Basi di Dati
Ottimizzazione del DBMS

Corso B

Base dati di esempio

PAZIENTI

COD	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

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

RICOVERI

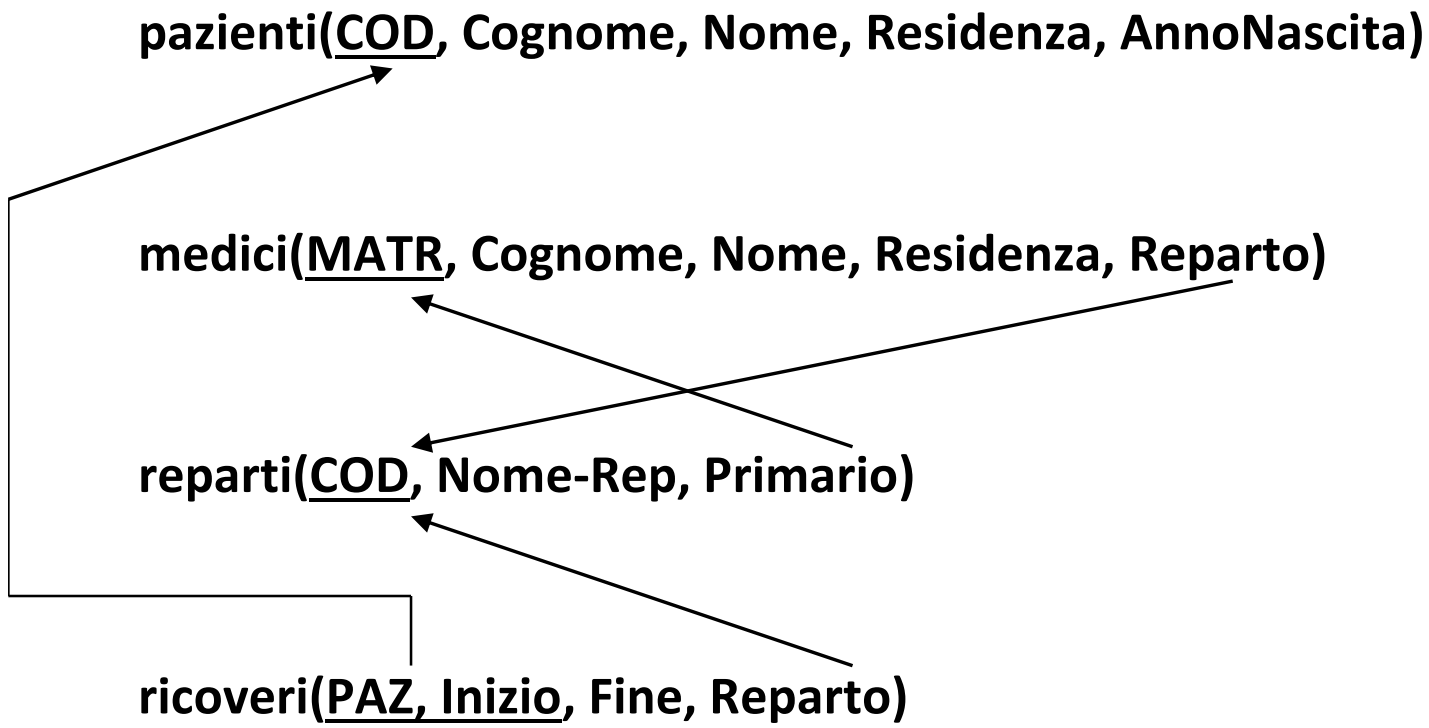
PAZ	Inizio	Fine	Reparto
A102	2/05/2004	9/05/2004	A
A102	2/12/2004	2/01/2005	A
S555	5/10/2014	3/12/2014	B
B444	1/12/2004	2/01/2005	B
S555	6/09/2015	1/11/2015	A

MEDICI

MATR	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 dati di esempio

Schema relazionale con vincoli di integrità referenziali:



Base di Dati "Impiegati"

IMPIEGATI

MATR	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)

Ottimizzazione delle interrogazioni

- Ottimizzare: fare in modo che l'implementazione dell'interrogazione risulti più veloce possibile

- Ci occuperemo di ottimizzazione in tempo (più che sull'ottimizzazione in memoria/spazio)

Contesto

- L'applicazione gira in memoria centrale e dialoga attraverso il DBMS
- Il DBMS dialoga con le periferiche di storage attraverso il gestore del buffer (gestisce pagine di memoria)
- Le pagine sono mappate sul dispositivo di storage
- Un'attività (processo) in memoria centrale lavora con tempi dell'ordine dei nanosecondi (10^{-9} secondi)

Contesto

- Quando l'applicazione ha bisogno di una tupla chiede al DBMS di fornirgliela
- **Il DBMS deve trasferire una pagina dalla periferica di storage alla memoria centrale (buffer)**
- Quando la pagina con la tupla richiesta è nel buffer, l'applicazione può elaborarla

Contesto

- **Il DBMS deve trasferire una pagina dalla periferica di storage alla memoria centrale (buffer)**

L'ordine di grandezza di questa operazione è il millisecondo (10^{-3} secondi)

Confrontato con i tempi di lavoro del processo in memoria principale, è **un milione di volte** più lenta!

Contesto

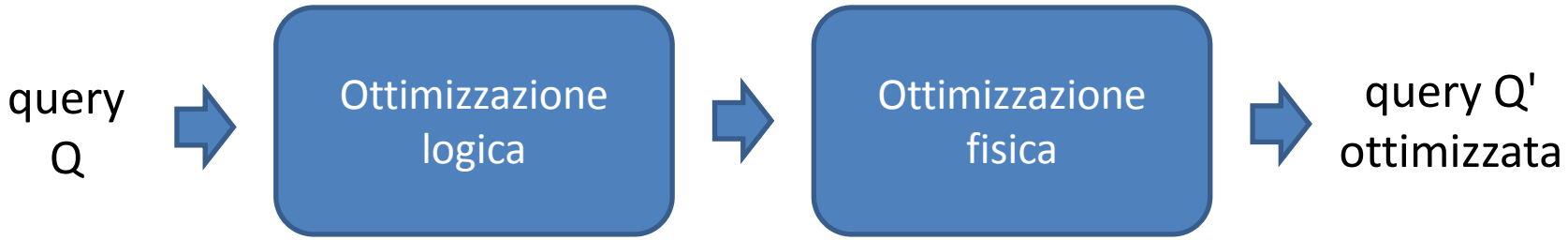
La complessità delle attività "algoritmiche" del DBMS in memoria principale è minima (solo loop, confronti, copia e trasferimento di dati)

Se i DBMS vogliono minimizzare i tempi di risposta devono minimizzare il trasferimento di pagine

L'obiettivo dell'ottimizzatore del DBMS è di minimizzare il trasferimento di pagine da e verso la memoria centrale

Ottimizzazione in due fasi

Tutti i DBMS realizzano l'ottimizzazione dell'interrogazione Q (espressa in algebra relazionale) in due fasi:



Q' è il piano operativo per l'esecuzione dell'interrogazione

Ottimizzazione logica

L'ottimizzazione logica è indipendente dalle strutture di memorizzazione

L'ottimizzazione logica prende in input l'albero di parsificazione (albero sintattico) scritto dall'utente e lo trasforma **sfruttando le proprietà dell'algebra relazionale**

L'albero in uscita è perfettamente equivalente all'interrogazione originale

Ottimizzazione fisica

- L'albero prodotto in output dall'ottimizzatore logico, diventa l'input dell'ottimizzatore fisico
- L'ottimizzatore fisico prenderà in considerazione le strutture interne di memorizzazione
- L'ottimizzatore fisico entra nei nodi (operatori) dell'albero di parsificazione, li esamina e in base alle strutture fisiche sceglie l'algoritmo ottimale per eseguirli
- Alla fine l'albero di parsificazione avrà, come nodi, gli algoritmi più adatti per l'esecuzione ottimale dei nodi

Esempio di albero

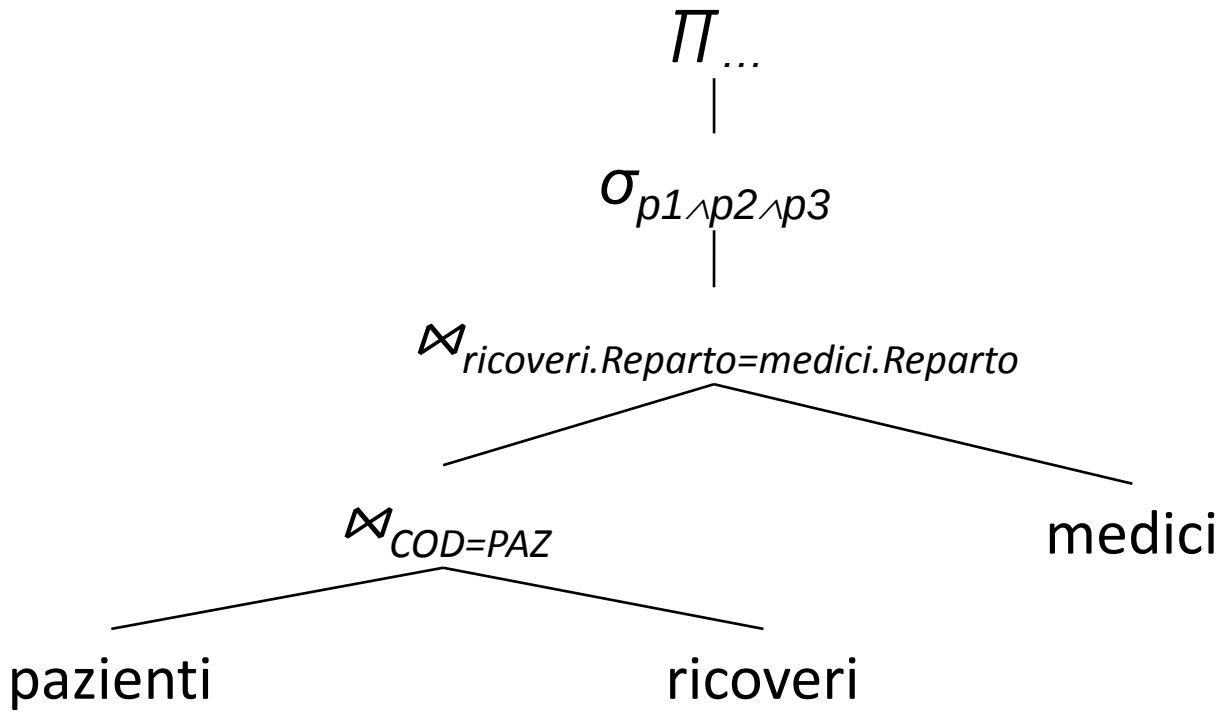
$$\Pi_{\ldots}(\sigma_{pazienti.Residenza='TO' \wedge Inizio='2010' \wedge MATR='405'}((pazienti \bowtie_{COD=PAZ} ricoveri) \bowtie_{ricoveri.Reparto=medici.Reparto} medici))$$

Chiamo

- *p1= pazienti.Residenza='TO'*
- *p2=Inizio='2010'*
- *p3=MATR='405'*

Esempio di albero

$$\Pi_{\ldots}(\sigma_{p1 \wedge p2 \wedge p3}((\text{pazienti} \bowtie_{COD=PAZ} \text{ricoveri}) \bowtie_{\text{ricoveri.Reparto=medici.Reparto}} \text{medici}))$$



Ottimizzazione logica

L'ottimizzatore è guidato da un semplice principio euristico:

- L'interrogazione da un punto di vista logico coinvolge una massa di tuple notevole (pensiamo al join)
- L'ottimizzatore cerca di ridurre il più drasticamente possibile il numero di tuple coinvolte dall'interrogazione

Obiettivo: ridurre la massa di tuple concettualmente coinvolte dall'interrogazione

Ottimizzazione logica

- Si è dimostrato nei fatti che quando l'ottimizzatore fisico deve lavorare sui dati, produce i risultati migliori se riceve in input un albero di parsificazione che ha ridotto il numero di tuple concettualmente coinvolte dall'interrogazione
- L'influenza positiva dell'ottimizzazione logica si riscontra nelle grandi applicazioni gestionali dei sistemi informativi (la maggioranza delle applicazioni su basi di dati)
- In generale, però, l'euristica non è sempre valida

Euristica dell'ottimizzatore

Quasi tutte le operazioni principali dei sistemi informativi che usiamo quotidianamente sono mirate a lavorare su pochissimi dati:

- lavoro sul mio piano di studi, non su tutti
- lavoro sui miei pazienti, non su tutti
- lavoro sulle operazioni del mio conto in banca, non su tutte

Entra quindi in gioco la **selezione**

Euristica dell'ottimizzazione

Dato che nelle applicazioni dei sistemi informativi l'attenzione è su un numero limitato di tuple, posso fare in modo che la selezione sia uno dei primi operatori da eseguire, riducendo subito il numero di tuple coinvolte?

