

# Algebra Relazionale

---

Lucia Ferrari

`lucia02.ferrari@edu.unife.it`

# Introduzione

---

L'insieme principale delle operazioni per il modello relazionale è l'algebra relazionale.

Il risultato di un'espressione è una nuova relazione che può essere usata all'interno di un'altra espressione.

Operazioni relazionali unarie:  
SELECT, PROJECT e RENAME

---

## Operazione: SELECT (selezione)

L'operazione di selezione è utilizzata per scegliere un sottoinsieme di tuple (un insieme di RIGHE) della relazione che soddisfano una certa condizione. Può essere vista come una partizione ORIZZONTALE della relazione in due insiemi: le tuple che soddisfano la condizioni e quelle che non lo fanno.

La sintassi è:

$$\sigma_{\langle \text{condizione di selezione} \rangle}(R),$$

dove  $\sigma$  (sigma) denota l'operatore di selezione e la condizione di selezione è un'espressione booleana (ha valore vero o falso) specificata sugli attributi della relazione  $R$ .

L'espressione booleana all'interno di  $\langle \rangle$  è formata da un certo numero di clausole che possono essere in due forme:

$\langle \text{attribute name} \rangle$  operazione  $\langle \text{constant value} \rangle$  oppure  
 $\langle \text{attribute name} \rangle$  operazione  $\langle \text{attribute name} \rangle$

dove:

- $\langle \text{attribute name} \rangle$  è un attributo di  $R$
- operazione  $\in \{<, \leq, =, \neq, \geq, >\}$
- $\langle \text{constant value} \rangle$  è una costante che appartiene al dominio dell'attributo

Le clausole possono essere connesse tra loro tramite: and ( $\wedge$ ), or ( $\vee$ ), o not ( $\neg$ ).

Ad esempio, considerando la seguente relazione  $R$ :

Nome	Cognome	Salario	Eta
<i>Maria</i>	<i>Rossi</i>	1500	22
<i>Giovanni</i>	<i>Giovanni</i>	900	20
<i>Anna</i>	<i>Bianchi</i>	5000	30
<i>Maria</i>	<i>Maria</i>	1200	45

1) Dopo l'operazione :  $\sigma_{(Salario > 3000)}(R)$  la relazione risultante è:

Nome	Cognome	Salario	Eta
<i>Anna</i>	<i>Bianchi</i>	5000	30

2) Dopo l'operazione :  $\sigma_{(Nome=Cognome) \wedge (Salario > 500)}(R)$  la relazione risultante è:

Nome	Cognome	Salario	Eta
<i>Giovanni</i>	<i>Giovanni</i>	900	20
<i>Maria</i>	<i>Maria</i>	1200	45

L'operatore di selezione è commutativo:

$$\sigma_{\langle \text{cond}_1 \rangle}(\sigma_{\langle \text{cond}_2 \rangle}(R)) = \sigma_{\langle \text{cond}_2 \rangle}(\sigma_{\langle \text{cond}_1 \rangle}(R)).$$

Esempio:

$$\sigma_{\langle \text{Nome} = \text{"Maria"} \rangle}(\sigma_{\langle \text{Eta} > 30 \rangle}(R)) = \sigma_{\langle \text{Eta} > 30 \rangle}(\sigma_{\langle \text{Nome} = \text{"Maria"} \rangle}(R)).$$

Ovvero:

Nome	Cognome	Salario	Eta
<i>Maria</i>	<i>Maria</i>	1200	45



L'operatore di proiezione seleziona alcune COLONNE dalla tabella e ne elimina delle altre.

Può essere vista come una partizione verticale: seleziona solo alcuni attributi della relazione.

La sintassi è:

$$\pi_{\langle \text{lista di attributi} \rangle}(R)$$

dove  $\pi$  (pi) è il simbolo usato per rappresentare l'operatore PROJECT e  $\langle \text{lista di attributi} \rangle$  è la lista di attributi (separati da una virgola) che si vuole ottenere dalla relazione  $R$ .

Il risultato dell'operazione di proiezione contiene solo gli attributi specificati nella  $\langle \text{lista attributi} \rangle$  e nello stesso ordine in cui appaiono nella lista.

