

Basi di dati cod. 861II [9 CFU]

Corso di Laurea in Ingegneria Informatica

Oracle MySQL
A.A. 2022-2023

Francesco Pistolesi
Dipartimento di Ingegneria dell'Informazione
Università di Pisa
francesco.pistolesi@unipi.it

Subquery



Subquery: cosa sono?

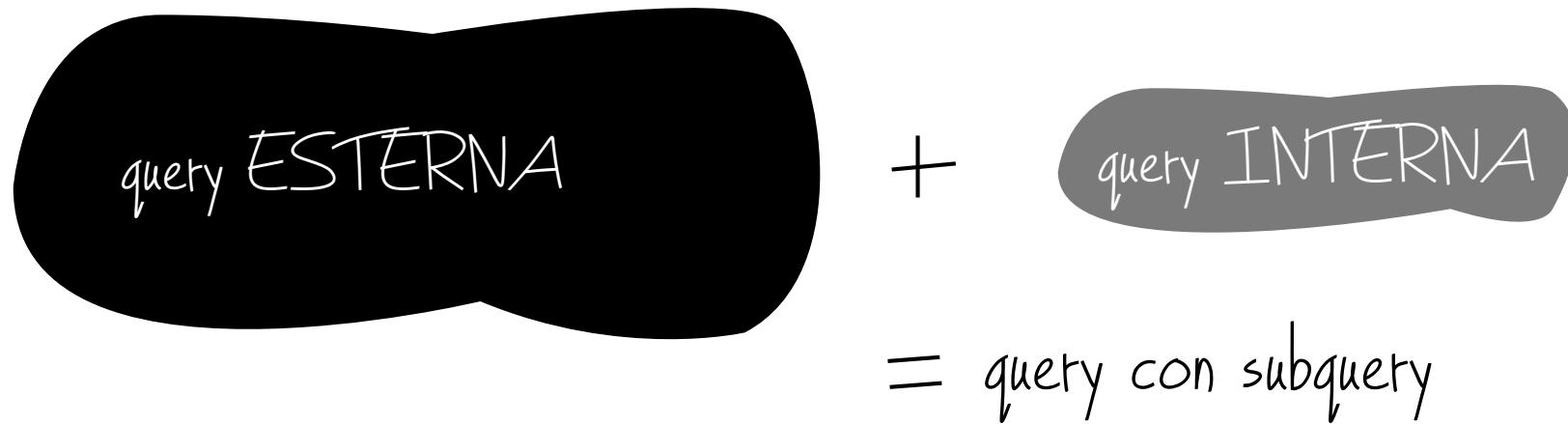
dette subquery

Sono query che possono essere **incapsulate** in un'altra query



in modo non correlato
o correlato...

Subquery: struttura



Subquery: perché?

Rappresentano **un'alternativa al join** per query su più tabelle



a volte sono più semplici da capire e da scrivere

Noncorrelated subquery



Noncorrelated subquery: cosa sono

eliminati alla fine dell'esecuzione

Si incapsulano nel WHERE per ottenere **risultati**
usati dalla outer query per determinare il result set finale

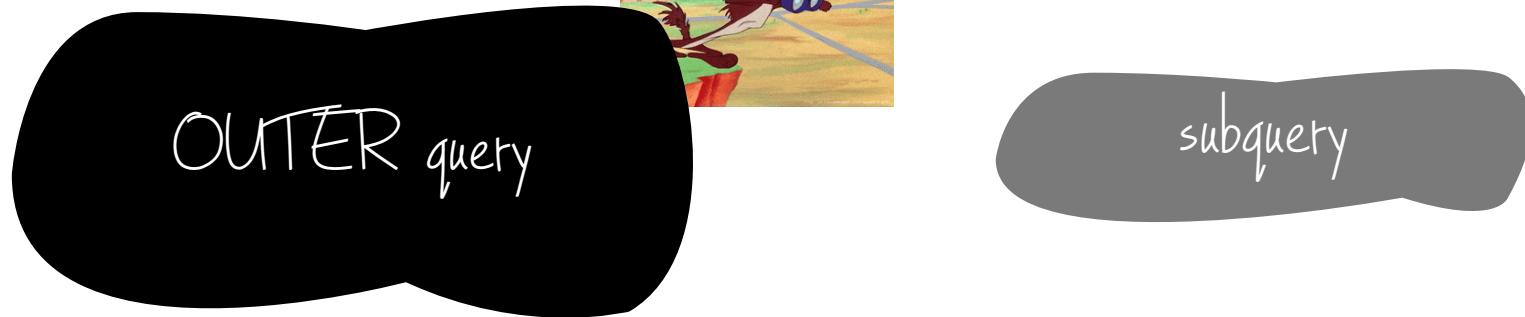


i record ottenuti dalla subquery non dipendono dalla outer query

Noncorrelated subquery: risultato

→ quello che vede l'utente
(il risultato della outer query)

Un record fa parte del risultato se i valori che assume su un sottoinsieme di attributi si trova **anche** in (almeno) un record del result set della subquery



Noncorrelated subquery in MySQL

Indicare nome, cognome e parcella degli ortopedici che hanno effettuato almeno una visita nell'anno 2013

```
SELECT M.Nome, M.Cognome, M.Parcella  
FROM Medico M  
WHERE M.Specializzazione = 'Ortopedia'  
AND M.Matricola IN (  
    SELECT V.Medico  
    FROM Visita V  
    WHERE YEAR(V.Data) = 2013)
```

qui si può spezzare

outer query

subquery

'IN' permette di controllare la presenza della matricola all'interno del risultato della subquery

Damn! How does it work?

Indicare nome, cognome e parcella degli ortopedici che hanno effettuato almeno una visita nell'anno 2013

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
29858	Neri	Rino	Dermatologia	110	Siena

Nome	Cognome	Parcella
Marta	Rossi	120



Matricola
18339
35512

IN ?

Medico
35512
35512
16220
35512



VISITA

```
SELECT M.Nome, M.Cognome, M.Parcella  
FROM Medico M  
WHERE M.Specializzazione = 'Ortopedia'  
AND M.Matricola IN  
(  
    SELECT V.Medico  
    FROM Visita V  
    WHERE YEAR(V.Data) = 2013  
)
```

Medico	Paziente	Data	Mutuata
35512	GTTRTA	2013-01-15	1
29858	MNZMBT	2012-11-30	0
18339	CPRLND	2012-10-28	1
35512	GTTRTA	2013-02-20	1
16220	MNZMBT	2013-01-25	0
35512	LPRNTA	2013-03-01	0
18339	CPRLND	2012-01-01	0

Noncorrelated subquery: negazione

Usando **NOT IN** un record della outer query va nel risultato solo **se non è presente** nel result set della subquery



IN controlla che il record sia anche nel risultato della subquery, mentre NOT IN che non ci sia

Negazione: esempio

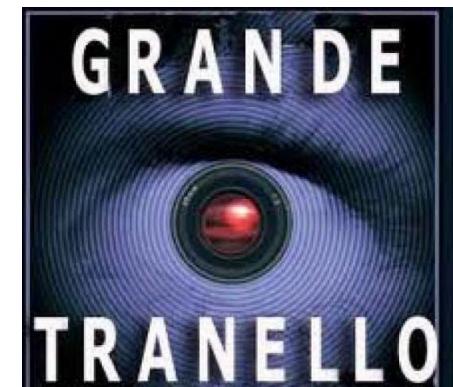
Indicare i cognomi dei pazienti che **non appartengono** anche a un medico

```
SELECT DISTINCT P.Cognome  
FROM Paziente P  
WHERE P.Cognome NOT IN (  
    SELECT M.Cognome  
    FROM Medico M  
);
```

il cognome di un paziente va nel risultato se non c'è
un medico a cui tale cognome appartiene

La query insidiosa

Indicare nome, cognome e specializzazione dei medici che hanno effettuato visite eccetto che il giorno 1° Marzo 2013



