

Operatori Algebrici Relazionali

Studiare anche da Unità 3 – Lezione 3 del vostro libro di testo

Le operazioni relazionali

- Sono le operazioni che consentono di interrogare una base di dati relazionale
- I linguaggi di programmazione utilizzati per l'interrogazione sono di tipo non procedurale e si basano sull'algebra lineare
- Sono uno strumento teorico, ma alla base delle strategie usate per l'interrogazione dei DB

Tipologie di operazioni

Gli operatori dell'algebra relazionale si classificano in **primitivi** e **derivati**.

Gli operatori primitivi sono:

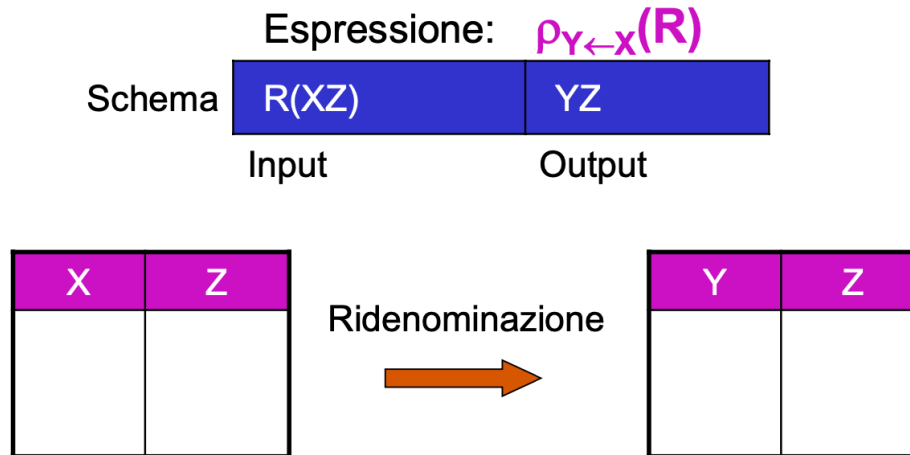
1. **ridenominazione**
2. **unione**
3. **differenza** di relazioni
4. **proiezione** di relazioni
5. **selezione** (o **restrizione**)
6. **prodotto**

Gli operatori derivati sono:

1. **intersezione**
2. **giunzione**

Ridenominazione

- L'operatore di **ridenominazione**, ρ , **modifica lo schema di una relazione, cambiando i nomi di uno o più attributi**
- La definizione formale, oltremodo complessa, si omette; è sufficiente ricordare che $\rho_{Y \leftarrow X}(r)$, con r su $R(XZ)$, cambia lo schema in YZ , lasciando invariati i valori delle tuple, e che nel caso si cambi più di un attributo, allora l'ordine in cui si elencano è significativo



Redditi

CF	Imponibile
BNCGRG78F21A	10000

$\rho_{\text{CodiceFiscale} \leftarrow \text{CF}}(\text{Redditi})$

CodiceFiscale	Imponibile
BNCGRG78F21A	10000

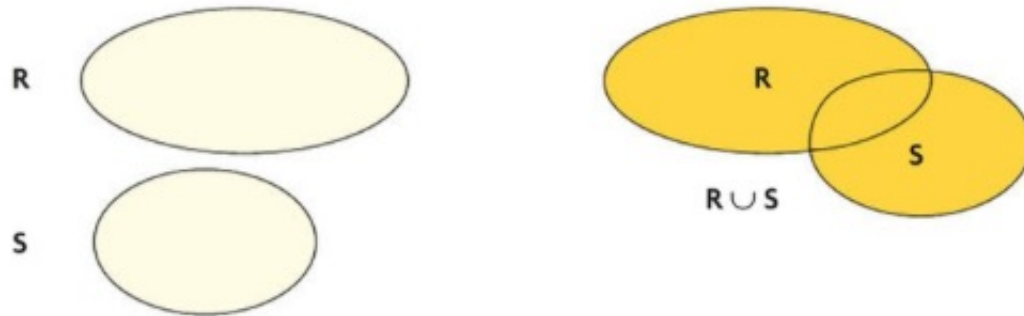
VoliNoSmoking

Numero	Giorno
SC278	28/07/2001
SC315	30/07/2001

$\rho_{\text{Codice, Data} \leftarrow \text{Numero, Giorno}}(\text{VoliNoSmoking})$

Codice	Data
SC278	28/07/2001
SC315	30/07/2001

Unione



Ad esempio, se R rappresenta i clienti del primo semestre di attività di un'azienda e S i clienti del secondo semestre, $R \cup S$ rappresenta i clienti dell'anno.