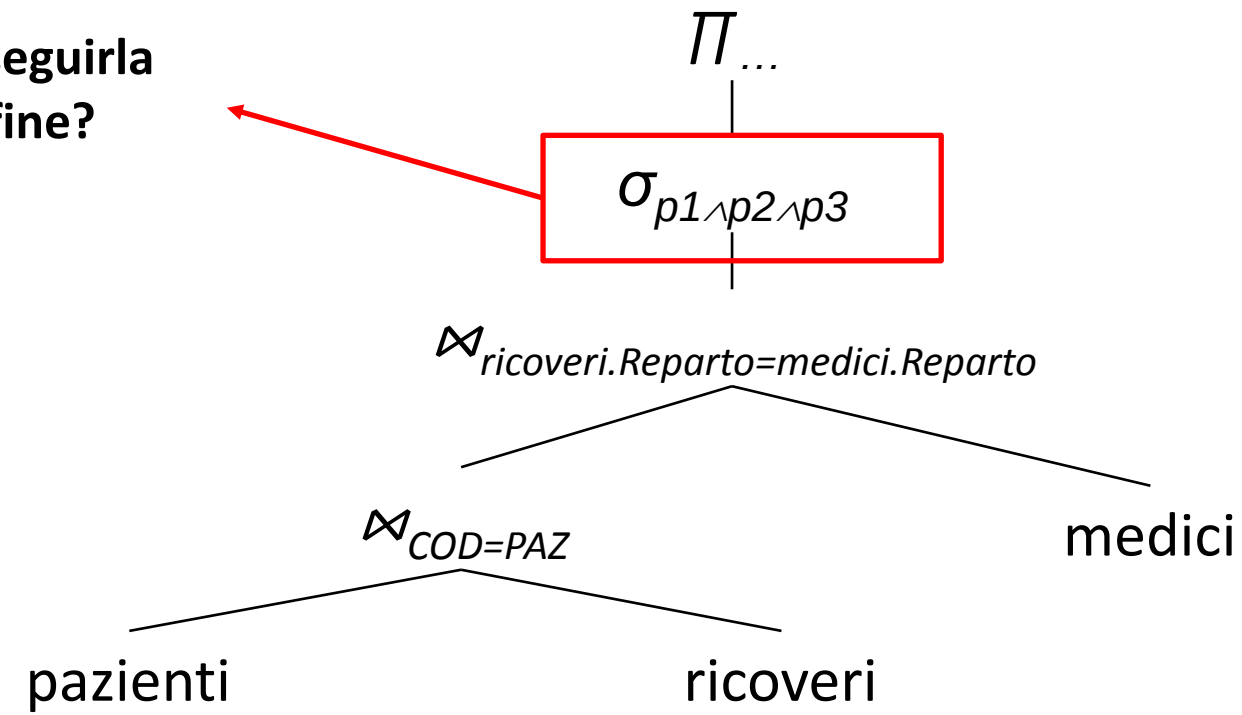
Posso farlo grazie alle **proprietà distributive della selezione**!

L'ottimizzatore cerca, con un algoritmo, di applicare questa semplice euristica

Esempio di albero

$$\Pi_{\ldots}(\sigma_{p1 \wedge p2 \wedge p3}((\text{pazienti} \bowtie_{COD=PAZ} \text{ricoveri}) \bowtie_{\text{ricoveri.Reparto=medici.Reparto}} \text{medici}))$$

Perché eseguirla solo alla fine?



Algoritmo di ottimizzazione logica

I predicati *p* della selezione sono in forma congiuntiva (non è una limitazione: qualsiasi predicato si può ricondurre alla forma congiuntiva con De Morgan)

1. Decomposizione degli AND
   - $\sigma_{p1 \wedge p2}(r) \rightarrow \sigma_{p1}(\sigma_{p2}(r))$

2. Trasferire le selezioni verso le foglie finché è possibile con le proprietà distributive della selezione

3. Trasferire le proiezioni verso le foglie finché è possibile con le proprietà distributive della proiezione

Algoritmo di ottimizzazione logica

4. Ricondurre ad un'unica selezione le selezioni multiple
   - $\sigma_{p1}(\sigma_{p2}(r)) \rightarrow \sigma_{p1 \wedge p2}(r)$

5. Riconoscere le sequenze di join
   - $\sigma_{\theta}(r \times s) \rightarrow r \bowtie_{\theta} s$

6. Ricondurre ad un'unica proiezione le proiezioni multiple
   - $\Pi_X \Pi_{X,Y}(r) \rightarrow \Pi_X(r)$

7. Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)

Spiegazione del passo 1

I predicati *p* della selezione sono in forma congiuntiva (non è una limitazione: qualsiasi predicato si può ricondurre alla forma congiuntiva con De Morgan)

1. **Decomposizione degli AND**
   - $\sigma_{p1 \wedge p2}(r) \rightarrow \sigma_{p1}(\sigma_{p2}(r))$

2. Trasferire le selezioni verso le foglie finché è possibile con le proprietà distributive della selezione

3. Trasferire le proiezioni verso le foglie finché è possibile con le proprietà distributive della proiezione

Spiegazione del passo 1

Vediamo la proprietà distributiva della selezione rispetto al join:

$$\sigma_p(r(A) \bowtie_\theta s(B)) \rightarrow \sigma_p(r(A)) \bowtie_\theta s(B)$$

Valida **solo se** gli attributi coinvolti da *p* sono contenuti nello schema *A* (o nello schema *B*)

Ma, ad una selezione con un predicato che pesca da più relazioni difficilmente può essere applicata la proprietà distributiva

Spiegazione del passo 1

Infatti, nell'esempio ho

$$\sigma_{p1 \wedge p2 \wedge p3}$$

con

*p1= pazienti.Residenza='TO' (che pesca da **pazienti**)*

*p2=Inizio='2010' (che pesca da **ricoveri**)*

*p3=MATR='405' (che pesca da **medici**)*

Questo predicato non consente di muovere la selezione da dov'è sfruttando la proprietà distributiva della selezione rispetto al join

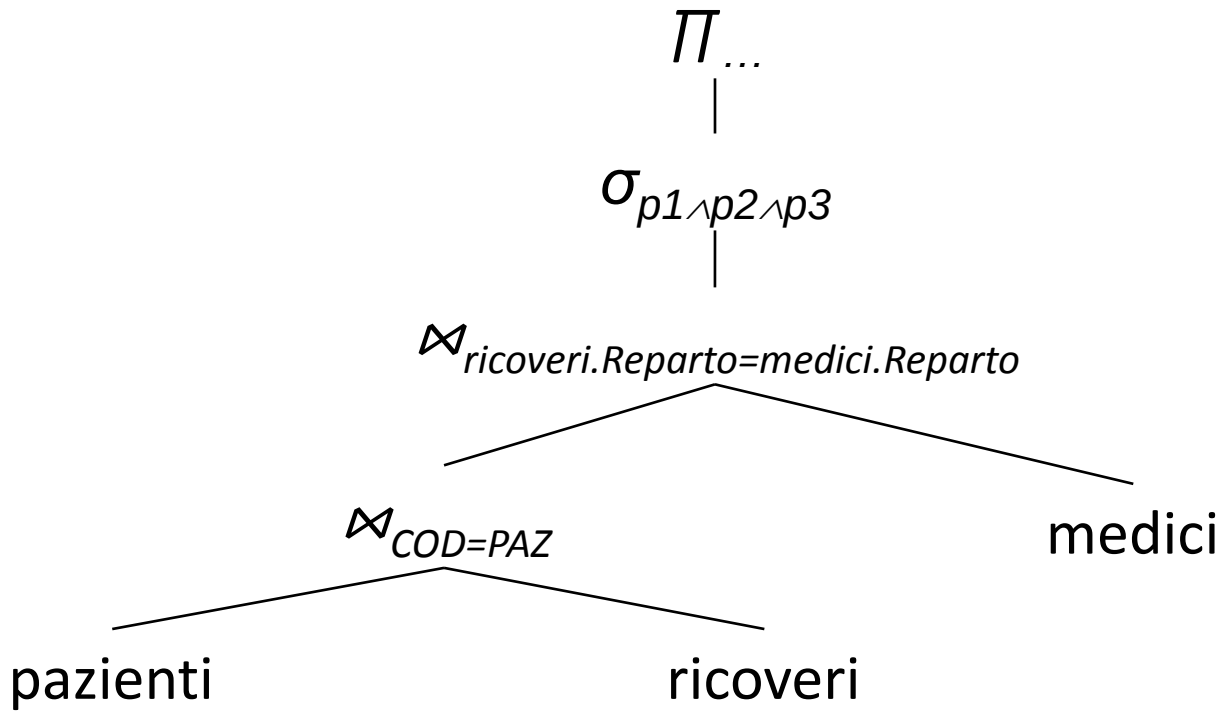
Spiegazione del passo 1

Ma se la spezzo in

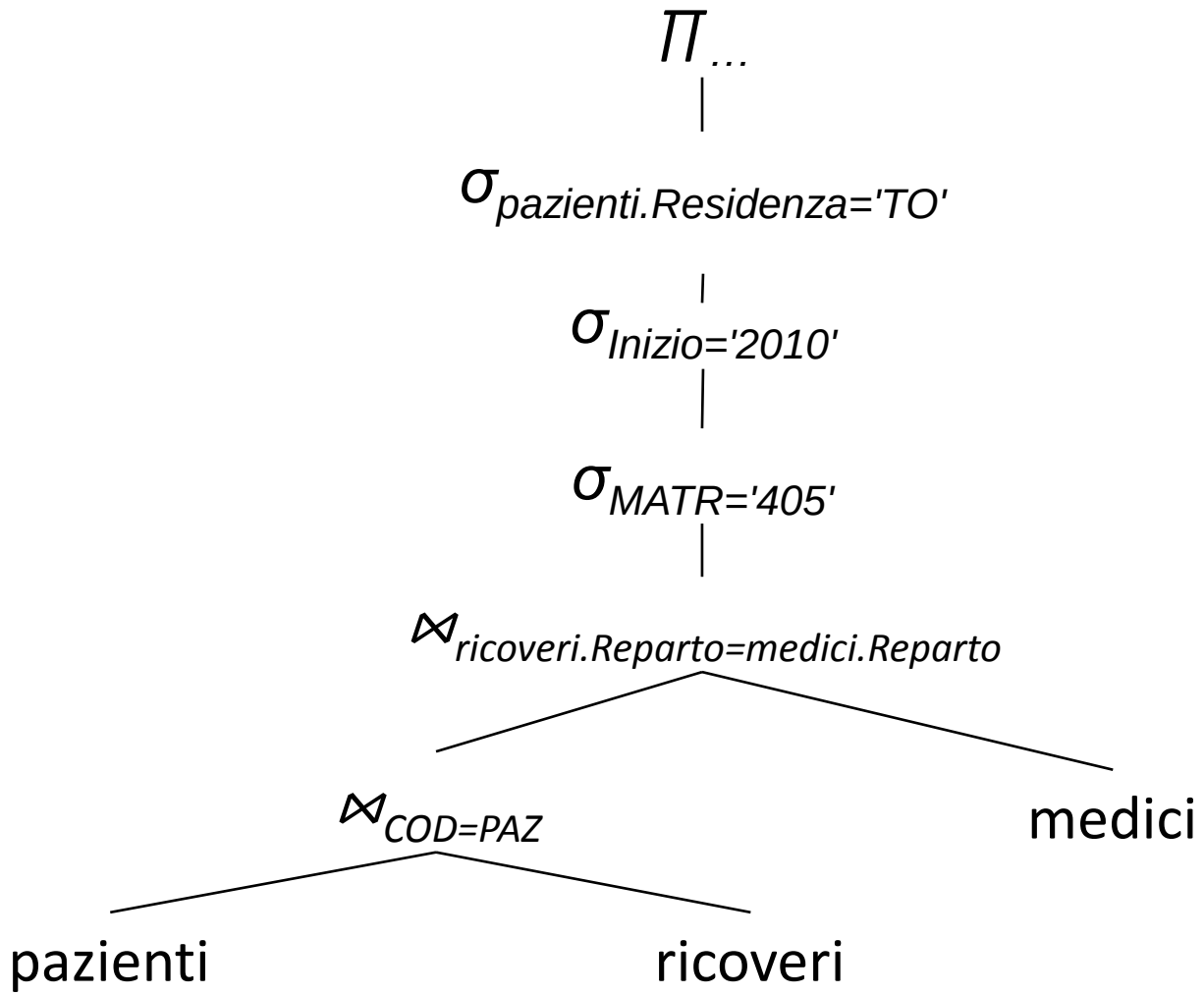
$$\sigma_{p1}\sigma_{p2}\sigma_{p3}$$

i singoli predicati sono più semplici e aumento la probabilità che essi soddisfino la condizione necessaria all'applicazione della proprietà distributiva