Inoltre l'operazione di proiezione RIMUOVE eventuali tuple duplicate (in modo che la relazione rimanga valida)

Ad esempio considerando:

Id	Articolo	Prezzo	Scaffale
1	<i>Torcia</i>	100	1
2	<i>Bilancia</i>	90	2
3	<i>Martello</i>	70	5
4	<i>Martello</i>	100	5

Dopo  $\pi_{Id, Articolo}(R)$  la relazione risultante è:

Id	Articolo
1	<i>Torcia</i>
2	<i>Bilancia</i>
3	<i>Martello</i>
4	<i>Martello</i>

Dopo  $\pi_{Articolo, Scaffale}(R)$  la relazione risultante è:

Articolo	Scaffale
<i>Torcia</i>	1
<i>Bilancia</i>	2
<i>Martello</i>	5

In generale per la maggiorparte di query abbiamo bisogno di applicare più operazioni di diverso tipo una dopo l'altra. Possiamo quindi:

- Scrivere l'operazione come una singola espressione innestando i risultati
- Applicare un'operazione alla volta salvando i risultati uno alla volta.

Per esempio la singola espressione:

$$\pi_{FName, LName, Salary}(\sigma_{DNo=5}(EMPLOYEE)),$$

può essere espressa da:

$$\begin{aligned} DEPT5\_EMPS &\leftarrow \sigma_{DNo=5}(EMPLOYEE) \\ RESULT &\leftarrow \pi_{FName, LName, Salary}(DEPT5\_EMPS), \end{aligned}$$

dove  $\leftarrow$  è l'operatore di assegnazione

## Operatore: RENAME

Se non è applicata nessuna operazione di rinomina il nome degli attributi nella relazione risultante è lo stesso di quelli della relazione di partenza e anche l'ordine (a meno che non venga modificato da un'operazione di proiezione). L'operazione RENAME applicata alla relazione  $R$  con grado  $n$  ha la seguente sintassi:

$$\rho_{S(B_1, \dots, B_n)}(R) \quad \circ \quad \rho_S(R) \quad \circ \quad \rho_{(B_1, \dots, B_n)}(R),$$

dove  $\rho$  (rho) è il simbolo usato per l'operatore RENAME,  $S$  è il nuovo nome della relazione, e  $B_1, \dots, B_n$  sono i nuovi nomi dei suoi attributi.

- La prima espressione rinomina sia la relazione che i suoi attributi
- La seconda solo la relazione
- La terza rinomina solo gli attributi.

Considerando la relazione  $R$ :

Nome	Cognome
<i>Mario</i>	<i>Rossi</i>
<i>Elena</i>	<i>Bianchi</i>

Dopo  $\rho_{S(NomeImp, CognomeImp)}(R)$ , otteniamo la relazione  $S$ :

NomeImp	CognomeImp
<i>Mario</i>	<i>Rossi</i>
<i>Elena</i>	<i>Bianchi</i>

# Algebra relazionale: operazioni insiemistiche

---

Le operazioni insiemistiche (UNIONE, INTERSEZIONE, DIFFERENZA) devono risultati compatibili all'unione, ovvero due relazioni  $R(A_1, \dots, A_n)$  e  $S(B_1, \dots, B_n)$  sono dette compatibili se:

- Hanno lo stesso grado  $n$  (stesso numero di attributi)
- $dom(A_i) = dom(B_i)$ ,  $1 \leq i \leq n$ .

Nella relazione finale per convenzione si tengono i nomi degli attributi di  $R$  (ovvero della prima)

- UNION: Il risultato dell'operazione, denotato da  $R \cup S$ , è una relazione che include tutte le tuple che sono: in  $R$  o in  $S$  oppure in entrambi. Tuple duplicate vengono eliminate.
- INTERSECTION: Il risultato, denotato da  $R \cap S$ , è una relazione che include tutte le tuple che sono sia in  $R$  che in  $S$ .
- SET DIFFERENCE: Il risultato dell'operazione, denotato da  $R - S$ , è una relazione che include tutte le tuple che sono in  $R$  ma non in  $S$ .



