

# Basi di dati

Corso di laurea in Ingegneria Informatica  
Scuola di Ingegneria - Università di Pisa

---

**Oracle MySQL**  
A.A. 2016-2017

Ing. Francesco Pistolesi

*Postdoctoral Researcher*

Dipartimento di Ingegneria dell'Informazione  
[francesco.pistolesi@iet.unipi.it](mailto:francesco.pistolesi@iet.unipi.it)

# Correlated subquery nella having clause

---



# Correlated subquery nella having clause

---

Una correlated subquery può essere usata anche per esprimere una  
**condizione sui gruppi che riferisce la outer query**

→ **SELECT**  
**HAVING**  
**SELECT**

può riferire solo attributi presenti nella  
select list della outer query

# A volte ritornano...

---

Indicare le specializzazioni che hanno **solo medici della stessa città**

# Soluzione

---

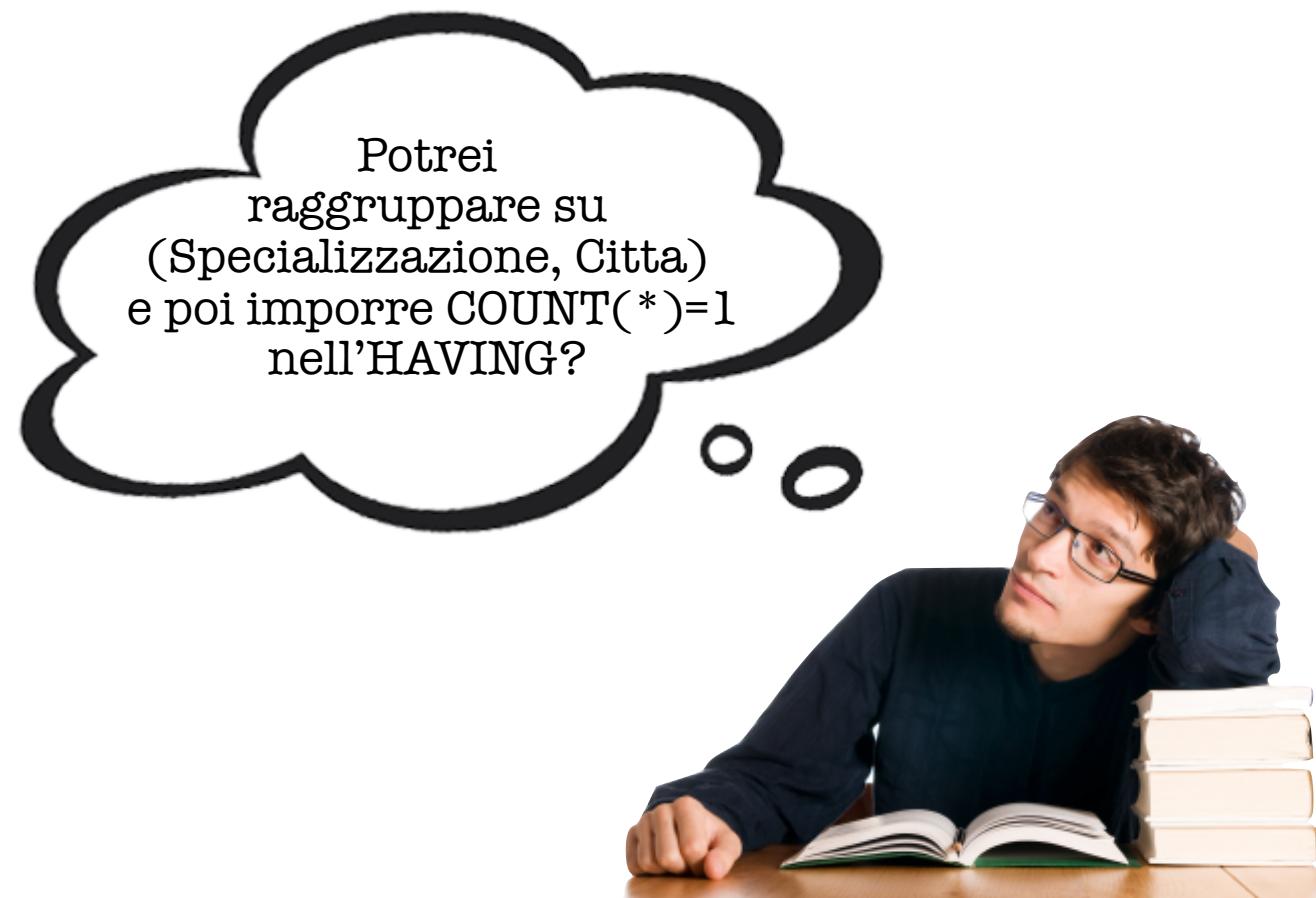
Indicare le specializzazioni che hanno **solo medici della stessa città**

```
SELECT Specializzazione  
FROM Medico  
GROUP BY Specializzazione  
HAVING COUNT(DISTINCT Citta) = 1;
```

# Il dubbio...

---

Indicare le specializzazioni che hanno **solo medici della stessa città**



# Vediamo...

---

Indicare le specializzazioni che hanno **solo medici della stessa città**

**MEDICO**

Matricola	Cognome	Nome	Specializzazione	Parcella	Città
18339	Verdi	Paolo	Ortopedia	150	Pisa
35512	Rossi	Marta	Ortopedia	120	Pisa
16220	Gialli	Rita	Cardiologia	135	Roma
58663	Turchesi	Fulvio	Cardiologia	155	Pisa
29858	Neri	Rino	Dermatologia	110	Siena

cosa vuol dire raggruppate su (Specializzazione, Città)?

# Soluzione

---

Indicare le specializzazioni che hanno **solo medici della stessa città**

~~SELECT Specializzazione  
FROM Medico  
GROUP BY Specializzazione, Citta  
HAVING COUNT(\*) = 1~~



**MEDICO**

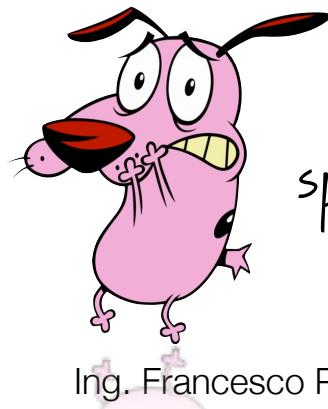
Matricola	Cognome	Nome	Specializzazione	Parcella	Citta
18339	Verdi	Paolo	Ortopedia	150	Pisa
35512	Rossi	Marta	Ortopedia	120	Pisa
16220	Gialli	Rita	Cardiologia	135	Roma
58663	Turchesi	Fulvio	Cardiologia	155	Pisa
29858	Neri	Rino	Dermatologia	110	Siena

# Qual è la condizione realmente espressa?

---

Indicare le specializzazioni che hanno **solo medici della stessa città**

```
SELECT Specializzazione  
      FROM Medico  
     GROUP BY Specializzazione, Citta  
    HAVING COUNT(*) = 1
```



specializzazioni aventi un solo medico per almeno una città

# Soluzione corretta

---

Indicare le specializzazioni che hanno **solo medici della stessa città**

```
SELECT M.Specializzazione  
FROM Medico M  
GROUP BY M.Specializzazione, M.Citta  
HAVING COUNT(*) = ( SELECT COUNT(*)  
                      FROM Medico M2  
                      WHERE M2.Specializzazione = M.Specializzazione );
```

correlated subquery

benché corretta, MySQL fornisce un risultato anomalo

# Soluzione corretta

Indicare le specializzazioni che hanno **solo medici della stessa città**



a causa di un bug, occorre correlare la subquery  
nella having clause a un alias, ridenominando  
l'attributo di correlazione nella outer query

```
SELECT M.Specializzazione AS Spec  
FROM Medico M  
GROUP BY M.Specializzazione, M.Citta  
HAVING COUNT(*) = (
```

```
SELECT COUNT(*)  
FROM Medico M2  
WHERE M2.Specializzazione = M.Spec );
```

correlated subquery

numero di medici della stessa  
specializzazione di M

# Fondamentale

---

Una correlated subquery nella  
having clause

può essere correlata

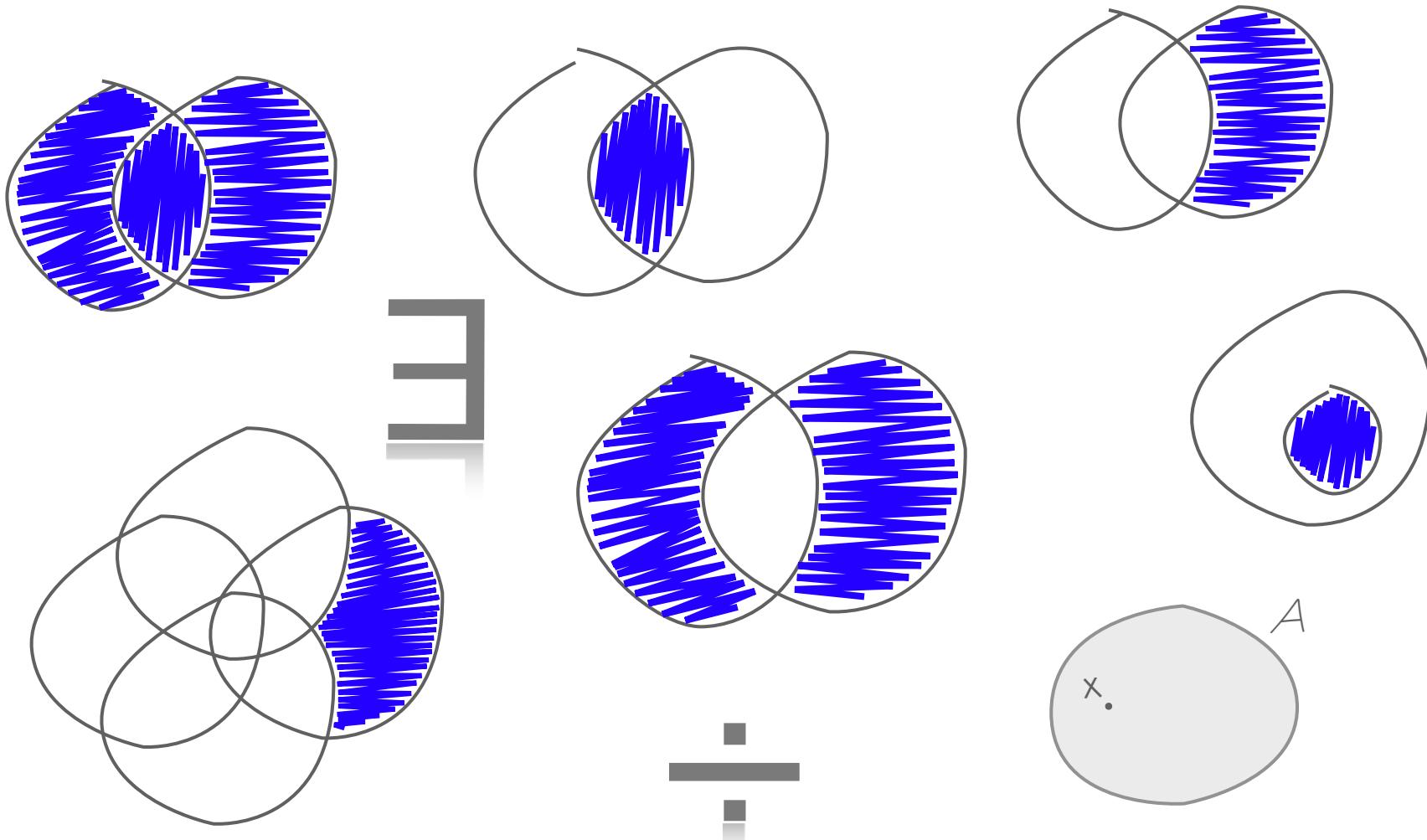
solo ad

alias nella select list

della outer query

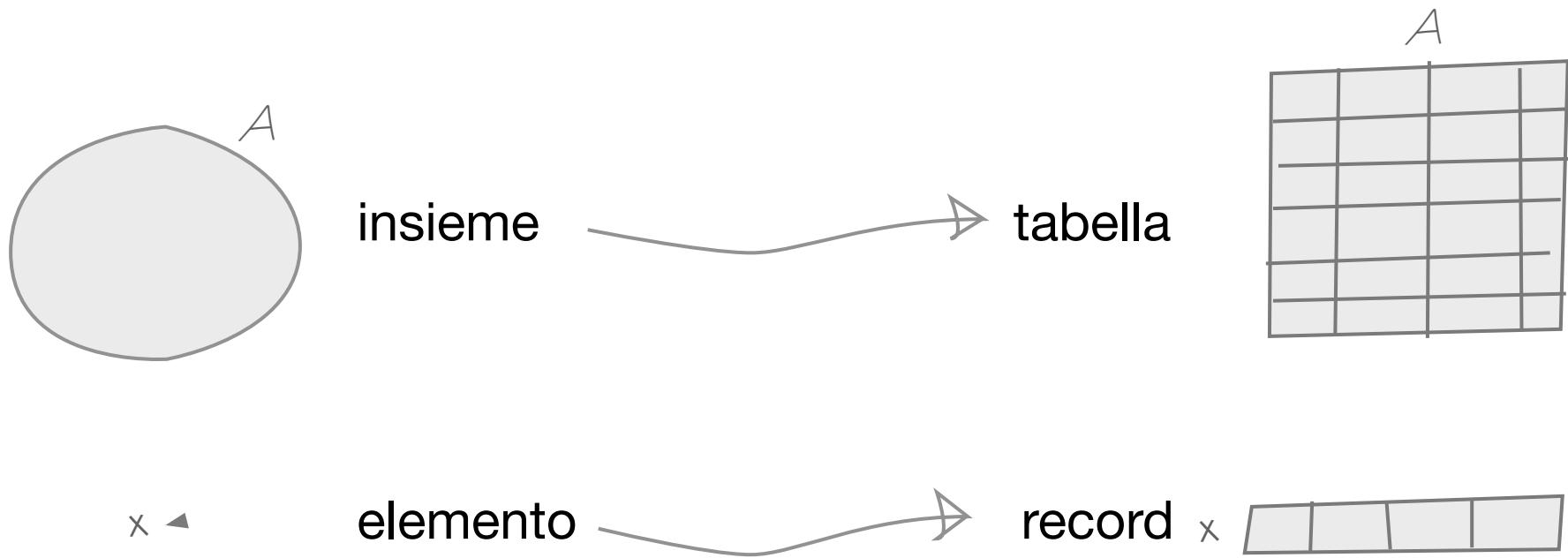
# Query insiemistiche

---



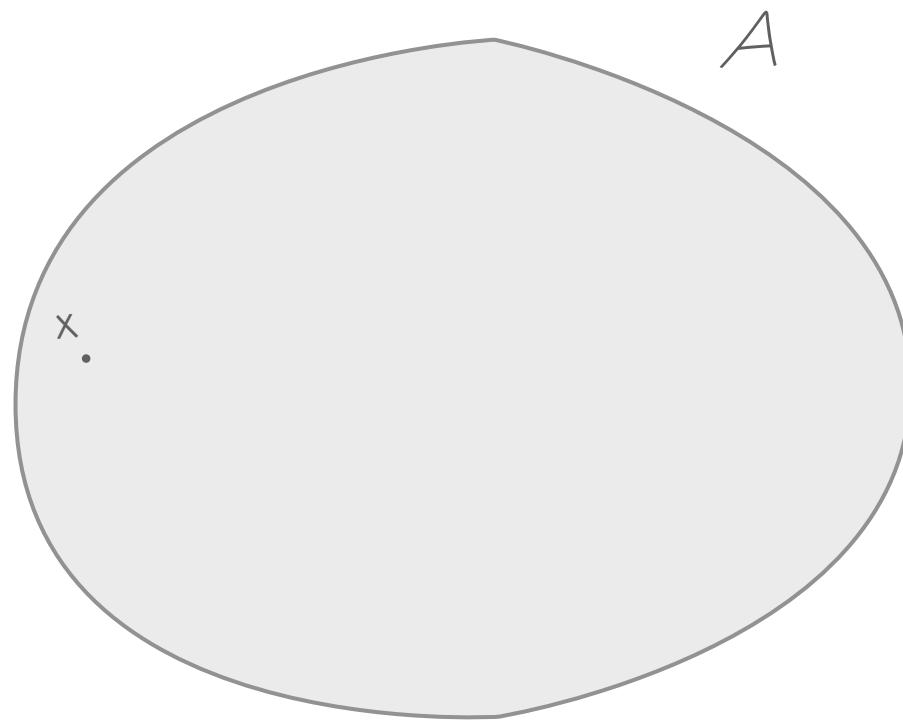
# Corrispondenze

---



# Appartenenza

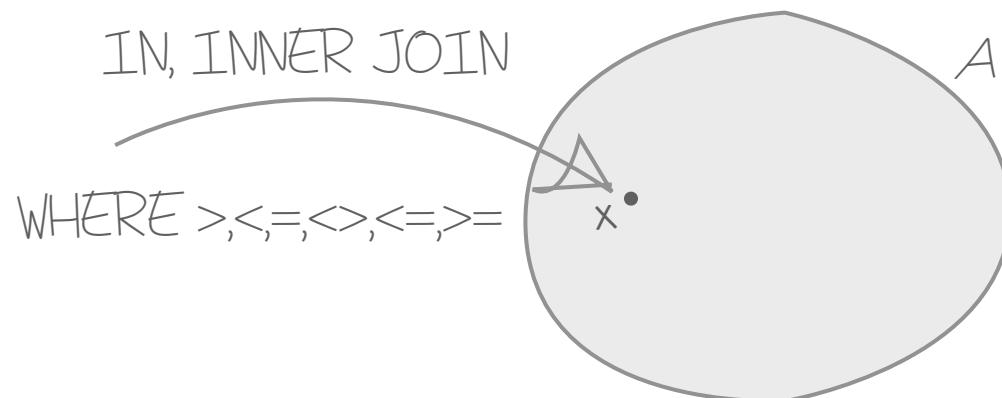
---



# Appartenenza

---

- un elemento  $x$  appartiene a un insieme  $A$  se è contenuto in esso
- introdotta da **proposizioni relative**
- in MySQL il controllo di appartenenza si effettua con **IN**, con operatori di confronto nel WHERE, oppure mediante **INNER JOIN**