Esecuzione del passo 1



Esecuzione del passo 1



Spiegazione del passo 2

I predicati *p* della selezione sono in forma congiuntiva (non è una limitazione: qualsiasi predicato si può ricondurre alla forma congiuntiva con De Morgan)

1. Decomposizione degli AND
   - $\sigma_{p1 \wedge p2}(r) \rightarrow \sigma_{p1}(\sigma_{p2}(r))$

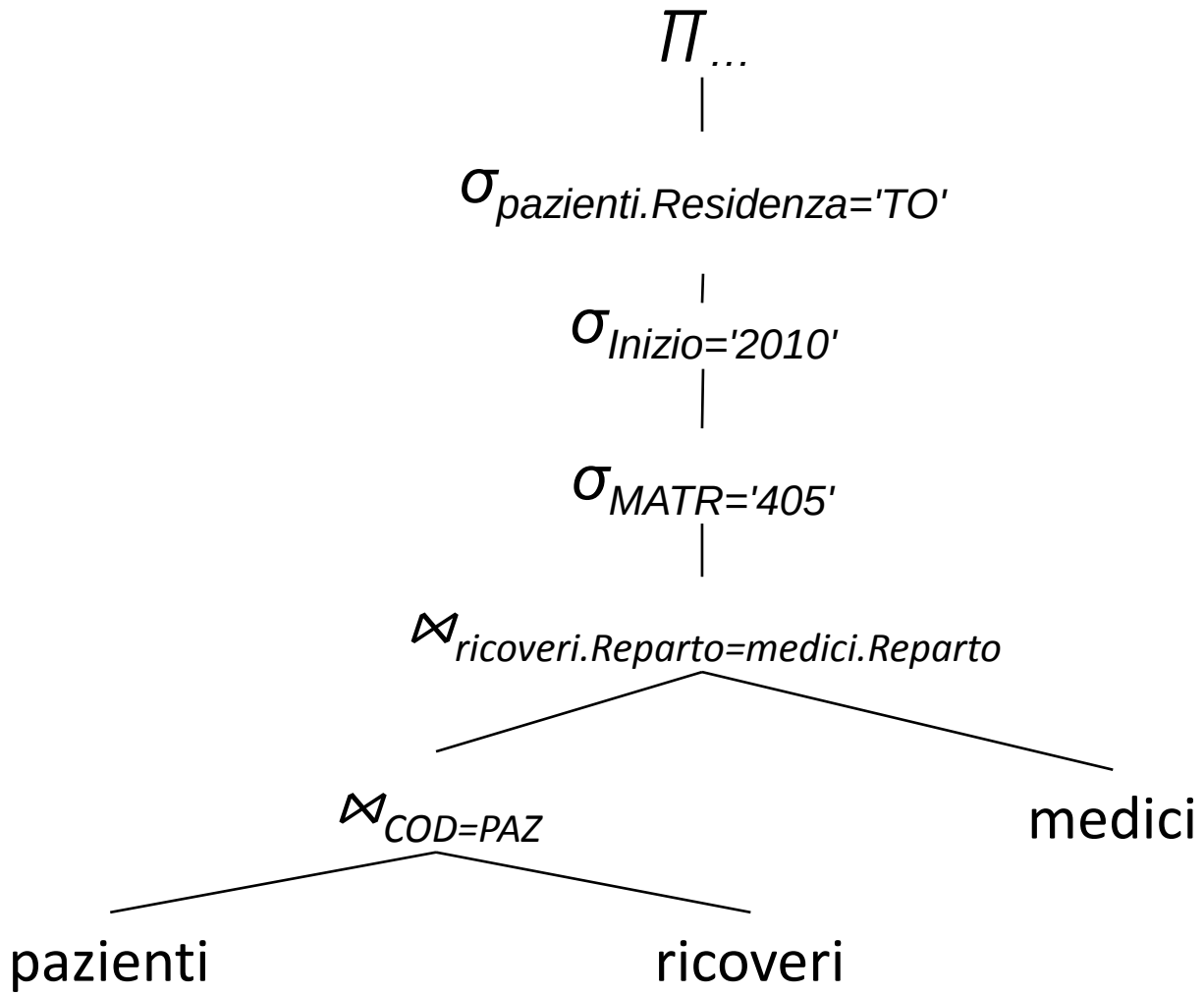
2. **Trasferire le selezioni verso le foglie finché è possibile con le proprietà distributive della selezione**

3. Trasferire le proiezioni verso le foglie finché è possibile con le proprietà distributive della proiezione

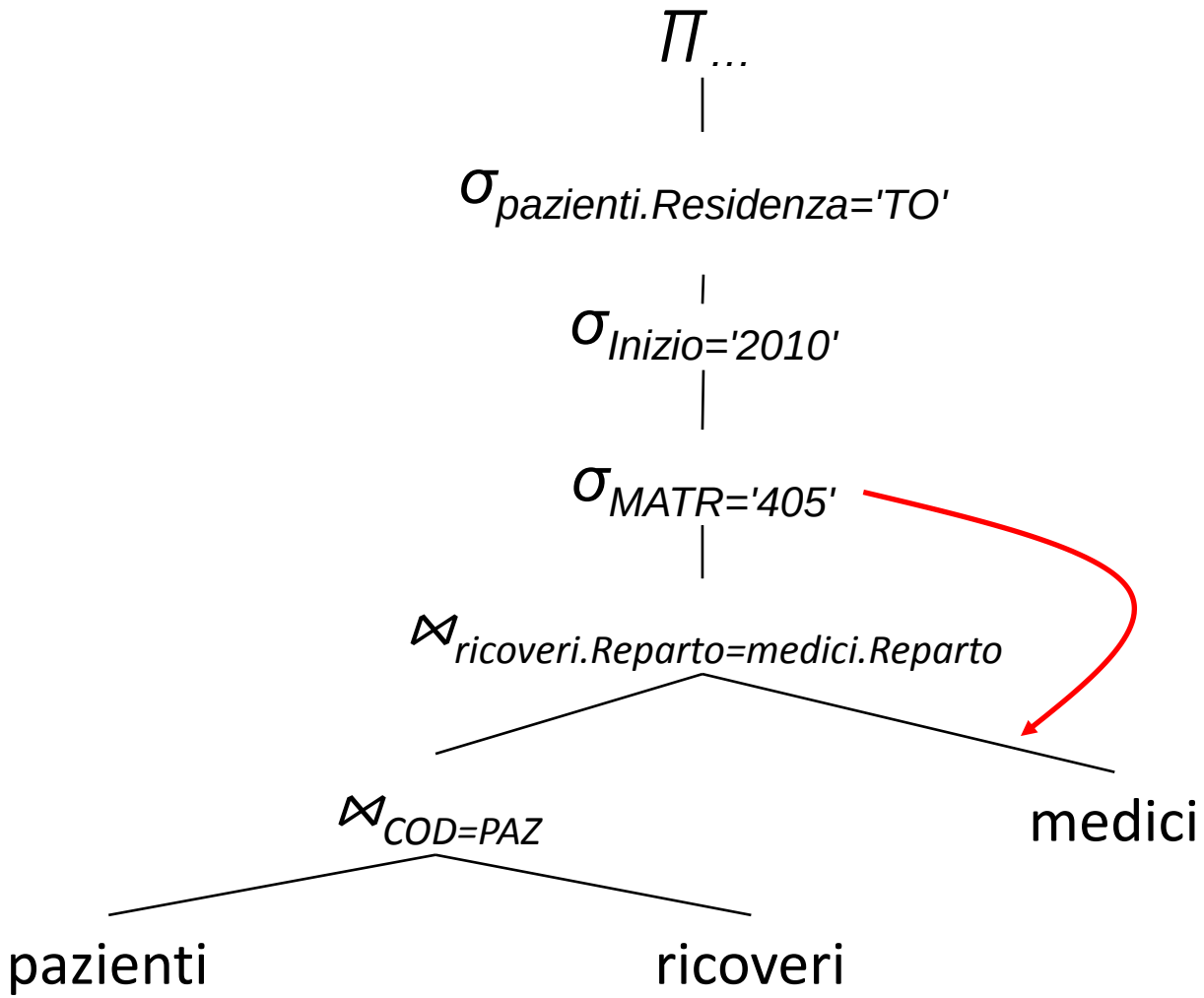
Spiegazione del punto 2

A questo punto si può applicare il punto 2, trasferire le selezioni verso le foglie