Considerando le seguenti relazioni  $R$  e  $S$ :

A	B
$\alpha$	10
$\alpha$	20
$\beta$	30

$R$

C	D
$\alpha$	20
$\beta$	40
$\beta$	50

$S$

Risultati:

A	B
$\alpha$	10
$\alpha$	20
$\beta$	30
$\beta$	40
$\beta$	50

$R \cup S$

A	B
$\alpha$	20

$R \cap S$

A	B
$\alpha$	10
$\beta$	30

$R - S$

- Unione e intersezione sono operazioni commutative:

$$R \cup S = S \cup R \quad \text{and} \quad R \cap S = S \cap R,$$

- Unione e intersezione sono anche associative:

$$R \cup (S \cup T) = (R \cup S) \cup T \quad \text{and} \quad R \cap (S \cap T) = (R \cap S) \cap T.$$

- La differenza NON è commutativa

$$R - S \neq S - R.$$

- L'intersezione può essere espressa tramite unione e differenza:

$$R \cap S = ((R \cup S) - (R - S)) - (S - R).$$

Il prodotto cartesiano, denotato da  $\times$ , è un'altra operazione insiemistica ma non ha bisogno di avere relazioni compatibili.

Il risultato di  $R(A_1, \dots, A_n) \times S(B_1, \dots, B_m)$  è una relazione

$$Q(A_1, \dots, A_n, B_1, \dots, B_m)$$

di grado  $n + m$  con gli attributi nello stesso ordine.

Se  $R$  ha  $n_R$  tuple e  $S$   $n_S$  tuple allora  $R \times S$  avrà  $n_R \cdot n_S$  tuple.

Considerando le relazioni  $R$  e  $S$ :

Nome	Progetti
<i>anna</i>	10
<i>barbara</i>	20
<i>claudia</i>	30

$R$

Codice	Dip
1	Math
2	CS

$S$

Allora:

Nome	Progetti	Codice	Dip
<i>anna</i>	10	1	Math
<i>anna</i>	10	2	CS
<i>barbara</i>	20	1	Math
<i>barbara</i>	20	2	CS
<i>claudia</i>	30	1	Math
<i>claudia</i>	30	2	CS

$R \times S$

## Operazioni relazionali binarie: JOIN e DIVISION

---

## Operazione: JOIN

L'operazione di JOIN effettua un prodotto cartesiano tra due relazioni e poi ci applica una selezione

La sintassi dell'operazione di JOIN tra due relazioni  $R(A_1, \dots, A_n)$  e  $S(B_1, \dots, B_m)$  è:

$$R \bowtie_{\langle \text{join condition} \rangle} S.$$

Il risultato dell'operazione di JOIN è una relazione  $Q$  con  $n + m$  attributi

$$Q(A_1, \dots, A_n, B_1, \dots, B_m)$$

$Q$  ha una tupla per ogni combinazione di tuple di  $R$  e  $S$  che soddisfano la condizione specificata.

Una condizione di join generale è nella forma:

$$\langle \text{cond} \rangle \wedge \langle \text{cond} \rangle \wedge \dots \wedge \langle \text{cond} \rangle,$$

dove ogni  $\langle \text{cond} \rangle$  è nella forma  $A_i \theta B_j$ ,  $A_i$  è un attributo di  $R$ ,  $B_j$  è un attributo di  $S$ ,  $A_i$  e  $B_j$  hanno lo stesso dominio, e  $\theta \in \{<, \leq, =, \neq, \geq, >\}$ .

Una JOIN con queste condizioni è chiamata THETA-JOIN.

Le tuple i cui attributi sono NULL o per cui le condizione di JOIN risulta falsa non appaiono nel risultato.

Quando due condizioni di JOIN hanno unicamente = come operatore sono chiamate EQUI-JOIN.

In questi casi nelle tuple risultanti appaiono sempre due o più coppie di attributi che hanno lo stesso identico valore

Per evitare la presenza di questi valori doppi si può usare una operazione di NATURAL JOIN, indicata da \*. Per poterla utilizzare le due relazioni devono avere degli attributi con lo stesso nome (o è necessaria una operazione di rinomina).



Considerando le due relazioni *Impiegato* e *Dipartimento*:

Nome	Iniziali	Id_dip
<i>Anna</i>	<i>AF</i>	1
<i>Barbara</i>	<i>BG</i>	1
<i>Claudio</i>	<i>CO</i>	3
<i>Davide</i>	<i>DL</i>	2
<i>Emma</i>	<i>EE</i>	2

*Impiegato*

Cod_dip	Indirizzo
1	<i>ViaM30</i>
2	<i>ViaP10</i>

*Dipartimento*

Utilizzando EQUI-JOIN diventa:

Nome	Iniziali	Id_dip	Cod_dip	Indirizzo
<i>Anna</i>	<i>AF</i>	1	1	<i>ViaM30</i>
<i>Barbara</i>	<i>BG</i>	1	1	<i>ViaM30</i>
<i>Davide</i>	<i>DL</i>	2	2	<i>ViaP10</i>
<i>Emma</i>	<i>EE</i>	2	2	<i>ViaP10</i>