# Appartenenza: esempi

---

- i medici **che**...
- il numero di visite **effettuate** dai medici di Pisa...
- i pazienti **aventi** il reddito più alto del reddito medio...



modi alternativi per dire che qualcosa deve appartenere a un certo insieme

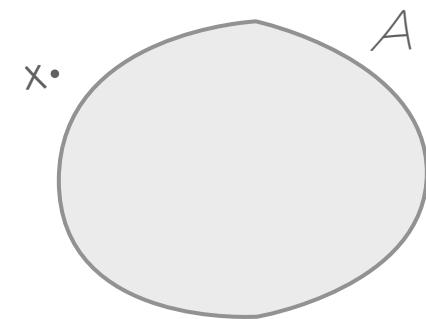


# Esclusione

---

- è il contrario dell'appartenenza
- introdotta da **proposizioni relative negative**
- in MySQL si impone con **NOT IN**, con operatori di confronto nel WHERE, oppure con **OUTER JOIN**

se si ha poca confidenza con i join esterni, usate "not in"



# Esclusione: esempi

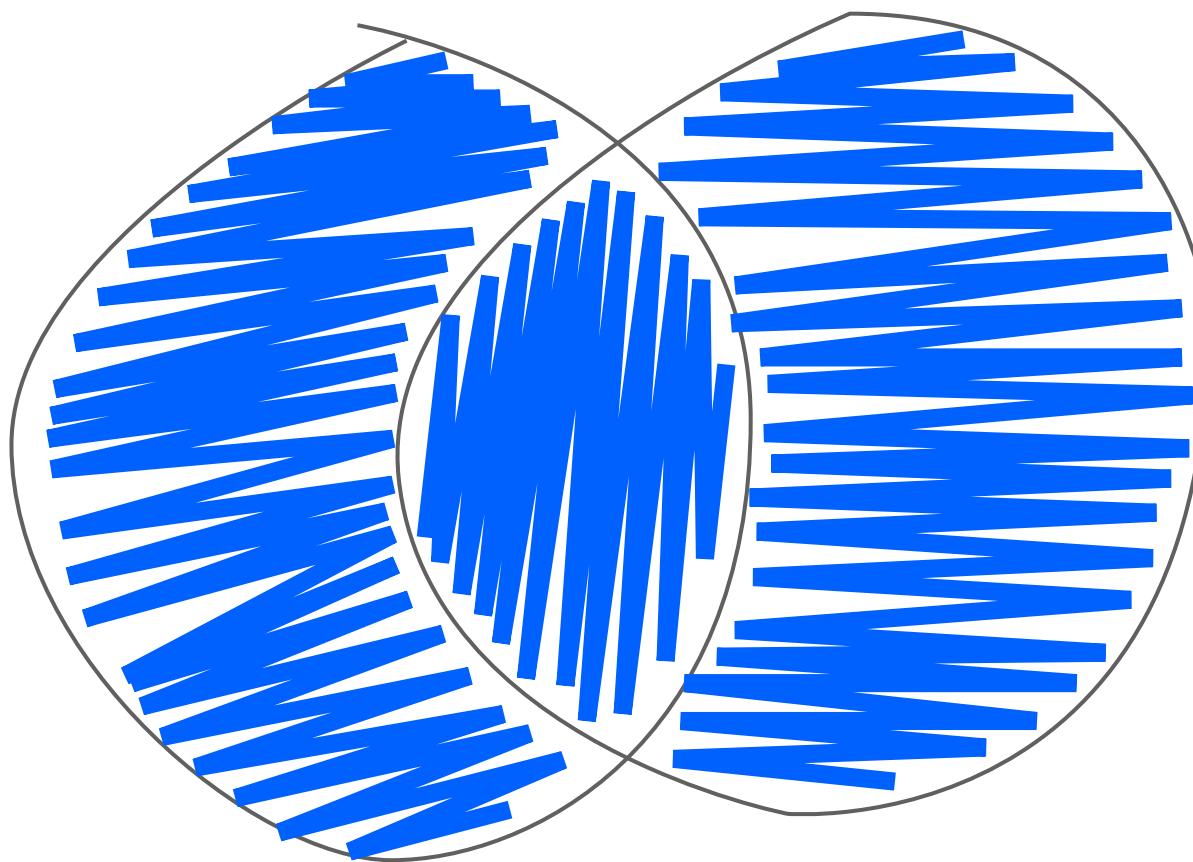
---

- i medici **che non...**
- il numero di visite **non effettuate** dai medici di Pisa...
- i pazienti **che non hanno superato** dieci visite...



# Unione

---



# Unione

---

- **unisce** i risultati espressi da più condizioni
- introdotta da **proposizioni disgiuntive** (o, oppure, ovvero...)
- talvolta introdotta da **proposizioni copulative**
- in MySQL si esprime con **OR** oppure **UNION**



Attenzione!! possono trarre in inganno

# Unione: esempi

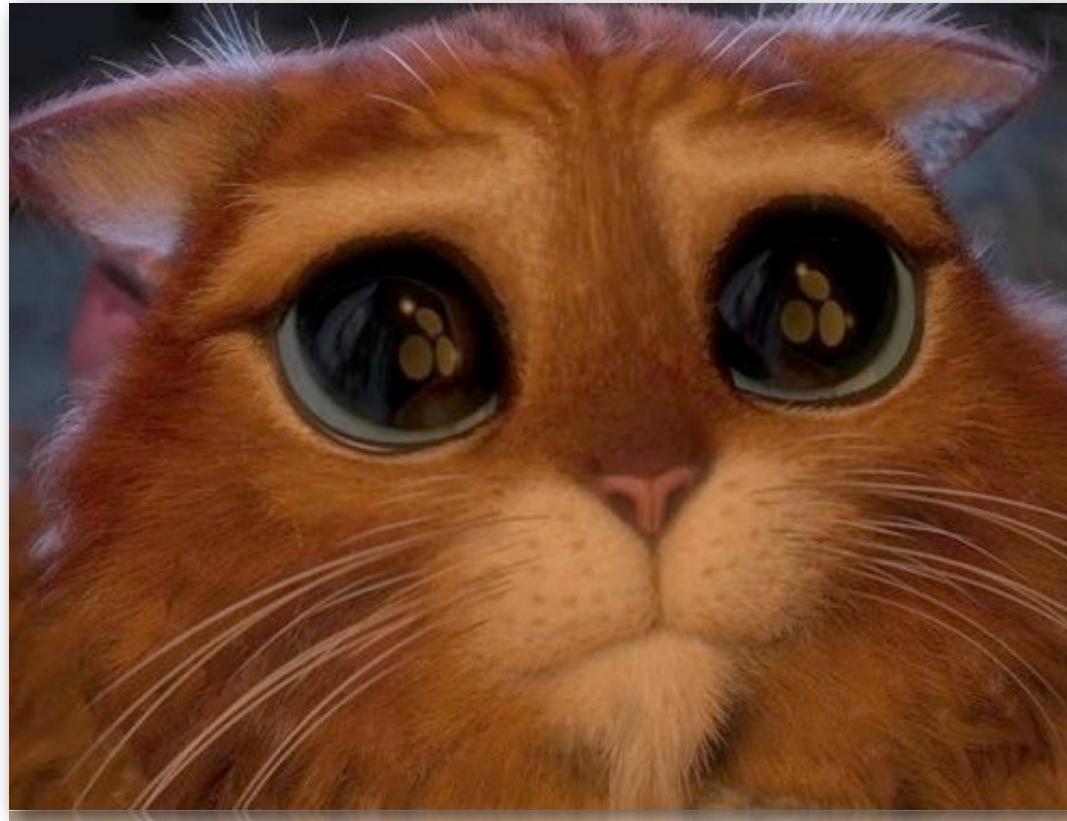
---

- i pazienti visitati dal dott. Verdi **o** dal dott. Rossi
- i medici che hanno visitato **almeno** un paziente di Pisa **o** Siena
- indicare la parcella media fra quella degli otorini **e** quella degli ortopedici

# Non lasciatevi ingannare...

---

Indicare la parcella media fra quella degli otorini e quella degli ortopedici



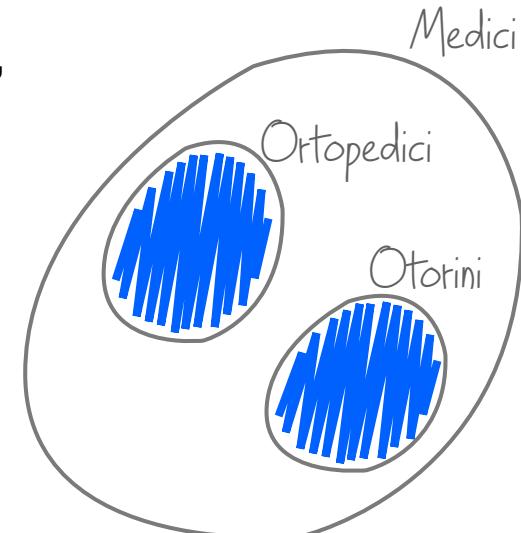
# Non lasciatevi ingannare...



Indicare la parcella media fra quella degli otorini **e** quella degli ortopedici

```
SELECT AVG(Parcella)
FROM Medico
WHERE Specializzazione = 'Otorinolaringoiatria'
      OR Specializzazione = 'Ortopedia';
```

questa è un'unione: un medico è un ototino oppure un ortopedico



# Leggere bene è importante

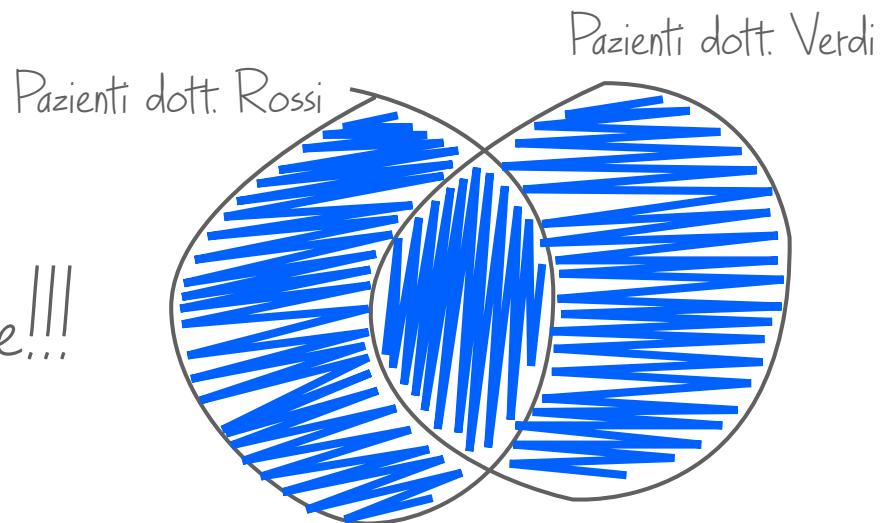
---

I pazienti visitati dal dott. Verdi o dal dott. Rossi



qual è il significato in italiano???

La disgiunzione non esclude l'intersezione!!!



# Uso di UNION

---

I pazienti visitati dal dott. Verdi **o** dal dott. Rossi

pazienti visitati da Verdi

{  
**SELECT** Paziente  
**FROM** Visita V **INNER JOIN** Medico M  
    **ON** V.Medico = M.Matricola  
**WHERE** M.Cognome = 'Verdi'

**UNION**

i pazienti nel risultato  
seguono l'ordine dell'unione

pazienti visitati da Rossi

{  
**SELECT** Paziente  
**FROM** Visita V **INNER JOIN** Medico M  
    **ON** V.Medico = M.Matricola  
**WHERE** M.Cognome = 'Rossi';

# ...e i duplicati?!

---

UNION elimina automaticamente i duplicati; possono essere mantenuti utilizzando **UNION ALL**

# Esempio con duplicati

Totale delle visite effettuate il lunedì dal dott. Verdi e il venerdì dal dott. Rossi

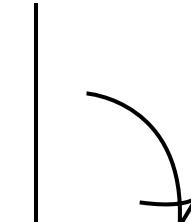
```
SELECT COUNT(*)  
FROM (
```

→ se non si mantengono i duplicati il conteggio è errato  
perché qui un duplicato di Paziente esprime una visita

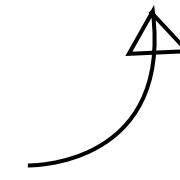
```
    SELECT Paziente  
    FROM Visita V INNER JOIN Medico M  
        ON V.Medico = M.Matricola  
    WHERE M.Cognome = 'Verdi'  
        AND DAYOFWEEK (V.Data) = 1
```

**UNION ALL**

```
    SELECT Paziente  
    FROM Visita V INNER JOIN Medico M  
        ON V.Medico = M.Matricola  
    WHERE M.Cognome = 'Rossi'  
        AND DAYOFWEEK (V.Data) = 5
```



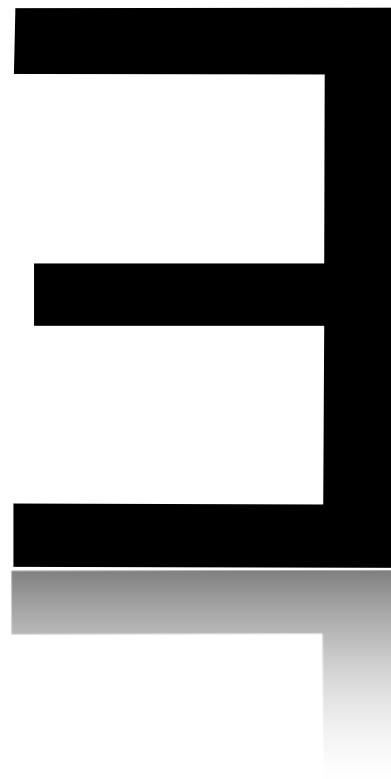
tutti i record sono uniti,  
mantenendo i duplicati



) AS D;

# Esistenza

---



# Esistenza

---

- esprime l'esistenza di **almeno un elemento** in un insieme
- introdotta da *almeno, a patto che, qualche...*
- in MySQL si esprime con **EXISTS**

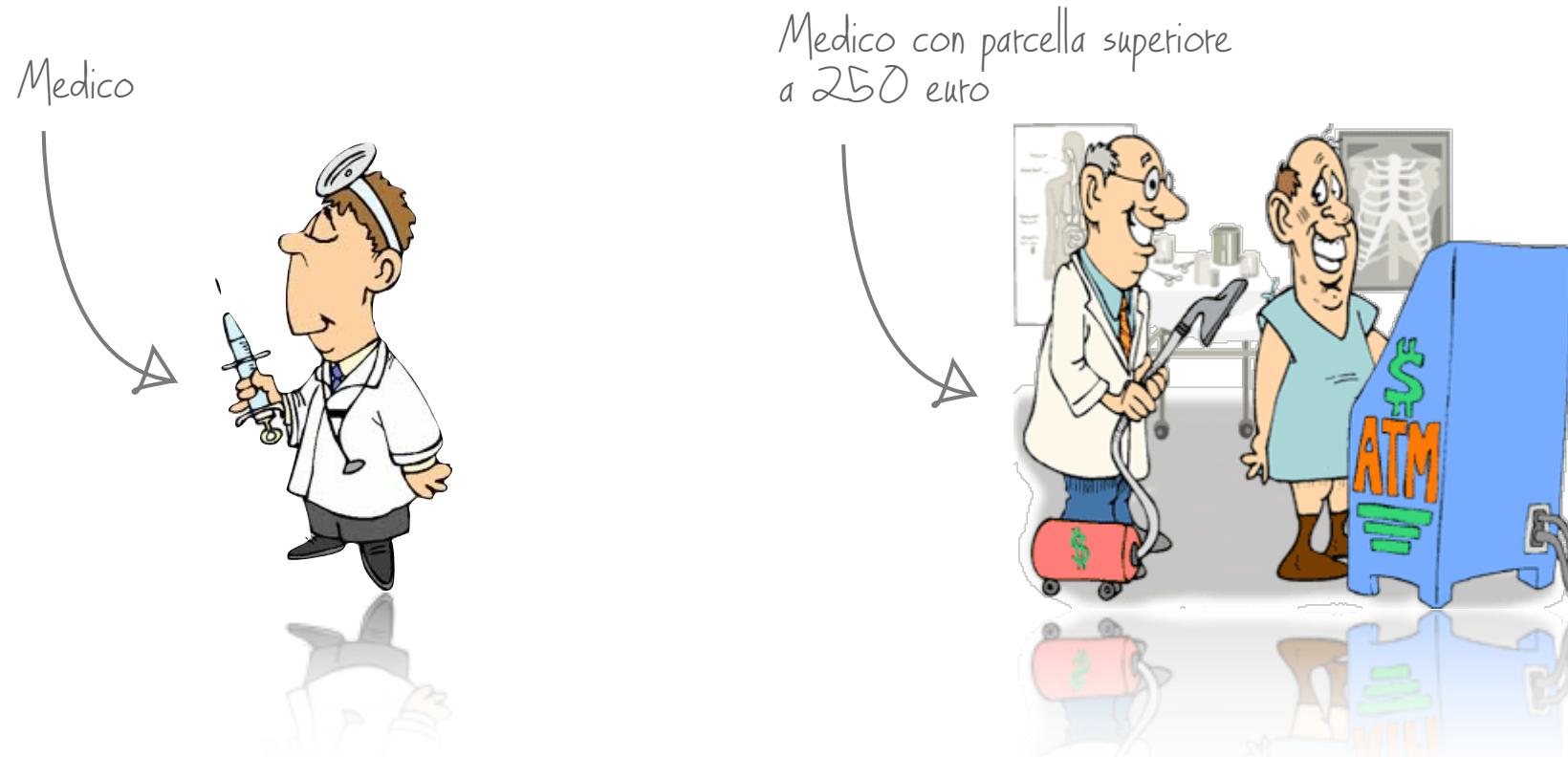
**ATTENTION!**

*exists va sempre usato in correlated subquery!!!*

# Esistenza: esempio

---

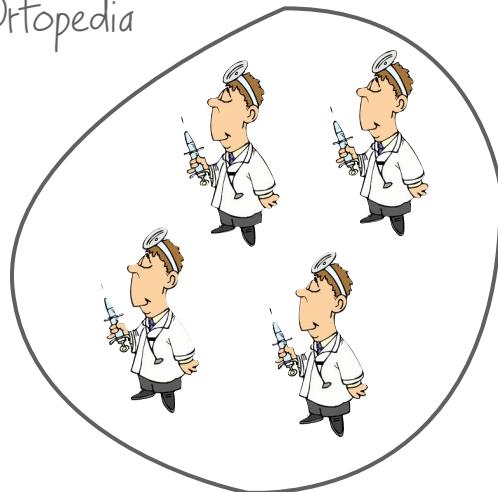
I pazienti visitati da un medico alla cui specializzazione appartenga  
**almeno un medico** con parcella superiore a 250 euro