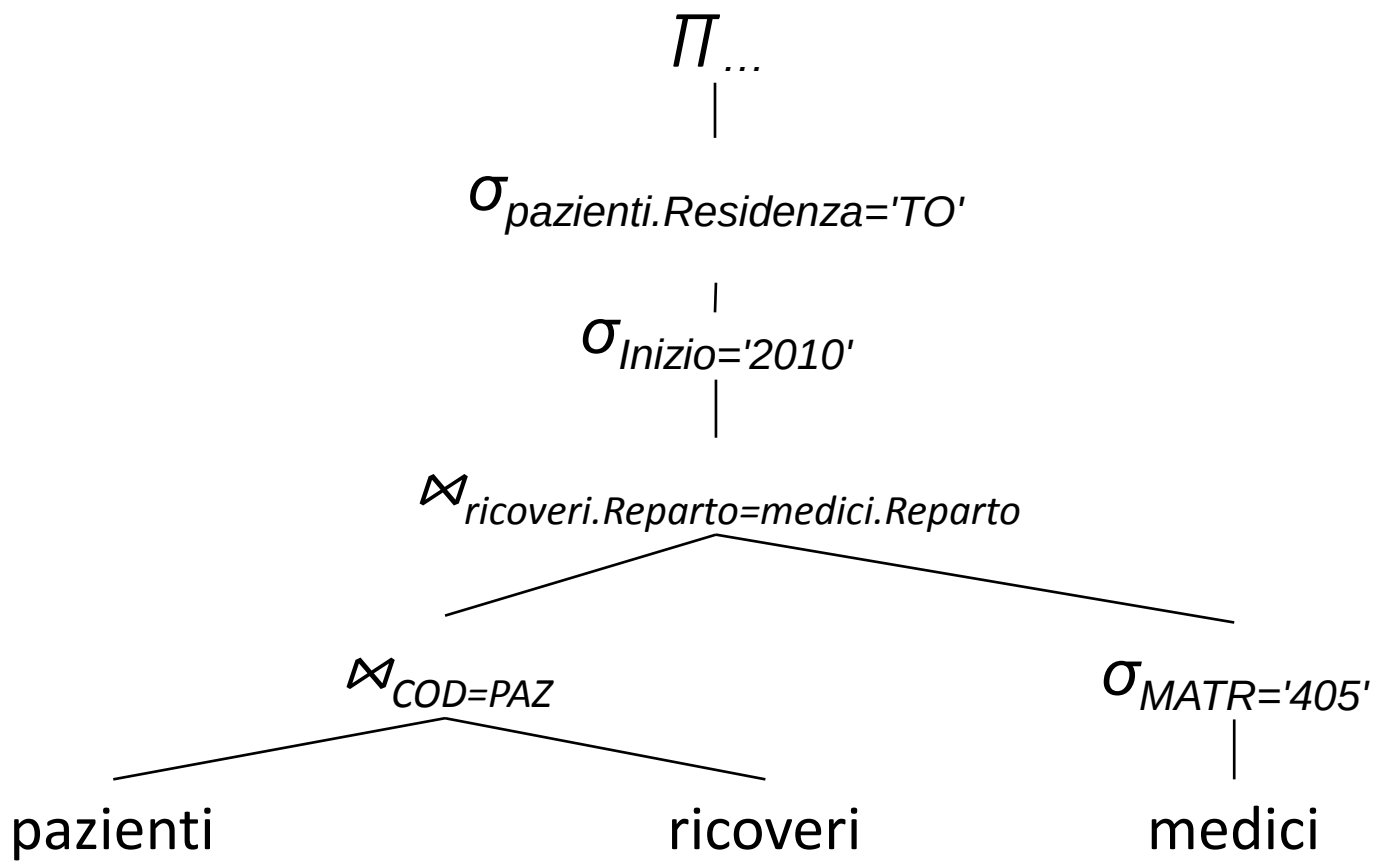
Esecuzione del passo 2



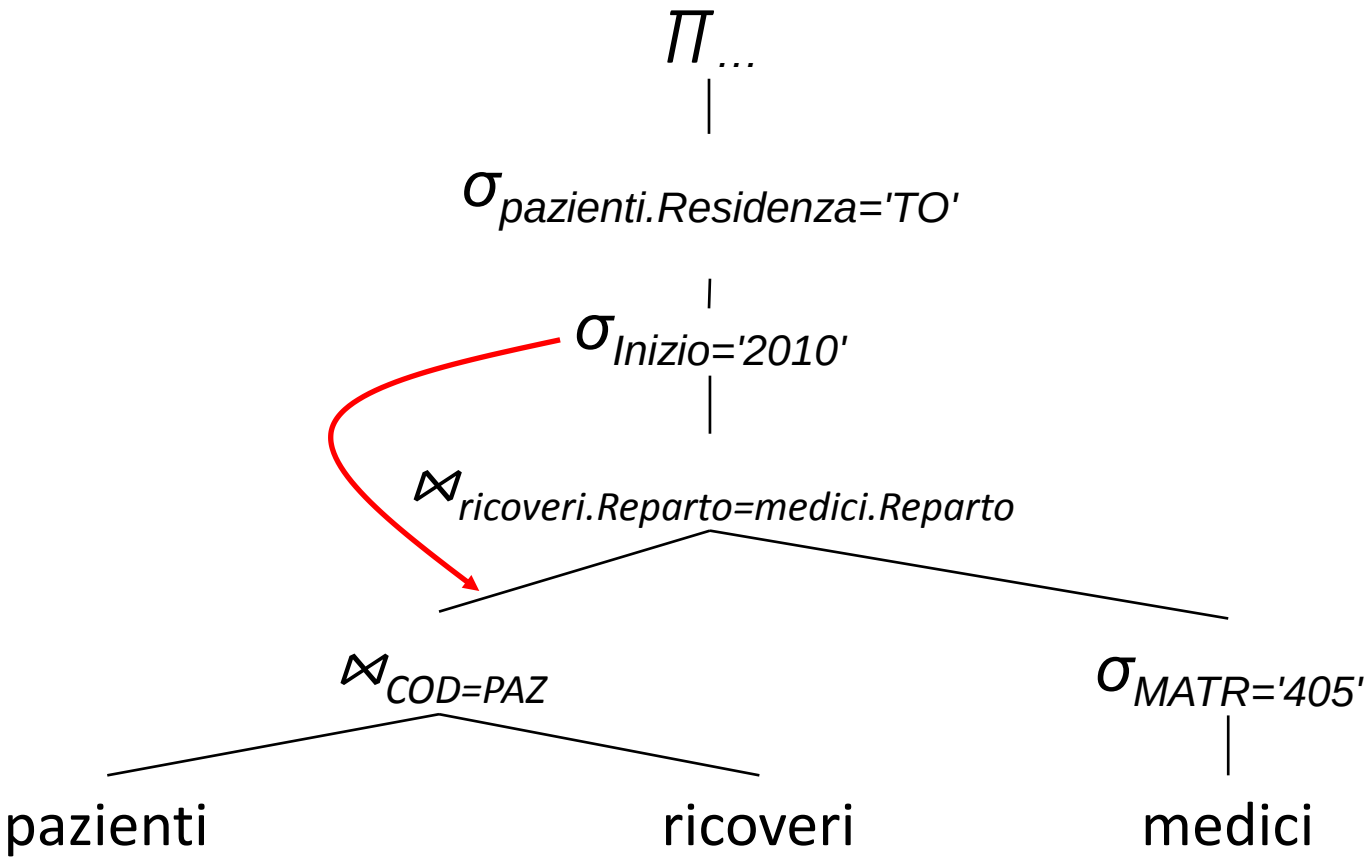
Esecuzione del passo 2



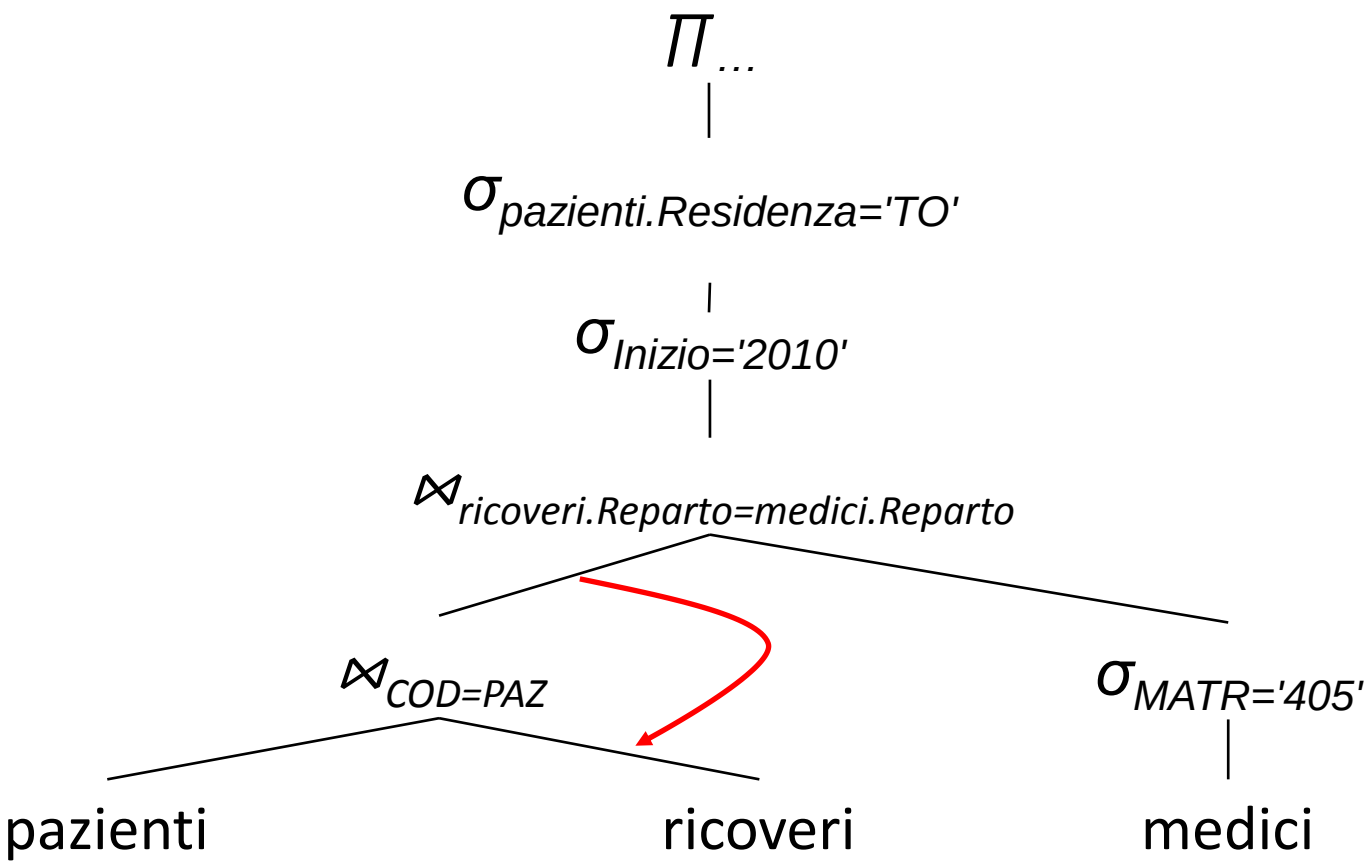
Esecuzione del passo 2



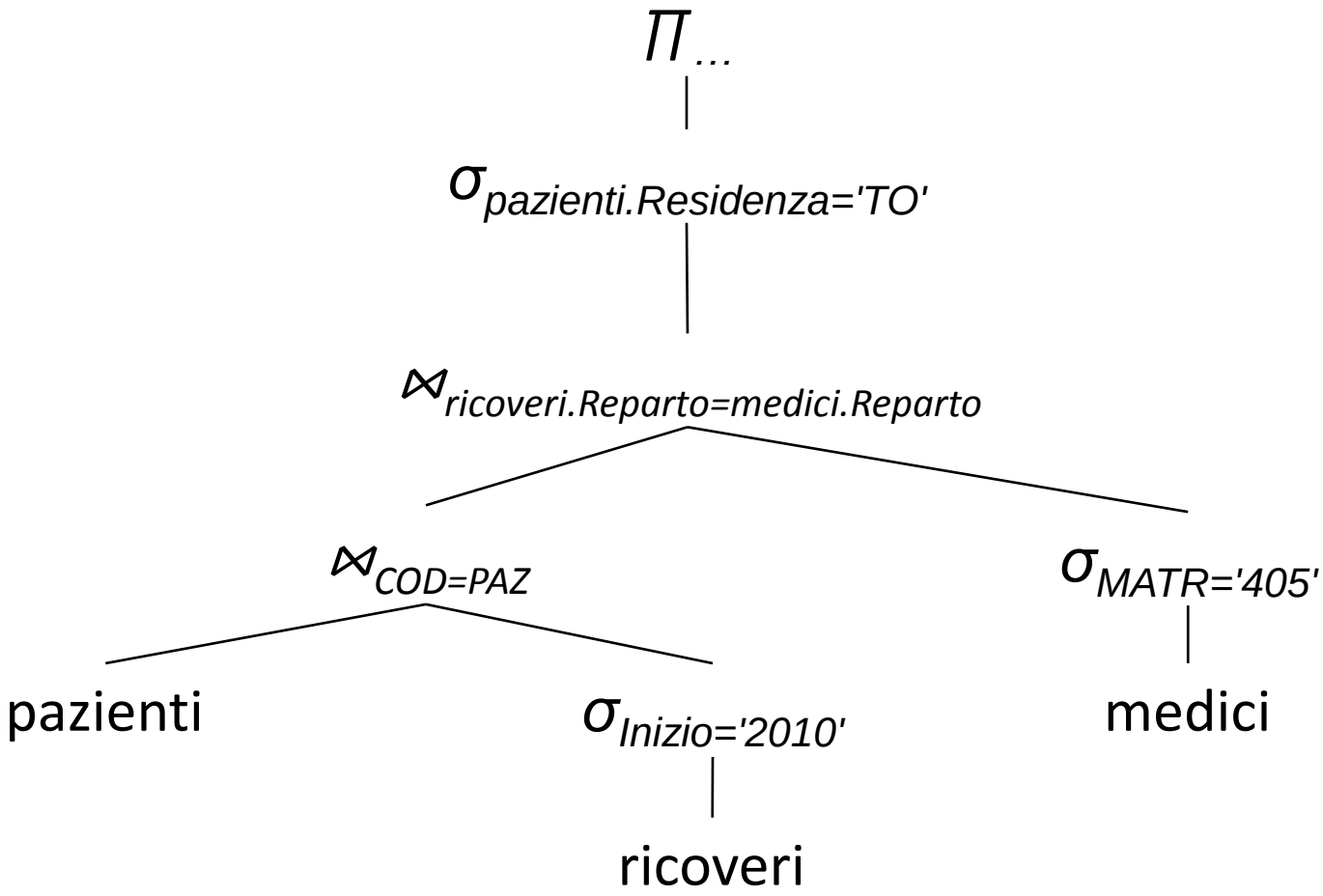
Esecuzione del passo 2



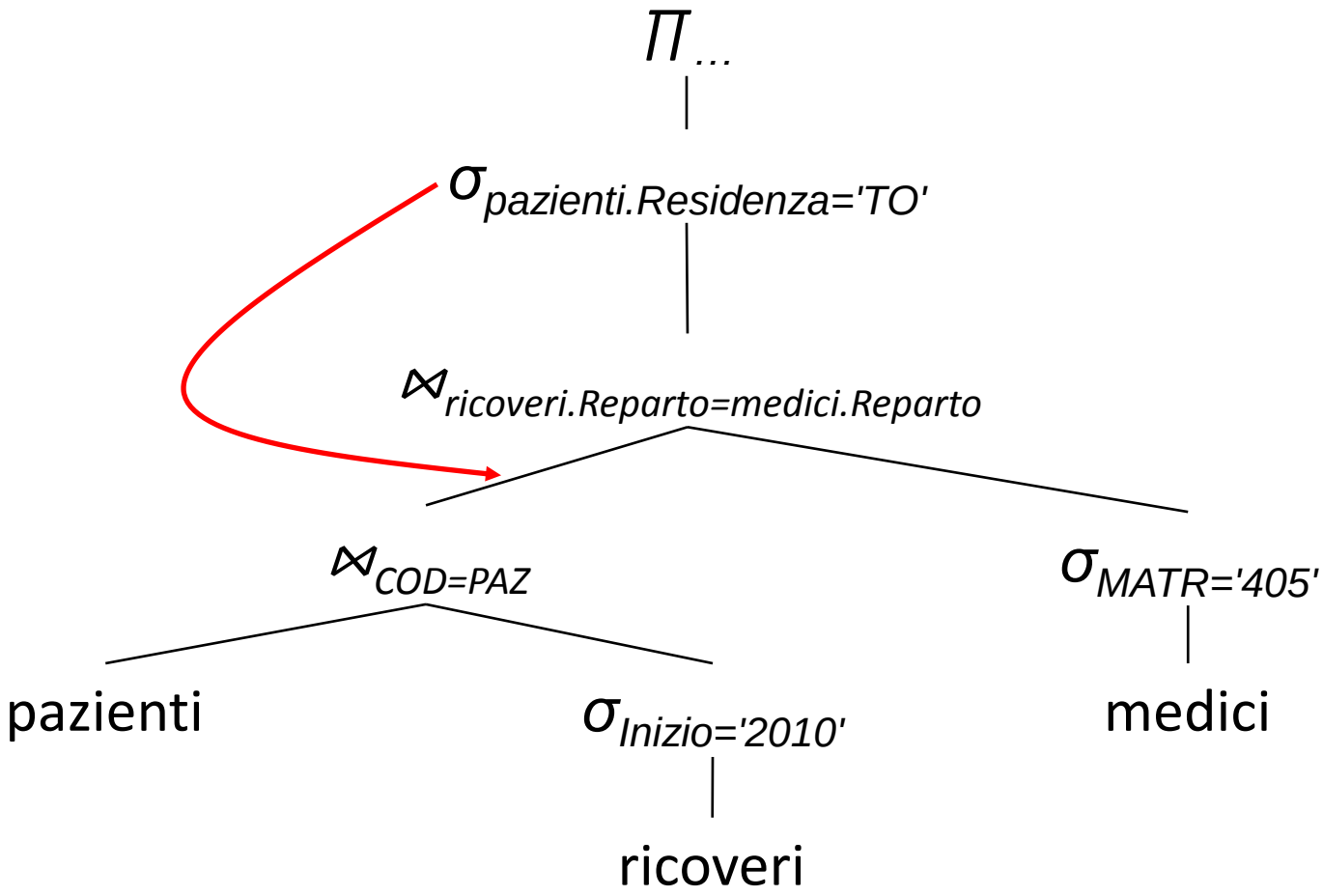
Esecuzione del passo 2



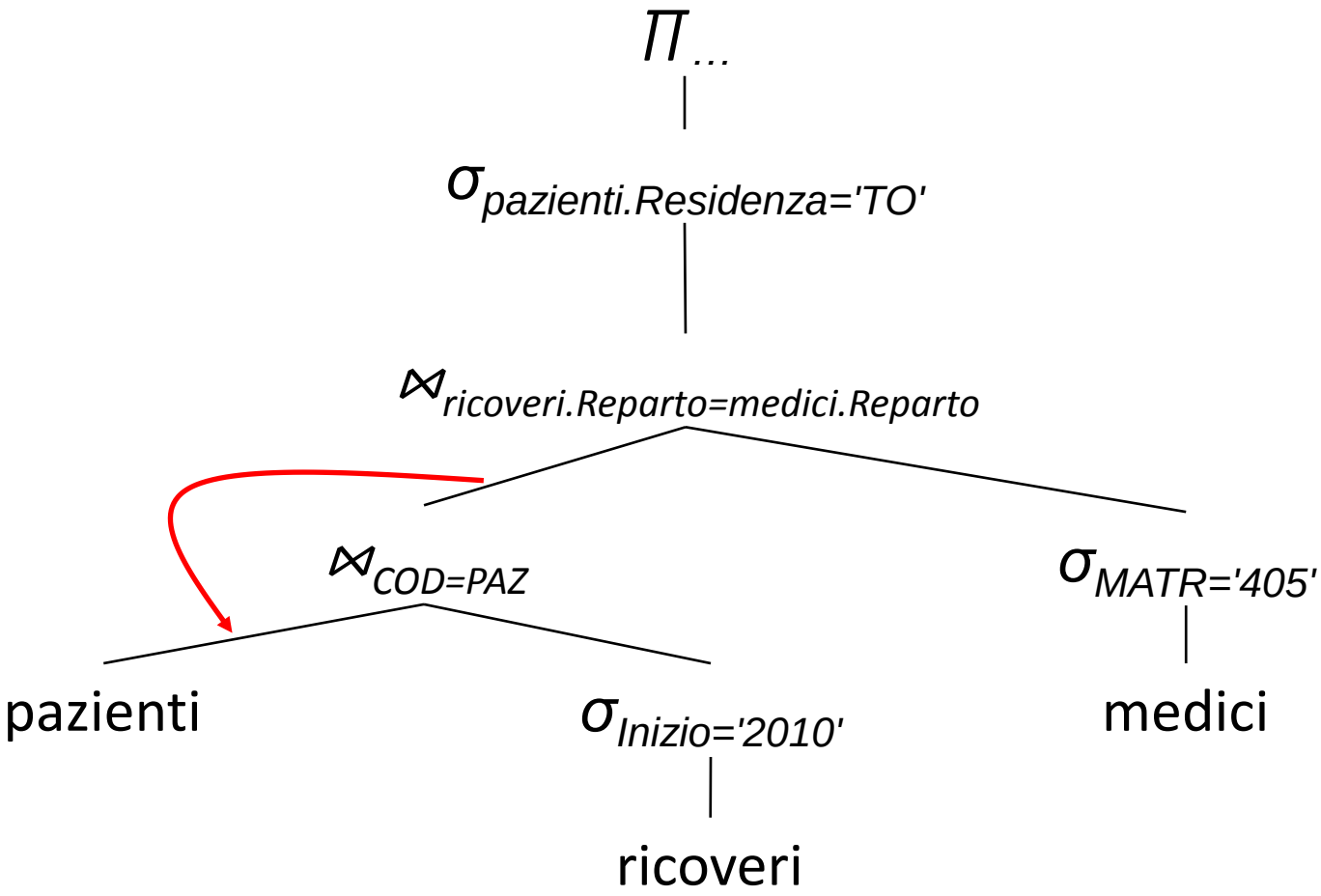
Esecuzione del passo 2



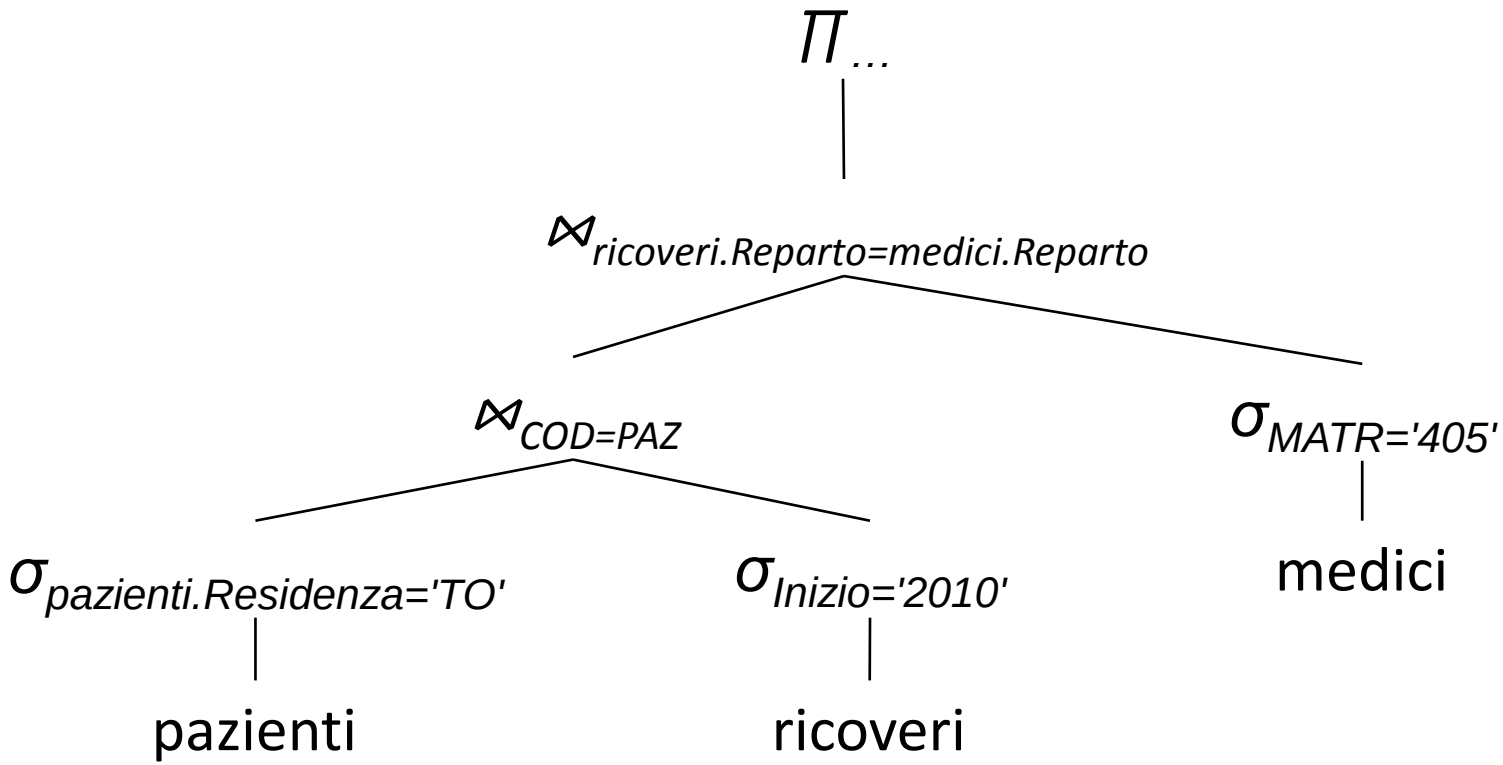
Esecuzione del passo 2



Esecuzione del passo 2



Esecuzione del passo 2



Spiegazione del passo 4

4. **Ricondurre ad un'unica selezione le selezioni multiple**
   - $\sigma_{p1}(\sigma_{p2}(r)) \rightarrow \sigma_{p1 \wedge p2}(r)$

5. Riconoscere le sequenze di join
   - $\sigma_{\theta}(r \times s) \rightarrow r \bowtie_{\theta} s$

