## CAP3\_ALGEBRA\_RELAZIONALE

Table of contents

- Algebra e calcolo relazionale
  - Algebra relazionale
    - Operatori insiemistici (\$ cup, cap, -\$)
    - Ridenominazione (\$rho\_{a/b}(R)\$)
    - Selezione (\$ sigma\_{varphi}(R)\$)
    - Proiezione (\$Pi\_{a\_1, dotsc, a\_n}(R)\$)
    - JOIN
      - JOIN NATURALE (\$R bowtie S\$)
      - JOIN COMPLETO
      - JOIN ESTERNO
      - JOIN E PROIEZIONI
      - PRODOTTO CARTESIANO (\$ times\$)
    - VISTE (\$:=\$)
- Esempi esercizi
- Esempi estratti da prove in itinere

# Algebra e calcolo relazionale

#algebra-relazionale #procedurali #ridenominazione #selezione #proiezione #join #viste

I linguaggi possono essere distinti in:

- dichiarativi, specificano le proprietà del risultato("che cosa")
  - · calcolo relazionale
  - SQL
  - Query By Example (QBE)
- procedurali, specificano le modalità di generazione del risultato ("come")
  - algebra relazionale

## Algebra relazionale

Insieme di operatori:

- su relazioni
- · che producono relazioni
- · possono essere composti

Con l'algebra relazionale lavoriamo su tabelle/relazioni e applichiamo strutture algebriche con semantiche ben fondate, per produrre altre tabelle.

## Operatori insiemistici ( $\cup, \cap, -$ )

Le relazioni sono degli insiemi di n-uple.

- unione  $A \cup B$ , unisce tutti gli attributi delle tabelle, i duplicati vengono eliminati;
- intersezione  $A \cap B$ , produce relazione di n-uple uguali tra entrambe le relazioni;
- differenza A-B, relazione di n-uple non contenute in B.

#### △ Nota sulla compatibilità

La possibilità di operare con  $\cup$  e  $\cap$  sussiste fintanto che le due relazioni in questione abbiano cardinalità uguale. Questo è dato dal fatto che l'intersezione è una unione con sottrazione; le due relazioni devono essere quindi compatibili per l'unione se vogliamo che l'intersezione sia possibile.

## Ridenominazione ( $ho_{a/b}(R)$ )

Operatore monadico (su una tabella) che *modifica lo schema*, non l'istanza, cambiando il nome di 1 o più attributi.

$$\text{REN}_{newName \leftarrow oldName}(Operando)$$

$$\rho_{A_1,\ldots,A_n\leftarrow a_1,\ldots,a_n}(R)$$

Gli attributi  $a_1,\ldots,a_n$  assumono nuovo nome  $A_1,\ldots,A_n$  per la relazione R.

#### **≔** Ridenominare 2 tabelle

L'unione tra 2 tabelle con attributi "Madre" e "Padre" non è possibile siccome il nome degli attributi è diverso, possiamo tuttavia ridenominare questi

 $\mathsf{REN}_{\mathsf{genitore} \leftarrow \mathsf{padre}}(\mathsf{Paternita}) \cup \mathsf{REN}_{\mathsf{genitore} \leftarrow \mathsf{madre}}(\mathsf{Maternita})$ 

## Selezione ( $\sigma_{arphi}(R)$ )

Operatore monadico (su una sola tabella) che produce un risultato con lo stesso schema dell'operando e contiene una selezione delle n-uple che soddisfano un predicato (TRUE, FALSE). Semplicemente: prende una condizione e ritorna i risultati soddisfacenti la condizione.

$$SEL_{Condizione}(Operando)$$

$$\sigma_{Condizione}(R)$$

dove Condizione è una formula proposizionale.

#### ∷≣ Impiegati che guadagnano più di 50

SEL<sub>stipendio</sub> > 50 (Impiegati)

### ≡ Impiegati che guadagnano più di 50 e lavorano a 'Milano'

SEL<sub>stipendio</sub> > 50 AND filiale = 'Milano' (Impiegati)

#### ⚠ Nota sui valori NULL

#### #NULL\_VALUES

Nell'algebra relazionale (quindi in psql) i valori NULL non sono distinti l'uno dall'altro. Questo significa che operazioni come  $A \neq B$  dove A=0 e B= NULL, restituiranno sempre unknown siccome NULL non è ben definito.

A	B	(A  eq B)	A IS DISTINCT FROM $B$
0	0	false	false

A	B	(A  eq B)	A IS DISTINCT FROM $B$
0	1	true	true
0	NULL	unknown	true
NULL	NULL	unknown	false

# Proiezione ( $\Pi_{a_1,\dots,a_n}\!(R)$ )

Decomposizione verticale, operatore ortogonale.