$Impiegato \bowtie_{(Impiegato.Id\_dip=Dipartimento.Cod\_dip)} Dipartimento$

Utilizzando JOIN NATURALE diventa:

Nome	Iniziali	Id_dip	Indirizzo
<i>Anna</i>	<i>AF</i>	1	<i>ViaM30</i>
<i>Barbara</i>	<i>BG</i>	1	<i>ViaM30</i>
<i>Davide</i>	<i>DL</i>	2	<i>ViaP10</i>
<i>Emma</i>	<i>EE</i>	2	<i>ViaP10</i>

$\rho_{(Id\_dip,Indirizzo)}(Dipartimento)$

$Impiegato *_{(Impiegato.Id\_dip=Dipartimento.Id\_dip)} Dipartimento$

Può essere mostrato che l'insieme  $\{\sigma, \pi, \rho, \cup, -, \times\}$  è un insieme completo. Ovvero ogni altra operazione dell'algebra relazionale può essere ottenuta con una combinazione di operazioni di questo insieme.

Per esempio:

$$\begin{aligned} R \cap S &\equiv (R \cup S) - ((R - S) \cup (S - R)) \\ R \bowtie_{\langle \text{condition} \rangle} S &\equiv \sigma_{\langle \text{condition} \rangle}(R \times S). \end{aligned}$$

L'operazione di divisione, denotata da  $\div$ , è applicata a due relazioni  $R(Z)$  e  $S(X)$ , dove gli attributi di  $S$  sono un sottoinsieme di quelli di  $R$ ; ovvero  $X \subseteq Z$ .

Prendiamo  $Y = Z - X$ .

Il risultato di  $R(Z) \div S(X)$  è una relazione  $T(Y)$  che include una tupla  $t$  della relazione  $R$  solo se  $t[Y]$  appare combinato con ognuna delle tuple di  $S$

Considerando le due relazioni  $R$  e  $S$ :

A	B
$\alpha$	1
$\beta$	1
$\alpha$	3
$\alpha$	2
$\beta$	2
$\beta$	4

$R$

A
$\alpha$
$\beta$

$S$

Diventa:

B
1
2

$R \div S$

Per ottenere il risultato della divisione:

1. prendiamo gli attributi che non sono presenti in  $S$  e che appaiono in  $R$
2. Prendiamo solo le tuple che hanno quel attributi combinato con TUTTI i valori di  $S$ .

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 *_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 \Join R_2$
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$ .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	$R_1(Z) \div R_2(Y)$

## Altre operazioni relazionali

---

Una funzione di aggregazione  $\mathcal{F}$ , è definita nel seguente modo:

$$\langle \text{grouping attributes} \rangle \mathcal{F}_{\langle \text{function list} \rangle} (R),$$

dove:

- $\langle \text{grouping attributes} \rangle$  è una lista di attributi della relazione R.
- $\langle \text{function list} \rangle$  è una lista di coppie ( $\langle \text{function} \rangle \langle \text{attribute} \rangle$ ).

In ognuna delle coppie:

- >  $\langle \text{function} \rangle$  è una delle funzioni consentite tra: SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT
- >  $\langle \text{attribute} \rangle$  è un attributo della relazione R.

Il risultato finale avrà gli attributi di raggruppamento + i risultati delle funzioni applicate



Considerando  $R$ :

SSN	Name	Salary	Dno
1	A	30000	5
2	B	40000	5
3	C	25000	4
4	D	30000	4
5	E	20000	4
6	F	35000	1

Allora:

Dno	N_Emp	Avg_Sal
5	2	35000
4	3	25000
1	1	35000

Count(SSN)	Average(Salary)
6	30000

2)  $\mathcal{F}_{COUNT(SSN), AVERAGE(Salary)}(R)$

1)  $\rho_{(Dno, N\_Emp, Avg\_Sal)}(Dno \mathcal{F}_{COUNT(SSN), AVERAGE(Salary)}(R))$

1) Calcola la media dei salari considerando i GRUPPI di tuple che hanno lo stesso Dno e conta quante tuple hanno lo stesso valore di Dno

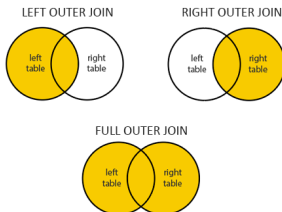
2) Conta tutte le tuple presenti nella relazione R e fa una media del salario. <sup>27/38</sup>

## Operazione: OUTER-JOIN

L'operazione di OUTER-JOIN è utilizzata per effettuare una join che non escluda le tuple che non hanno corrispondenza nell'altra relazione o che hanno valore NULL sulla condizione di join.