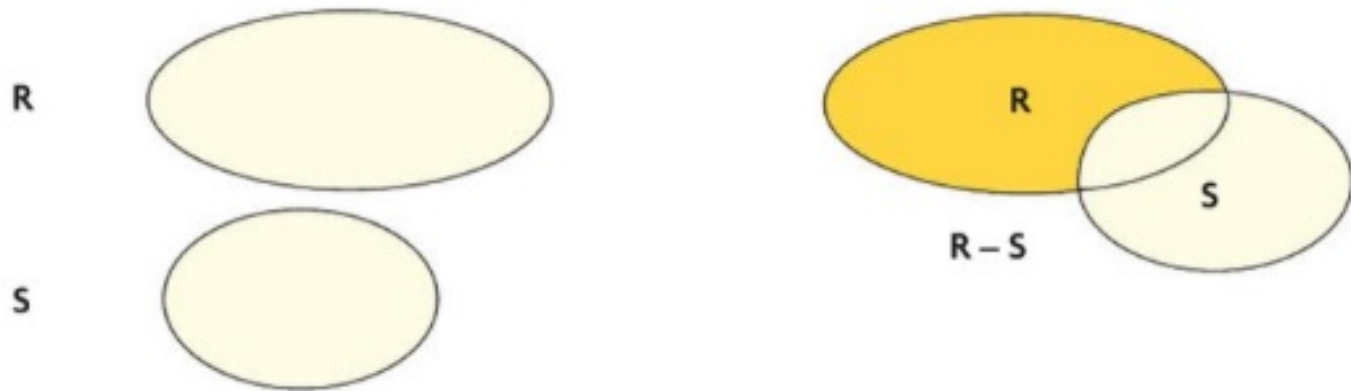
R = Clienti1Semestre				
R	Cognome			
	Rossi			
	Bianchi			
	Verdi			

S = Clienti2Semestre				
S	Cognome			
	Gialli			
	Bianchi			
	Neri			

$R \cup S = \text{Clienti} = \text{Clienti1Semestre} \cup \text{Clienti2Semestre}$

$R \cup S$	Cognome			
	Rossi			
	Bianchi			
	Neri			
	Verdi			
	Gialli			

Differenza



Ad esempio, se R rappresenta tutti i clienti di una certa azienda e S rappresenta i clienti dell'anno 2009, $R - S$ rappresenta tutti i clienti esclusi quelli relativi al 2009.

R = Clienti				
R	Cognome			
	Rossi			
	Bianchi			
	Neri			

S = Clienti09				
S	Cognome			
	Gialli			
	Bianchi			
	Neri			

R - S = Clienti - Clienti09				
R - S	Cognome			
	Rossi			

Selezione

R = Clienti

R	<u>CodCli</u>	Cognome	Nome	Indirizzo	Citta
	C001	Bianchi	Andrea	Via Po, 23	Torino
	C002	Neri	Paolo	Via Roma, 12	Milano
	C003	Verdi	Gianfranco	Via Europa, 150	Torino

↓

S = $\sigma_{Citta="Torino"}(R)$

S	<u>CodCli</u>	Cognome	Nome	Indirizzo	Citta
	C001	Bianchi	Andrea	Via Po, 23	Torino
	C003	Verdi	Gianfranco	Via Europa, 150	Torino

Selezione: esempi (1)

Esami

Matricola	CodCorso	Voto	Lode
29323	483	28	NO
39654	729	30	Sì
29323	913	26	NO
35467	913	30	NO
31283	729	30	NO