Anche lui operatore monadico, parametrico.

Semplicemente: prende una lista di attributi riguardante a una tabella e restituisce solo quelli specificati.

$$PROJ_{ListaAttributi}(Operando)$$

≡ Cognome e filiale di tutti gli impiegati

PROJ<sub>cognome.nome</sub>(Impiegati)

Una proiezione contiene al più tante n-uple quante l'operando, può contenerne di meno. Se X è una superchiave di R, allora  $\mathrm{PROJ}_X(R)$  contiene esattamente tante n-uple quante R. Possiamo usare selezione e proiezione insieme, per restituire risultati di una selezione per delle colonne specifiche solo del SELECT:

≡ Matricola e cognome degli impiegati che guadagnano più di 50

PROJ<sub>matricola,cognome</sub>(SEL<sub>stipendio</sub> > 50(Impiegati))

Non possiamo correlare informazioni presenti in relazioni diverse, né informazioni in n-upla diverse di una stessa relazione.

#### JOIN

Permette di correlare dati in relazioni diverse.

Cardinalità:

- il join di  $R_1$  e  $R_2$  contiene un numero di n-uple:
  - ullet compreso fra 0 e il prodotto di  $|R_1|$  e  $|R_2|$
- se coinvolge una chiave di  $R_2$  allora il numero di n-uple è:
  - compreso fra 0 e  $|R_1|$
- ullet se il join coinvolge una chiave di  $R_2$  e vincolo d'integrità referenziale, allora il numero di n-uple è
  - pari a  $|R_1|$

 $R_1$  JOIN  $R_2$  e' una relazione su  $X_1X_2$  (intesa come unione):

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

Per ogni riga che si trova nella tabella di sinistra, guardiamo quante di righe hanno un attributo in comune con la tabella di destra e uniamo nel caso in cui questa incidenza esista.

### JOIN NATURALE ( $R \bowtie S$ )

Immaginiamo di avere una due tabelle e volessimo unire le due, seguendo un criterio: numero deve essere contenuto in entrambe.

Possiamo farlo con il join naturale dove i miei attributi coincidono su un attributo. Noi non dobbiamo fare nulla, il join e' automatico se l'attributo comune esiste.

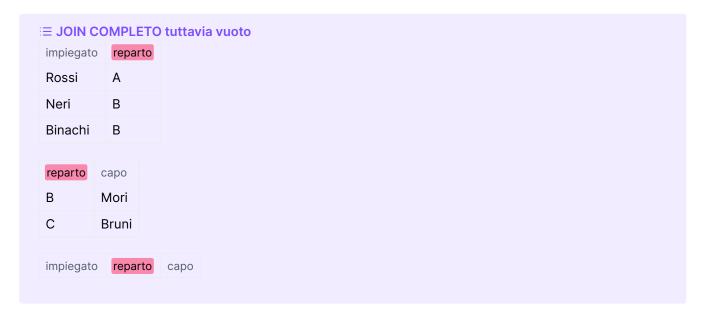


#### Produce un risultato:

- · sull'unione degli attributi degli operandi;
- con n-uple costruite ciascuna a partire da una n-upla di ognuno degli operandi;

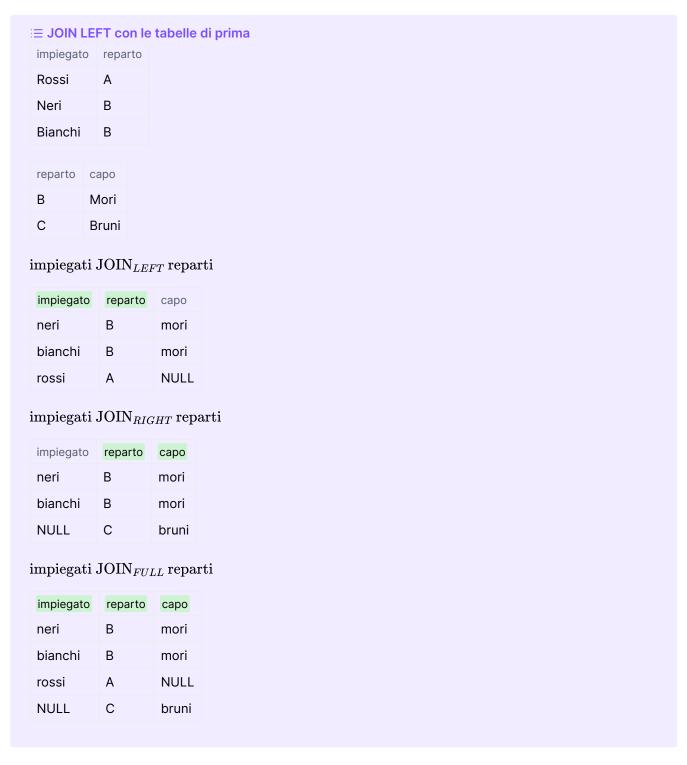
#### **JOIN COMPLETO**

Ogni n-upla contribuisce al risultato. Nessuna viene eliminata. Tuttavia se non troviamo attributi uguali, il join diventa *incompleto*.