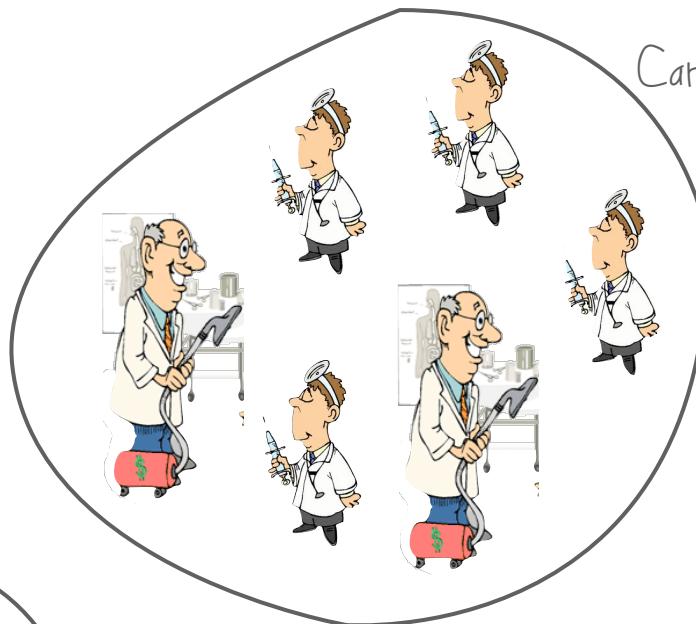
# Esistenza: esempio

---

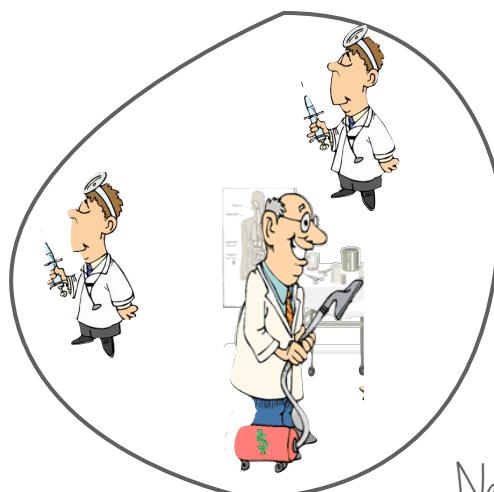
Ortopedia



Cardiologia



Neurologia



# Esistenza: esempio

---

I pazienti visitati da un medico alla cui specializzazione appartenga  
**almeno un medico** con parcella superiore a 250 euro

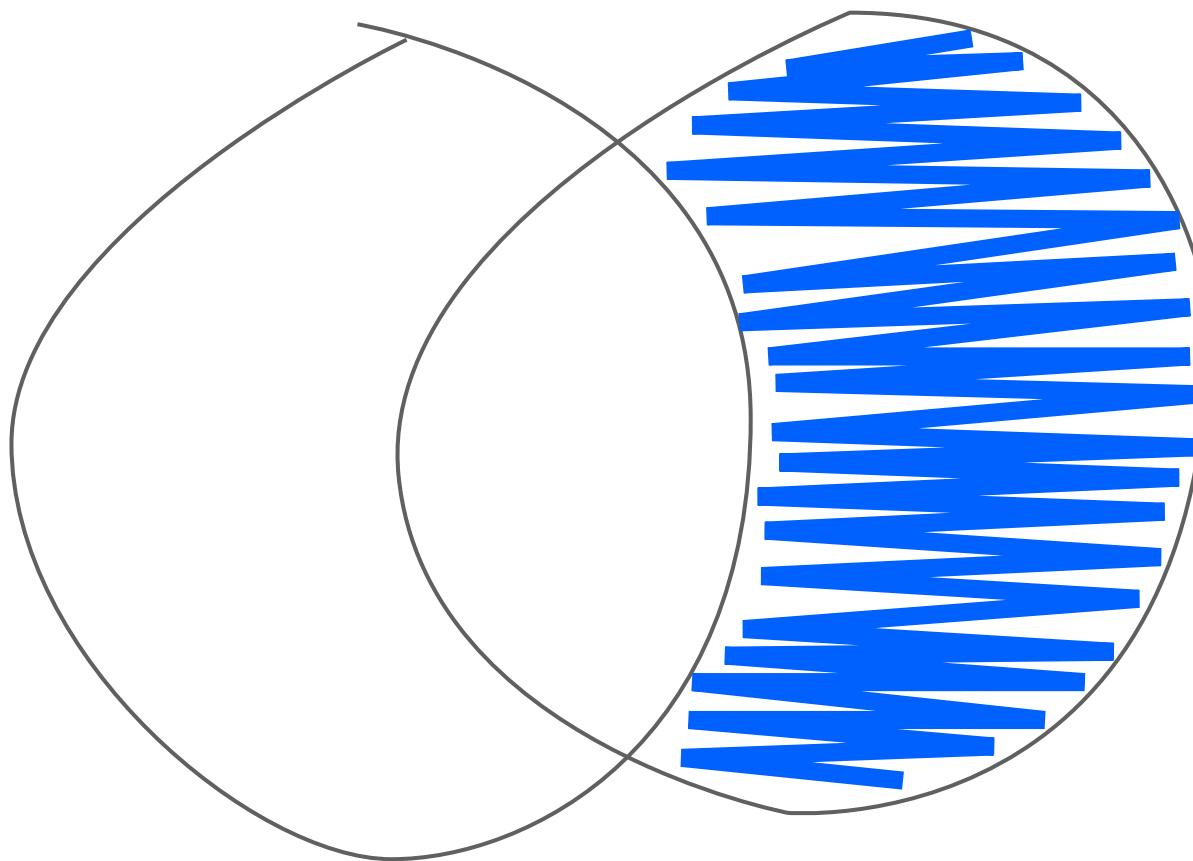
```
SELECT DISTINCT V.Paziente
FROM Visita V INNER JOIN Medico M1
WHERE EXISTS ( SELECT *
                FROM Medico M2
                WHERE M2.Specializzazione = M1.Specializzazione
                  AND M2.Parcella > 250 );
```

dato un record della outer query, questo  
insieme **DEVE ESISTERE**

Il record va nel risultato se la subquery  
restituisce **ALMENO UN RECORD**

# Differenza

---



# Differenza

---

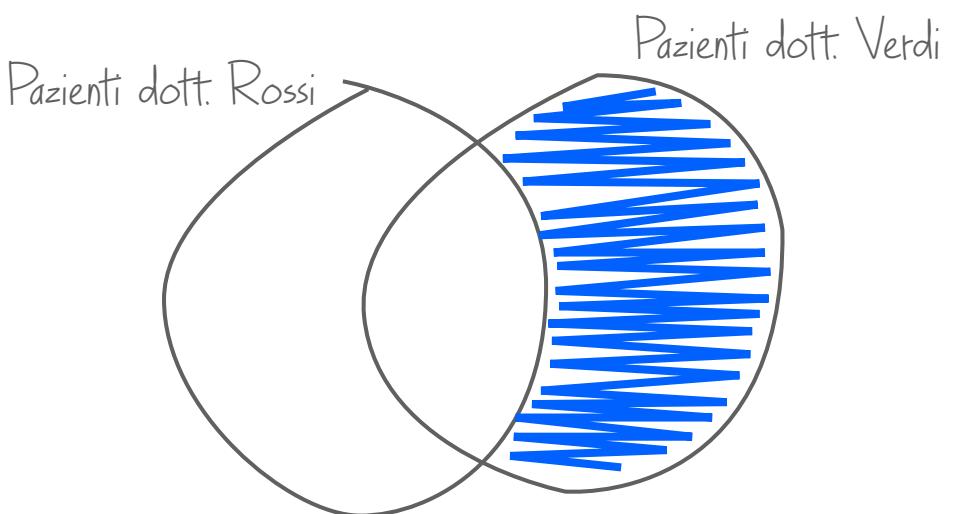
- esprime il **complemento** di un insieme rispetto a un altro
- introdotta da **proposizioni eccettuative** (*tranne, eccetto che...*); da avverbi come *solamente, esclusivamente*; da abbinamenti di congiunzioni come *ma+non*
- in MySQL si esprime con **NOT IN, NOT EXISTS**, oppure tramite **OUTER JOIN**

# Differenza: esempio

---

I pazienti visitati dal dott. Verdi **ma non** dal dott. Rossi

I pazienti visitati **ANCHE**  
dal dott. Rossi **vanno eliminati!!**



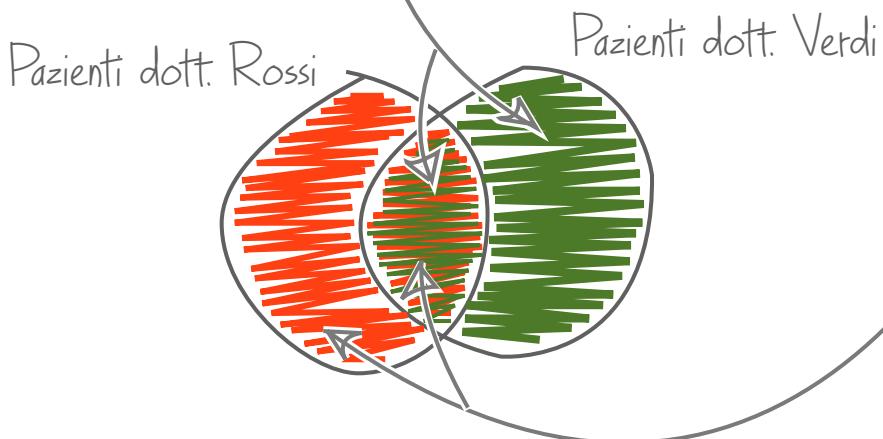
# Differenza: esempio

---

I pazienti visitati dal dott. Verdi **ma non** dal dott. Rossi

```
SELECT DISTINCT V.Paziente  
FROM Visita V INNER JOIN Medico M  
    ON V.Medico = M.Matricola  
WHERE M.Cognome = 'Verdi'  
    AND V.Paziente NOT IN (
```

```
    SELECT Paziente  
    FROM Visita V2 INNER JOIN Medico M2  
        ON V2.Medico = M2.Matricola  
    WHERE M2.Cognome = 'Rossi');
```



# Ancora malvagità

---

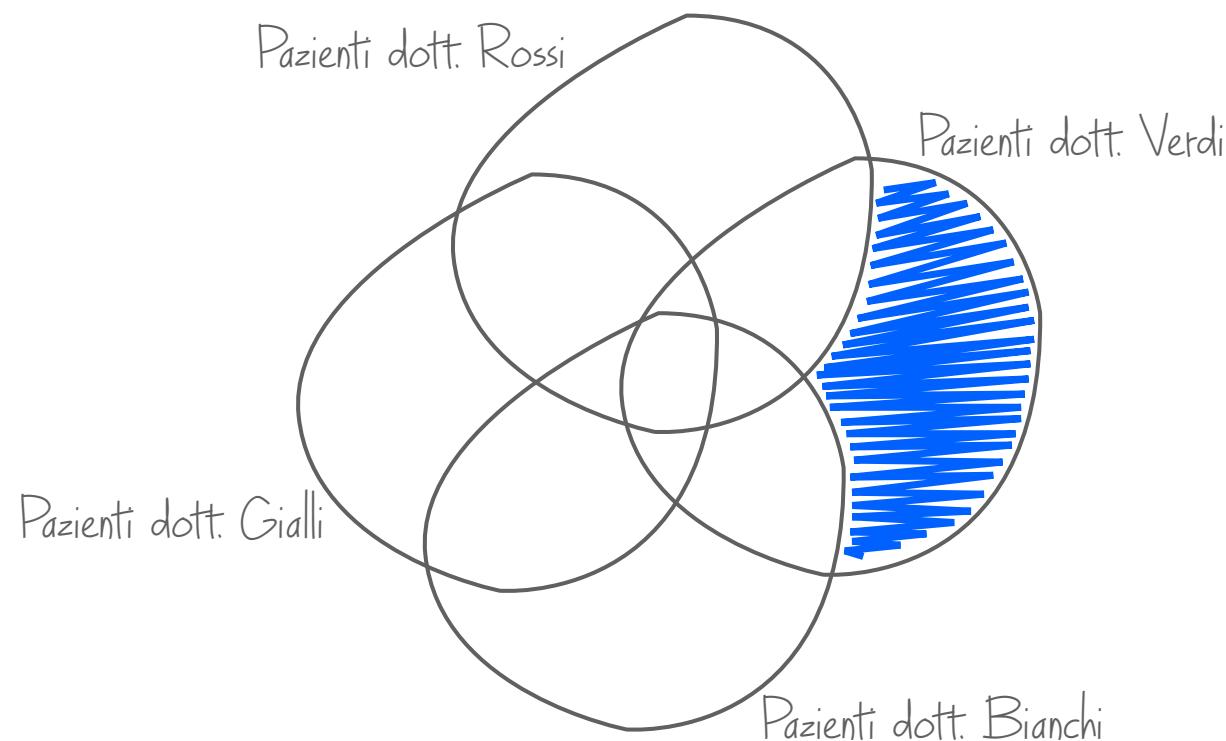
I pazienti visitati **solamente** dal dott. Verdi



# Differenza a più insiemi

---

I pazienti visitati **solamente** dal dott. Verdi

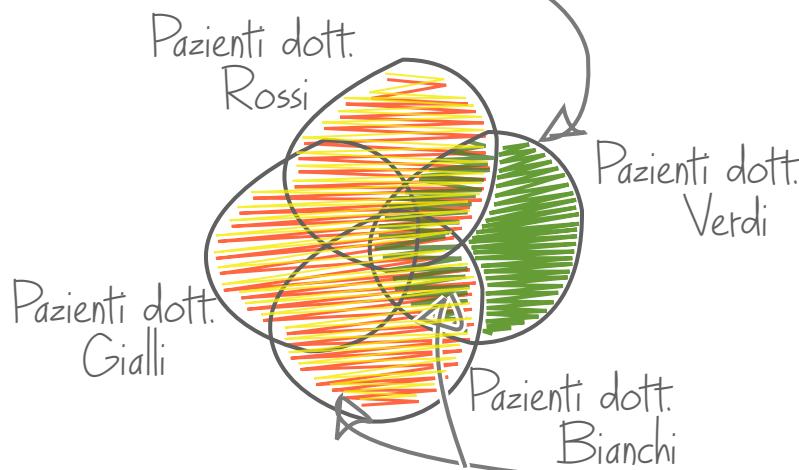


# Soluzione con NOT IN + correlated subquery

I pazienti visitati **solamente** dal dott. Verdi

```
SELECT DISTINCT V.Paziente  
FROM Visita V INNER JOIN Medico M  
    ON V.Medico = M.Matricola  
WHERE M.Cognome = 'Verdi'  
    AND V.Paziente NOT IN (
```

```
SELECT Paziente  
FROM Visita V2 INNER JOIN Medico M2  
    ON V2.Medico = M2.Matricola  
WHERE M2.Cognome <> 'Verdi');
```



# Soluzione con join esterno

---

I pazienti visitati **soltanto** dal dott. Verdi

```
SELECT DISTINCT V1.Paziente
FROM Visita V1 INNER JOIN Medico M1 ON V1.Medico = M1.Matricola
NATURAL LEFT OUTER JOIN
(
    SELECT V2.Paziente
    FROM Visita V2 INNER JOIN Medico M2
        ON V2.Medico = M2.Matricola
    WHERE M2.Cognome <> 'Verdi'
) AS D
WHERE M1.Cognome = 'Verdi' AND D.Paziente IS NULL
```

} pazienti visitati da medici diversi da Verdi

se un paziente è visitato solo da Verdi, nessun record di V1 (visite fatte da Verdi) è joinabile con record in D (visite fatte da altri medici sullo stesso paziente)

# Soluzione con NOT EXISTS

---

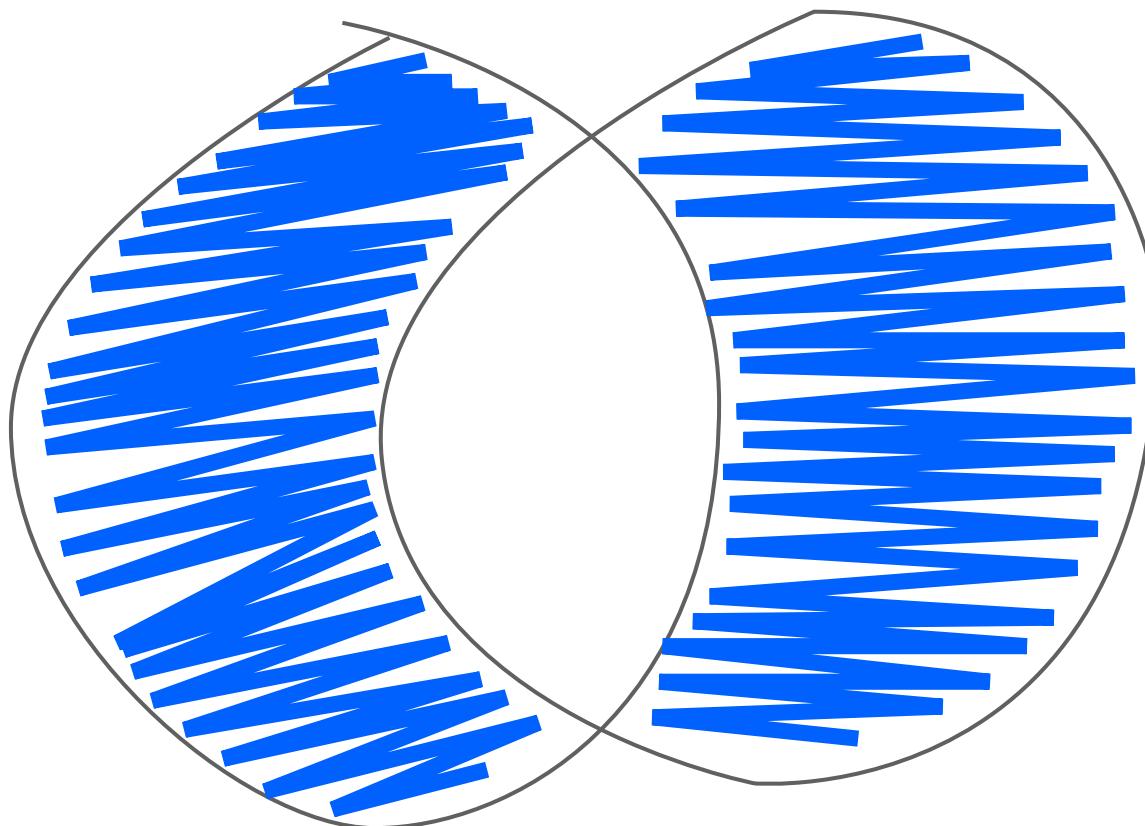
I pazienti visitati **solamente** dal dott. Verdi

```
SELECT DISTINCT Paziente
  FROM Visita V1 INNER JOIN Medico M1
        ON V1.Medico = M1.Matricola
 WHERE M1.Cognome = 'Verdi'
       AND NOT EXISTS (
           SELECT *
             FROM Visita V2 INNER JOIN Medico M2
                   ON V2.Medico = M2.Matricola
             WHERE V2.Paziente = V1.Paziente
                   AND M2.Cognome <> 'Verdi'
       );
```

preso una visita fatta da Verdi, non deve esistere alcuna visita che coinvolga lo stesso paziente, fatta da un medico diverso da Verdi