$\sigma_{(\text{Voto} = 30) \text{ AND } (\text{Lode} = \text{NO})}(\text{Esami})$

Matricola	CodCorso	Voto	Lode
35467	913	30	NO
31283	729	30	NO

$\sigma_{(\text{CodCorso} = 729) \text{ OR } (\text{Voto} = 30)}(\text{Esami})$

Matricola	CodCorso	Voto	Lode
39654	729	30	Sì
35467	913	30	NO
31283	729	30	NO

Partite

Giornata	Casa	Ospite	GolCasa	GolOspite
4	Venezia	Bologna	0	1
5	Brescia	Atalanta	3	3
5	Inter	Bologna	1	0
5	Lazio	Parma	0	0

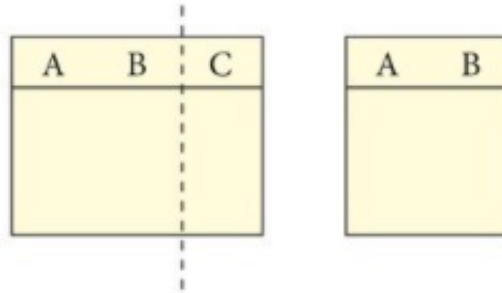
$\sigma_{(Giornata = 5) \text{ AND } (GolCasa = GolOspite)}(Partite)$

Giornata	Casa	Ospite	GolCasa	GolOspite
5	Brescia	Atalanta	3	3
5	Lazio	Parma	0	0

$\sigma_{(Ospite = Bologna) \text{ AND } (GolCasa < GolOspite)}(Partite)$

Giornata	Casa	Ospite	GolCasa	GolOspite
4	Venezia	Bologna	0	1

Proiezione



Consideriamo la seguente relazione CLIENTI relativa ai clienti di un'azienda e supponiamo di voler estrapolare solo il cognome e il nome dei clienti.

R = Clienti

R	<u>CodCli</u>	Cognome	Nome	Indirizzo	Citta
	C001	Bianchi	Andrea	Via Po, 23	Torino
	C002	Neri	Paolo	Via Roma, 12	Milano
	C003	Verdi	Gianfranco	Via Europa, 150	Torino

↓

S = $\pi_{\text{Cognome, Nome}}(R)$

S	Cognome	Nome
	Bianchi	Andrea
	Neri	Paolo
	Verdi	Gianfranco

Card(S) ≤ Card(R)

Card(S) ≤ Card(R), infatti le tuple presenti nella proiezione possono anche essere di numero inferiore a quelle di R, poiché le tuple duplicate vengono scartate.

Corsi

CodCorso	Titolo	Docente	Anno
483	Analisi	Biondi	1
729	Analisi	Neri	1
913	Sistemi Informativi	Castani	2

$\pi_{\text{CodCorso, Docente}}(\text{Corsi})$

CodCorso	Docente
483	Biondi
729	Neri
913	Castani

$\pi_{\text{CodCorso, Anno}}(\text{Corsi})$

CodCorso	Anno
483	1
729	1
913	2

Corsi

CodCorso	Titolo	Docente	Anno
483	Analisi	Biondi	1
729	Analisi	Neri	1
913	Sistemi Informativi	Castani	2

$\pi_{\text{Titolo}}(\text{Corsi})$

Titolo
Analisi
Sistemi Informativi

$\pi_{\text{Docente}}(\text{Corsi})$

Docente
Biondi
Neri
Castani

Congiunzione

Intersezione

Date due relazioni compatibili R e S, l'**intersezione** di R e S restituisce la relazione composta da tutte le tuple presenti sia in R sia in S.

Scriveremo:

$$R \cap S = \{ t \mid t \in R \text{ AND } t \in S \}$$

Supponiamo di avere le informazioni relative ai clienti del 2008 e a quelli del 2009 della nostra azienda. Vogliamo ottenere una tabella con le persone che sono state nostre clienti sia nel 2008 sia nel 2009.