Estende con valori NULL le n-uple che verrebbero tagliate fuori da un join interno, si può fare sulla sinistra, destra o completo:

- *sinistro* mantiene tutte le *n*-uple del primo operando;
- destro del secondo operando;
- completo su entrambi gli operandi.



#### **JOIN E PROIEZIONI**

Se prendessimo due tabelle e facessimo INNER JOIN (JOIN NATURALE), con una successiva PROIEZIONE, non e' detto che si ritorni alla tabella originale. Quando il JOIN non e' completo, allora accade.

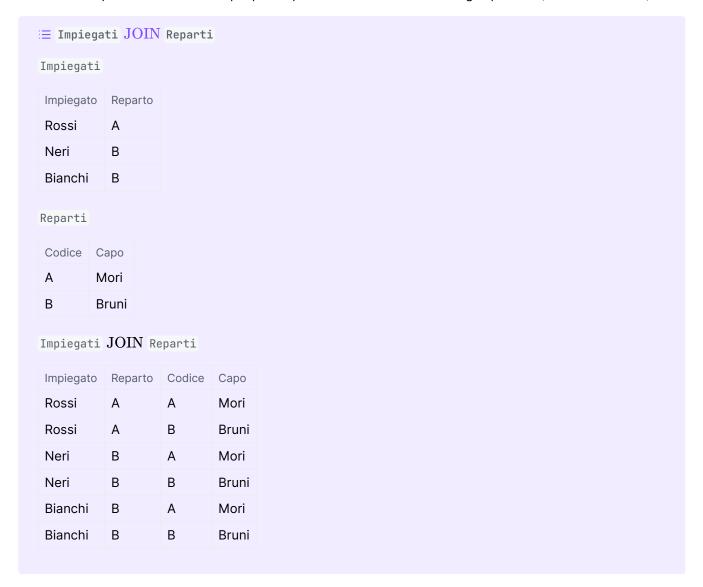
$$\operatorname{PROJ}_{X_1}(R_1 \operatorname{JOIN} R_2) \subseteq R_1$$

Se facessimo l'operazione inversa (prima due PROIEZIONI e poi il JOIN), otterremmo piu'n-uple di quelle di partenza.

### PRODOTTO CARTESIANO ( $\times$ )

Sarebbe un JOIN NATURALE su relazioni senza attributi in comune.

Contiene sempre un numero di n-uple pari al prodotto delle cardinalita' degli operandi (tutte combinabili).



Di solito viene susseguito con un SELECT se vogliamo dargli un senso:

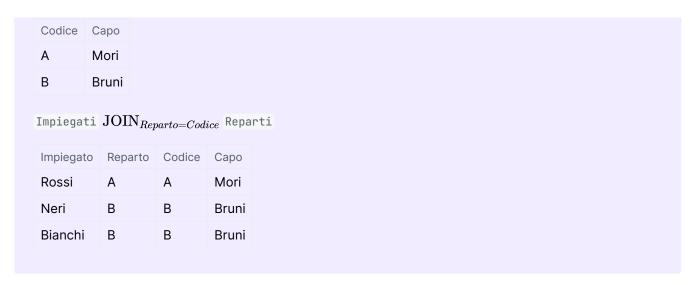
$$\operatorname{SEL}_{condizione}(R_1 \operatorname{JOIN} R_2)$$

L'operazione viene chiamata theta-join ( $R\bowtie_{\theta} S$ ), JOIN con condizione:

$$R_1 ext{ JOIN}_{condizione} R_2$$

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





### **VISTE (:=)**

Sono rappresentazioni dei dati per schema esterno.

- relazioni derivate, cui contenuto è funzione di altre relazioni;
- · relazioni di base, a contenuto autonomo.