# Differenza simmetrica

---



# Differenza simmetrica

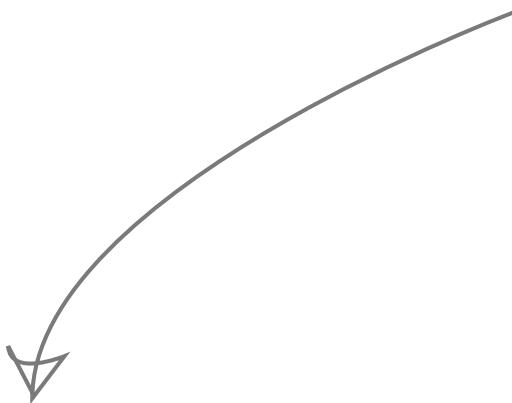
---

- dati due insiemi, contiene gli elementi presenti **solo in uno di essi**
- introdotta da avverbi come *solamente*, *esclusivamente*, in abbinamento a **congiunzioni disgiuntive** (*o*, *oppure*)
- in MySQL si esprime con **NOT IN**, con **NOT IN+UNION** oppure mediante **OUTER JOIN**

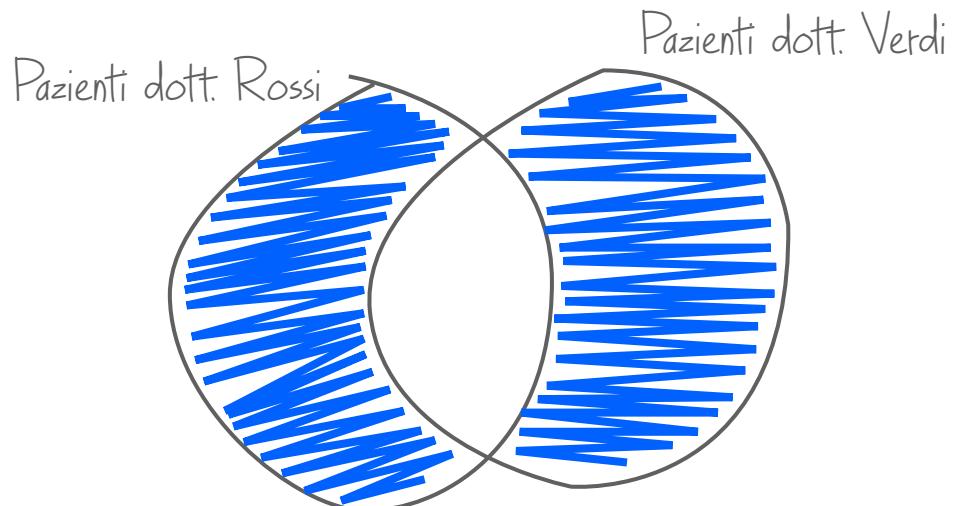
# Differenza simmetrica

---

I pazienti visitati **esclusivamente** dal dott. Verdi **o** dal dott. Rossi



non mi interessa se il paziente è  
stato visitato anche da altri medici



# Differenza simmetrica: join e sottoquery scorrelata

I pazienti visitati **esclusivamente** dal dott. Verdi **o** dal dott. Rossi

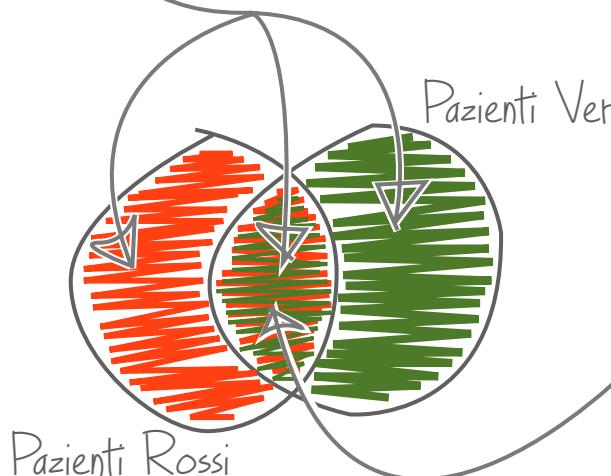
```
SELECT DISTINCT V.Paziente  
FROM Visita V INNER JOIN Medico M  
ON V.Medico = M.Matricola  
WHERE (M.Cognome = 'Verdi' OR M.Cognome = 'Rossi')  
AND V.Paziente NOT IN (
```

pazienti visitati da Verdi, da Rossi e  
da entrambi

```
SELECT V.Paziente  
FROM Visita V INNER JOIN Medico M  
ON V.Medico = M.Matricola  
WHERE M.Cognome = 'Verdi'  
AND V.Paziente IN (
```

pazienti visitati almeno una volta  
sia da Verdi che da Rossi

```
SELECT Paziente  
FROM Visita V2 INNER JOIN Medico M2  
ON V2.Medico = M2.Matricola  
WHERE M2.Cognome = 'Rossi')
```



);

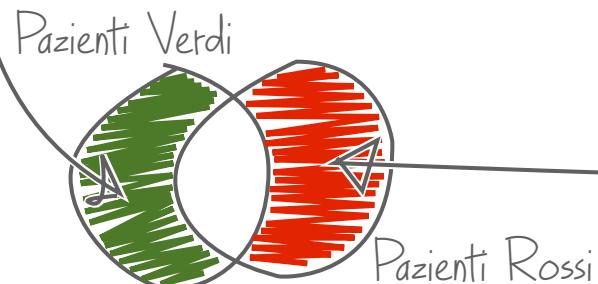
# Differenza simmetrica: UNION

I pazienti visitati **esclusivamente** dal dott. Verdi **o** dal dott. Rossi

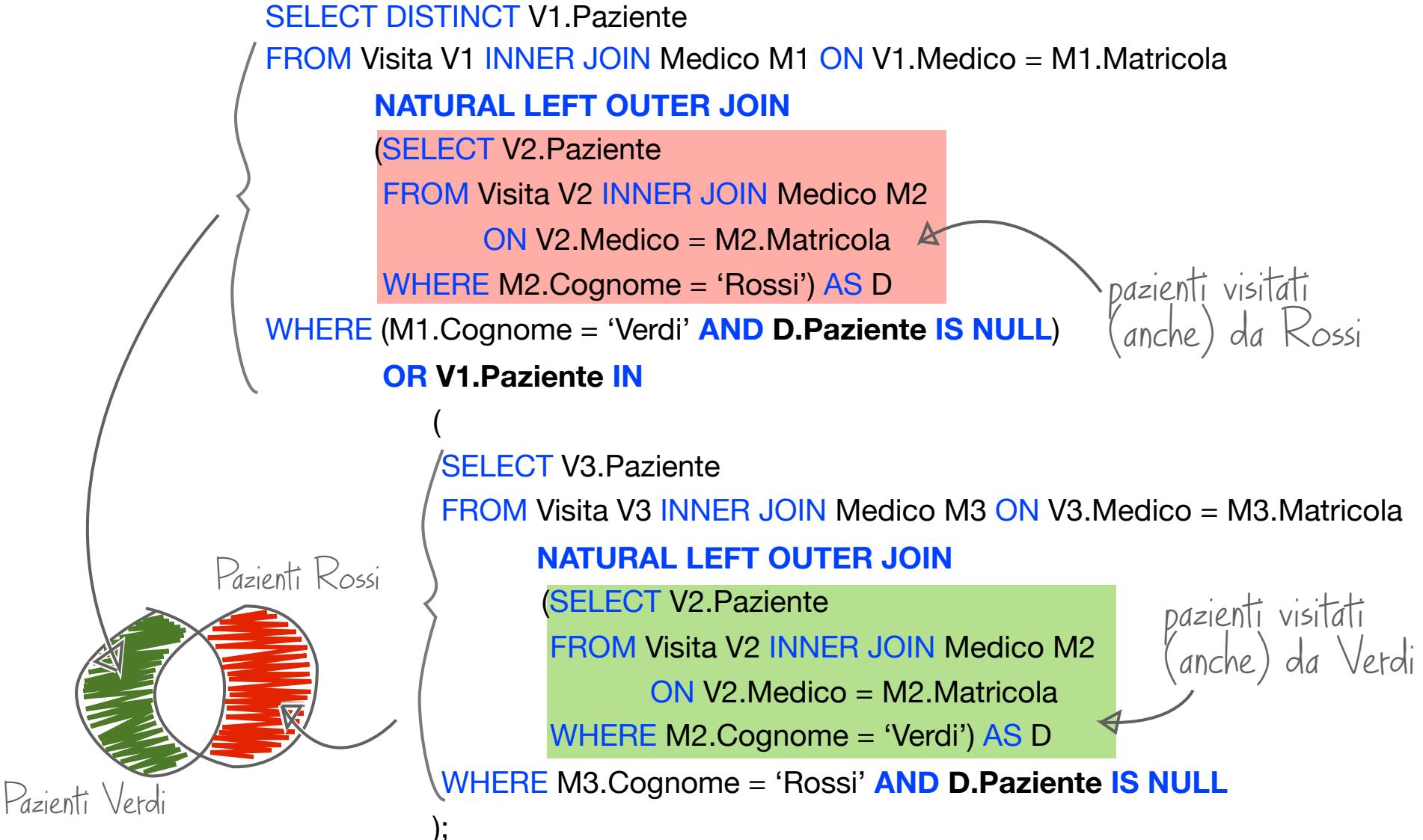
```
SELECT V.Paziente  
FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
WHERE M.Cognome = 'Verdi'  
AND V.Paziente NOT IN ( SELECT Paziente  
FROM Visita V2 INNER JOIN Medico M2  
ON V2.Medico = M2.Matricola  
WHERE M2.Cognome = 'Rossi')
```

**UNION**

```
SELECT V.Paziente  
FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
WHERE M.Cognome = 'Rossi'  
AND V.Paziente NOT IN ( SELECT Paziente  
FROM Visita V2 INNER JOIN Medico M2  
ON V2.Medico = M2.Matricola  
WHERE M2.Cognome = 'Verdi');
```

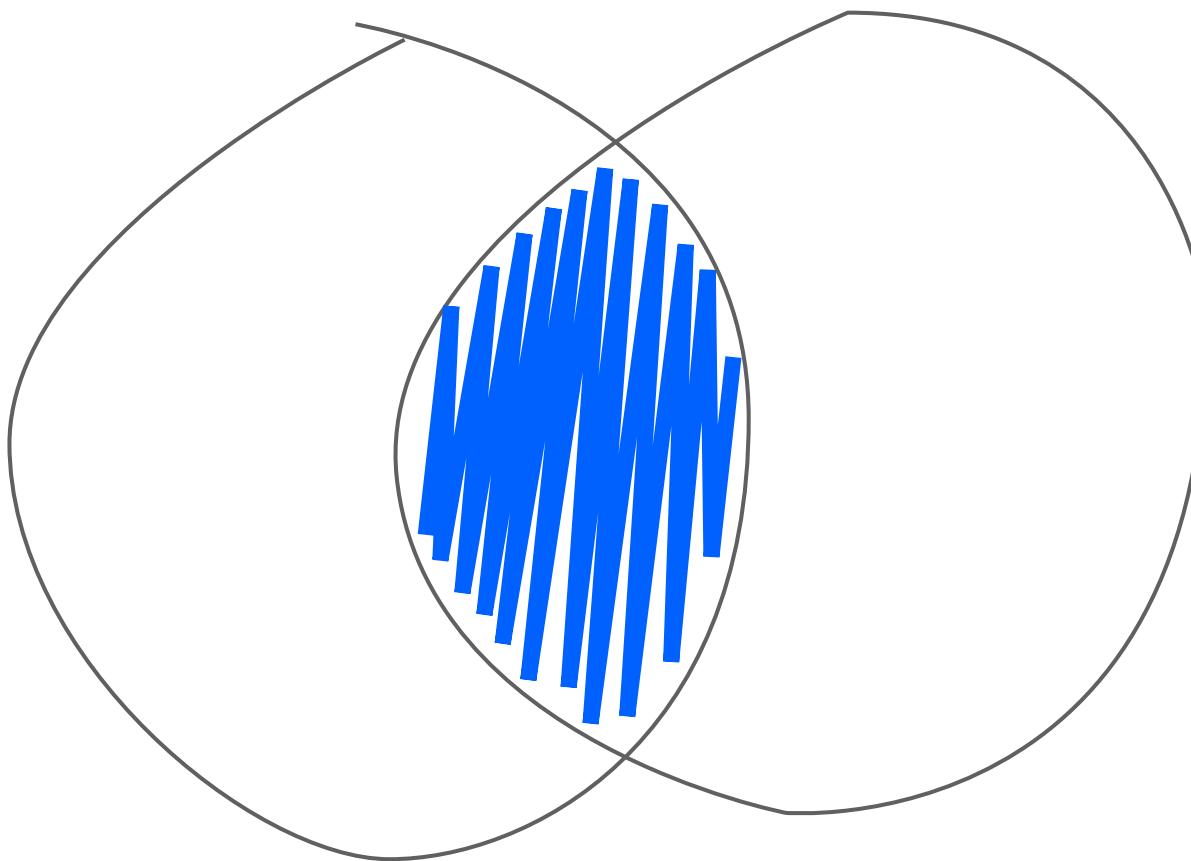


# Differenza simmetrica: self outer join e subquery



# Intersezione

---



# Intersezione

---

- un elemento vi appartiene se appartiene a **entrambi** gli insiemi
- introdotta da **congiunzioni correlative** (*sia/che, ma anche...*)
- in MySQL si esprime con **IN** oppure mediante **INNER JOIN**

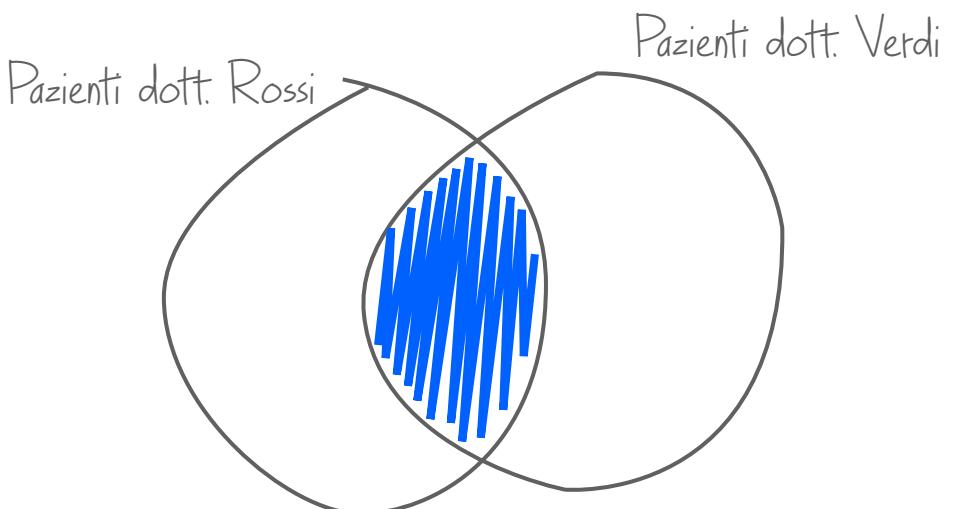
# Intersezione

---

I pazienti visitati **sia** dal dott. Verdi **che** dal dott. Rossi



Dato un paziente, questo deve essere stato visitato  
almeno una volta da **ENTRAMBI** i medici

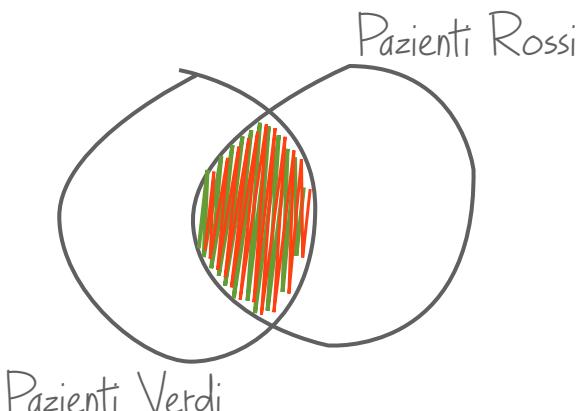


# Intersezione con subquery

---

I pazienti visitati **sia** dal dott. Verdi **che** dal dott. Rossi

```
SELECT DISTINCT V.Paziente
FROM Visita V INNER JOIN Medico M
    ON V.Medico = M.Matricola
WHERE M.Cognome = 'Verdi'
    AND V.Paziente IN (
        SELECT V2.Paziente
        FROM Visita V2 INNER JOIN Medico M2
            ON V2.Medico = M2.Matricola
        WHERE M2.Cognome = 'Rossi'
    );
```



# Intersezione con self(ie) join

---

I pazienti visitati **sia** dal dott. Verdi **che** dal dott. Rossi

```
SELECT DISTINCT V1.Paziente  
FROM (Visita V1 INNER JOIN Medico M1 ON V1.Medico = M1.Matricola)  
      INNER JOIN  
      (Visita V2 INNER JOIN Medico M2 ON V2.Medico = M2.Matricola)  
      USING (Paziente)  
WHERE M1.Cognome = 'Verdi'  
      AND M2.Cognome = 'Rossi';
```



# Divisione

---



# Divisione

---

medici che hanno visitato tutti i pazienti

pazienti visitati una volta in tutti i mesi dell'anno

Operatore insiemistico derivato utile per interrogazioni che contengono **condizioni esaustive** espresse mediante l'avverbio *tutti*

The word "TUTTI" is repeated five times in different colors: blue, orange, purple, red, and yellow. Each color is placed above a larger, bold black "TUTTI". Below each large "TUTTI" is a smaller, colored "TUTTI".