Soluzione

Indicare nome, cognome e specializzazione dei medici **che hanno effettuato visite** eccetto che il giorno 1° Marzo 2013

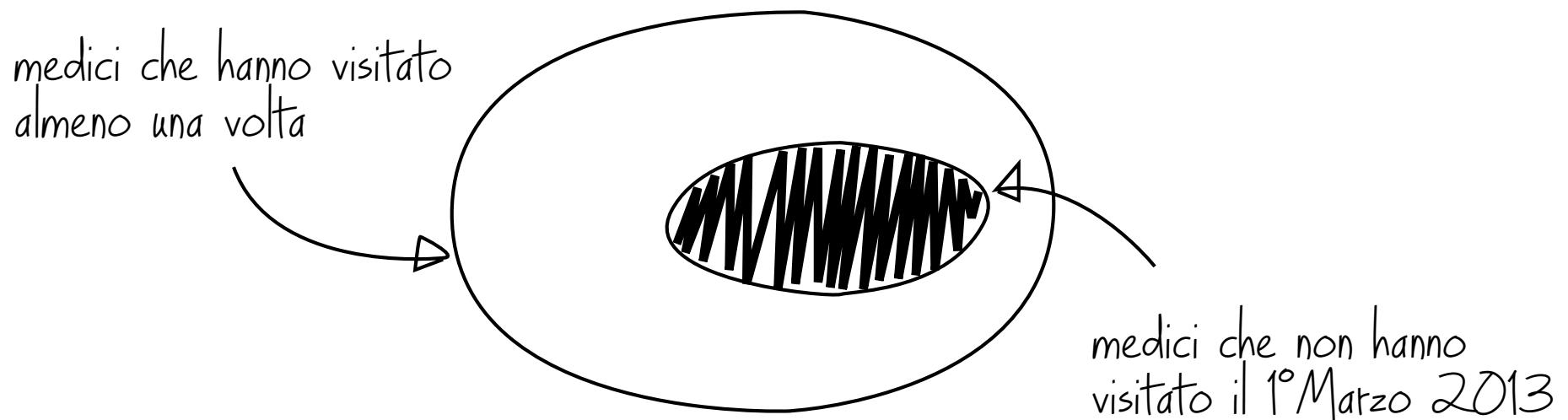
```
SELECT M.Nome, M.Cognome, Specializzazione  
FROM Medico M  
WHERE Matricola NOT IN (SELECT V.Medico  
                           FROM Visita V  
                           WHERE V.Data = '2013-03-01'  
);
```



se un medico non ha effettuato alcuna visita va nel risultato!

Perché è sbagliata?

Indicare nome, cognome e specializzazione dei medici che hanno effettuato visite eccetto che il giorno 1° Marzo 2013



Soluzione corretta

Indicare nome, cognome e specializzazione dei medici che hanno effettuato visite eccetto che il giorno 1° Marzo 2013

```
SELECT M.Nome, M.Cognome, M.Specializzazione  
FROM Medico M  
WHERE M.Matricola IN (  
    SELECT V.Medico  
    FROM Visita V  
)  
AND M.Matricola NOT IN (  
    SELECT V.Medico  
    FROM Visita V  
    WHERE V.Data = '2013-03-01'
```



se un medico non ha effettuato alcuna visita **NON** va nel risultato!

Versione join-equivalente

Indicare nome, cognome e specializzazione dei medici che hanno effettuato visite eccetto che il giorno 1° Marzo 2013

```
SELECT M.Nome, M.Cognome, M.Specializzazione  
FROM Medico M  
WHERE M.Matricola IN ( SELECT V.Medico  
                        FROM Visita V )  
      AND M.Matricola NOT IN ( SELECT V.Medico  
                                FROM Visita V  
                                WHERE V.Data = '2013-03-01' )
```

con subquery

versione join-equivalente

```
SELECT M.Nome, M.Cognome, M.Specializzazione  
FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
LEFT OUTER JOIN( SELECT *  
                  FROM Visita  
                  WHERE Data = '2013-03-01' ) AS D  
ON V.Medico = D.Medico  
WHERE D.Medico IS NULL
```

Equivalenza join-subquery

Data una query con subquery è **sempre** possibile passare alla versione basata su join, e viceversa



il query optimizer sfrutta questa equivalenza per riscrivere le query il cui codice avrebbe bassa performance

Versione join-equivalente (slide 35)

Indicare nome, cognome e specializzazione dei medici che hanno effettuato visite eccetto che il giorno 1° Marzo 2013

```
SELECT DISTINCT M.Nome, M.Cognome, M.Specializzazione  
FROM Visita V INNER JOIN Medico M ON V.Medico = M.Matricola  
LEFT OUTER JOIN(  
    SELECT *  
    FROM Visita  
    WHERE Data = '2013-03-01'  
) AS D  
ON V.Medico = D.Medico  
WHERE D.Medico IS NULL;
```

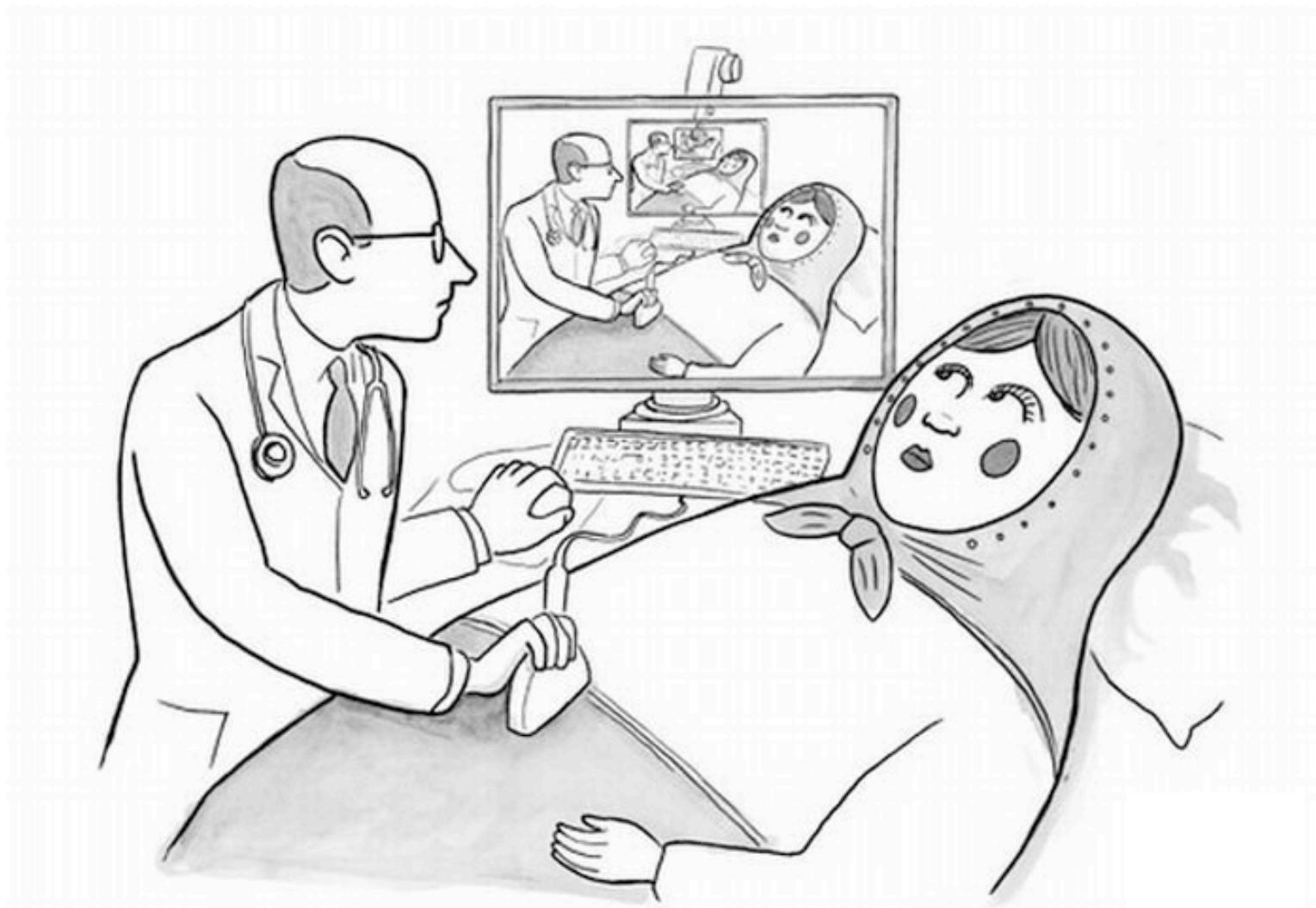


data una query con subquery è sempre possibile passare alla versione basata su join, e viceversa

...la domanda sorge spontanea



Quante subquery?