6. Ricondurre ad un'unica proiezione le proiezioni multiple
   - $\Pi_X \Pi_{X,Y}(r) \rightarrow \Pi_X(r)$

7. Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)

Spiegazione del passo 4

- A forza di trasferire le selezioni, le selezioni si fermano sui sottoalberi di competenza
- Potremmo quindi trovare delle selezioni in cascata (selezioni multiple)
- L'ottimizzatore ricompone insieme le selezioni multiple, applicando una sola selezione con una congiunzione di predicati

Spiegazione del passo 5

4. Ricondurre ad un'unica selezione le selezioni multiple
   - $\sigma_{p1}(\sigma_{p2}(r)) \rightarrow \sigma_{p1 \wedge p2}(r)$

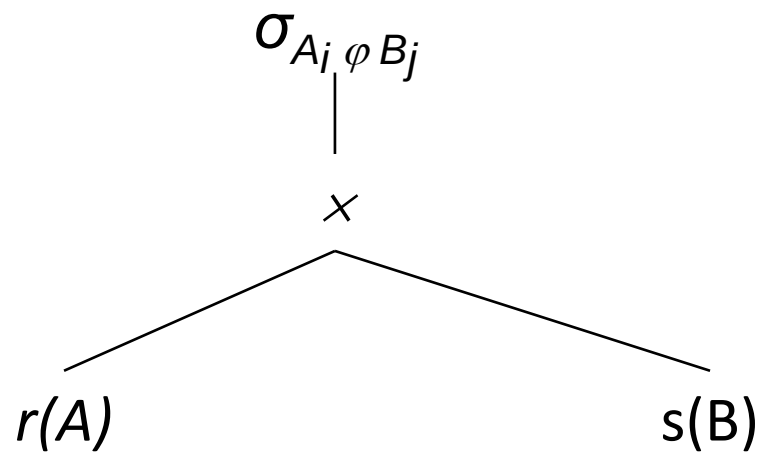
5. **Riconoscere le sequenze di join**
   - $\boldsymbol{\sigma_{\theta}(r \times s) \rightarrow r \bowtie_{\theta} s}$