Tipologie:

- LEFT-OUTER-JOIN tra  $R$  e  $S$ , denotata da  $R \bowtie_{\langle \text{cond} \rangle} S$ , mantiene tutte le tuple di  $R$  (della tabella di sinistra).
- RIGHT-OUTER-JOIN tra  $R$  e  $S$ , denotata da  $R \bowtie_{\langle \text{cond} \rangle} S$ , tiene tutte le tuple di  $S$  (della tabella di destra).
- FULL-OUTER-JOIN tra  $R$  e  $S$ , denotata da  $R \bowtie_{\langle \text{cond} \rangle} S$ , tiene tutte le tuple sia di  $R$  che di  $S$  (entrambe le tabelle).



Considerando la seguente relazione *Imp* e *Dip*:

Nome	Cod
<i>Andrea</i>	1
<i>Barbara</i>	2
<i>Claudia</i>	4

*Imp*

Cod	Ind
1	<i>ViaA1</i>
2	<i>ViaB2</i>
3	<i>ViaC3</i>

*Dip*

Allora  $Imp \bowtie_{Imp.Cod=Dip.Cod} Dip$  rispettivamente con LEFT-JOIN, RIGHT-JOIN E FULL-JOIN:

Nome	Cod	Ind
<i>Andrea</i>	1	<i>ViaA1</i>
<i>Barbara</i>	2	<i>ViaB2</i>
<i>Claudia</i>	4	NULL

Nome	Cod	Ind
<i>Andrea</i>	1	<i>ViaA1</i>
<i>Barbara</i>	2	<i>ViaB2</i>
NULL	3	<i>ViaC3</i>

Nome	Cod	Ind
<i>Andrea</i>	1	<i>ViaA1</i>
<i>Barbara</i>	2	<i>ViaB2</i>
NULL	3	<i>ViaC3</i>
<i>Claudia</i>	4	NULL

## Esercizi

---

Sia dato il seguente schema relazionale:

*Person*(*name*, *age*, *gender*)

*Frequents*(*name*, *pizzeria*)

*Eats*(*name*, *pizza*)

*Serves*(*pizzeria*, *pizza*, *price*).

Esprimere in algebra relazionale le seguenti interrogazioni senza usare l'operatore di divisione  $\div$ ).

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

1. Determinare i nomi di donne che mangiano la pizza con i funghi o la margherita o entrambe.

2. Determinare i nomi di donne che mangiano sia pizza margherita che funghi.

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

3. Tutte le pizzerie che servono almeno una pizza che Amy mangia e che costa meno di 10 euro.

4. Nomi di pizzerie frequentate da almeno una persona minorenn

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

5. Tutte le pizzerie frequentate solo da donne o solo da uomini

6. Nomi di tutte le pizzerie che Dan non ha mai frequentato e in cui può comprare pizze che mangia.



Person(name, age, gender)  
Frequents(name, pizzeria)  
Eats(name, pizza)  
Serves(pizzeria, pizza, price).

7. Individuare per ogni persona le pizze che mangia e che NON sono servite dalle pizzerie che frequenta. (risultati nella forma: nome persona - pizza)

8. Nomi di tutte le persone che frequentano SOLO pizzerie che servono almeno una pizza che mangiano

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

9. Nomi delle persone che frequentano TUTTE le pizzerie che servono almeno una pizza che mangiano

10. Ricavare il prezzo medio delle pizze

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

11. Ricavare il prezzo medio delle pizze per ognuna delle pizzerie

12. Ricavare il prezzo medio di ogni pizza che mangia Amy (nome pizza - prezzo medio )

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

13. Ricavare il numero di persone che mangia ogni pizza (numero persone - pizza)

14. Ricavare il prezzo della pizza con costo massimo e il prezzo della pizza con costo minimo

Person(name, age, gender)

Frequents(name, pizzeria)

Eats(name, pizza)

Serves(pizzeria, pizza, price).

15. Ricavare per ogni pizzeria la pizza che costa di più e il suo prezzo  
(pizzeria - pizza - prezzo)