge e.Corso=o.Corso \wedge e.Indirizzo=o.Indirizzo))\}$$

# Pattern di soluzione

Elencare gli studenti che hanno superato tutti gli esami del loro indirizzo

$$\{s.* \mid s(S) \mid \forall o(O)(o.Indirizzo=s.Indirizzo \Rightarrow \exists e(E)(e.MATR=s.MATR \wedge e.Corso=o.Corso \wedge e.Indirizzo=o.Indirizzo))\}$$

## Spiegazione:

- con la quantificazione universale scorro tutti i corsi dell'offerta formativa
- con l'antecedente dell'implicazione mi soffermo sui corsi dell'indirizzo dello studente
- con il conseguente dell'implicazione verifico che lo studente abbia superato l'esame

# Pattern di soluzione

Elencare gli studenti che hanno superato tutti gli esami del loro indirizzo

$$\{s.* \mid s(S) \mid \forall o(O)(o.Indirizzo=s.Indirizzo \Rightarrow \exists e(E)(e.MATR=s.MATR \wedge e.Corso=o.Corso \wedge e.Indirizzo=o.Indirizzo))\}$$

## Spiegazione:

- Se scorro corsi che non sono dell'indirizzo dello studente, l'antecedente è falso e l'implicazione rimane quindi vera (il quantificatore universale non viene falsificato)

# Esempio

## Impiegati

<u>MATR</u>	Cognome	Nome	Età	Stipendio
203	Neri	Piero	50	40
574	Bisi	Mario	60	60
461	Bargio	Sergio	30	61
530	Belli	Nicola	40	38
405	Mizzi	Nicola	55	60
501	Monti	Mario	25	35

IMPIEGATI(MATR, Cognome, Nome, Età, Stipendio)

ORGANIGRAMMA(Capo, Impiegato)



## Organigramma

Capo	Impiegato
203	405
203	501
574	203
574	530
405	461



# Esempio

Elencare i capi i cui subalterni guadagnano tutti più del capo

Possiamo usare la quantificazione universale e l'implicazione

# Esempio

Elencare i capi i cui subalterni guadagnano tutti più del capo

Possiamo usare le quantificazioni universale ed esistenziale e l'implicazione

$$\{i.* \mid i(\text{Impiegati}) \mid \forall i'(\text{Impiegati}) ( \\ (\exists o(\text{Organigramma})(o.\text{Capo}=i.\text{MATR} \wedge o.\text{Impiegato} = i'.\text{MATR}) ) \\ \Rightarrow i'.\text{Stipendio} > i.\text{Stipendio})\}$$

# Esempio

Elencare i capi i cui subalterni guadagnano tutti più del capo

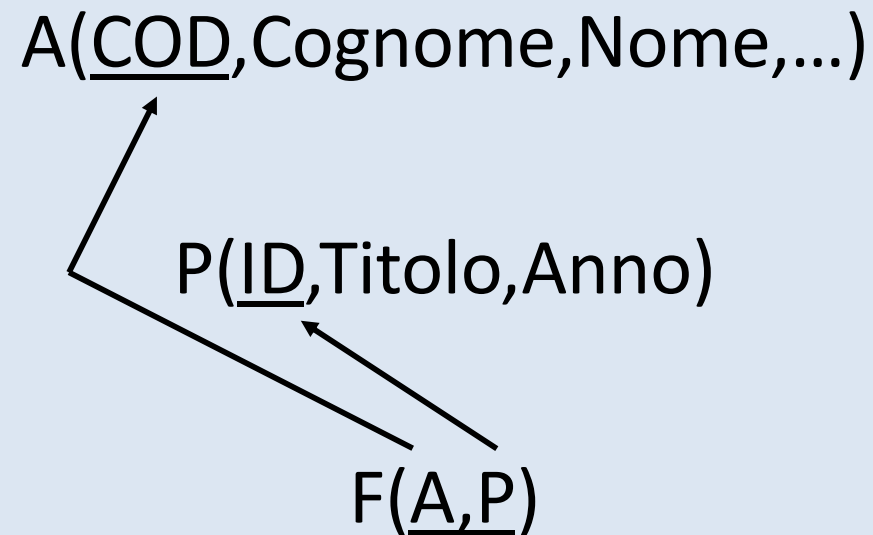
In algebra relazionale:

$$\begin{aligned} & \pi_{Capo}(organigramma) - \\ & \pi_{I1.Capo}(\sigma_{I2.Stipendio \leq I1.Stipendio}(\rho_{I1 \leftarrow impiegati}(impiegati) \\ & \quad \bowtie_{I1.MATR=Capo} organigramma \\ & \quad \bowtie_{I2.MATR=Impiegato} \rho_{I2 \leftarrow impiegati}(impiegati))) \end{aligned}$$