6. Ricondurre ad un'unica proiezione le proiezioni multiple
   - $\Pi_X \Pi_{X,Y}(r) \rightarrow \Pi_X(r)$

7. Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)

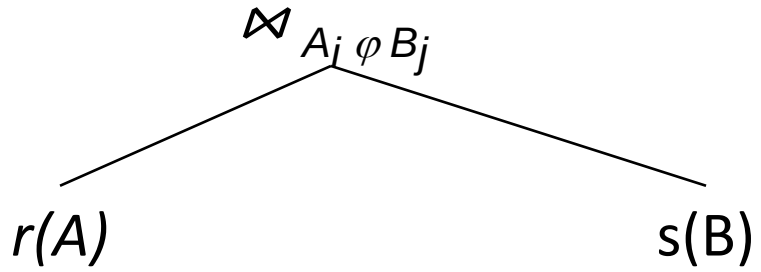
Spiegazione del passo 5

Capita molto spesso, anche in SQL, di trovarsi con delle selezioni con sottoalbero un prodotto cartesiano



Spiegazione del passo 5

Il passo 5 trasforma quindi il sottoalbero in un theta-join (tutti i DBMS hanno infatti algoritmi ottimizzati per eseguire i join)



Spiegazione del passo 6

4. Ricondurre ad un'unica selezione le selezioni multiple
   - $\sigma_{p1}(\sigma_{p2}(r)) \rightarrow \sigma_{p1 \wedge p2}(r)$

5. Riconoscere le sequenze di join
   - $\sigma_{\theta}(r \times s) \rightarrow r \bowtie_{\theta} s$

6. Ricondurre ad un'unica proiezione le proiezioni multiple
   - $\Pi_X \Pi_{X,Y}(r) \rightarrow \Pi_X(r)$

7. Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)

Spiegazione del passo 6

La spiegazione è intuitiva!

Aspetti quantitativi delle interrogazioni

I DBMS mantengono nel dizionario dati una serie di informazioni di tipo statistico su ogni tavola *r*, in particolare:

- cardinalità della relazione $CARD(r) = |r|$
- ampiezza della tupla in byte *SIZE(t)*
- $VAL(A_i,r)$ il numero di valori distinti che appare nella colonna A_i all'interno della tavola *r, ovvero*

$$VAL(A_i,r) = |\Pi_{A_i}(r)|$$

  - Ad esempio *VAL(Reparto,ricoveri) = 2*
  - Se A_i è chiave, $VAL(A_i,r) = CARD(r)$

Aspetti quantitativi delle interrogazioni

I DBMS mantengono nel dizionario dati una serie di informazioni di tipo statistico su ogni tavola *r*, in particolare:

- *MIN(A_i,r),* il valore minimo di A_i contenuto in *r*
- *MAX(A_i,r),* il valore massimo di A_i contenuto in *r*

- *NPAGE(r) = CARD(r)/fattore_di_bloccaggio*
  - *fattore_di_bloccaggio è il numero massimo di tuple che una pagina può contenere*

Analisi dei costi delle interrogazioni

L'analisi quantitativa dell'interrogazione permette di predire *ex-ante* il risultato della cardinalità della relazione risultato senza eseguirla

Stima del costo della selezione

Data la selezione $\sigma_p(r)$, conoscendo l'intervallo di variabilità della selezione $\sigma_p(r)$

$$0 \leq |\sigma_p(r)| \leq |r|$$

si può modellare la cardinalità della selezione $\sigma_p(r)$ con un **fattore di selettività f_p** per la cardinalità di r

$$|\sigma_p(r)| = f_p \cdot |r|$$

Il fattore di selettività è legato al solo predicato p di selezione e varia tra 0 e 1

Fattore di selettività

Il fattore di selettività f_p può essere interpretato come la probabilità che una tupla in *r* soddisfi il predicato di selezione *p,* ovvero la stima della percentuale di tuple che soddisfano il predicato di selezione

Come possiamo stimare f_p?

Fattore di selettività

I DBMS hanno a disposizione un formulario anche molto avanzato per il calcolo del fattore di selettività f_p

Noi forniremo una versione più grossolana, ma sufficiente ai nostri scopi

Considereremo quindi una **distribuzione uniforme** dei valori all'interno delle varie colonne

Tabella del fattore di selettività

- Predicati atomici

Predicato p	**f_p**
$A_i = v$	$1/VAL(A_i, r)$
$A_i \leq v$	$(v - MIN(A_i, r))/(MAX(A_i, r) - MIN(A_i, r))$
$A_i \geq v$	$(MAX(A_i, r) - v)/(MAX(A_i, r) - MIN(A_i, r))$
$v_1 \leq A_i \leq v_2$	$(v_2 - v_1)/(MAX(A_i, r) - MIN(A_i, r))$

Tabella del fattore di selettività

- Predicati composti

Predicato p	**f_p**
$p_1 \wedge p_2 \wedge \ldots \wedge p_n$	$f_{p_1} \cdot f_{p_2} \cdot \ldots \cdot f_{p_n}$
$\neg p$	$1 - f_p$
$p_1 \vee p_2 \vee \ldots \vee p_n$	$1 - ((1 - f_{p_1}) \cdot (1 - f_{p_2}) \cdot \ldots \cdot (1 - f_{p_n}))$

L'OR si dimostra con De Morgan:

$\neg\neg(p_1 \vee p_2 \vee \ldots \vee p_n) = \neg(\neg p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n)$

Stima della cardinalità del join

Per i nostri scopi, ci limiteremo a studiare l'equi-join

$$|r(A) \bowtie_{A_i = B_j} s(B)|$$

- Se esiste un vincolo di integrità referenziale $r(\ldots A_i \ldots) \rightarrow s(\ldots B_j \ldots)$, la cardinalità dell'equi-join è

$$|r(A) \bowtie_{A_i = B_j} s(B)| = |r(A)| = CARD(r)$$

Stima della cardinalità del join

In generale, per stimare

$$|r(A) \bowtie_{A_i = B_j} s(B)|$$

Immaginiamo di prendere **una tupla** di *r* che in corrispondenza dell'attributo A_i trova un ben preciso valore *v*, e in *s* ci siano diverse tuple con valore *v*

L'equi-join produrrà in uscita la giustapposizione della tupla di r per cui $A_i = v$ con le tuple di *s* per cui $B_j = v$

La cardinalità della selezione con predicato $p: B_j = v$ è $(1/VAL(B_j, s)) \cdot CARD(s)$, numero delle tuple giustapposte a *t*

Stima della cardinalità del join

Possiamo ora stimare la cardinalità completa del join estendendolo a tutte le tuple *t* della relazione *r*

La cardinalità del join diventa quindi:

$$|r(A) \bowtie_{A_i = B_j} s(B)| = (1/VAL(B_j, s)) \cdot CARD(s) \cdot CARD(r)$$

Non ho ancora finito!

Stima della cardinalità del join

Posso partire da s

Immaginiamo di prendere **una tupla** di *s* che in corrispondenza dell'attributo B_j trova un ben preciso valore *v*, e in *r* ci siano diverse tuple con valore *v*

L'equi-join produrrà in uscita la giustapposizione della tupla di s per cui $B_j = v$ con le tuple di *r* per cui $A_i = v$

La cardinalità della selezione con predicato *p:* $A_i = v$ è $(1/VAL(A_i, r)) \cdot CARD(r)$, numero delle tuple giustapposte a *t*

Stima della cardinalità del join

Possiamo ora stimare la cardinalità completa del join estendendolo a tutte le tuple *t* della relazione *s*

La cardinalità del join diventa quindi:

$$|r(A) \bowtie_{A_i = B_j} s(B)| = (1/VAL(A_i, r)) \cdot CARD(r) \cdot CARD(s)$$

Cambia solo il fattore di selettività:

$1/VAL(A_i, r)$ oppure $1/VAL(B_j, s)$?

Stima della cardinalità del join

Entrambe le formule sono delle sovrastime, possiamo quindi considerare il minimo delle due

La cardinalità del join diventa infine:

$$|r(A) \bowtie_{A_i = B_j} s(B)| = \min\{1/VAL(A_i, r), 1/VAL(B_j, s)\} \cdot CARD(r) \cdot CARD(s)$$

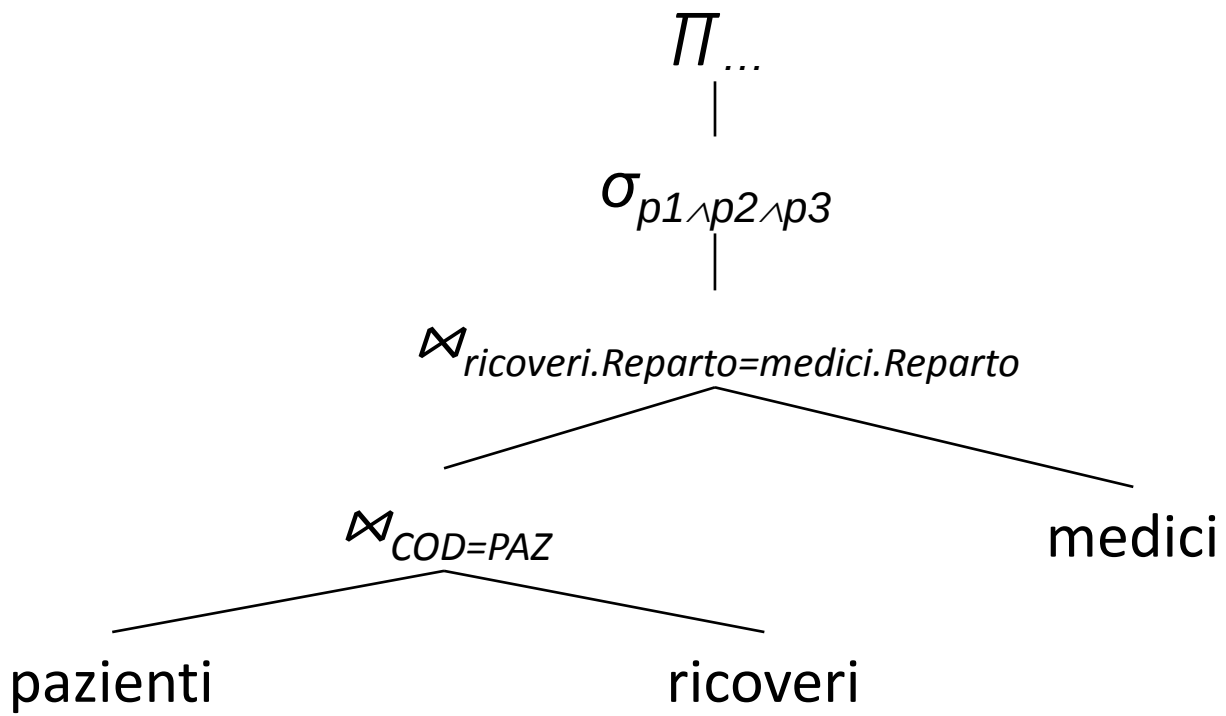
Caso di studio

$$\Pi_{\ldots}(\sigma_{pazienti.Residenza='TO' \wedge Inizio='2010' \wedge MATR='405'}((pazienti \bowtie_{COD=PAZ} ricoveri) \bowtie_{ricoveri.Reparto=medici.Reparto} medici))$$

Con

- *p1= pazienti.Residenza='TO'*
- *p2=Inizio='2010'*
- *p3=MATR='405'*

Caso di studio



Caso di studio

Dizionario dati:

- CARD(pazienti) = 10^5
- CARD(ricoveri)= 10^5
- CARD(medici)= 100
- VAL(Inizio,ricoveri) = 10 (dieci anni)
- VAL(Reparto,ricoveri) = 10
- VAL(Residenza,pazienti) = 100
- VAL(PAZ,ricoveri) = 10^5
- VAL(Reparto,medici) = 10

Caso di studio

Posso già calcolare i fattori di selettività dei predicati

- *p1= pazienti.Residenza='TO'*
- *p2=Inizio='2010'*
- *p3=MATR='405'*

cioè

- f_{p1} *= 1/(VAL(Residenza,pazienti) = 1/100*
- f_{p2} *= 1/(VAL(Inizio,ricoveri) = 1/10*
- f_{p3} *= 1/(VAL(MATR,medici) =* ***1/CARD(medici)*** *= 1/100*

Caso di studio

Semplificazione: da un punto di vista concettuale il join

$$pazienti \bowtie_{COD=PAZ} ricoveri$$

elabora un numero di tuple pari al prodotto di delle cardinalità delle due relazioni, quindi:

$$10^5 \cdot 10^5 = 10^{10} \text{ (dieci miliardi)}$$

per produrre quante tuple?