# Divisione: esempio

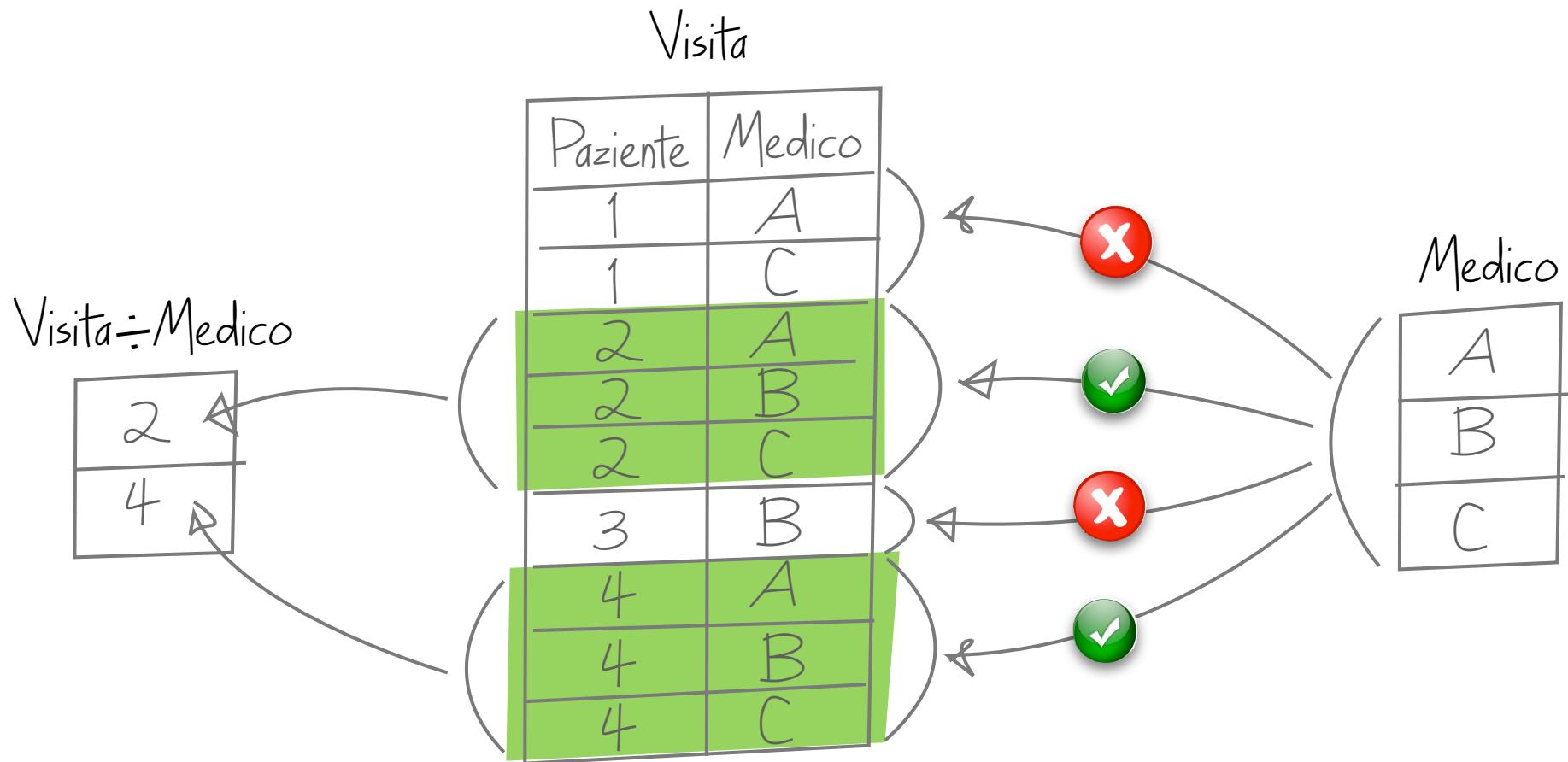
---

Indicare i pazienti visitati **da tutti i medici**



# Divisione: come funziona

Indicare i pazienti visitati **da tutti i medici**



# Divisione: con doppio NOT EXISTS

---

Indicare i pazienti visitati **da tutti i medici**

```
SELECT P.CodFiscale
FROM Paziente P
WHERE NOT EXISTS
(
    SELECT *
    FROM Medico M
    WHERE NOT EXISTS
    (
        SELECT *
        FROM Visita V
        WHERE V.Medico = M.Matricola
            AND V.Paziente = P.CodFiscale
    )
);
```

# Qual è il significato?

---

Indicare i pazienti visitati **da tutti i medici**

```
SELECT P.CodFiscale  
FROM Paziente P  
WHERE NOT EXISTS
```

(

```
SELECT *  
FROM Medico M  
WHERE NOT EXISTS
```

(

```
SELECT *  
FROM Visita V  
WHERE V.Medico = M.Matricola  
AND V.Paziente = P.CodFiscale
```

)

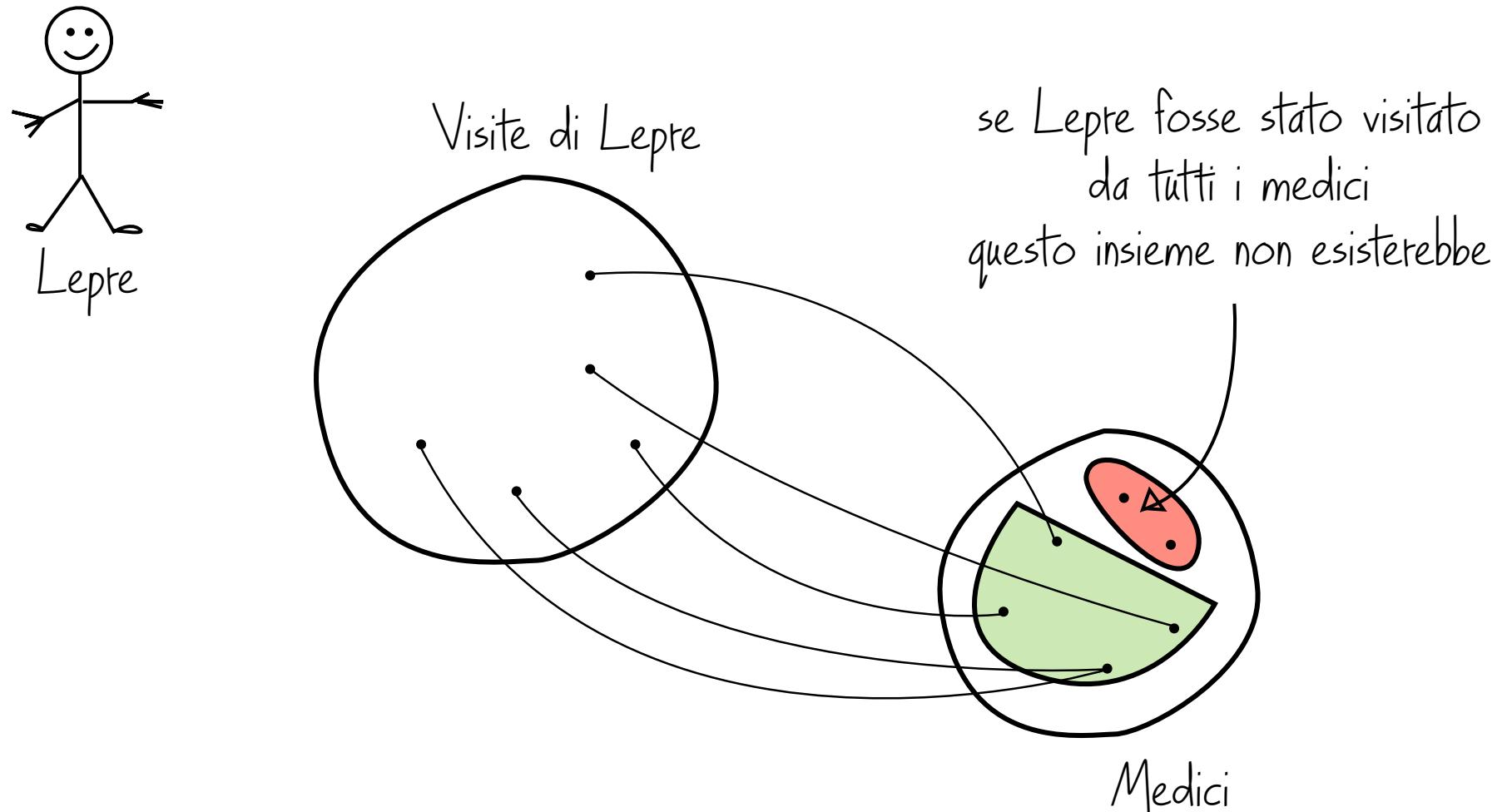
);

se considerato il paziente P non riesco a trovare  
alcun medico che non abbia mai visitato P  
significa che P è stato visitato da tutti i medici

medici che non hanno mai  
visitato il paziente P

# Cosa significa?

---



# Divisione: raggruppamento e subquery having

---

Indicare i pazienti visitati **da tutti i medici**

```
SELECT V.Paziente  
FROM Visita V  
GROUP BY V.Paziente  
HAVING COUNT(DISTINCT V.Medico) =
```

totale dei medici

```
( SELECT COUNT (*)  
  FROM Medico );
```

medici diversi che hanno visitato il paziente

# Modificatori

---



# Modificatori

---

→ valore1 [ $<$ ,  $\leq$ ,  $\neq$ ,  $=$ ,  $\geq$ ,  $>$ ] valore2

Modificano il confronto che precede una subquery, imponendone la validità per **tutti** oppure **almeno un record** del result set della subquery

# Almeno un record del result set

---

Indicare il nome e cognome dei medici la cui parcella è superiore a quella di almeno un cardiologo di Pisa

```
SELECT M.Nome,  
       M.Cognome  
FROM Medico M  
WHERE M.Parcella > ANY (  
                           SELECT M2.Parcella  
                           FROM Medico M2  
                           WHERE M2.Specializzazione = 'Cardiologia'  
                                 AND M2.Citta = 'Pisa'  
);
```

un record della outer query va nel risultato se la condizione ha successo per *ALMENO UN* record della subquery

# Tutti i record del result set

---

Indicare il nome e cognome dei medici di Siena la cui parcella è inferiore a quella di tutti i medici di Siena

NOTA: il minore stretto qui è giusto perché il medico della outer query deve andare nel risultato solo se la sua parcella è più bassa di tutti i medici di Siena

```
SELECT M.Nome, M.Cognome
FROM Medico M
WHERE M.Parcella < ALL (
    SELECT M2.Parcella
    FROM Medico M2
    WHERE M2.Citta = 'Siena'
);
```

il record esterno va nel risultato se la condizione ha successo per TUTTI i record della subquery

# Altro esempio con ALL

---

Indicare matricola e cognome del medico avente la parcella più bassa  
delle parcelle di tutti i medici

```
SELECT M1.Matricola,  
       M1. Cognome  
FROM Medico M1  
WHERE M1.Parcella < ALL (  
                           SELECT M2.Parcella  
                           FROM Medico M2  
                           );
```



# Questa volta è errata

---

Indicare matricola e cognome del medico avente la parcella più bassa  
delle parcelle di tutti i medici



```
SELECT M1.Matricola,  
       M1.Cognome  
  FROM Medico M1  
 WHERE M1.Parcella < ALL (  
           SELECT M2.Parcella  
             FROM Medico M2  
          );
```

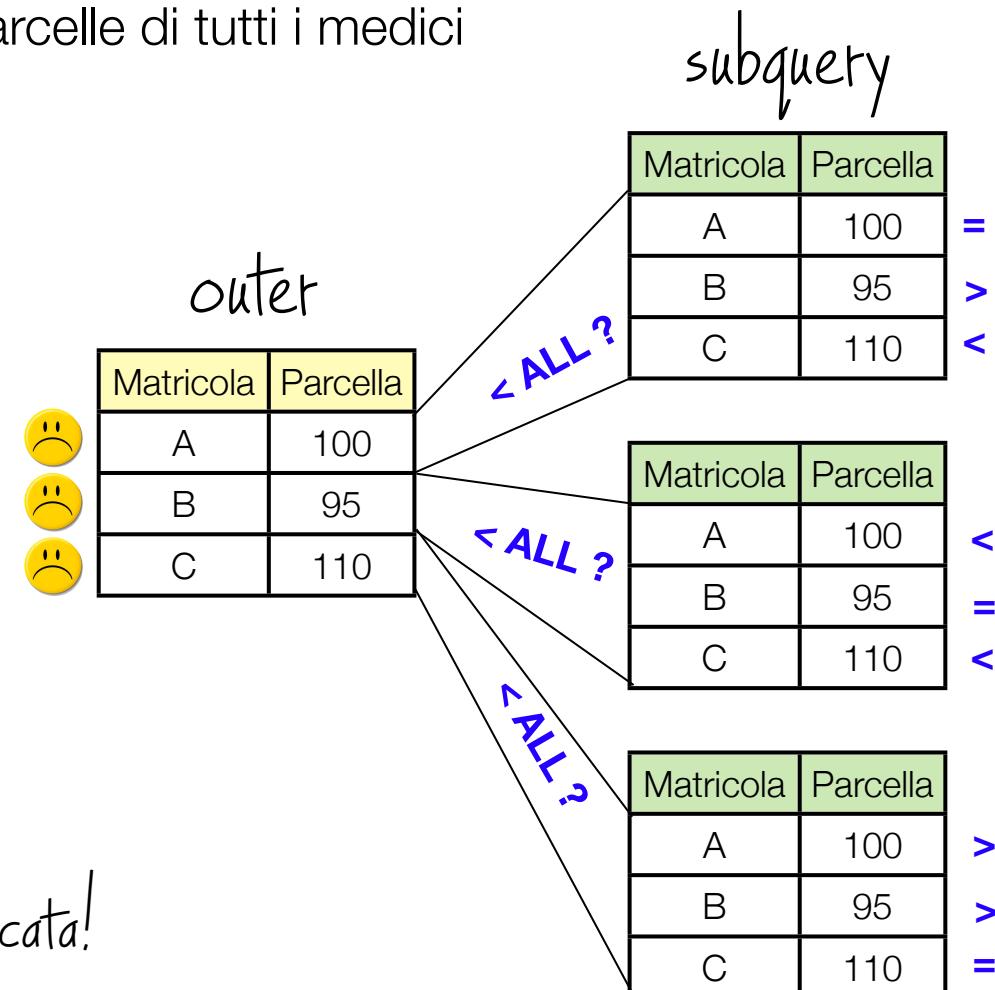
~~EMPTY  
SET~~

la subquery contiene anche il record della outer query e quindi la outer query restituirà sempre result set vuoto!

# Vediamo i result set intermedi

Indicare matricola e cognome del medico avente la parcella più bassa delle parcelle di tutti i medici

```
SELECT M1.Matricola,  
       M1.Cognome  
  FROM Medico M1  
 WHERE M1.Parcella < ALL  
       (  
           SELECT M2.Parcella  
             FROM Medico M2  
       );
```

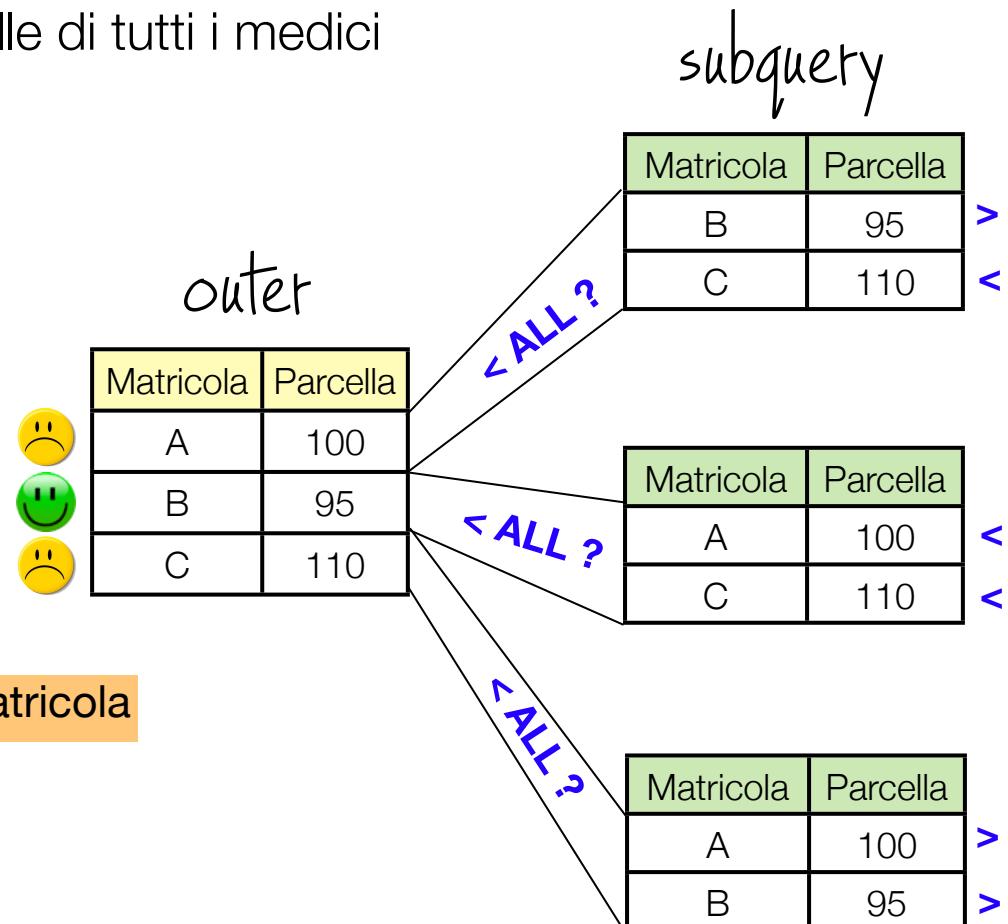


la condizione non è mai verificata!

