

SQL - DDL - QL

/ 23-03

linguaggi di manipolazione dati, in particolare di interrogazione (ricerca dati). Non specifico "cosa voglio", ma una serie di vincoli da cui i dati in output devono soddisfare, co' grattie a un linguaggio didattico, inoltre non specifico il come.

Per specificare quanto sopra ci si basa su un' **algebra relazionale** e permette di ottimizzare le interrogazioni.

Tali linguaggi sono chiusi: stesso input → stesso output
relazioni → **relazione / tabella** **RISULTATO**
tabelle → **VIRTUALE**
(NON MODIFICA TABELE)

La relazione è identificata da R in algebra

In SQL R non è una query, bensì:

SELECT * → cosa voglio estrarre, * = tutto (tutte le tuple)
FROM R

Operatori: (S + i → cambiano l'intestazione) di **QUERY** (**non modifica tabella**)

(P RIDENOMINAZIONE)
Π PROIEZIONE
G SELEZIONE } unari: prendono in input una sola relazione

X PRODOTTO CARTESIANO
U UNIONE
Δ DIFFERENZA } binari

query SPJ: basta
guardare la tupla x
copiare le operazioni,
utilizzando solo
SELECT, FROM, WHERE

Ve ne sono altri derivati da questi sopra:

INTERSEZIONE

JOIN

DIVISIONE



binari
derivati

Significato

- $\rho : R$ $\frac{A \ B \ C}{\dots \ \dots \ \dots}$ voglio combiere: nomi a una o più colonne
RIDENOMINAZIONE



per essere corretto: $A, B \in U_R$, e non posso avere stessa nome di colonna (no: $C \in BC!$)

$N, M \notin U_R$

in SQL:

```
SELECT A as N,  
       B as M,  
       C  
FROM R
```

Note: La ridenominazione, come gli altri operatori, non modifica la tabella. Bensi' presentano il risultato
(la ridenominazione $A, B \in N, \pi$ vale solo x la query corrente, in una successiva ho sempre nomi A e B!).

• Π PROIEZIONE

specifica come risultato solo le proprietà che mi interessano

$\Pi_{A,B}(R)$ proietto R su A,B

R

A	B	C
a	b	3
d	a	5
c	b	27

$A, B \in U_R$

esempio

A	B
a	b
d	a
c	b

grado 2 c
lunghezza
uguale
(4 o tuple)

la proiezione e' unica ma in due casi: estrezzione delle colonne,
eventuale rimozione di duplicati

$\Pi_B(R)$:

B

il grado e' 1 ma le lunghezze e' diverse

b

dell'originale:

a

il risultato e' una relazione, quindi

X

le tuple duplicate vengono rimosse
(ovviamente non succede per le chiavi)

in SQL

possicura che il risultato non contiene duplicati

SELECT (DISTINCT) A, B 1^ fase, estrae

FROM R 2^ fase
elimina
duplicati

se rimovo distinct significa che sono sicuro
di non avere duplicati (\rightarrow l'acetto), ad
ogni modo il sistema non controllo e

risparmia tempo

A,B not null

(posso ometterlo se: $A \neq B \neq AB$ unique)

(DISTINCT compare al mass una volta)

Note: i valori nulli non sono

distinti, vengono collassati

(anche se ha più tuple con un otr.
null)

SELECT B
FROM R

$$\frac{B}{\begin{matrix} b \\ a \\ b \end{matrix}}$$

SELECT DISTINCT B
FROM R

$$\frac{B}{\begin{matrix} b \\ a \end{matrix}}$$

- 6 SELEZIONE (\neq SELECT)

seleziona le colonne

permette di filtrare le righe in
base a dei vincoli (e scatola oltre)

condizione inviolata
alla voglio

$G_p(R)$

SIGMA

R	A	B	C
	a	b	3 ✓
	d	a	5 ✗
	c	b	27 ✗

F: • $A \circledcirc v \quad A \in U_R$ appartenere alla relazione

$v \in \text{dom}(A)$ valore possibile di A (dom compatibili)

$\circledcirc \in \{ =, \neq, >, <, \geq, \leq \}$

"ommettendo
valori comuni"

• $A \circledcirc A' \quad A, A' \in U_R$ attributi

es. $\Rightarrow G_{C=3}(R)$

ottengo

A	B	C
a	b	3 ✓
d	a	5 ✗
c	b	27 ✗

| scorso

• $G_{A>B}(R)$

ottengo

A	B	C
a	b	3
d	a	5
c	b	27

A	B	C
d	a	5
c	b	27

Tutti i operatori possono essere combinati con operazioni logiche (\wedge , \vee , \neg)