Dobbiamo valutare la cardinalità del join

Caso di studio

Valutiamo la cardinalità del join

$$pazienti \bowtie_{COD=PAZ} ricoveri$$

E' possibile usare la formula precisa in quanto *COD* è chiave di pazienti, ed esiste un vincolo di integrità referenziale tra ricoveri e pazienti, quindi

$$|pazienti \bowtie_{COD=PAZ} ricoveri| = CARD(ricoveri) = 10^5$$

Caso di studio

Ora devo effettuare il join del risultato del join precedente (*r1*) con medici

$$r1 \bowtie_{ricoveri.Reparto=medici.Reparto} medici$$

Le tuple coinvolte concettualmente sono

$$10^5 \cdot 10^2 = 10^7 \text{ (dieci milioni)}$$

per produrre quante tuple?

Caso di studio

Valutiamo la cardinalità del join

$$r1 \bowtie_{ricoveri.Reparto=medici.Reparto} medici$$

Non esiste nessun vincolo e non ci sono chiavi coinvolte, quindi dobbiamo stimare la cardinalità:

$$|r1 \bowtie_{ricoveri.Reparto=medici.Reparto} medici| =$$

$$min\{1/VAL(Reparto, r1), 1/VAL(Reparto, medici)\} \cdot CARD(r1) \cdot CARD(medici)$$

$$= min\{1/10, 1/10\} \cdot 10^5 \cdot 10^2$$

$$= 1/10 \cdot 10^5 \cdot 10^2 = 10^6$$

Caso di studio

A questo punto abbiamo la selezione $\sigma_{p1 \wedge p2 \wedge p3}$ che elabora le 10^6 tuple del risultato precedente (r2) e produce:

$$|\sigma_{p1 \wedge p2 \wedge p3}| =$$

$$f_{p1} \cdot f_{p2} \cdot f_{p3} \cdot CARD(r2) = 1/100 \cdot 1/10 \cdot 1/100 \cdot 10^6$$

$$= 10$$

Caso di studio

Qual è la massa di tuple coinvolte dall'interrogazione?

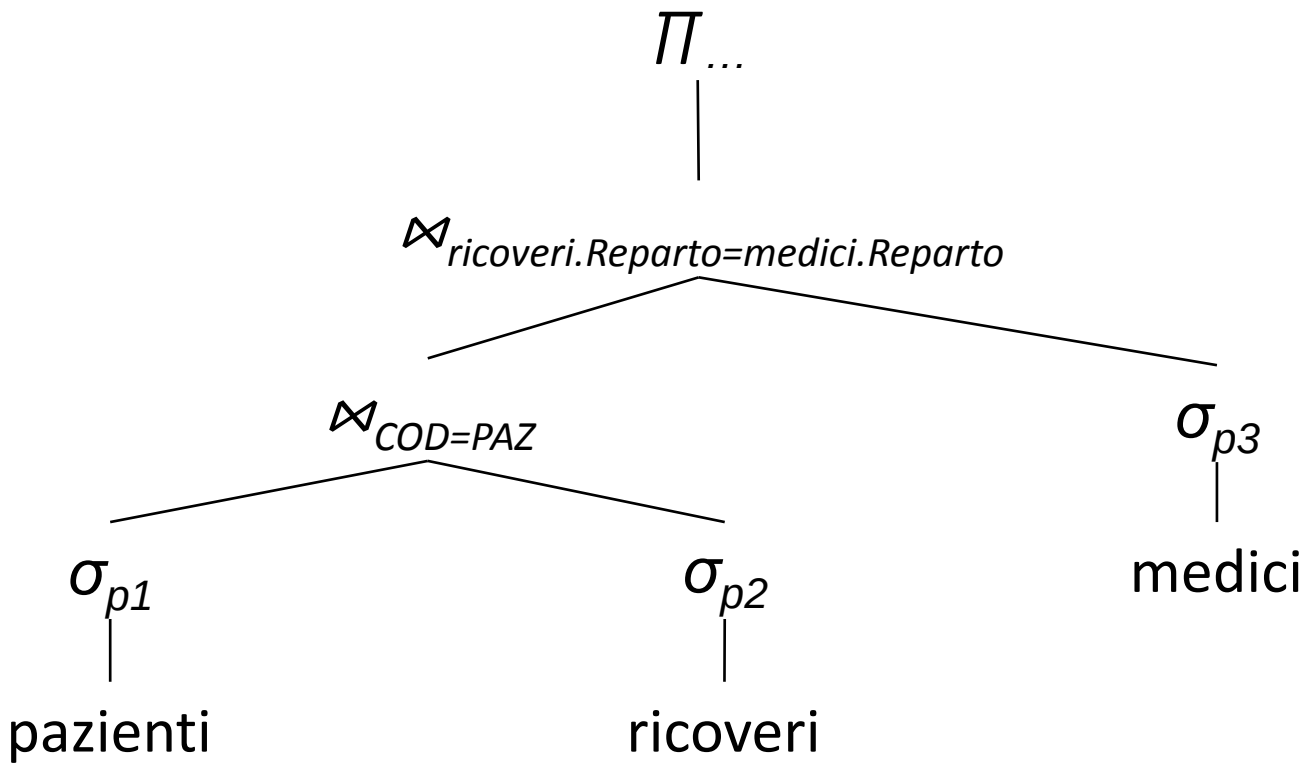
Bisogna sommare le tuple coinvolte dal primo join + tuple coinvolte dal secondo join + tuple coinvolte dalla selezione: $10^{10} + 10^{7} + 10^{6}$

E' sufficiente considerare l'ordine di grandezza, ovvero:

$$10^{10}$$

Caso di studio

Consideriamo ora il risultato dell'ottimizzazione logica



Caso di studio

Quante sono le tuple coinvolte dalla selezione σ_{p1}?

Sono esattamente le tuple della relazione *pazienti*, ovvero 10^5

Quante tuple produce?

$$|\sigma_{p1}| = f_{p1} \cdot CARD(pazienti) = 1/100 \cdot 10^5 = 10^3$$

Caso di studio

Quante sono le tuple coinvolte dalla selezione σ_{p2}?

Sono esattamente le tuple della relazione *ricoveri*, ovvero 10^5

Quante tuple produce?

$$|\sigma_{p2}| = f_{p2} \cdot CARD(ricoveri) = 1/10 \cdot 10^5 = 10^4$$

Caso di studio

Quante sono le tuple coinvolte dal join

$$\sigma_{p1}(pazienti) \bowtie_{COD=PAZ} \sigma_{p2}(ricoveri)$$?

Questa volta il join lavora su $10^3 \cdot 10^4 = 10^7$ tuple

Quante tuple produce?

Caso di studio

Quante tuple produce

$$\sigma_{p1}(pazienti) \bowtie_{COD=PAZ} \sigma_{p2}(ricoveri)$$?

Stavolta non si può più sfruttare il vincolo di integrità referenziale, in quanto non è più detto che le tuple in *ricoveri* trovino delle tuple corrispondenti in *pazienti*

La selezione ha alterato il vincolo!

Caso di studio

Devo quindi stimare l'equi-join

$$|\sigma_{p1}(pazienti) \bowtie_{COD=PAZ} \sigma_{p2}(ricoveri)| =$$

$$min\{1/VAL(COD, \boldsymbol{\sigma_{p1}(pazienti)}), 1/VAL(PAZ, \boldsymbol{\sigma_{p2}(ricoveri)})\} \cdot$$
$$CARD(\boldsymbol{\sigma_{p1}(pazienti)}) \cdot CARD(\boldsymbol{\sigma_{p2}(ricoveri)})$$

Non ho però $VAL(COD, \boldsymbol{\sigma_{p1}(pazienti)})$

La selezione σ_{p1} non riguarda COD, ma il DBMS conosce questi dati:

- $VAL(COD,pazienti) = 10^5$
- $CARD(\sigma_{p1}(pazienti)) = f_{p1} \cdot CARD(pazienti) = 1/100 \cdot 10^5 = 10^3$

Euristica del DBMS per la stima

L'euristica che il DBMS assume nello stimare il numero di valori distinti di codice che proviene dalla selezione con predicato *p1* sulla tavola pazienti è di prendere il

$$min\{ 10^5, 10^3 \}$$

Infatti non possono esserci più di 10^3 valori distinti

La stima si effettua cioè con la seguente formula:

$$VAL(COD, \boldsymbol{\sigma_{p1}(pazienti)}) =$$
$$min\{VAL(COD,pazienti), CARD(\sigma_{p1}(pazienti))\}$$

Euristica del DBMS per la stima

Formula generale per la stima di

$$|\sigma_{p1}(r(A)) \bowtie_{A_i = B_j} \sigma_{p2}(s(B))| =$$
$$min\{1/VAL(A_i, \sigma_{p1}(r)), 1/VAL(B_j, \sigma_{p2}(s))\}$$
$$\cdot CARD(\sigma_{p1}(r)) \cdot CARD(\sigma_{p2}(s))$$

con

- $VAL(A_i, \sigma_{p1}(r)) = min\{ VAL(A_i,r), CARD(\sigma_{p1}(r)) \}$
- $VAL(B_j, \sigma_{p2}(s)) = min\{ VAL(B_j,r), CARD(\sigma_{p2}(s)) \}$

quando A_i e B_j non sono coinvolti nei predicati di selezione

Caso di studio

Non ho neanche il $VAL(PAZ, \boldsymbol{\sigma_{p2}(ricoveri)})$

Il DBMS conosce questi dati:

- $VAL(PAZ,ricoveri) = 10^5$
- $CARD(\sigma_{p2}(ricoveri)) = f_{p2} \cdot CARD(ricoveri) = 1/10 \cdot 10^5 = 10^4$

Di conseguenza

$$VAL(PAZ, \boldsymbol{\sigma_{p2}(ricoveri)}) =$$

$$min\{VAL(PAZ,ricoveri), CARD(\sigma_{p2}(ricoveri))\} =$$

$$min\{10^5, 10^4\} = 10^4$$

Caso di studio

Quante tuple produce

$$\sigma_{p1}(pazienti) \bowtie_{COD=PAZ} \sigma_{p2}(ricoveri) \ ?$$

$$|\sigma_{p1}(pazienti) \bowtie_{COD=PAZ} \sigma_{p2}(ricoveri)|$$

$$=$$

$$min\{1/VAL(COD, \boldsymbol{\sigma_{p1}(pazienti)}), 1/VAL(PAZ, \boldsymbol{\sigma_{p2}(ricoveri)})\} \cdot CARD(\boldsymbol{\sigma_{p1}(pazienti)}) \cdot CARD(\boldsymbol{\sigma_{p2}(ricoveri)})$$

$$=$$

$$min\{1/10^3, 1/10^4\} \cdot 10^3 \cdot 10^4 = 1/10^4 \cdot 10^3 \cdot 10^4 = \boldsymbol{10^3}$$

Caso di studio

Quante sono le tuple coinvolte dalla selezione σ_{p3}?

Sono esattamente le tuple della relazione *medici*, ovvero 10^2

Quante tuple produce?

$$|\sigma_{p3}| = 1$$

(è una selezione su MATR che è chiave, quindi produce una sola tupla)

Caso di studio

Quante sono le tuple coinvolte dal join del risultato del join precedente (r1) con *medici*?

$$r1 \bowtie_{ricoveri.Reparto=medici.Reparto} \sigma_{p3}(medici)$$

Questa volta il join lavora su $10^3 \cdot 1 = 10^3$ tuple

Quante tuple produce?

Non è interessante (esercizio)

Caso di studio

Qual è la massa di tuple coinvolte dall'interrogazione?

Bisogna sommare le tuple coinvolte dalle tre selezioni + tuple coinvolte dal primo join + tuple coinvolte dal secondo join: $10^5 + 10^5 + 10^7 + 10^2 + 10^3$

E' sufficiente considerare il valore dominante, ovvero:

10^7

Che è **mille volte meno** della massa di tuple coinvolte senza ottimizzazione logica!