R = Clienti08

R	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C006	Bianchi	MI	Via Po, 23
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

S = Clienti09

S	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C016	Verdi	CO	Via Moro, 13
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

$R \cap S$

$R \cap S$	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

Esercizio

trova unione, intersezione, differenza fra laureati e quadri

laureati

Matr	Cognome	Età
7345	Bianchi	37
3492	Verdi	40
3877	Neri	25

quadri

Matr	Cognome	Età
7111	Viola	50
3492	Verdi	40
3877	Neri	25

E indica, per ciascuna relazione risultante,
GRADO E CARDINALITA'

Soluzione

laureati

Matr	Cognome	Età
7345	Bianchi	37
3492	Verdi	40
3877	Neri	25

quadri

Matr	Cognome	Età
7111	Viola	50
3492	Verdi	40
3877	Neri	25

laureati \cup quadri

Matr	Cognome	Età
7111	Viola	50
3492	Verdi	40
3877	Neri	25
7345	Bianchi	37

UNIONE

Grado = 3

CARD=4

laureati \cap quadri

Matr	Cognome	Età
3492	Verdi	40
3877	Neri	25

INTERSEZIONE

Grado = 3

CARD=2

laureati - quadri

Matr	Cognome	Età
7345	Bianchi	37

DIFFERENZA

Grado = 3

CARD=1

Join

Tra le operazioni derivate, quelle che rivestono maggiore utilità sono quelle di giunzione (*join*) che consentono di costruire una relazione partendo da due relazioni e applicando uno specifico criterio di restrizione sul loro prodotto cartesiano. In base alla natura specifica del criterio di restrizione e al risultato che si intende ottenere, si distinguono diversi tipi di join:

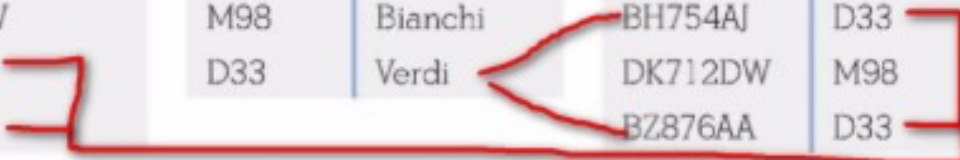


Il θ -join è la combinazione di prodotto Cartesiano e selezione

Equi-Join

- È la corrispondenza di valori uguali per attributi comuni nelle due tabelle → ridondante
- La/e colonna/e che sono confrontate per ricercare valori identici vengono duplicate

			Proprietari		Auto	
Cod_F	cognome	targa	Cod_P	cognome	targa	Cod_P
P45	Rossi	AY888BD	P45	Rossi	AY888BD	P45
M98	Bianchi	DK712DW	M98	Bianchi	BH754AJ	D33
D33	Verdi	BH754AJ	D33	Verdi	DK712DW	M98
D33	Verdi	BZ876AA			BZ876AA	D33