Annidamento multiplo

Non c'è limite *teoricamente* al numero di query che si possono annidare l'una nell'altra

Annidamento multiplo: esempio

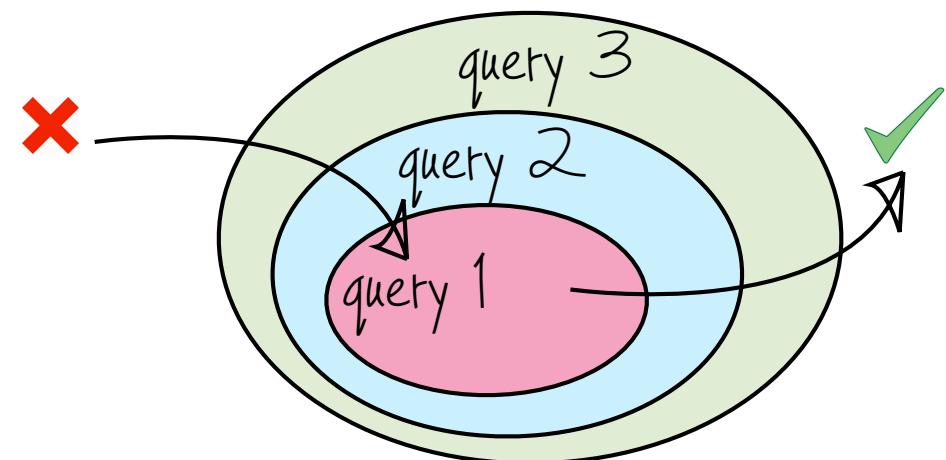
Indicare il numero di pazienti di Siena, mai visitati da ortopedici
da Paziente ↗ da Visita ↗ ↗ da Medico

```
SELECT COUNT(*)  
FROM Paziente P  
WHERE P.Citta = 'Siena'  
AND P.CodFiscale NOT IN (  
    SELECT V.Paziente  
    FROM Visita V  
    WHERE V.Medico IN (  
        SELECT M.Matricola  
        FROM Medico M  
        WHERE M.Specializzazione = 'Ortopedia' )  
);
```

Visibilità

In una subquery si possono riferire **tutti gli attributi delle query esterne**, a qualsiasi livello di annidamento esterno essi si trovino

si esce ma non si entra!



Subquery scalari

une seule

solo uno

nut ein

Restituiscono **un unico record**, composto da un **unico attributo**

just one

uno e basta!

Subquery vs subquery scalari

The image shows two pages from a notebook comparing subqueries and scalar subqueries.

Subquery scalare

produce un unico record con un unico attributo.

Se è numerico, si usano $<$, \leq , $=$, \geq , $>$, \neq per il confronto con i record della query esterna

Subquery

produce un insieme di record, si lega alla query esterna mediante IN o NOT IN.

Se produce un solo record si usa $=$ o \neq

10

11

Subquery scalare e operatori di confronto

Indicare il numero degli otorini **aventi parcella più alta della media** delle parcelle dei medici della loro specializzazione.

Soluzione

Indicare il numero degli otorini **aventi parcella più alta della media** delle parcelle dei medici della loro specializzazione.

```
SELECT COUNT(*)  
FROM Medico M1  
WHERE M1.Specializzazione = 'Otorinolaringoiatria'  
AND M1.Parcella > (  
    SELECT AVG(M2.Parcella)  
    FROM Medico M2  
    WHERE M2.Specializzazione = 'Otorinolaringoiatria'  
);
```

La subquery restituisce un numero!!!

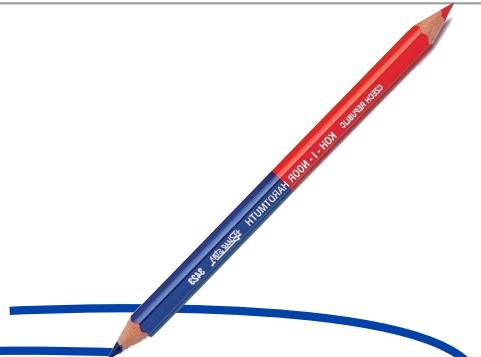
A volte ritornano...

Indicare qual è il reddito massimo, **e il nome e cognome di chi lo detiene**

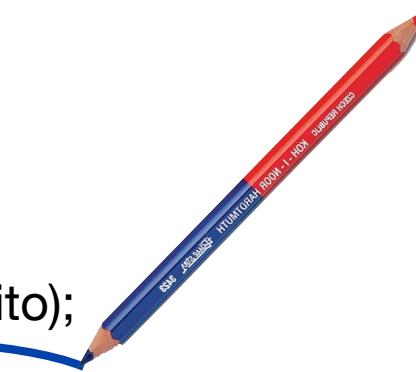


Gli errori blu, la follia

`SELECT MAX(Reddotto), Nome, Cognome
FROM Paziente;`



`SELECT Nome, Cognome
FROM Paziente
WHERE Reddotto = MAX(Reddotto);`



**cose da
PAZZI**

Soluzione

Indicare qual è il reddito massimo, **e il nome e cognome di chi lo detiene**

```
SELECT Reddito,  
       Nome,  
       Cognome  
  FROM Paziente  
 WHERE Reddito = (  
                   SELECT MAX(Reddito)  
                     FROM Paziente  
                   );
```



è l'unico modo corretto di risolvere il problema del record correlato al valore massimo su un attributo

Ma il male...