$G_{c=3 \wedge A > B} (R)$

A	B	C	=	\emptyset
a	b	3		relazione vuota
d	x	5		
c	b	27		

$G_{c=5 \wedge A > B} (R)$

A	B	C
d	x	5

in SQL

{ SELECT *
FROM R
↓ obbligatori x essere query
WHERE F

Se F atomica (è solo), se invece è composta da \wedge \vee \neg
allora tali simboli si traducono in SQL: AND, OR, NOT

$G_{c=3 \wedge A > B} (R)$ zero. SELECT *
FROM R
WHERE C = 3 AND A > B

E' una definizione induttiva, posso annidarlo quanto voglio, eventualmente applicando parentesi

Posso combinare nella stessa query più operatori:

$\pi_{A,B} (G_{C=5 \wedge A>B} (R))$ prima filtro, poi proiezione

ottengo $A \quad B$, quindi le tuple d'è
 $\downarrow \quad \downarrow$

```
SELECT DISTINCT A, B  
FROM R  
WHERE C = 5 AND A > B
```

Può combinare il contenuto di tabella diverse: operatori binari.

• ~~X~~ PRODOTTO CARTESIANO

R	A	B	C	S	D	E
	a	b	3		d	1
	d	a	5		f	5
	c	b	27			

si tratta di connettere le tuple in tutti i modi possibili:

$R \times S$	A	B	C	D	E	(tuple con copie)
	a	b	3	d	1	
$U_R \cap U_S = \emptyset$ no attributi comuni	a	b	3	f	5	cardinalità
(per scrivere la ridenominazione)	d	a	5	d	1	(no tuple - solo)
	d	a	5	f	5	= prodotto
	c	b	27	d	1	cardinalità
	c	b	27	f	5	

grado $R \times S = \text{somma}$

dei gradi

in SQL

SELECT *

FROM R, S → l'effetto è fare il prodotto cartesiano

è differenza dell'algebra qui SQL non costringe alla ridenominazione
→ SQL trova di default i nomi delle colonne preponendo
quello della relazione: R.A, R.B, R.C, S.D, S.E
nel caso di nomi uguali c'è: R.A, S.A, come
diversi

in algebra:

$$R(A, B, C)$$

$$R \times S_{\text{C=}}(S)$$

$$\begin{array}{c} R \\ \underbrace{A \quad B \quad C} \\ T \end{array}$$

in SQL ottengo

$$\begin{array}{cccccc} A & & B & & D \\ \parallel & & \parallel & & \parallel \\ R.A & R.B & R.C & T.C & T.D \end{array}$$

lascio per non
dare ambiguità

→ in algebra prima ridenomino, poi faccio il prodotto

→ SQL di base parte dal FROM, legge le istruzioni optionali
e poi finisce sul SELECT (con eventuali ridenominazioni)

/06-06

Valori null

ins. degli op. di confronto

;

Dati gli operatori \neq , \geq , \leq ... se un argomento (o tutti e due) sono null allora e₁, Θ e₂ restituisce UNKNOWN
(non soffriamo elice se un argomento è diverso / maggiore .. di null)

es. AND	T	F	U	OR	T	F	U
	T	T	F	U	T	T	T
	F	F	F	(F) \rightarrow l'and è falso	F	T	F
	U	U	(F)	caso un org. falso	U	T	U

$\textcircled{1} \rightarrow$ basta un seg. vero

Se uno tuple viene valutato unknown non viene restituita

Quindi nei confronti non posso dire "e_i = null?" (ottengo unknown)
bensi' le porticelle $e_i \text{ IS NULL}$

SELECT *

FROM film

WHERE voto ≥ 3)
OR
voto < 3) \equiv voto IS NOT NULL

restituisce tutti i film

Altri operatori

• \cup UNIONE



ottengo tutte le tuple di R e di S, ma tali tuple devono essere compatibili, quindi $U_R = U_S$ ovvero hanno stesso schema

R₁

A	B	C
a	b	b
d	a	s
c	b	27

stesso schema =

R₂

A	B	C
a	b	b
j	a	s

ottengo comunque una relazione (non ho tuple duplicate, e b3 solo una volta)

(grado unione = grado di pertenza (stesso n° di ott. tanti))
Cardinalità tra i due somma di tutte le tuple

in SQL

(tutti gli operatori insiemistici lavorano fuori dalla query)
/fuu

SELECT * → prende tutto R₁

FROM R₁

UNION

SELECT *

FROM R₂

} unisco le due
query

→ in SQL se i domini sono compatibili, anche se ho attributi con uguali nomi, si prende come riferimento il primo argomento (in algebra deno rinominare)