Spiegazione del passo 7

4. Ricondurre ad un'unica selezione le selezioni multiple
   - $\sigma_{p1}(\sigma_{p2}(r)) \rightarrow \sigma_{p1 \wedge p2}(r)$

5. Riconoscere le sequenze di join
   - $\sigma_{\Theta}(r \times s) \rightarrow r \bowtie_{\Theta} s$

6. Ricondurre ad un'unica proiezione le proiezioni multiple
   - $\Pi_X \Pi_{X,Y}(r) \rightarrow \Pi_X(r)$

7. **Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)**

Spiegazione del passo 7

Esame delle varianti dell'albero di parsificazione dovute alle proprietà associative (scegliere la variante di costo minimo)

- L'ottimizzatore cerca una configurazione dell'albero di parsificazione che riduca ulteriormente la massa di tuple concettualmente elaborate
- Principalmente questo passo consiste nella ricerca di parentesizzazioni alternative, e meno costose, di join multipli

Spiegazione del passo 7

Questo passo ha delle difficoltà computazionali non indifferenti

Consideriamo una sequenza di join:

$$r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

- Le configurazioni possibili dovute alle proprietà associative crescono esponenzialmente con n
  - n=3, 2 configurazioni
  - n=4, 5 configurazioni
  - n=5, 14 configurazioni
  - n=10, 4862 configurazioni

$$configurazioni = \frac{1}{n}\binom{2(n-1)}{n-1}$$

Esempio con 4 relazioni

$$((r_1 \bowtie r_2) \bowtie r_3) \bowtie r_4$$

$$(r_1 \bowtie (r_2 \bowtie r_3)) \bowtie r_4$$

$$r_1 \bowtie ((r_2 \bowtie r_3) \bowtie r_4)$$

$$r_1 \bowtie (r_2 \bowtie (r_3 \bowtie r_4))$$

$$(r_1 \bowtie r_2) \bowtie (r_3 \bowtie r_4)$$

Spiegazione del passo 7

- Gli ottimizzatori non esplorano tutte le possibili configurazioni ma cercano una configurazione subottimale

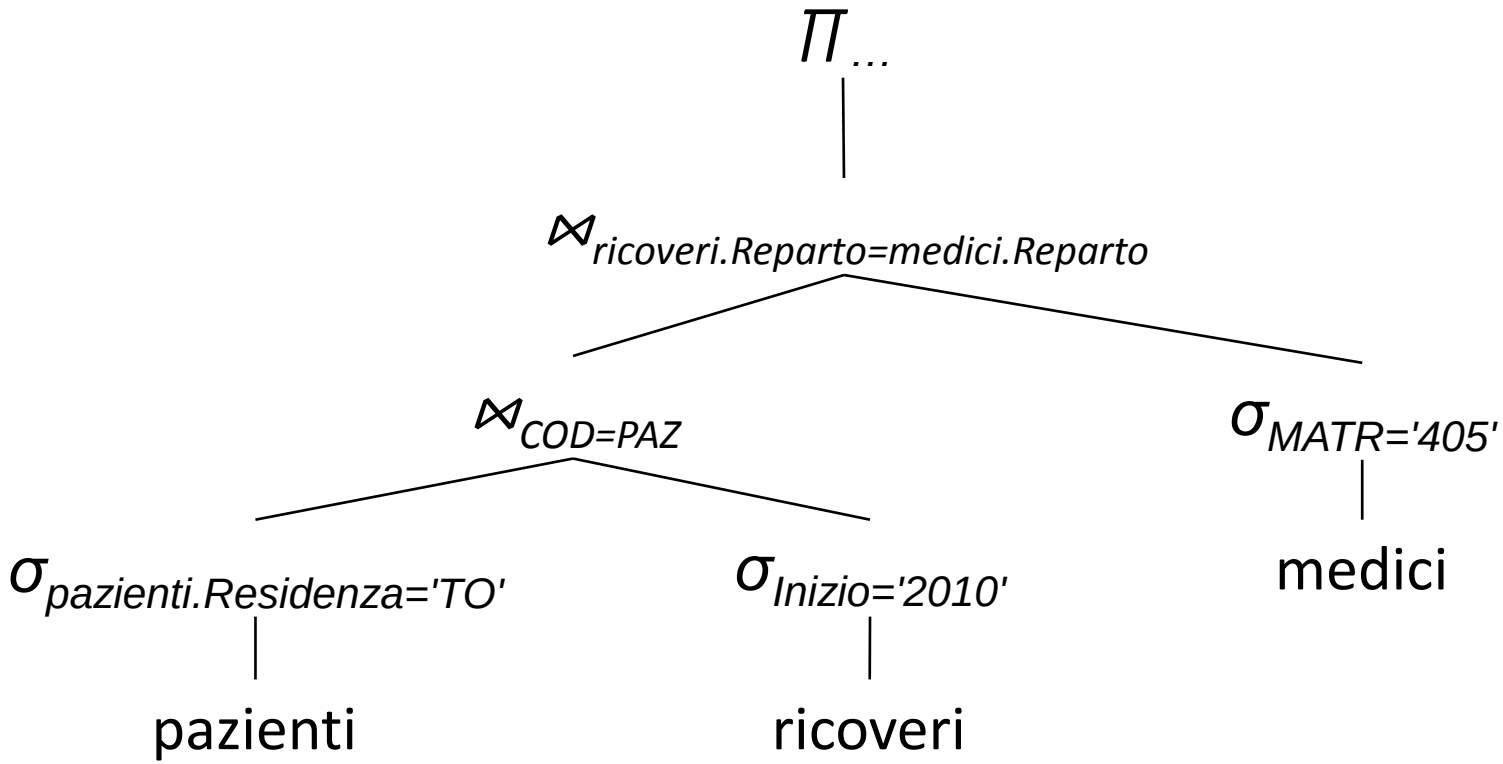
- La scelta dell'ottimizzatore è quella di anticipare il prima possibile i join tra relazioni con cardinalità bassa

- Si cerca il join a cardinalità minima dei sottoalberi e si esegue per primo e così via

Spiegazione del passo 7

E' vero che il join può lavorare su una massa di tuple enorme, ma solitamente, in uscita, produce un numero di tuple limitato (dovuto alla selettività del suo predicato theta)

Spiegazione del passo 7



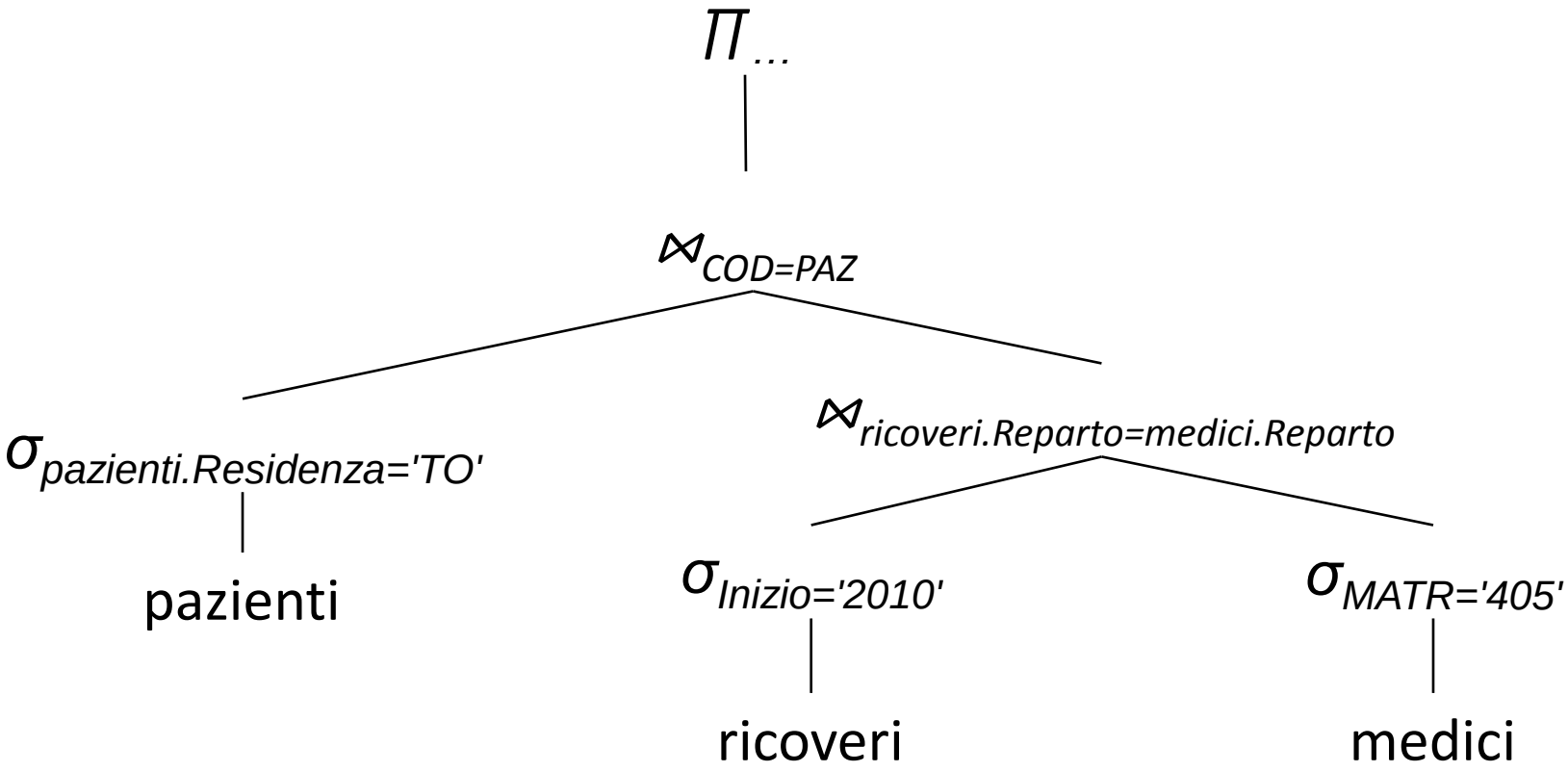
Spiegazione del passo 7

La cardinalità di $\sigma_{p3}(medici)$ è 1, quindi l'ottimizzatore cercherà di eseguire prima un join che coinvolge σ_{p3}(medici)

Tra gli altri due sottoalberi, quello a cardinalità minima (10^3) è σ_{p1}(pazienti)

Ma medici non può fare join con pazienti, quindi l'ottimizzatore è costretto a scegliere σ_{p2}(ricoveri)

Spiegazione del passo 7



Analisi quantitativa

I dati delle tre selezioni sono identici

- σ_{p1} elabora 10^5 tuple e $|\sigma_{p1}| = 10^3$
- σ_{p2} elabora 10^5 tuple $|\sigma_{p2}| = 10^4$
- σ_{p3} elabora 10^2 tuple $|\sigma_{p3}| = 1$

Il join $\sigma_{p2} \bowtie_{ricoveri.Reparto=medici.Reparto} \sigma_{p3}$ dovrà elaborare
$10^4 \cdot 1 = 10^4$ tuple e produrrà

$$min\{1/VAL(Reparto, \sigma_{p3}(medici)), 1/VAL(Reparto, \sigma_{p2}(ricoveri))\} \cdot CARD(\sigma_{p3}(medici)) \cdot CARD(\sigma_{p2}(ricoveri)) = min\{1/1, 1/10\} \cdot 1 \cdot 10^4 =$$

$$1/10 \cdot 10^4 = 10^3$$

Analisi quantitativa

Per l'ultimo join tra σ_{p1}*(pazienti)* e il risultato del join precedente
$10^3 \cdot 10^3 = 10^6$ tuple elaborate

Qual è la massa di tuple coinvolte dall'interrogazione?

Bisogna sommare le tuple coinvolte dalle tre selezioni + tuple coinvolte dal primo join + tuple coinvolte dal secondo join:
$10^2 + 10^5 + 10^5 + 10^6$

E' sufficiente considerare il valore dominante, ovvero:

10^6

In seguito…

- Quando l'ottimizzatore logico finisce il suo lavoro e invia l'albero di parsificazione all'ottimizzatore fisico, quest'ultimo mette, al posto dei nodi, dei veri e propri blocchi di codice (ad esempio: il codice per la selezione, il codice per il join…)

- L'ottimizzatore fisico sceglie, all'interno delle sue librerie, il codice che corrisponde all'algoritmo che muove meno pagine