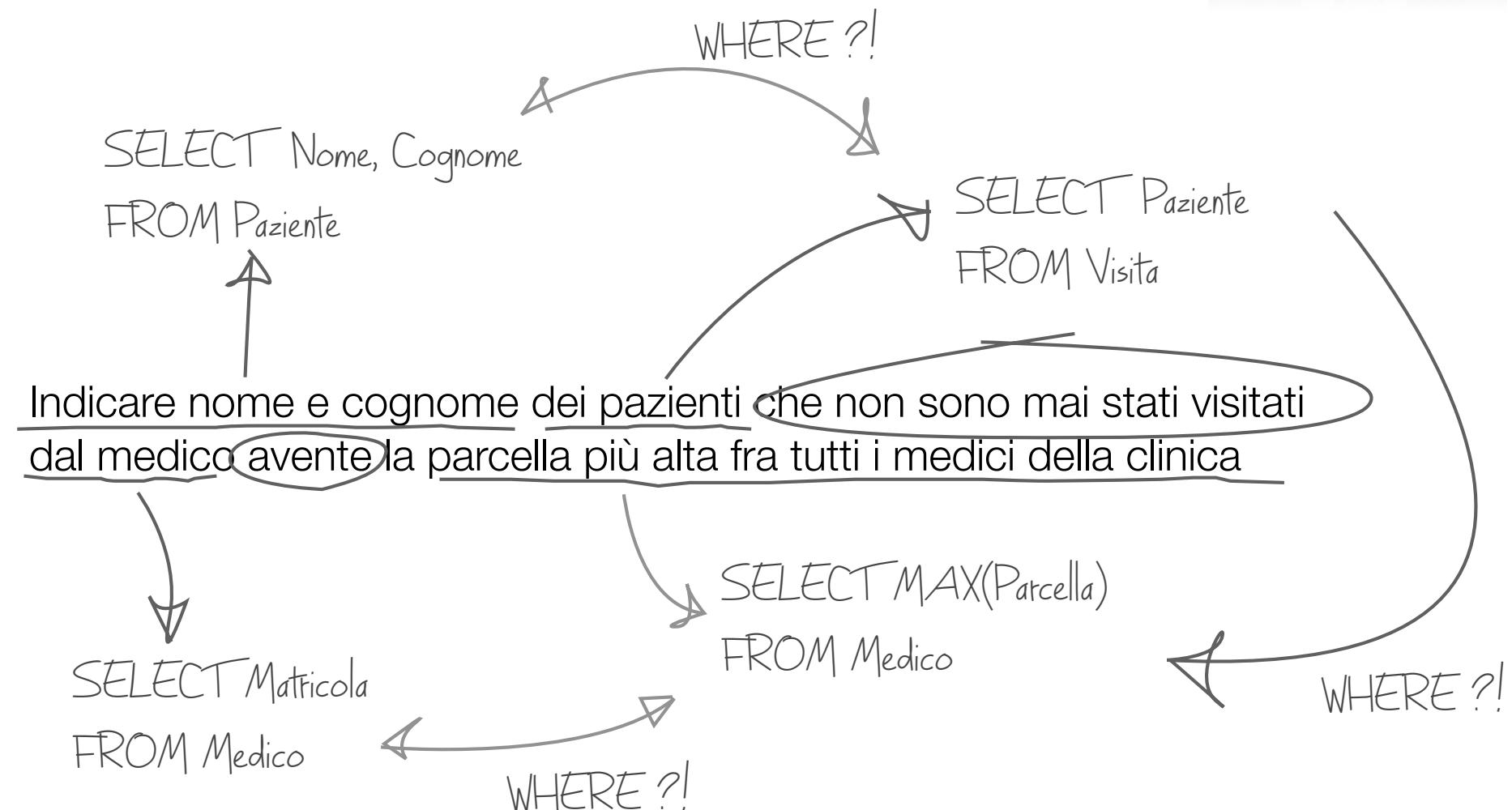
Indicare nome e cognome dei pazienti che non sono mai stati visitati dal medico
avente la parcella più alta fra tutti i medici della clinica



Sconfiggiamola!



Soluzione



Soluzione

Indicare nome e cognome dei pazienti che non sono mai stati visitati dal medico
avente la parcella più alta fra tutti i medici della clinica

A cartoon illustration of a nerd with glasses, a bow tie, and suspenders, looking up from behind a keyboard.

```
SELECT MAX(Parcella)
FROM Medico)
```

...ho nascosto un errore, non lo troverete mai!!!

Soluzione

Indicare nome e cognome dei pazienti che non sono mai stati visitati dal medico
avente la parcella più alta fra tutti i medici della clinica

```
SELECT Nome, Cognome  
FROM Paziente  
WHERE CodFiscale NOT IN (  
    SELECT Paziente  
    FROM Visita  
    WHERE Medico = (  
        )  
    );
```

questa subquery in generale può restituire più di un record!

```
    SELECT Matricola  
    FROM Medico  
    WHERE Parcella = ( SELECT MAX(Parcella)  
        FROM Medico)
```

più di un medico può avere la parcella massima, si deve usare IN!!!

Soluzione

Indicare nome e cognome dei pazienti che non sono mai stati visitati dal medico
avente la parcella più alta fra tutti i medici della clinica

```
SELECT Nome, Cognome
FROM Paziente
WHERE CodFiscale NOT IN (
    SELECT Paziente
    FROM Visita
    WHERE Medico IN (
        SELECT Matricola
        FROM Medico
        WHERE Parcella = ( SELECT MAX(Parcella)
                           FROM Medico)
    )
);
```

'IN' ammette che più di un medico abbia la parcella massima

Risoluzione mista subquery-join



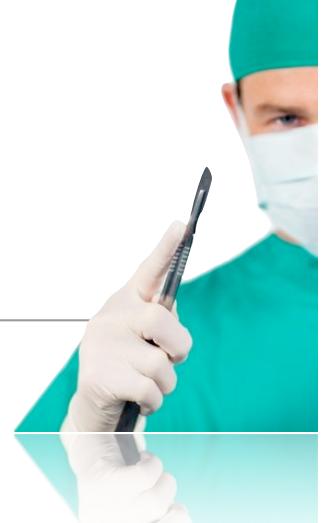
Una query può essere risolta anche **combinando** le subquery e join



Esempio

Indicare il numero di pazienti di età superiore a 50 anni visitati dai cardiologi di Pisa aventi parcella inferiore alla media delle parcelle dei cardiologi.

Bisturi!



SELECT COUNT(??)

Indicare il numero di pazienti di età superiore a 50 anni visitati

```
SELECT CodFiscale  
FROM Paziente P INNER JOIN Visita V  
ON P.CodFiscale=V.Paziente  
WHERE « Eta superiore a 50 anni »  
AND V.Medico IN
```

dai cardiologi di Pisa con parcella inferiore alla media delle parcelle dei cardiologi.

```
SELECT Matricola  
FROM Medico  
WHERE Specializzazione="Cardiologia"  
AND Citta="Pisa"
```

```
SELECT AVG(Parcella)  
FROM Medico  
WHERE Specializzazione="Cardiologia"  
AND Parcella <
```

...e adesso ricuciamo

Indicare il numero di pazienti di età superiore a 50 anni visitati dai cardiologi di Pisa aventi parcella inferiore alla media delle parcelle dei cardiologi.

```
SELECT COUNT(DISTINCT CodFiscale)
FROM Paziente P INNER JOIN Visita V
    ON P.CodFiscale = V.Paziente
WHERE CURRENT_DATE > P.DataNascita + INTERVAL 50 YEAR
    AND V.Medico IN (
        SELECT M.Matricola
        FROM Medico M
        WHERE M.Specializzazione = 'Cardiologia'
            AND M.Citta = 'Pisa'
            AND M.Parcella < ( SELECT AVG(M2.Parcella)
                FROM Medico M2
                WHERE M2.Specializzazione = 'Cardiologia' )
    );
```



Correlated subquery



Correlated subquery

uno per ogni tupla della query esterna

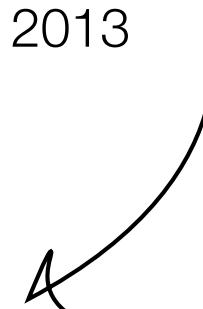
Il loro risultato **dipende** da ciascuna tupla della query esterna

concetto simile a una chiamata di funzione



Correlated subquery: esempio

Indicare matricola e parcella dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013



il paziente non è MAI stato visitato prima da quel medico

Correlated subquery: esempio

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013



Correlated subquery: esempio

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013

VISITA		
Medico	Paziente	Data
35512	GTTFBL	2013-10-20
29858	MNZMBT	2012-11-30
18339	CPRLND	2012-09-28
35512	GTTFBL	2014-01-19
16220	MNZMBT	2013-10-25
35512	LPRNTA	2013-03-01
18339	CPRLND	2013-10-01

è sufficiente la sola tabella Visita!