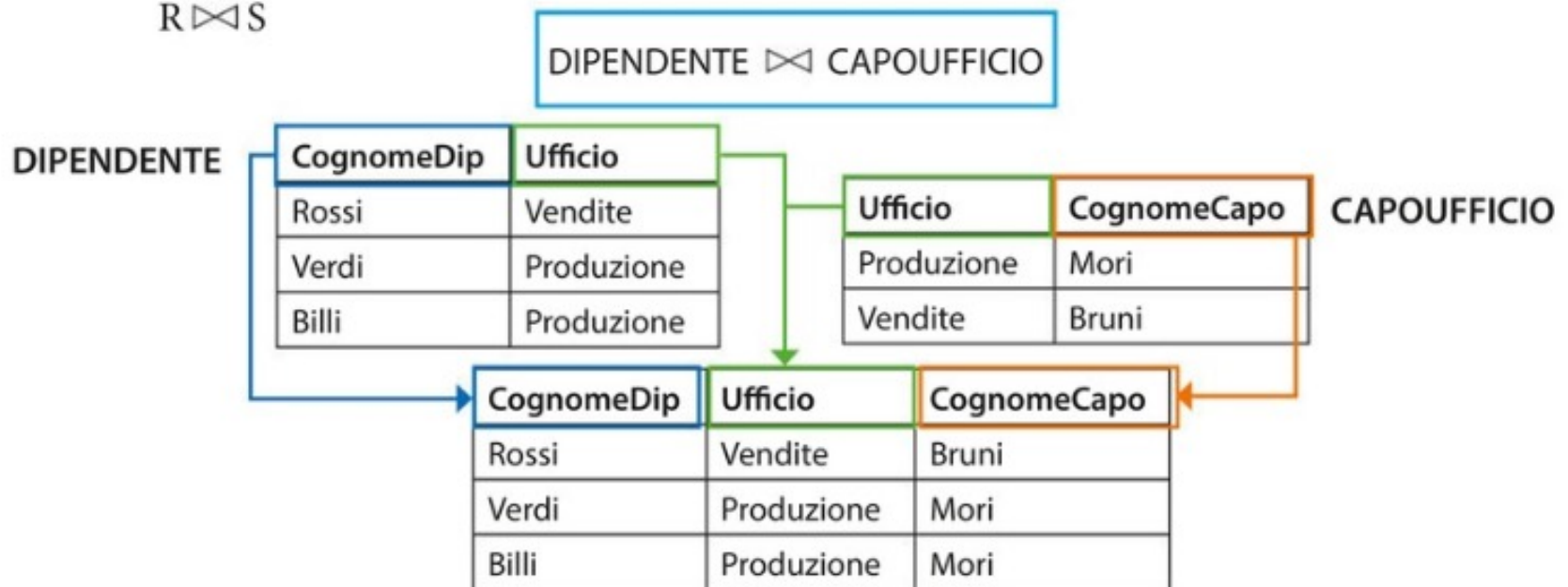
Natural Join

È un operatore che correla i dati di due relazioni R e S sulla base di **valori uguali in attributi con lo stesso nome** in R e S definiti sugli stessi domini. La relazione risultante:

- ha per attributi l'unione degli attributi delle relazioni di partenza;
- le sue tuple sono ottenute selezionando le tuple delle relazioni con valori uguali negli attributi comuni.

In simboli:

$$R \bowtie S$$



Outer join

Outer join

Cerchiamo di comprendere, ora, in che cosa consiste il **join esterno** (*outer join*). Per poterne spiegare al meglio le funzionalità e il significato, serviamoci di un esempio basato sulle due relazioni *Persona* e *Automobile*.

PERSONA

<u>Codice</u>	Nome	Cognome
1	Mario	Rossi
2	Luigi	Bianchi
3	Giuseppe	Neri

AUTOMOBILE

<u>Targa</u>	Modello	Codice
AASSGG	Tipo1	1
UUJJHH	Tipo2	1
PPLLBB	Tipo3	2
WWYYXX	Tipo4	

Effettuiamo un join naturale sul campo *Codice* comune alle due relazioni. Otteniamo la tabella:

Targa	Modello	Nome	Cognome
AASSGG	Tipo1	Mario	Rossi
UUJJHH	Tipo2	Mario	Rossi
PPLLBB	Tipo3	Luigi	Bianchi

Particolare attenzione deve essere rivolta al ruolo che i campi nulli ricoprono negli inner join. L'automobile di targa *WWYYXX*, così come *Giuseppe Neri*, non sono presenti nella tabella risultante dall'inner join. Se avessimo voluto informazioni sui proprietari di tutte le automobili, avremmo cercato il risultato visibile qui a lato.

Targa	Modello	Codice
AASSGG	Tipo1	1
UUJJHH	Tipo2	1
PPLLBB	Tipo3	2
WWYYXX	Tipo4	Null

L'operazione di giunzione naturale esclude automaticamente tutti i record aventi valore nullo lungo la colonna impiegata come elemento di giunzione.
Per includere anche tali valori, è stato messo a punto il join esterno (*outer join*).

Right outer Join

Right outer join

Date due relazioni A e B, il **right outer join** restituisce una relazione selezionando tutte le tuple di A che corrispondono con B, più i record di B (ossia la relazione di destra) che non corrispondono. Le tuple che non corrispondono vengono valorizzate a NULL.