Ci sono 2 tipi di relazioni derivate:

- <u>viste materializzate</u>, funzionano molto bene fintanto che le relazioni rimangono costanti nel tempo, ovvero non cambiano troppo frequentemente (che non vedremo);
- <u>relazioni virtuali (viste)</u>, supportate da tutti i DBMS, un'interrogazione su una vista viene eseguita "ricalcolando" la vista;

Rimpiazzare pezzi grossi in un nome che mi dà significato, aiuta nella comprensione delle interrogazioni da farsi. Nello schema esterno ogni utente vede solo:

- ciò che gli interessa;
- · ciò che è autorizzato a vedere.

```
nomeVista_{listaAttributi} := PROJ_{attributi}(Operando) \text{ UNION } \cdots
```

≡ Modello e prezzo di tutti i prodotti costruiti da un produttore

tuttiProdotti(model, price) :=  $PROJ_{model, price}(PC)$  UNION  $PROJ_{model, price}(LAPTOP)$  UNION  $PROJ_{model, price}(PRINTER)$ 

# Esempi esercizi

#### Impiegati

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

• Impiegati che guadagnano piu' di 50 e lavorano a Milano

$${
m SEL}_{stipendio>50~{
m AND}~Filiale='Milano'}(Impiegati)$$

• Matricola e cognome di tutti gli impiegati

$$PROJ_{matricola, cognome}(Impiegati)$$

• Matricola e cognome degli impiegati che guadagnano piu' di 50

$$PROJ_{matricola,cognome}(SEL_{stipendio>50}(Impiegati))$$

#### Impiegati

Matricola	Nome	Eta	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

• Nome e stipendio dei capi degli impiegati che guadagnano piu' di 50

$$\mathrm{PROJ}_{nome, stipendio}(Supervisione\ \mathrm{JOIN}_{capo=matricola}\ (\mathrm{SEL}_{stipendio} > 50 (Impiegati)))$$

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

```
{
m PROJ}_{matricola,nome,stipendio,matricolaC,nomeC,stipendioC} \ ({
m SEL}_{stipendio>stipendioC} \ ({
m REN}_{matricolaC,nomeC,stipendioC\leftarrow matricola,nome,stipendio}(Impiegati) \ {
m JOIN}_{matricolaC=capo} \ ({
m Supervisione\ JOIN}_{impiegato=matricola\ Impiegati})))
```

# Esempi estratti da prove in itinere

• Dato lo schema di relazione R(X), sotto quali condizioni l'espressione dell'algebra relazionale  $\sigma_{A=B}(R)$  è ben definita, cioè non causa un errore?

Nell'algebra relazionale il simbolo = indica la clausola WHERE di SQL. Nessun errore si presenta fintanto che non siano presenti valori NULL.

- Date due tabelle con schemi  $R_1(X_1)$ ,  $R_2(X_2)$ , dove  $X_1 \cup X_2 = \{A\}$ , sapendo che  $\#(r_1) = n$  e  $\#(r_2) = 0$  (cioè l'istanza di R2 è vuota), indicare le cardinalità delle seguenti espressioni dell'algebra relazionale:
  - $R1 \bowtie_{NAT} R2$  (join naturale)  $\rightarrow$  0
  - $R1\bowtie_{LEFT}R2$  (left outer join) ightarrow n
  - $R1 \bowtie_{FULL} R2$  (full outer join)  $\rightarrow$  n + 0
- Fornire un esempio di una coppia di valori (per A e B) per la quale i due predicati ( $A \neq B$ ) e (A IS DISTINCT FROM B) forniscono risultati diversi.

Vedere tabella #NULL\_VALUES

• Date due tabelle con schemi  $R_1(X_1)$ ,  $R_2(X_2)$ , sotto quali condizioni l'espressione dell'algebra relazionale  $R1 \cap R2$  è ben definita, cioè non causa un errore?

Non causa errore fintanto che le due relazioni abbiano la stessa cardinalità.

- Date due tabelle con schemi  $R_1(X_1)$ ,  $R_2(X_2)$ , dove  $X_1 \cap X_2 = \emptyset$ , sapendo che  $\#(r_1) = 0$  e  $\#(r_2) = n_2$  (cioè l'istanza di  $R_1$  è vuota), indicare le cardinalità delle seguenti espressioni dell'algebra relazionale:
  - $R_1 \times R_2$  (prodotto cartesiano)  $\rightarrow$  0
  - $R_1 \bowtie_{RIGHT} R_2$  (right outer join)  $\rightarrow$  n<sub>2</sub>
  - $R_1 \bowtie_{FULL} R_2$  (full outer join)  $\rightarrow$  n<sub>2</sub> + 0
- Date le relazioni  $R(\underline{A}, B^*, C)$  e  $S(\underline{D}, E, F^*)$ , dove #R = n e #S = m, quante ennuple compongono il risultato della query

```
SELECT * FROM R LEFT OUTER JOIN S ON A = D?
```

La cardinalità corrisponde al numero di n-uple di R: n.

Cosa differenzia la proiezione dell'algebra relazionale rispetto a quella implementata in SQL?

La proiezione PROJ dell'algebra relazionale ha la caratteristica di eliminare di default, le n-uple duplicate. La sua implementazione in SQL invece richiede l'aggiunta della clausola <code>DISTINCT</code>.