# Approccio corretto col minore stretto

Indicare matricola e cognome del medico avente la parcella più bassa delle parcelle di tutti i medici

```
SELECT M1.Matricola,  
       M1.Cognome  
FROM Medico M1  
WHERE M1.Parcella < ALL  
      (  
        SELECT M2.Parcella  
        FROM Medico M2  
        WHERE M2.Matricola <> M1.Matricola  
      );
```



# E se ci fossero dei pari merito?

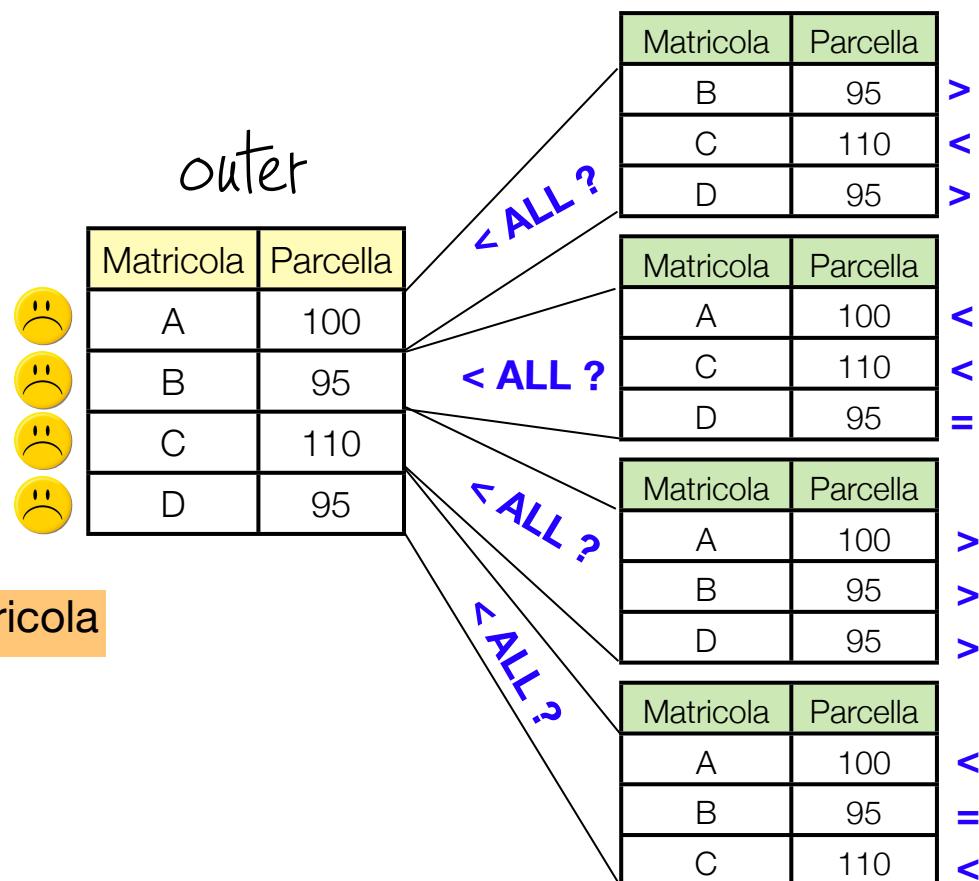
Indicare matricola e cognome del medico avente la parcella più bassa  
delle parcelle di tutti i medici

subquery

```
SELECT M1.Matricola,  
       M1.Cognome  
FROM Medico M1  
WHERE M1.Parcella < ALL  
      (  
          SELECT M2.Parcella  
          FROM Medico M2  
          WHERE M2.Matricola <> M1.Matricola  
      );
```

∅

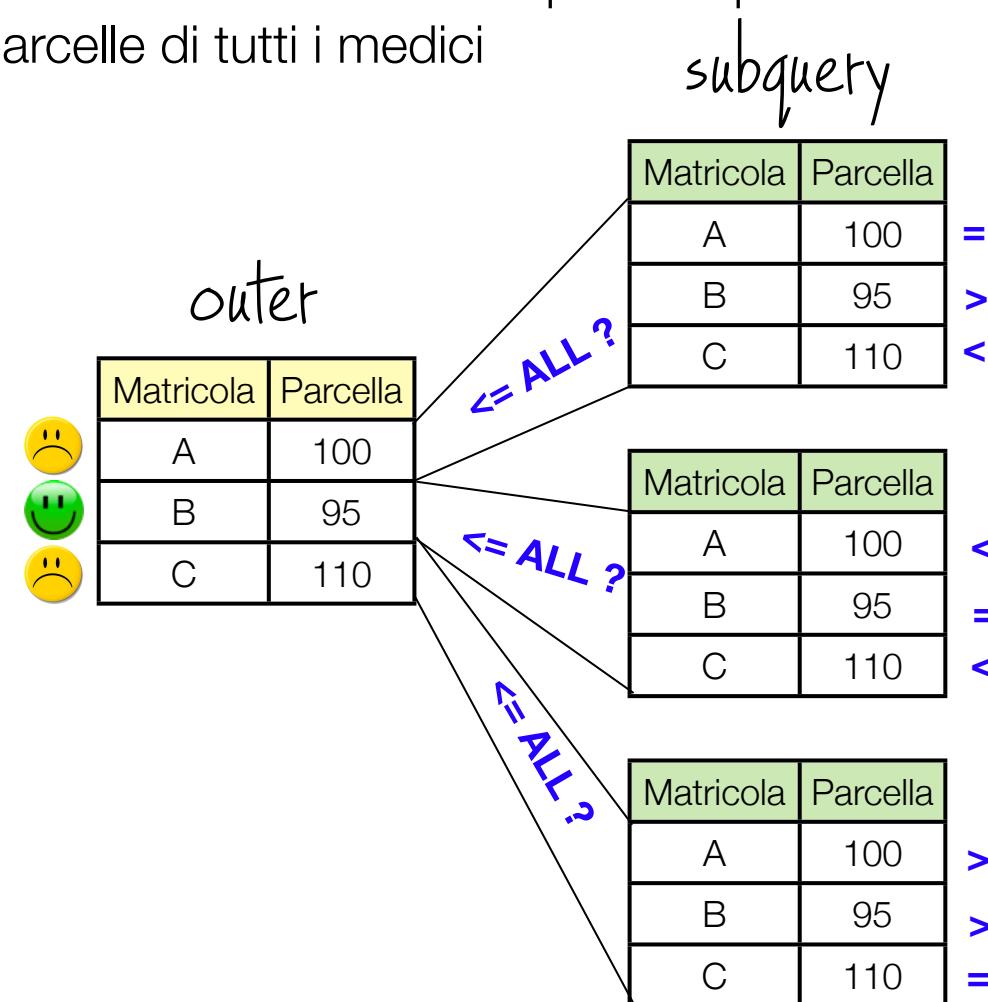
result set vuoto.



# Approccio con minore o uguale

Indicare matricola e cognome del medico avente la parcella più bassa  
delle parcelle di tutti i medici

```
SELECT M1.Matricola,  
       M1.Cognome  
FROM Medico M1  
WHERE M1.Parcella <= ALL  
  (  
    SELECT M2.Parcella  
    FROM Medico M2  
  );
```

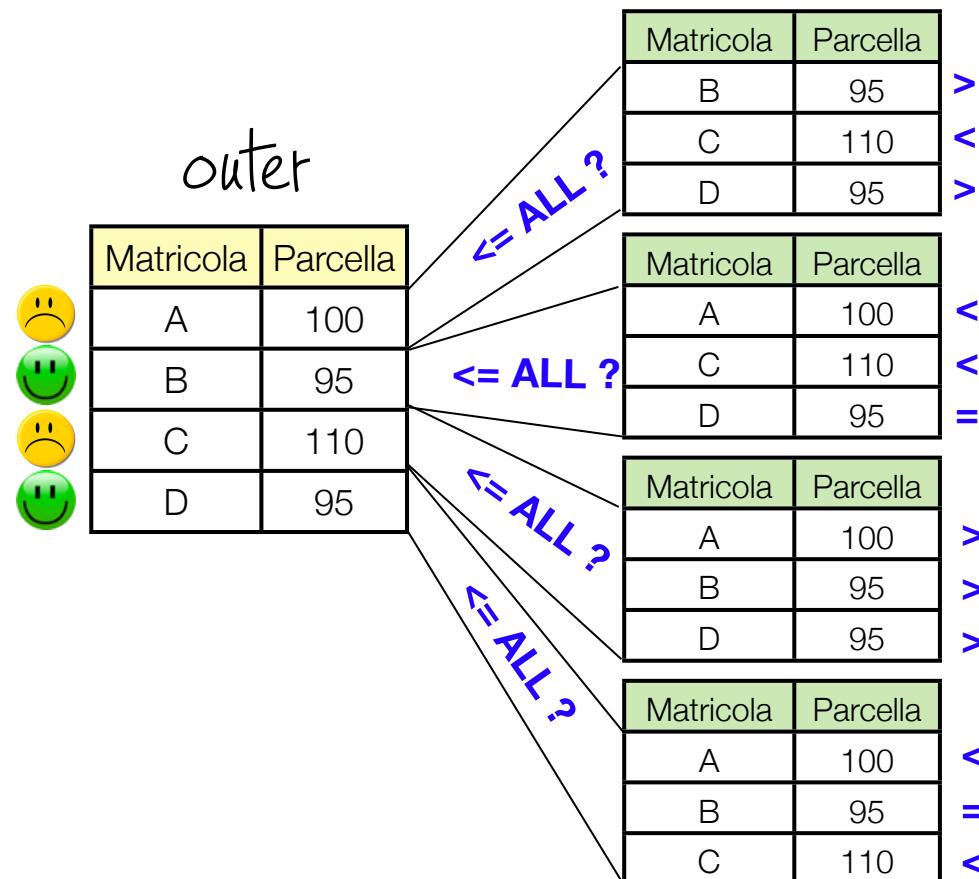


# Con pari merito

Indicare matricola e cognome del medico avente la parcella più bassa  
delle parcelle di tutti i medici

subquery

```
SELECT M1.Matricola,  
       M1.Cognome  
  FROM Medico M1  
 WHERE M1.Parcella <= ALL  
       (  
           SELECT M2.Parcella  
             FROM Medico M2  
       );
```



mantiene tutti gli ex aequo!

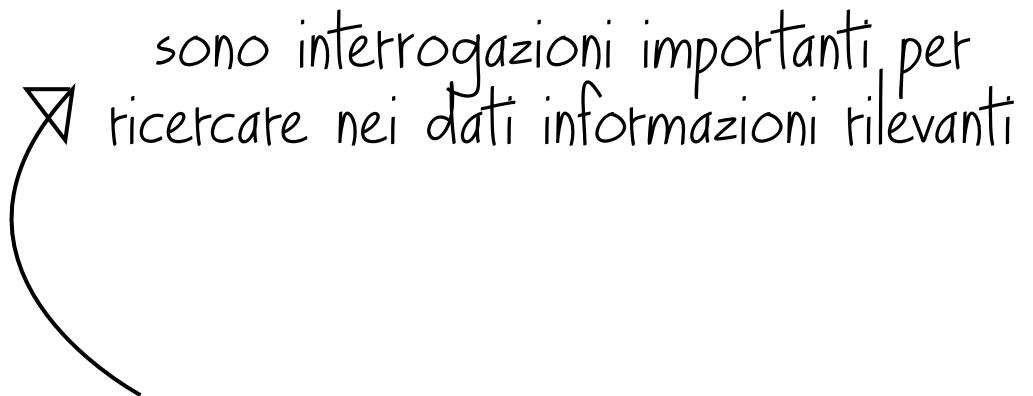
# Query complesse

---



# Query complesse

---

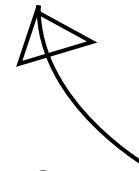


Hanno la funzione di rispondere a **interrogazioni articolate** che fondono in maniera complessa svariati aspetti logici della realtà

# Perché le query complesse?

---

trend di vendita, abitudini dei clienti, prodotti il cui acquisto provoca l'acquisto di altri prodotti, criticità di gestione di un business, e così via.



In un database ci sono enormi quantità di **informazioni nascoste** che, se individuate, permettono un'analisi profonda dei dati e della realtà



business intelligence e data mining sono basati su query complesse per la fase di ricerca delle informazioni nascoste



# Metodo di risoluzione

---

Una query complessa si risolve attraverso i seguenti passi:

1. **Leggere** approfonditamente il testo
2. **Dividere** il problema in sottoproblemi
3. **Disegnare** gli insiemi in gioco e tracciare schemi di ragionamento
4. **Analizzare** logicamente i sottoproblemi
5. **Tradurre** in codice ogni sottoproblema
6. **Pegare** Santa Rita
7. **Legare** logicamente le soluzioni dei sottoproblemi
8. **Scrivere il codice** della query risolutiva

# Query complesse: esempio

---

Indicare le specializzazioni contenenti solo medici provenienti dalla stessa città, di cui almeno uno abbia superato, nell'anno 2011, un totale di visite pari al 10% di tutte le visite della sua specializzazione nello stesso anno

La prima reazione...

---

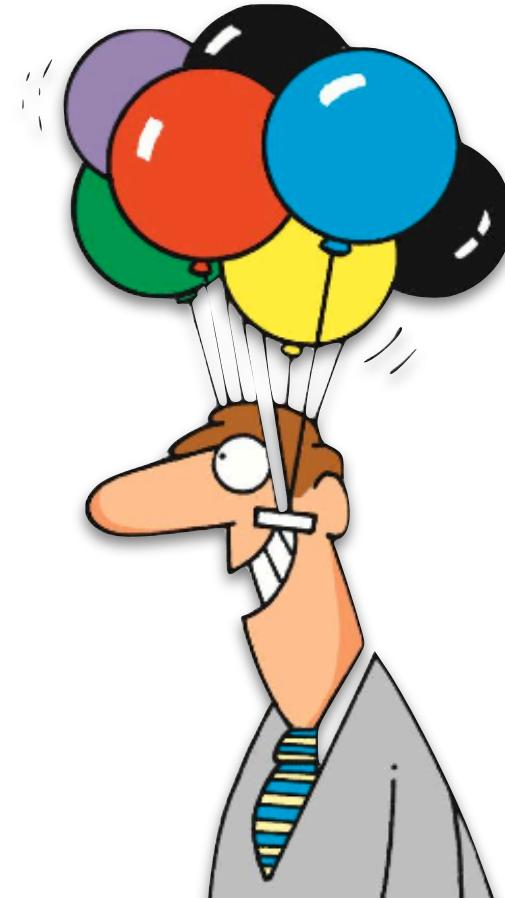


**NO MARIA, IO ESCO**

# Prima cosa da fare

---

Non farsi prendere dallo scottaggiamento!



# Leggere bene il testo e decomporre

Indicare le specializzazioni contenenti solamente medici provenienti dalla stessa città, di cui almeno uno abbia superato nell'anno 2011 un totale di visite pari al 10% di tutte le visite della sua specializzazione nello stesso anno

1. Considerare tutte le specializzazioni e i loro medici
2. In ogni specializzazione, contare il numero di città diverse a cui appartengono i medici che ne fanno parte
3. Imporre che ciascun conteggio precedente valga UNO
4. In ogni specializzazione contenente solo medici della stessa città, contare le visite effettuate nel 2011 da ciascun medico che ne fa parte
5. In ogni specializzazione contenente solo medici della stessa città, calcolare il totale delle visite effettuate nel 2011 da tutti i medici che ne fanno parte
6. Imporre che esista almeno un medico che abbia effettuato nel 2011 più del 10% del totale delle visite effettuate nel 2011 da tutti i medici della sua specializzazione

# Tradurre in SQL ogni parte

---

1. Considerate tutte le specializzazioni e i loro medici
- 2 In ogni specializzazione, contate il numero di città diverse a cui appartengono i medici che ne fanno parte
3. Imporre che ciascun conteggio precedente valga uno

Risultato "A"

```
CREATE OR REPLACE VIEW SpecializzazioniTarget AS
SELECT Specializzazione
FROM Medico
GROUP BY Specializzazione
HAVING COUNT(DISTINCT Citta) = 1;
```

# Tradurre in SQL ogni parte

---

5. In ogni specializzazione contenente solo medici della stessa città, contate le visite effettuate nel 2011 da ciascun medico che ne fa parte

```
CREATE OR REPLACE VIEW TotaleVisiteMedici AS
SELECT M.Specializzazione,
       V.Medico,
       COUNT(*) AS QuanteVisite
FROM Medico M INNER JOIN Visita V ON M.Matricola = V.Medico
WHERE M.Specializzazione IN (A)
      AND YEAR(V.Data) = 2011
GROUP BY M.Specializzazione, V.Medico;
```

Risultato "B"

# Tradurre in SQL ogni parte

---

4. In ogni specializzazione contenente solo medici della stessa città,  
calcolare il totale delle visite effettuate nel 2011 da tutti i medici che ne fanno parte

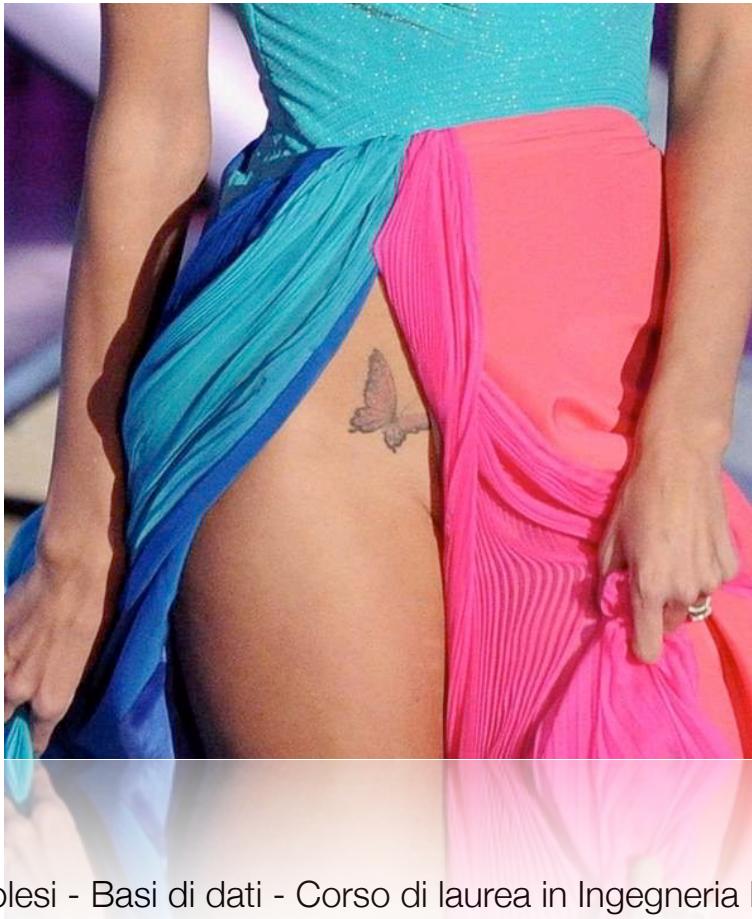
```
CREATE OR REPLACE VIEW TotaliVisiteSpecializzazioni AS  
SELECT B.Specializzazione,  
       SUM( B.QuanteVisite) AS TotaleVisite  
FROM B  
GROUP BY B.Specializzazione;
```

Risultato "C"

# Tradurre in SQL ogni parte

---

6. Importare che esista almeno un medico che ha effettuato nel 2011 più del 10% del totale delle visite effettuate nel 2011 da tutti i medici della sua specializzazione



# Tradurre in SQL ogni parte

6. Imporre che esista almeno un medico che abbia effettuato nel 2011 più del 10% del totale delle visite effettuate nel 2011 da tutti i medici della sua specializzazione

→ Data una specializzazione, si affianca al numero di visite effettuate, nel 2011, da un medico che ne fa parte (in B), il totale di visite effettuate da tutti i medici della sua specializzazione nel 2011 (in C). Dopodiché, si impone che il valore di sinistra sia superiore al 10% del valore di destra.