Targa	Modello	Nome	Cognome
AASSGG	Tipo1	Mario	Rossi
UUJJHH	Tipo2	Mario	Rossi
PPLLBB	Tipo3	Luigi	Bianchi
Null	Null	Giuseppe	Neri

Utilizzando un right outer join fra le tabelle *Automobile* e *Persona* sul comune campo *Codice* si ottiene il risultato visibile a lato.

Come è facile osservare, fa parte dell'insieme delle righe restituite anche *Giuseppe Neri*. Il suo record proviene dalla relazione di destra (*right*, appunto). Benché non abbia corrispondenze con la relazione di sinistra, è stato ugualmente incluso. Chiaramente, non è possibile determinare dei valori significativi per i due campi *Targa* e *Modello*, pertanto il DBMS ha completato la riga servendosi di due valori nulli, scelti appositamente per rappresentare l'assenza di informazione.

Left outer join

Left outer join

Date due relazioni A e B, il **left outer join** restituisce una relazione selezionando tutte le tuple di A che corrispondono con B, più i record di A (ossia la relazione di sinistra) che non corrispondono. Le tuple che non corrispondono vengono valorizzate a NULL.

Utilizzando un left outer join si ottiene il risultato visibile a lato. Anche in questo caso, i campi che non hanno corrispondenze sono stati completati con dei valori nulli.

Targa	Modello	Nome	Cognome
AASSGG	Tipo1	Mario	Rossi
UUJJHH	Tipo2	Mario	Rossi
PPLLBB	Tipo3	Luigi	Bianchi
WWYYXX	Tipo4	Null	Null

Full outer join

Full outer join

Il full outer join applica contemporaneamente sia il left outer join sia il right outer join. Utilizzando un full outer join si ottiene il seguente risultato:

Targa	Modello	Nome	Cognome
AASSGG	Tipo1	Mario	Rossi
UUJJHH	Tipo2	Mario	Rossi
PPLLBB	Tipo3	Luigi	Bianchi
Null	Null	Giuseppe	Neri
WWYYXX	Tipo4	Null	Null

Outer join: esempi

- Esistono tre varianti

- **Left** ($=\triangleright\triangleleft$):

- **Right** ($\triangleright\triangleleft=$):

- **Full** ($=\triangleright\triangleleft=$):

Ricercatori

Nome	CodProgetto
Rossi	HK27
Bianchi	HK27
Verdi	HK28

Progetti

CodProgetto	Responsabile
HK27	Bianchi
HAL2000	Neri

Ricercatori $=\triangleright\triangleleft$ Progetti

Nome	CodProgetto	Responsabile
Rossi	HK27	Bianchi
Bianchi	HK27	Bianchi
Verdi	HK28	NULL