Correlated subquery: come funzionano

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013

Visite di Ottobre 2013

Medico	Paziente	Data
35512	GTTFBL	2013-10-20
16220	MNZMBT	2013-10-25
18339	CPRLND	2013-10-01

Tutte le visite

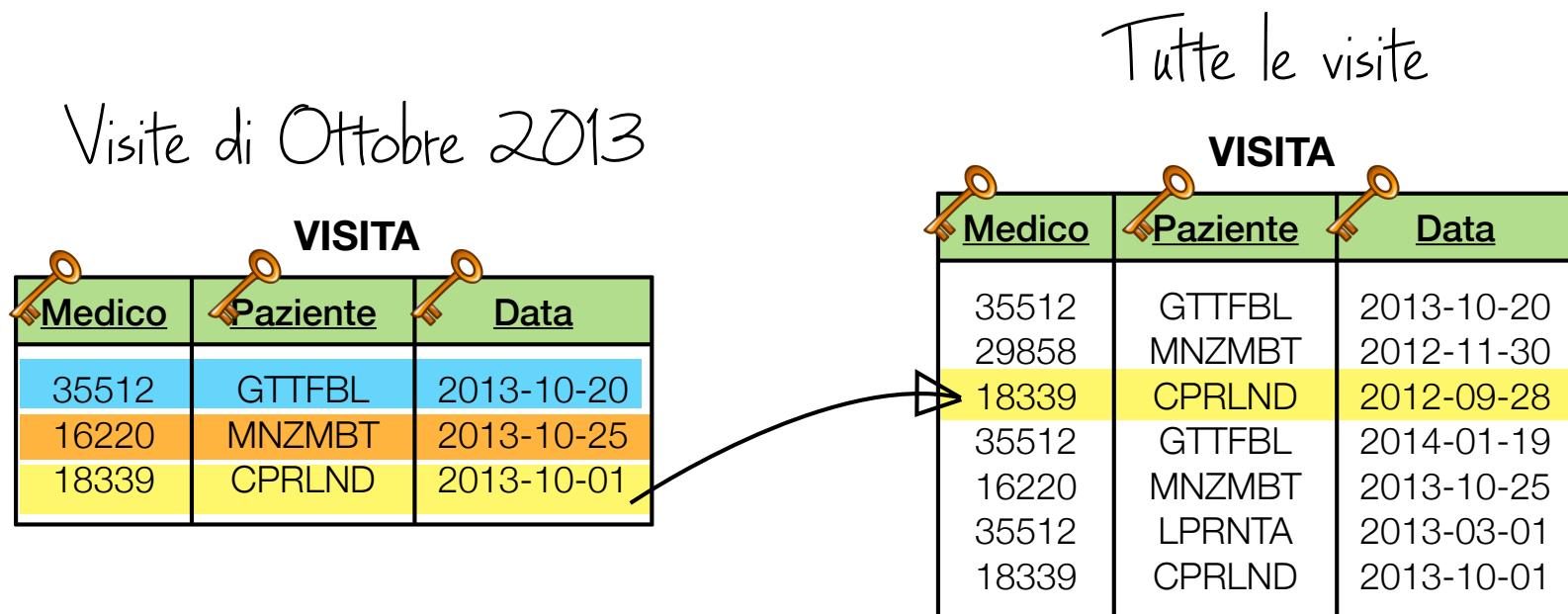
Medico	Paziente	Data
35512	GTTFBL	2013-10-20
29858	MNZMBT	2012-11-30
18339	CPRLND	2012-09-28
35512	GTTFBL	2014-01-19
16220	MNZMBT	2013-10-25
35512	LPRNTA	2013-03-01
18339	CPRLND	2013-10-01

Risultato

Medico
35512
16220

Correlated subquery: come funzionano

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013



Risultato

<u>Medico</u>
35512
16220

Correlated subquery in MySQL

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013

```
SELECT DISTINCT V1.Medico  
FROM Visita V1  
WHERE YEAR(V1.Data) = 2013  
    AND MONTH(V1.Data) = 10
```

query esterna
visite di Ottobre 2013

```
    AND V1.Paziente NOT IN ( SELECT V2.Paziente  
                            FROM Visita V2  
                            WHERE V2.Medico = V1.Medico  
                                AND V2.Data < V1.Data )
```

correlated subquery
eseguita per ogni tupla selezionata dalla query esterna

Keep in mind

Una correlated subquery

è eseguita

PER OGNI record

della query esterna e il suo
risultato ne dipende

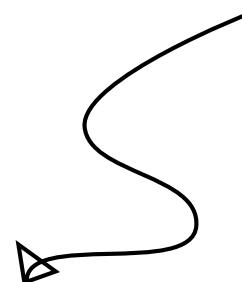
Correlated subquery nel SELECT

Una correlated subquery può essere inserita **anche nel SELECT** per calcolare “al volo” un valore da inserire, come attributo, nel risultato

deve essere SCALARE!!!

Esempio

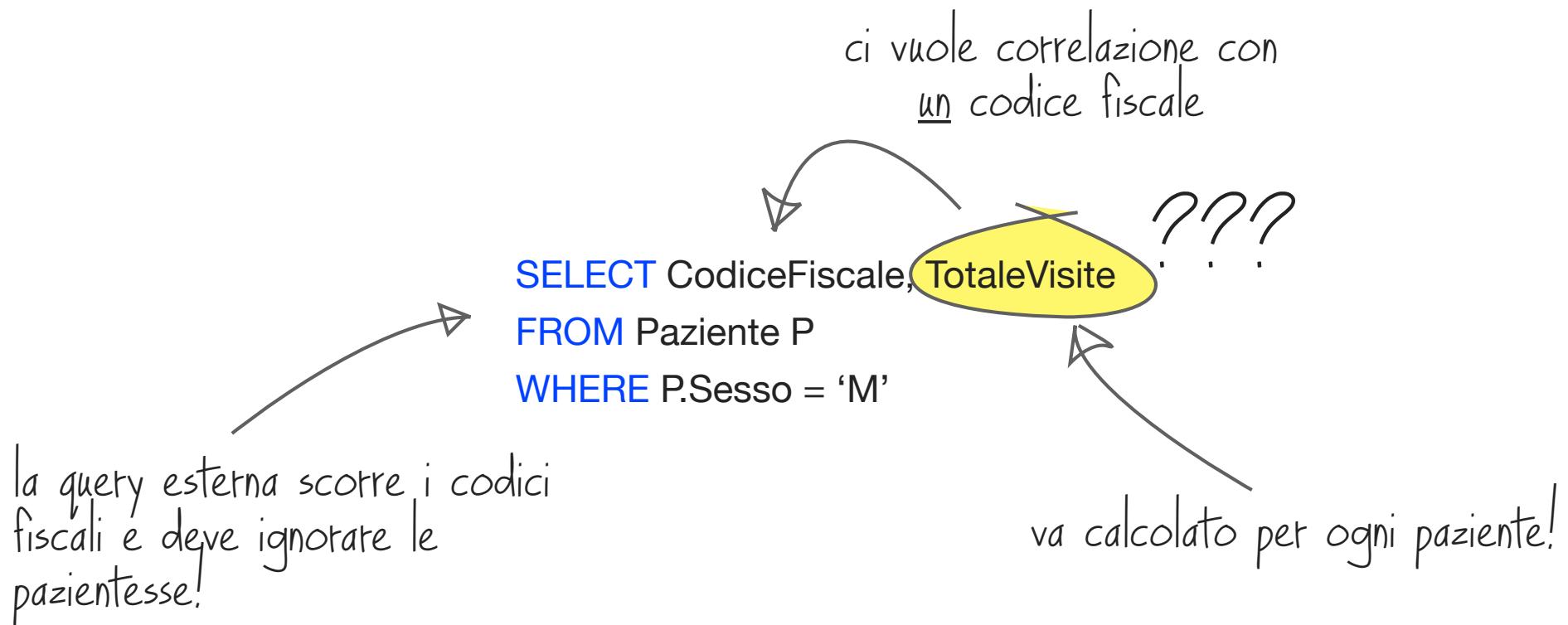
Considerato ciascun paziente di sesso maschile, indicarne il nome
e il numero di visite effettuate



è un'informazione da proiettare, non una condizione: va nel select!