# Esempio

Autori (A), Pubblicazioni (P), Firma (F)

Con schemi:



# Esempio

Trovare gli autori che hanno pubblicato solo in collaborazione (cioè autori che non hanno mai pubblicato una pubblicazione in cui sono gli unici firmatari)

# Esempio

Trovare gli autori che hanno pubblicato solo in collaborazione (cioè autori che non hanno mai pubblicato una pubblicazione in cui sono gli unici firmatari)

$$\{a.* \mid a(A) \mid \forall f(F)(f.A=a.COD \Rightarrow \exists f'(F)(f'.P=f.P \wedge f'.A \neq f.A))\}$$

# Esempio

Trovare gli autori che hanno pubblicato solo in collaborazione  
(autori che non hanno mai pubblicato una pubblicazione in cui  
sono gli unici firmatari)

Confrontare con algebra relazionale:

$$\pi_A(U) - \pi_A(U - P) = \\ \pi_A(f) - \pi_A(f - (\pi_{f1.A, f1.P}(\rho_{F1 \leftarrow F}(f) \bowtie_{f1.P=f2.P \wedge f1.A \neq f2.A} \rho_{F2 \leftarrow F}(f))))$$

$U$ : Elenco delle firme

$P$ : Elenco delle firme in collaborazione

$U - P$ : Elenco delle firme **uniche**

$\pi_A(U - P)$ : Elenco degli autori che hanno **(anche)** firme uniche

$\pi_A(U) - \pi_A(U - P)$ : Elenco degli autori che hanno **solo** firme in  
collaborazione

# Considerazione finale

Con i formalismi dell'algebra relazionale e del calcolo relazionale, sono in grado di esprimere tutte le interrogazioni potenzialmente computabili?

No, perché manca la ricorsione!

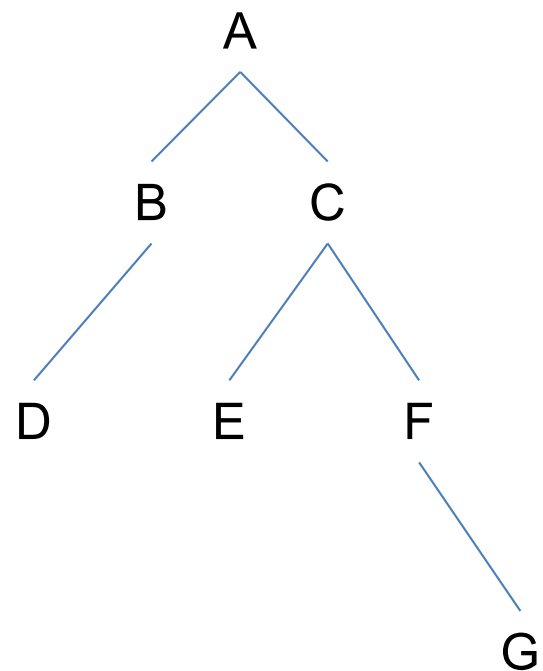


# Considerazione finale

Esempio: albero genealogico genitore-figlio

ag

Genitore	Figlio
A	C
A	B
B	D
C	E
C	F
F	G



# Considerazione finale

Relazione **nonno-nipote** facile da catturare:

$$\pi_{AG1.Genitore, AG2.Figlio}(\rho_{AG1 \leftarrow AG}(ag) \bowtie_{AG1.Figlio=AG2.Genitore} \rho_{AG2 \leftarrow AG}(ag))$$

- Con un altro join possiamo trovare i trisavoli
- In generale possiamo trovare un grado di parentela **purché il livello di profondità sia determinato a priori!**
- Le espressioni algebriche hanno quindi un numero di join che deve essere noto a priori

# Considerazione finale

Estrarre una relazione di questo tipo

*discendente(Persona1, Persona2)*

che contiene tutti i rapporti di discendenza, ovvero:

G	F
A	C
A	B
B	D
C	E
C	F
F	G



Persona1	Persona2
A	C
A	B
A	D
A	E
A	F
A	G
B	D
C	E
C	F
C	G
F	G

# Considerazione finale

Estrarre una relazione di questo tipo

*discendente(Persona1, Persona2)*

che contiene tutti i rapporti di discendenza, ovvero:

- Questa interrogazione non è esprimibile né algebra relazionale né in calcolo relazionale in quanto richiede un impianto ricorsivo
- Si tratta di una chiusura transitiva

# Considerazione finale

Estrarre una relazione di questo tipo

*discendente(Persona1,Persona2)*

che contiene tutti i rapporti di discendenza, ovvero:

- Le ultime versioni di SQL implementano una forma di **ricorsione attraverso le chiusure transitive**