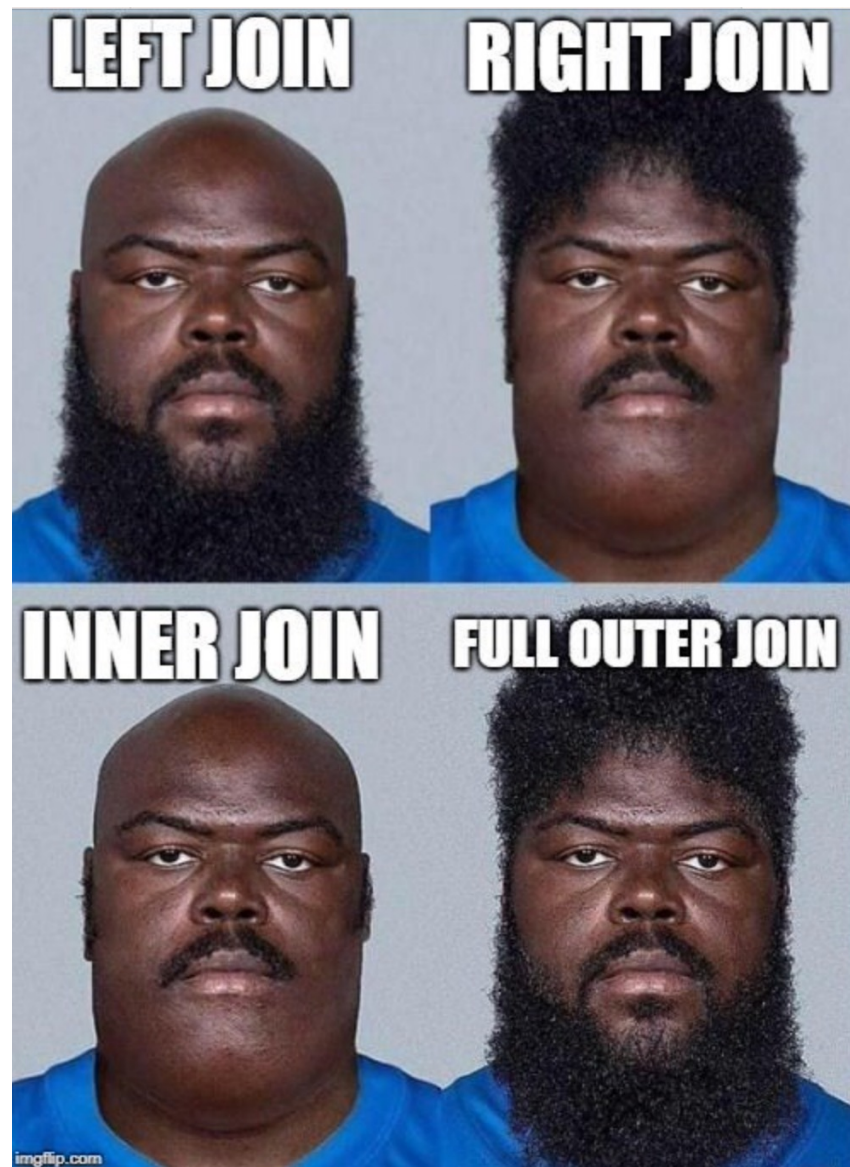
Ricercatori $\triangleright\triangleleft=$ Progetti

Nome	CodProgetto	Responsabile
Rossi	HK27	Bianchi
Bianchi	HK27	Bianchi
NULL	HAL2000	Neri

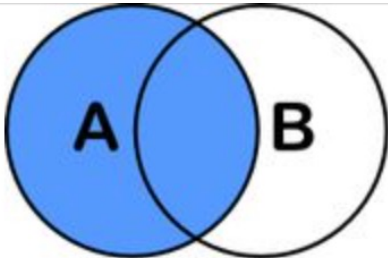
Ricercatori $=\triangleright\triangleleft=$ Progetti

Nome	CodProgetto	Responsabile
Rossi	HK27	Bianchi
Bianchi	HK27	Bianchi
Verdi	HK28	NULL
NULL	HAL2000	Neri

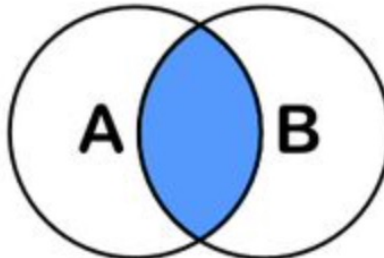
Riassunto col meme per memorizzare



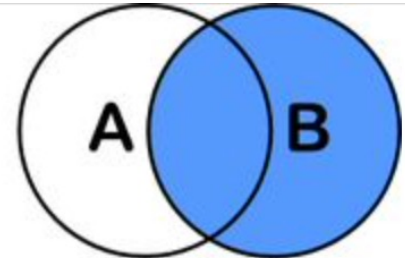
CHEATSHEET
**SQL
JOINS**



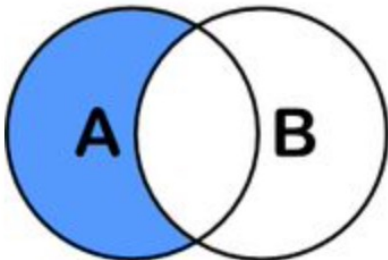
```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```



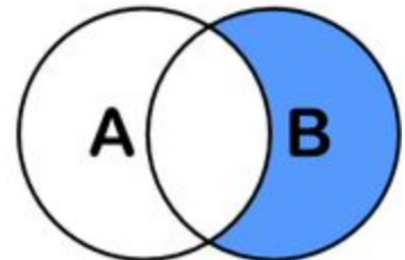
```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