Esempio

Considerato ciascun paziente di sesso maschile, indicarne il nome e
il numero di visite effettuate



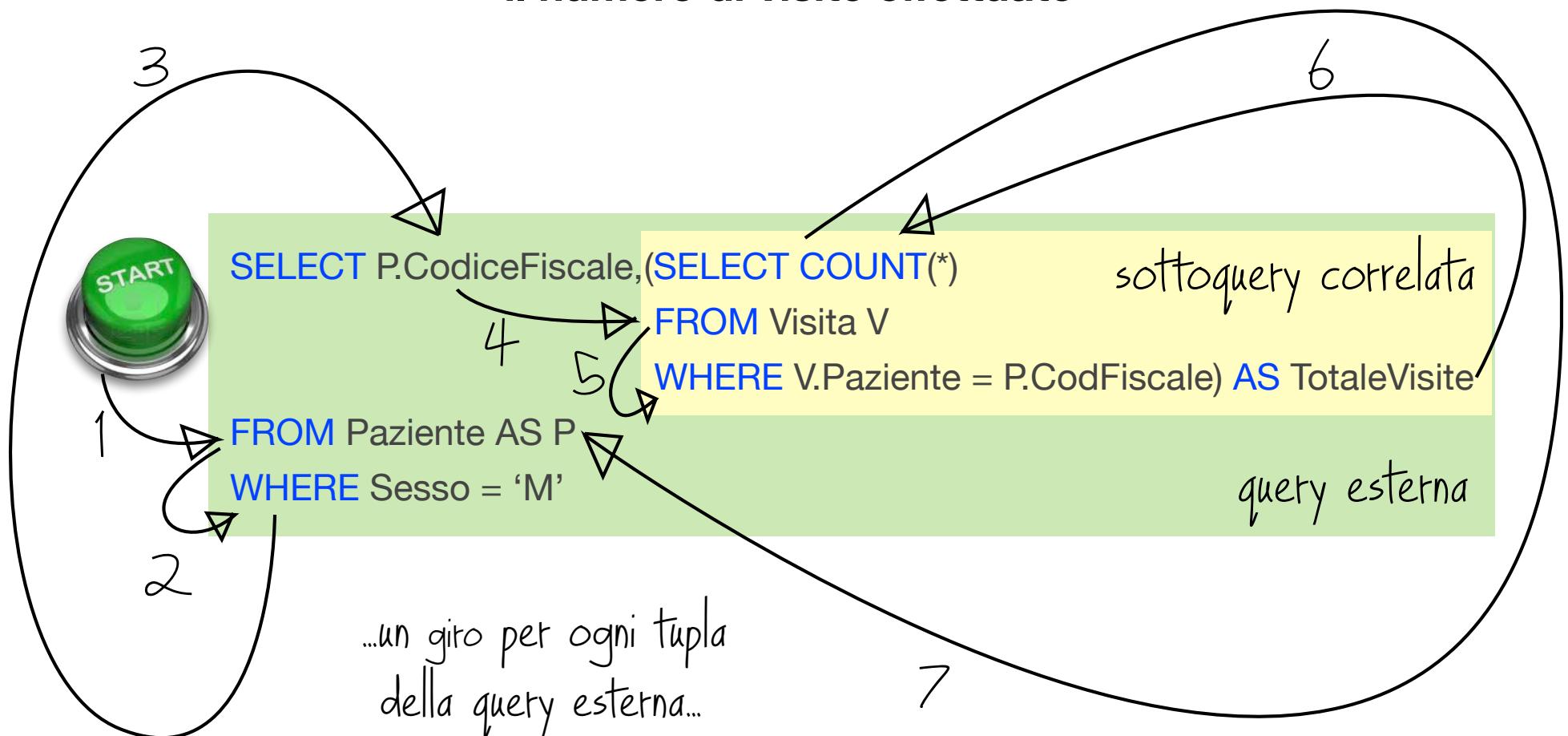
Esempio

Considerato ciascun paziente di sesso maschile, indicarne il nome e
il numero di visite effettuate

```
SELECT P.CodiceFiscale, (SELECT COUNT(*)          correlated subquery
                           FROM Visita V
                           WHERE V.Paziente = P.CodFiscale) AS TotaleVisite
FROM Paziente P
WHERE P.Sesso = 'M'                                query esterna
```

Correlated subquery nel SELECT: esecuzione

Considerato ciascun paziente di sesso maschile, indicarne il nome e
il numero di visite effettuate

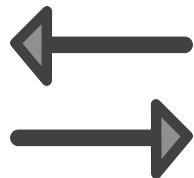


Costrutto EXISTS



Costrutto EXISTS

Permette di verificare che il result set di una correlated subquery contenga
almeno un record



la sua negazione controlla che il result set sia vuoto

EXISTS: esempio

Una visita di controllo è una visita in cui un medico visita un paziente **già visitato precedentemente** almeno una volta. Indicare medico, paziente e data delle visite di controllo del mese di Gennaio 2016



non interessa sapete quante volte
(una visita precedente o c'è o non c'è)

EXISTS: esempio (cont.)

Una visita di controllo è una visita in cui un medico visita un paziente **già visitato precedentemente** almeno una volta. Indicare medico, paziente e data delle visite di controllo del mese di Gennaio 2016

```
SELECT V1.Medico, V1.Paziente, V1.Data
FROM Visita V1
WHERE MONTH(V1.Data) = 1
      AND YEAR(V1.Data) = 2016
      AND EXISTS
(
    SELECT *
    FROM Visita V2
    WHERE V2.Medico = V1.Medico
          AND V2.Paziente = V1.Paziente
          AND V2.Data < V1.Data
);
```

i record del risultato hanno almeno un record nel result set della subquery

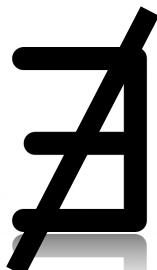
NOT EXISTS (soluzione slide 42)

Indicare la matricola dei medici che hanno visitato **per la prima volta** almeno un paziente nel mese di Ottobre 2013

```
SELECT DISTINCT V1.Medico  
FROM Visita V1  
WHERE YEAR(V1.Data) = 2013  
      AND MONTH(V1.Data) = 10  
      AND NOT EXISTS (
```

```
        SELECT *  
        FROM Visita V2  
        WHERE V2.Medico = V1.Medico  
              AND V2.Paziente = V1.Paziente  
              AND V2.Data < V1.Data
```

```
);
```