Specializzazione	Medico	QuanteVisite2011	Specializzazione	TotaleVisite2011
Otorinolaringoiatria	030	28	Otorinolaringoiatria	100
Otorinolaringoiatria	039	63	Otorinolaringoiatria	100
Otorinolaringoiatria	032	9	Otorinolaringoiatria	100
Cardiologia	041	45	Cardiologia	75
Cardiologia	048	30	Cardiologia	75

# Incollare tutti i pezzi

---

A

```
CREATE OR REPLACE VIEW SpecializzazioniTarget AS  
SELECT Specializzazione  
FROM Medico  
GROUP BY Specializzazione  
HAVING COUNT(DISTINCT Citta) = 1;
```

B

```
CREATE OR REPLACE VIEW TotaleVisiteMedici AS  
SELECT M.Specializzazione, V.Medico, COUNT(*) AS QuanteVisite  
FROM Medico M INNER JOIN Visita V ON M.Matricola = V.Medico  
WHERE M.Specializzazione IN (A)  
    AND YEAR(V.Data) = 2011  
GROUP BY M.Specializzazione, V.Medico;
```

C

```
CREATE OR REPLACE VIEW TotaliVisiteSpecializzazioni AS  
SELECT B.Specializzazione,  
       SUM(B.QuanteVisite) AS TotaleVisite  
FROM B  
GROUP BY B.Specializzazione;
```

# Incollare tutti i pezzi

---

B

```
CREATE OR REPLACE VIEW TotaleVisiteMedici AS  
SELECT M.Specializzazione, V.Medico, COUNT(*) AS QuanteVisite  
FROM Medico M INNER JOIN Visita V ON M.Matricola = V.Medico  
WHERE M.Specializzazione IN (A)  
      AND YEAR(V.Data) = 2011  
GROUP BY M.Specializzazione, V.Medico;
```

NATURAL JOIN



...più 'nature' di così...

C

```
CREATE OR REPLACE VIEW TotaliVisiteSpecializzazioni AS  
SELECT B.Specializzazione,  
      SUM(B.QuanteVisite) AS TotaleVisite  
FROM B  
GROUP BY B.Specializzazione;
```

# Incollare tutti i pezzi

```
CREATE OR REPLACE VIEW SpecializzazioniTarget AS  
SELECT Specializzazione  
FROM Medico  
GROUP BY Specializzazione  
HAVING COUNT(DISTINCT Citta) = 1;
```

A

```
CREATE OR REPLACE VIEW TotaleVisiteMedici AS  
SELECT M.Specializzazione, V.Medico, COUNT(*) AS QuanteVisite  
FROM Medico M INNER JOIN Visita V ON M.Matricola = V.Medico  
WHERE M.Specializzazione IN (A)  
      AND YEAR(V.Data) = 2011  
GROUP BY M.Specializzazione, V.Medico;
```

B

```
CREATE OR REPLACE VIEW TotaliVisiteSpecializzazioni AS  
SELECT B.Specializzazione,  
       SUM(B.QuanteVisite) AS TotaleVisite  
FROM B  
GROUP BY B.Specializzazione;
```

C

```
SELECT DISTINCT TVM.Specializzazione  
FROM TotaleVisiteMedici TVM  
      NATURAL JOIN  
TotaleVisiteSpecializzazioni TVS  
WHERE TVM.QuanteVisite > 0.1*TVS.TotaleVisite;
```

# Codice completo

---

```
CREATE OR REPLACE VIEW SpecializzazioniTarget AS
SELECT Specializzazione
FROM Medico
GROUP BY Specializzazione
HAVING COUNT(DISTINCT Citta) = 1;
```

```
CREATE OR REPLACE VIEW TotaleVisiteMedici AS
SELECT M.Specializzazione, V.Medico, COUNT(*) AS QuanteVisite
FROM Medico M
INNER JOIN Visita V ON M.Matricola = V.Medico
WHERE M.Specializzazione IN
(
    SELECT ST.Specializzazione
    FROM SpecializzazioniTarget ST
)
AND YEAR(V.Data) = 2011
GROUP BY M.Specializzazione, V.Medico;
```

# Codice completo (cont.)

---

```
CREATE OR REPLACE VIEW TotaliVisiteSpecializzazioni AS
SELECT TVM.Specializzazione,
       SUM( TVM.QuanteVisite) AS TotaleVisite
FROM TotaleVisiteMedici TVM
GROUP BY TVM.Specializzazione;
```

```
SELECT DISTINCT TVM.Specializzazione
FROM TotaleVisiteMedici TVM
      NATURAL JOIN
      TotaleVisiteSpecializzazioni TVS
WHERE TVM.QuanteVisite > 0.1*TVS.TotaleVisite;
```

# Data Manipulation Language (DML)

---



# Data Manipulation Language: cos'è?

---

Parte del linguaggio MySQL che permette di **inserire**, **cancellare** e **aggiornare** interi record o valori di attributi delle tabelle di un database

comandi **INSERT – DELETE – UPDATE**

# Inserimento

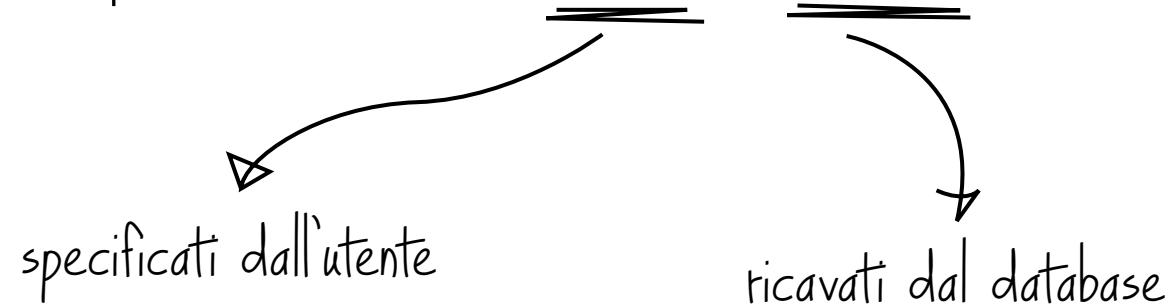
---



# Inserimento

---

Considerata una tabella, permette di **inserire un nuovo record** i cui valori degli attributi possono essere sia statici che ricavati



# Inserimento con valori statici: esempio

---

Inserire nel database un nuovo paziente, le cui informazioni sono:

- Nome: Elvira
- Cognome: Passerotti
- Sesso: F
- Data di nascita: 27 Ottobre 1965
- Città: Pisa
- Reddito: 1500 €
- Codice fiscale: PSSLVR65R67G702U

**INSERT INTO** *Paziente*

**VALUES** ('PSSLVR65R67G702U', 'Elvira', 'Passerotti', 'F',  
 '1965-10-27', 'Pisa', 1500);

*Non importa specificare  
lo schema se inserisco tutti i campi  
rispettando l'ordine dello schema*

# Inserimento: mancanza di informazioni

---

La scorsa settimana la dottoressa Clelia Celesti ha visitato il paziente Edoardo Lepre, codice fiscale ‘slq6’. La visita non è stata inserita e lo si vuole fare in ritardo non conoscendo più la data della visita. La visita non era mutuata.

*Attributi oggetto dell'inserimento*

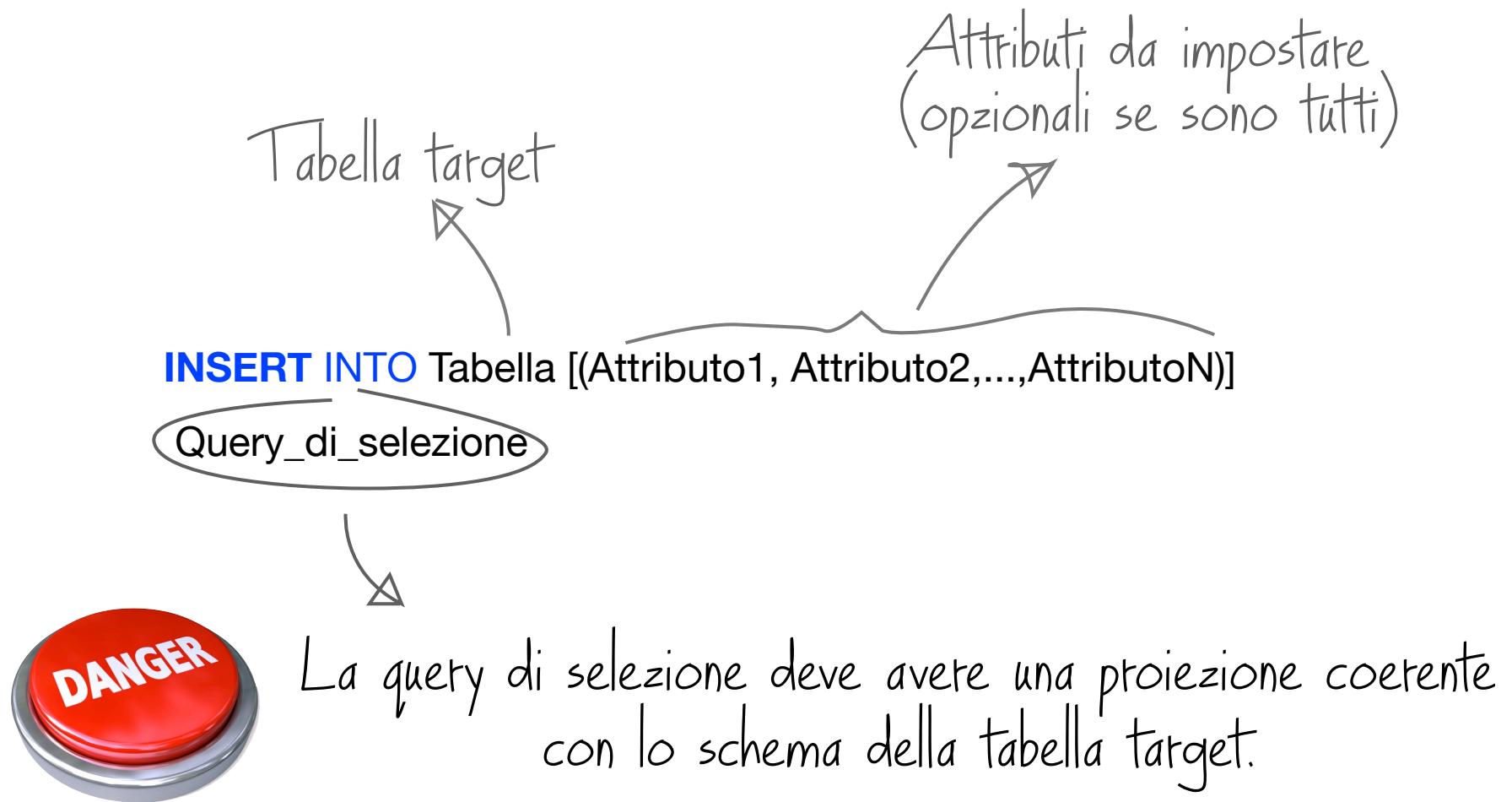


```
INSERT INTO Visita(Medico, Paziente, Mutuata)  
VALUES (010, 'slq6', FALSE);
```

*La data della visita (informazione perduta) sarà impostata automaticamente al default value*

# Inserimento con valori ricavati

---



# Inserimento con valori ricavati: esempio

---

Considerata la tabella VISITAVECCHIA avente lo schema riportato sotto, inserire in VISITAVECCHIA tutte le visite effettuate prima di due anni fa.

VISITAVECCHIA (CognomePaziente, CognomeMedico, DataVisita, Specializzazione)

```
INSERT INTO VisitaVecchia
    SELECT P.Cognome AS CognomePaziente,
           M.Cognome AS CognomeMedico,
           V.Data AS DataVisita,
           M.Specializzazione
      FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola
                INNER JOIN Paziente P ON V.Paziente = P.CodFiscale
     WHERE YEAR(V.Data) < YEAR(CURRENT_DATE) - 2;
```

*qual è la chiave primaria?*

# Aggiornamento

---



# Aggiornamento

---

Permette di **modificare il valore di uno o più attributi** di uno o più record  
con valori statici oppure ricavati

l'insieme di record si seleziona  
mediante una condizione

# Aggiornamento: esempio

---

Modificare in “mutuata” tutte le visite del mese corrente, effettuate da  
pazienti nati prima del 1925

**UPDATE** Visita

**SET** Mutuata = 1

**WHERE MONTH(Data) = MONTH(CURRENT\_DATE)**

**AND Paziente IN (SELECT CodFiscale**

**FROM Paziente**

**WHERE YEAR(DataNascita) < 1925);**

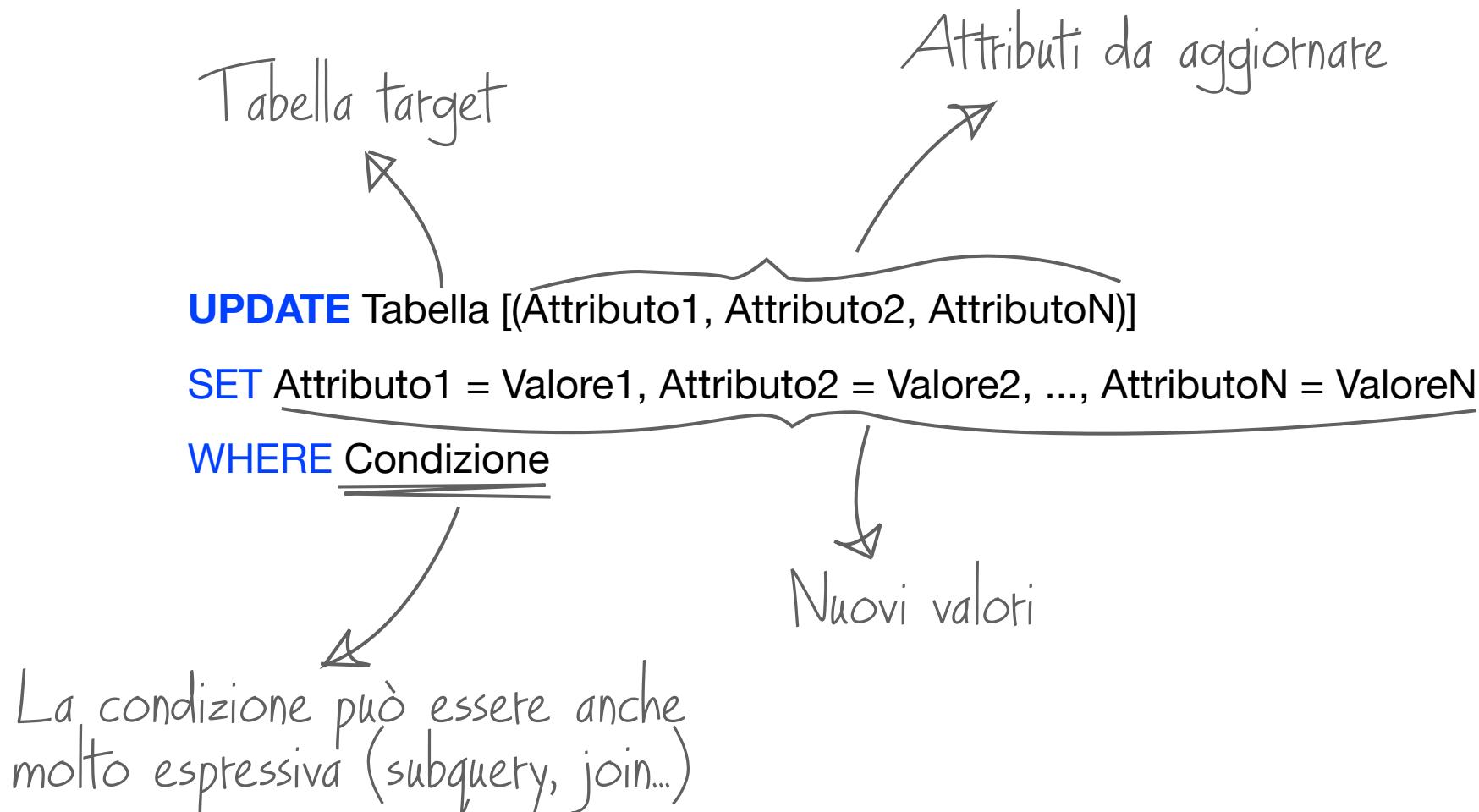
esprimibile con subquery



per le condizioni, stessa sintassi delle query di selezione

# Aggiornamento: sintassi MySQL

---



# Cancellazione

---



# Cancellazione

---

Permette di **cancellare uno o più record** dipendentemente dalla veridicità  
di una condizione, anche articolata

# Cancellazione: esempio

---

Il dottor Ettore Grigi ha lasciato la clinica. Rimuoverlo dal database.