→ l'unione togli i duplicati (e quindi assumo che mi interessino)

Monotonie

Una query e' MONOTONA se preso un $\Delta R \subseteq \Delta R'$ ($\forall R_i, R'_i \subseteq R'_i$) allora $Q(\Delta R) \subseteq Q(\Delta R')$
(l'inserimento di una nuova tupla puo' solo aumentare la dimensione, o rimanere uguale, non puo' alterare le tuple già presenti)

Gli operatori visti finora sono monotoni: se aggiunge un ^{input/tupla} argomento il risultato puo' solo aumentare o rimanere uguale.

L'unico operatore non monotono dell'algebra e' la differenza e non puo' descriverla solo con SFW

• \ DIFFERENZA (dom compatibili)



R ₁	R ₂
A B C	A B C
a b z	a b z
d a s	j a s
c b z?	c b z?

folgo tutte le tuple di R₁ che $\in R_2$

caso di non monotonia, se aggiungessi c b z? a R₂ (l'input) il risultato diminuisce (viene preso anche c b z? da R₁)

in SQL

SELECT *

FROM R₁

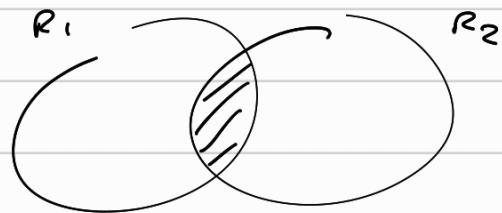
EXCEPT (o MINUS)

SELECT *

FROM R₁

• \cap INTERSEZIONE (o per derivato dalla differenza)

$$R \setminus (R \setminus S) \equiv R \cap S$$



avendo tenuto le tuple comuni

R_1	R_2
A B C	A B C
a b 3	a b 3
d a 5	d a 5
c b 27	c b 27

R_2
A B C
a b 3
d a 5
c b 27

R_1	R_2	$R_1 \cap R_2$
A B C	A B C	a b 3

l'intersezione è monotone

in SQL

SELECT *

FROM R1

INTERSECT

SELECT *

FROM R1

Operatore di join

R	B	C
a	b	3
d	a	5
c	b	27

E	F
1	a
6	k

$G_{n=f} (R \times S)$

A	B	C	E	F
a	b	3	1	a

se volessi combinare le colonne A ed F dove hanno stesso valore

Da qui posso osservare che le tuple di interesse sono poche, mentre il prodotto cartesiano intermedio è grande. Pur potendolo fare.
Una notazione alternativa è il Θ join (theta join)

$$R \text{ } \cancel{\times} \text{ } S = \sigma_{A \neq A'} (R \times S)$$

$A \in U_R$

$A' \in U_S$

Condizione di selezione

gli schemi devono essere distinti

(perché poi vengono sovrapposti, gli altri devono essere diversi, no uguali colonne)

nel caso di confronti di uguaglianze si dice equi - join $R \text{ } \cancel{\times} \text{ } S$
 $A = F$

Note: il theta join si basa su una singola condizione di selezione
(no formule complesse, no OR / AND, solo formule atomiche)
→ altrimenti ha errori sintattici o risultati non attesi!

es.	$R \text{ } \cancel{\times} \text{ } S$	$\begin{array}{cccccc} A & B & C & E & F \\ \hline e & b & 3 & 1 & a \\ d & a & 5 & 1 & a \\ c & b & 27 & 1 & a \\ c & b & 27 & 6 & k \end{array}$
	$C > E$	

in SQL

SELECT *
FROM R, S → prod cartesiano
WHERE C > E → selezione

oppure

SELECT *
FROM R CROSS JOIN S
WHERE C > E

oppure

non ci sono differenze fra le varie notazioni

SELECT *

FROM

R JOIN S

ON C > E

Operatore di join naturale / 05-06

ATTRIBUTI

Assunzione: colonne su stessa nome contengono stessi valori
dunque gli schemi possono non essere disgiunti
(ma devono avere stesso valore x stesso attributi)

Indicato con $| \times |$ o \bowtie senza predici

R \bowtie S

$U_R \cap U_S = \{A_1, \dots, A_n\}$

attr. in comune

prendo $\{B_1, \dots, B_n\}$ attr. nuovi (nuovi \neq)