i record del risultato hanno il result set della subquery vuoto

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*

TUTTI

TUTTI

TUTTI

TUTTI

TUTTI

TUTTI

TUTTI

TUTTI

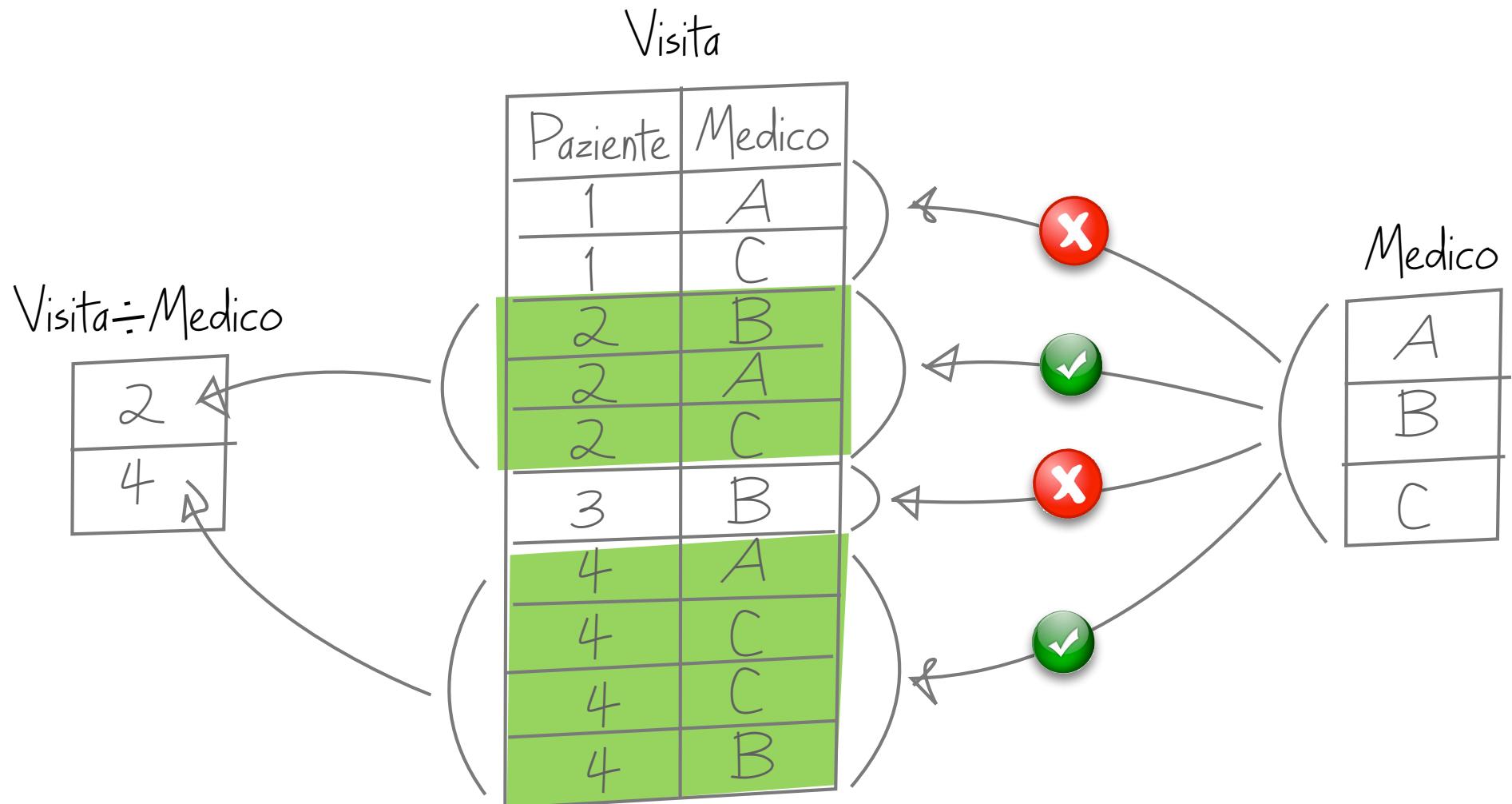
Divisione: esempio

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**

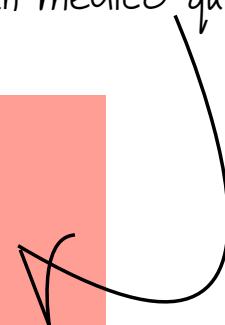
```
SELECT P.CodFiscale  
FROM Paziente P  
WHERE NOT EXISTS
```

```
(
```

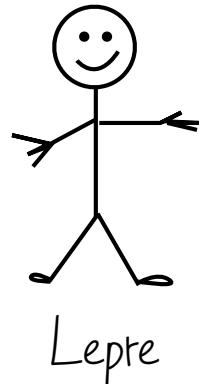
```
    SELECT *  
    FROM Medico M  
    WHERE NOT EXISTS  
(  
        SELECT *  
        FROM Visita V  
        WHERE V.Medico = M.Matricola  
            AND V.Paziente = P.CodFiscale
```

```
)  
);
```

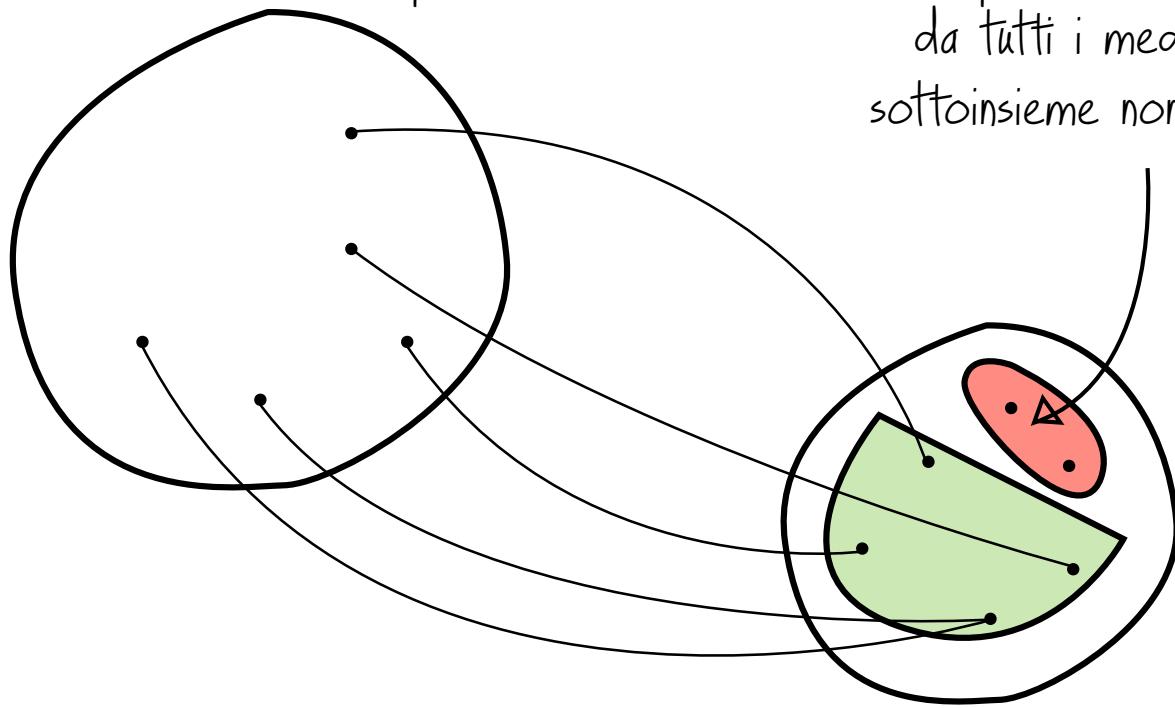
Prendi ogni paziente P per il quale
non trovi nemmeno un medico M
che non abbia visitato P
(cioè nemmeno un medico qui)



Cosa significa?



Visite di Lepre



se Lepre fosse stato visitato
da tutti i medici, questo
sottoinsieme non esisterebbe!

```
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
    )
);
```