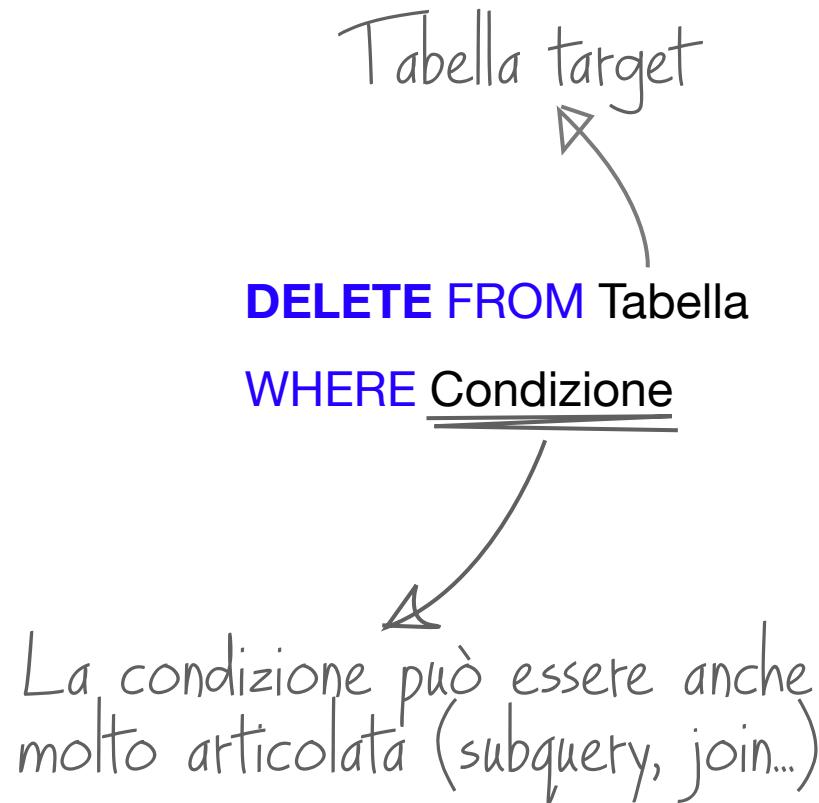
```
DELETE FROM Medico  
WHERE Nome = 'Ettore'  
AND Cognome = 'Grigi';
```



la rimozione fatta in questo modo presuppone che ci sia un solo medico di nome Ettore Grigi. È comunque sempre meglio usare la chiave primaria per identificare il record da eliminare.

# Cancellazione: sintassi MySQL

---



# ...e le visite di Ettore Grigi?!

---

Il dottor Ettore Grigi ha lasciato la clinica. Rimuoverlo dal database.

Non deve rimanere  
traccia di lui!



Occorre eliminare anche tutte le sue visite!

# Cancellazione completa

Tabella target

A		
a	b	c
A		

Tabelle "referenced"

A		
a		
A		

A		
a		
A		

A		
a		
a		
A		



# Politiche di cancellazione completa

---

- cancellazione **a mano** sulle tabelle “referenced”
- politica **ON DELETE NO ACTION | SET DEFAULT  
SET NULL | CASCADE**



Stabilità in fase di creazione delle tabelle mediante i vincoli di integrità referenziale.

# Cancellazione a mano

---

Il dottor Ettore Grigi ha lasciato la clinica. Rimuoverlo dal database sapendo che non ha omonimi fra i medici della clinica, e cancellare tutte le sue visite.

```
DELETE FROM Medico  
WHERE Nome = 'Ettore'  
      AND Cognome = 'Grigi';
```

```
DELETE FROM Visita  
WHERE Medico =  
  (  
    SELECT Matricola  
    FROM Medico  
    WHERE Nome = 'Ettore'  
      AND Cognome = 'Grigi'  
  );
```

# Riferimento alla target table nella condizione...

---

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

# Soluzione

---

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE FROM Medico
WHERE Matricola IN
(
    SELECT M1.Matricola
    FROM Medico M1
    LEFT OUTER JOIN
    (
        SELECT V2.Medico AS MedicoMutuato
        FROM Visita V2
        INNER JOIN
            Medico M2 ON V2.Medico = M2.Matricola
        WHERE M2.Citta = 'Pisa'
            AND V2.Mutuata IS TRUE
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE) - 1
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
    ) AS D
    ON M1.Matricola = D.MedicoMutuato
    WHERE D.MedicoMutuato IS NULL
);
```

# ...e il delfino si arrabbia!

---

error : You can't specify target table 'MEDICO' for update  
in FROM clause

Non si può mettere la tabella target  
nel FROM della query contenuta nella clausola WHERE!  
Il DBMS dovrebbe fare due accessi simultanei in lettura e  
scrittura alla target table, leggendola mentre essa viene  
modificata. Ciò è pericoloso e quindi proibito. Occorre prima  
individuare i record da eliminare, dopodiché accedere in  
scrittura alla target table ed eliminarli effettivamente.



# Cosa non gli piace?

Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE FROM Medico
WHERE Matricola IN
(
    SELECT M1.Matricola
    FROM Medico M1
        LEFT OUTER JOIN
    (
        SELECT V2.Medico AS MedicoMutuato
        FROM Visita V2
            INNER JOIN
        Medico M2 ON V2.Medico = M2.Matricola
        WHERE M2.Citta = 'Pisa'
            AND V2.Mutuata IS TRUE
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE) - 1
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
    ) AS D
    ON M1.Matricola = D.MedicoMutuato
    WHERE D.MedicoMutuato IS NULL
);
```



# Al mattino, pane e volpe!

---



# Derived table



```
DELETE FROM Medico
WHERE Matricola IN
(
    SELECT *
    FROM
    (
        SELECT M1.Matricola
        FROM Medico M1
        LEFT OUTER JOIN
        (
            SELECT V2.Medico AS MedicoMutuato
            FROM Visita V2
            INNER JOIN
            Medico M2 ON V2.Medico = M2.Matricola
            WHERE M2.Citta = 'Pisa'
                AND V2.Mutuata IS TRUE
                AND MONTH(V2.Data) = MONTH(CURRENT_DATE) - 1
                AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
        ) AS D
        ON M1.Matricola = D.MedicoMutuato
        WHERE D.MedicoMutuato IS NULL
    ) AS DD
);
```

La derived table DD è calcolata e memorizzata. Prendendo da DD i record da cancellare in Medico, non si fa doppio accesso a Medico

# Join anticipato



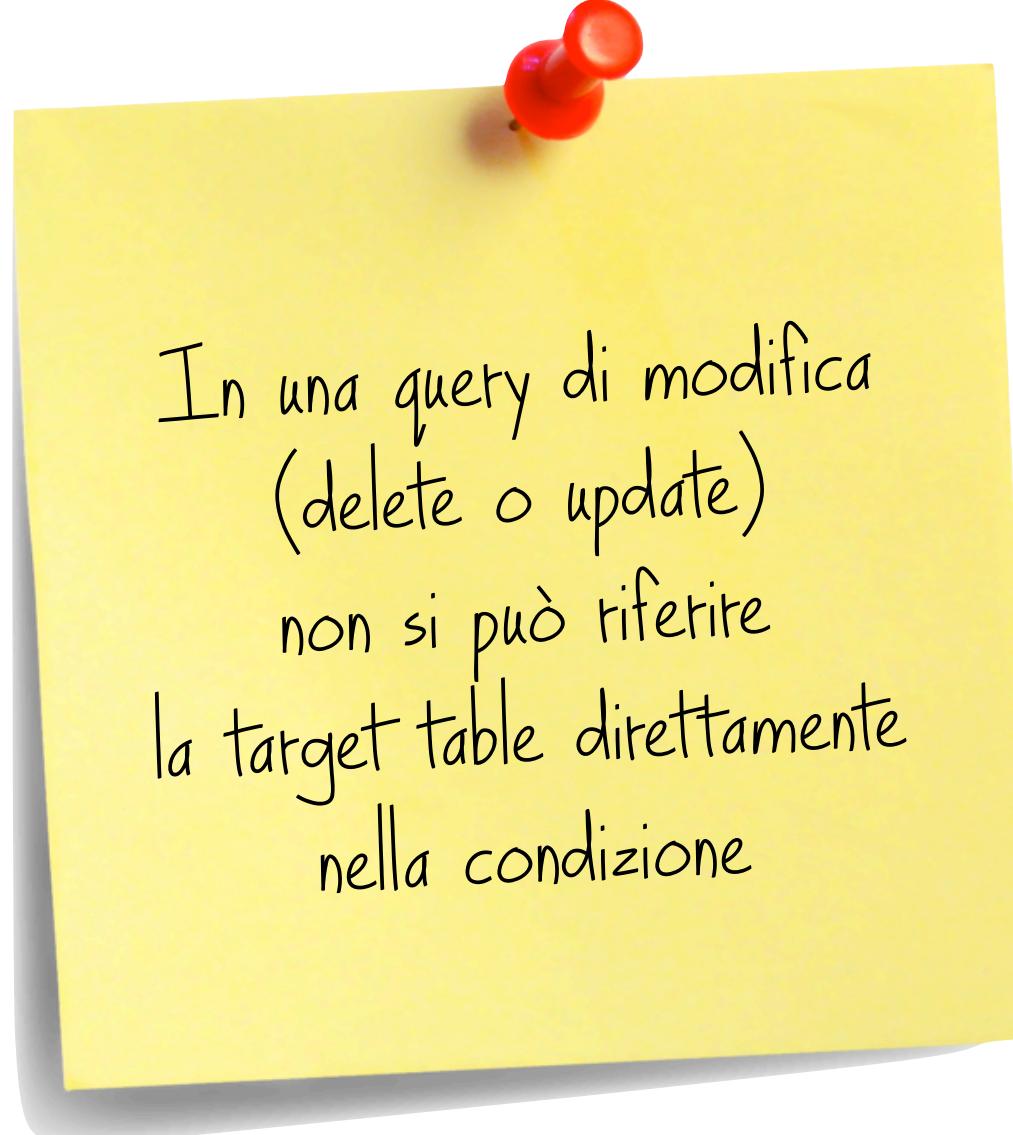
Rimuovere dal database tutti i medici di Pisa che non hanno effettuato visite mutuate il mese scorso.

```
DELETE M1.*  
FROM Medico M1  
      LEFT OUTER JOIN  
      (  
          SELECT V2.Medico AS MedicoMutuato  
          FROM Visita V2  
            INNER JOIN  
            Medico M2 ON V2.Medico = M2.Matricola  
          WHERE M2.Citta = 'Pisa'  
            AND V2.Mutuata IS TRUE  
            AND MONTH(V2.Data) = MONTH(CURRENT_DATE) - 1  
            AND YEAR(V2.Data) = YEAR(CURRENT_DATE)  
      ) AS D  
      ON M1.Matricola = D.MedicoMutuato  
WHERE D.MedicoMutuato IS NULL;
```

occorre scrivete M1.\* perché facendo il join anticipato il DBMS ha bisogno di sapere qual è la target table

# Importante

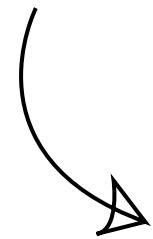
---



# ...e anche in aggiornamento

---

Anche quando si effettua un aggiornamento (comando update) MySQL  
**vieta l'inserimento della tabella target nella condizione WHERE.**



si possono usare le derived table  
oppure il join anticipato

# Esempio

---

Aumentare del 5% la parcella ai medici che hanno effettuato nell'ultimo anno più della metà delle visite della loro specializzazione.

# Modo sbagliato

Aumentare del 5% la parcella ai medici che hanno effettuato nell'ultimo anno più della metà delle visite della loro specializzazione.

```
UPDATE Medico M
SET M.Parcella = M.Parcella*1.05
WHERE (
    SELECT COUNT(*)
    FROM Visita V
    WHERE V.Medico = M.Matricola
        AND YEAR(V.Data) = YEAR(CURRENT_DATE)
)
>
0.5*(
    SELECT COUNT(*)
    FROM Visita V2 INNER JOIN Medico M2
        ON V2.Medico = M2.Matricola
    WHERE M2.Specializzazione = M.Specializzazione
        AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
);
```

La tabella Medico viene bloccata  
dal comando update e non la si può  
riferire nella condizione



Query	Message
UPDATE Medico M	error : You can't specify target table 'M' for update in FROM clause

# Aggiornamento con join anticipato

---

Aumentare del 5% la parcella ai medici che hanno effettuato nell'ultimo anno più della metà delle visite della loro specializzazione.

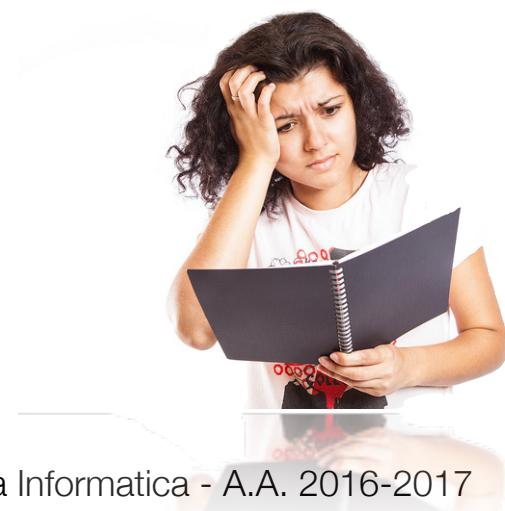
```
UPDATE Medico M
    NATURAL JOIN
    (
        SELECT V1.Medico AS Matricola, COUNT(*) AS TotaleVisiteMedico,
        (
            SELECT COUNT(*)
            FROM Visita V2 INNER JOIN Medico M2
                ON V2.Medico = M2.Matricola
            WHERE M2.Specializzazione = M1.Specializzazione
                AND YEAR(V2.Data) = YEAR(CURRENT_DATE)
        )
        AS TotaleVisiteSpecializzazione
        FROM Visita V1 INNER JOIN Medico M1 ON V1.Medico = M1.Matricola
        WHERE YEAR(V1.Data) = YEAR(CURRENT_DATE)
        GROUP BY V1.Medico
    ) AS D
SET M.Parcella = M.Parcella*1.05
WHERE D.TotaleVisiteMedico > D.TotaleVisiteSpecializzazione*0.5;
```

# Leggibilità e view

---

Per evitare di non capirci più nulla fra un mese e rendere il codice più modulare, nonché più leggibile, si raccomanda di **utilizzare le view**

Le view si possono usare sia in update  
che in delete, per frammentare il codice



# Con le view

```
CREATE OR REPLACE VIEW VisiteSpecializzazione AS
SELECT M.Specializzazione,
       COUNT(*) AS TotaleVisiteSpecializzazione
  FROM Visita V INNER JOIN Medico M
    ON V.Medico = M.Matricola
   WHERE YEAR(V.Data) = YEAR(CURRENT_DATE)
 GROUP BY M.Specializzazione;
```

```
CREATE OR REPLACE VIEW VisiteMedico AS
SELECT V.Medico,
       COUNT(*) AS TotaleVisiteMedico,
  FROM Visita V INNER JOIN Medico M
    ON V.Medico = M.Matricola
   WHERE YEAR(V.Data) = YEAR(CURRENT_DATE)
 GROUP BY V.Medico;
```

```
UPDATE Medico M
      NATURAL JOIN
      VisiteMedico VM
      NATURAL JOIN
      VisiteSpecializzazione VS
     SET M.Parcella = 1.05*M.Parcella
    WHERE VM.TotaleVisiteMedico > VS.TotaleVisiteSpecializzazione*0.5;
```

# Avvisi

---

Come di consueto, per casa vi consiglio di **rivedere attentamente i concetti** della lezione, dopodiché provate a risolvere sia gli esercizi che trovate nel corso delle slide che quelli per casa. **È molto importante che facciate gli esercizi**, anche solo un paio.

Ho inserito alcune **nuove slide relative al costrutto ALL** (dalla 65 alla 72) in base alle domande che mi avete fatto a lezione. Spero che siano chiare e che risolvano i vostri dubbi, specie sui pari merito, quelli sono importanti. Le ultime 5 slide mostrano invece un esempio della non riferibilità della target table in aggiornamento (lo stesso visto per la delete). Ho lasciato anche 5 slide (non fatte a lezione) sulla differenza simmetrica. Consideratele come esercizio.

Il querying termina con questa lezione. Prossima **lezione** giovedì **27 Aprile** in aula F9 dalle ore 10:30 alle ore 12:30. La lezione sarà relativa alle stored procedure. Se ci saranno variazioni riceverete una comunicazione per email.

# Esercizi

---

1. Indicare cognome e nome dei pazienti visitati almeno una volta da tutti i cardiologi di Pisa nel primo trimestre del 2015. [Risolvere in due modi diversi]
2. Selezionare cognome e specializzazione dei medici la cui parcella è superiore alla media delle parcelle della loro specializzazione e che, nell'anno 2011, hanno visitato almeno un paziente che non avevano mai visitato prima.
3. Impostare come mutuate (attributo omonimo pari a 1) tutte le visite ortopediche che coinvolgono pazienti già visitati precedentemente almeno due volte dallo stesso medico.
4. Scrivere una query che restituisca nome e cognome del medico che, al 31/12/2014, aveva visitato un numero di pazienti superiore a quelli visitati da ciascun medico della sua stessa specializzazione.
5. Scrivere una query che restituisca il codice fiscale dei pazienti che sono stati visitati sempre dal medico avente la parcella più alta, in tutte le specializzazioni. Se, anche per una sola specializzazione, non vi è un unico medico avente la parcella più alta, la query non deve restituire alcun risultato.