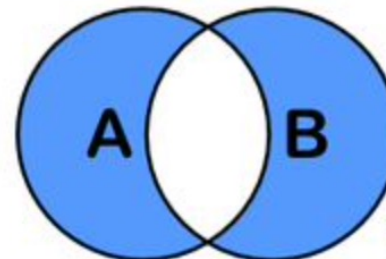
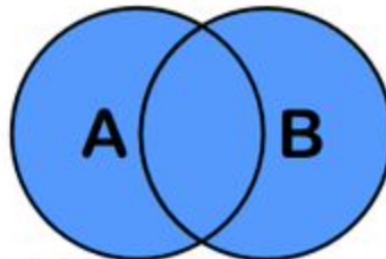
```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```



```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```



```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```

```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```



Esempio concreto in PHPMYADMIN

Il full joins in MySQL non esiste 😱😱😱

Soluzione:

Usa UNION

```
SELECT * FROM t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM t1
RIGHT JOIN t2 ON t1.id = t2.id
```

* Con **union all** vengono mantenuti i duplicati UNION

- L'operatore UNION viene utilizzato per combinare il set di risultati di due o più istruzioni SELECT.
- Ogni istruzione SELECT all'interno di UNION deve avere lo stesso numero di colonne
- Le colonne devono avere anche tipi di dati simili
- Anche le colonne in ogni istruzione SELECT devono essere nello stesso ordine
- Con UNION i duplicati **vengono eliminati (eccetto con UNION ALL*)**

```
SELECT 'Customer' AS Type, ContactName, City, Country
FROM Customers
UNION
SELECT 'Supplier', ContactName, City, Country
FROM Suppliers;
```

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

Esercizio da svolgere in lab 20-12-23

- Creare le seguenti tabelle in phpmyadmin

maternita

madre	figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

paternita

padre	figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

persone

nome	eta	reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Query da eseguire

- Mostra le persone che guadagnano più dei rispettivi padri e monstra nome, reddito e reddito del padre

```
select f.nome, f.reddito, p.reddito
from   persone p, paternita t, persone f
where  p.nome = t.padre and
       t.figlio = f.nome and
       f.reddito > p.reddito
```

```
select f.nome, f.reddito, p.reddito
from   persone p join paternita t on
                                p.nome = t.padre
       join persone f on t.figlio = f.nome
where  f.reddito > p.reddito
```

Query da eseguire

- Mostra il padre e se nota la madre per ogni figlio

```
select paternita.figlio, padre, madre  
from    paternita left outer join maternita  
        on paternita.figlio = maternita.figlio
```

NOTA: “outer” è opzionale

```
select paternita.figlio, padre, madre  
from    paternita left join maternita  
        on paternita.figlio = maternita.figlio
```

Query da eseguire

- Mostra il padre e la madre per ogni figlio di cui vedo il nome (NO DUPLICATI)

```
select padre as genitore, figlio
from paternita
union
select madre as genitore, figlio
from maternita
```


Prova a vedere come cambia con full/right/left join

Outer join, esempi

```
select paternita.figlio, padre, madre
from   maternita join paternita
       on maternita.figlio = paternita.figlio
```

```
select paternita.figlio, padre, madre
from   maternita left outer join paternita
       on maternita.figlio = paternita.figlio
```

```
select paternita.figlio, padre, madre
from   maternita right outer join paternita
       on maternita.figlio = paternita.figlio
```

```
select nome, padre, madre
from   persone full outer join maternita on
       persone.nome = maternita.figlio
       full outer join paternita on
       persone.nome = paternita.figlio
```