/ \

$\{B_1, \dots, B_n\} \cap U_R \quad \{B_1, \dots, B_n\} \cap U_S$
 $\neq \emptyset \quad \neq \emptyset$

si produce in:

$$\prod_{U_R \cup U_S} \left(\bigcap_{\substack{A_1 = B_1 \\ \dots \\ A_n = B_n}} (R \times P_{A_1, \dots, A_n \leftarrow B_1, \dots, B_n} (S)) \right)$$

Ottengo un tabella con tutti gli attributi, prendendo
quelli comuni, le entità sono il prodotto cartesiano
che hanno valore uguale di attributo

Esempio

R:

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

S:

B	C	D
b	c	d
b	c	e
a	d	b

possiamo fare $R \times_1 S$? Lo possiamo fare sempre \leftarrow grado
 $= U_R \cup U_S$

$$U_R \cap U_S = \{B, C\}$$

$$\pi_{n, B, C, D} \left(\sigma_{\begin{array}{l} B=B' \\ C=C' \end{array}} \left(R \times \underbrace{\rho_{B, C} \leftarrow_{B', C'} (S)}_{12 \text{ tuple}} \right) \right)$$

↙ ↘

A	B	C	B'	C'	D
a	b	c	b	c	d
a	b	c	b	c	e
a	b	c	b	c	d
d	b	c	b	c	e
c	a	d	d	a	b

copie inutili, la proiezione li taglia

Se disjunti \rightarrow si riduce al prodotto cartesiano

Se stesso schema \rightarrow si riduce all'intersezione

in SQL

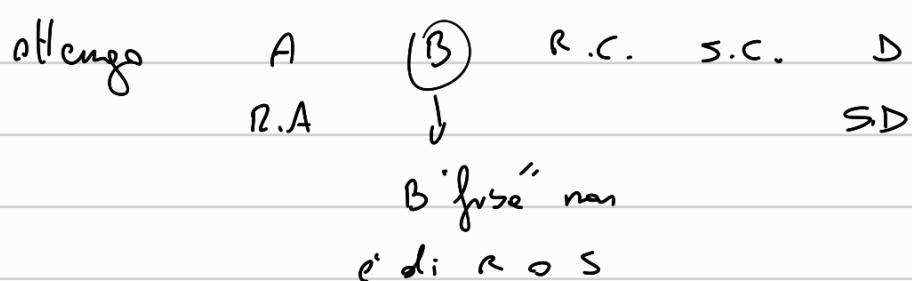
SELECT *
FROM R NATURAL JOIN S

Utile per ottenere relazioni che hanno chiavi esterne fra loro

Comuni e JOIN-USING

E' una specie di natural join (stessi valori in colonne omogenee) ma posso selezionare le colonne (comunque comuni) su cui applicarlo. Ad esempio se ho due relazioni con l'attributo Nome (rigori e contesti diversi), posso eseguire il join su quel-l'attributo

es. SELECT *
from R JOIN S USING (B) $R(A, B, C)$
 $S(B, C, D)$



Order by

Con tale clausola posso scegliere con quale ordine il DB visualizza il risultato. Di base l'ordine è relativo al piano di esecuzione scelto dal DB

Posto in fondo: ORDER BY A₁, A₂, A₃, ... ASC | DEC
 ↓
 puo' essere specificato crescente
 /decrescente per ogni attributo

l'ordine certo! Se ordino prima per A₁ (es. Cognome)

ordino A₂ solo per i valori di A₁ uguali

(Rossi Luigi, Rossi Mario). Se A₁ ha tanti valori distinti,
 gli ordini successivi non contano

Note sul JOIN

Passo avere tuple che non vengono considerate (non selezionate)

Dunque alcune tuple non partecipano al join. Se vogli si tenere
 traccia di tali tuple, invece di fare un INNER JOIN (definito se
 non specificato), effettua un OUTER JOIN (vole x join ti qualsiasi tipo)
 e le colonne che non hanno corrispondenze metti null

A	B	C	B'	C'	D
a	b	c	b	c	d
a	b	c	b	c	e
a	b	c	b	c	d
d	b	c	b	c	e
c	a	d	d	a	b
b	b	f	null	null	null

se non specifico l'outer join
 e' FULL, posso specificare
 LEFT o RIGHT se voglio considerare
 solo le tuple senza corrispondenza
 rispettivamente di R o S
 (full prende entrambe)

non visualizzata

Clausola limit

Specificando **limit** (valore), specifica il n° di tuple visualizzate.

Se ordino e poi applico **limit (1)**, ottengo, supponendo valori differenti per il dato attributo, le tuple con valore di attr. min o max.

↳ ordinamenti il DB sceglie a caso
fra quelli di uguale valore (es un max)