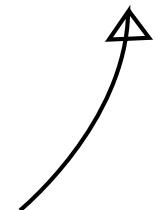
Divisione: con raggruppamento

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

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

medici diversi che hanno visitato il paziente

totale dei medici



Stored procedure



Stored procedure

invocate tramite chiamata e
possono ricevere e/o restituire valori

Sono **programmi dichiarativo-procedurali** memorizzati nel DBMS



supportate dalla versione 5.0 di MySQL

Accesso diretto ai dati

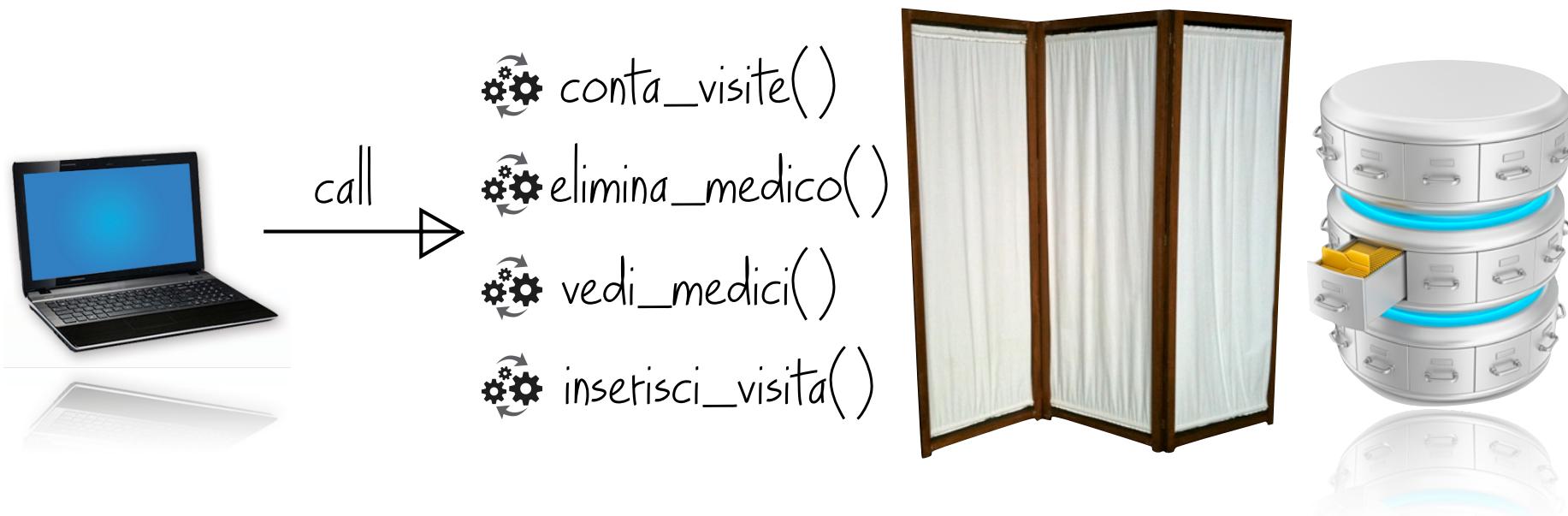


select from where...



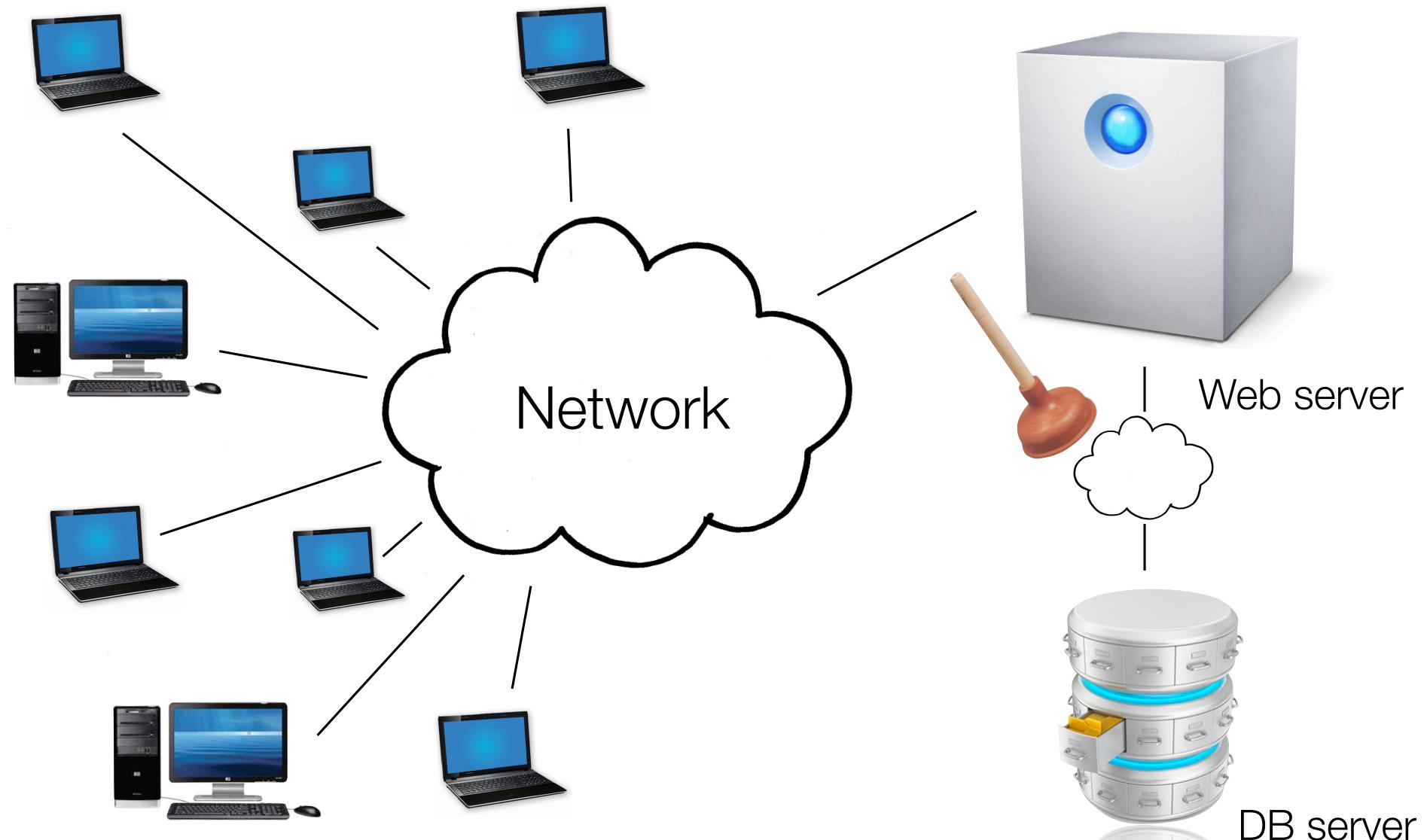
i dati devono essere accessibili
e si deve padroneggiare l'uso di SQL

Accesso mediante stored procedure



dati mascherati e codice mascherato
(sicurezza e protezione da attacchi)

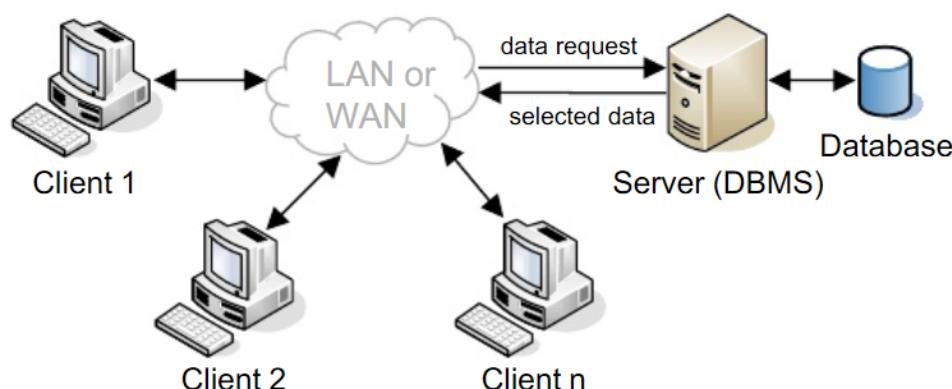
Architettura multi-tier



Prestazioni

il carico è spostato sul server (DBMS)

Mediante le stored procedure, le applicazioni **inviano solo una chiamata**, non il codice della query.



il traffico sulla rete è
drasticamente ridotto, e l'utente non deve
scrivere codice SQL complesso

Sicurezza

il codice può essere mascherato

i dati "grezzi" non sono visibili

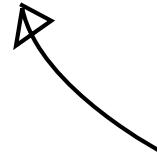
Le applicazioni possono essere autorizzate a eseguire stored procedure,
ma **avere accesso vietato alle tabelle**

restrizioni impostate con opportuni grant



Riuso del codice

devono avere i permessi per invocare le procedure



Una stored procedure è come un servizio che gli utenti delle applicazioni che usano il database possono **utilizzare senza scrivere codice**

Un semplice esempio

Scrivere una stored procedure che restituisca le specializzazioni mediche offerte dalla clinica

Un semplice esempio

Scrivere una stored procedure che stampi le specializzazioni mediche offerte dalla clinica

```
DROP PROCEDURE IF EXISTS mostra_specializzazioni;  
DELIMITER $$  
CREATE PROCEDURE mostra_specializzazioni()  
  BEGIN  
    SELECT DISTINCT Specializzazione  
    FROM Medico;  
  END $$  
DELIMITER ;
```

body

nome

Un select statement in una stored procedure equivale a "stampare" su standard output

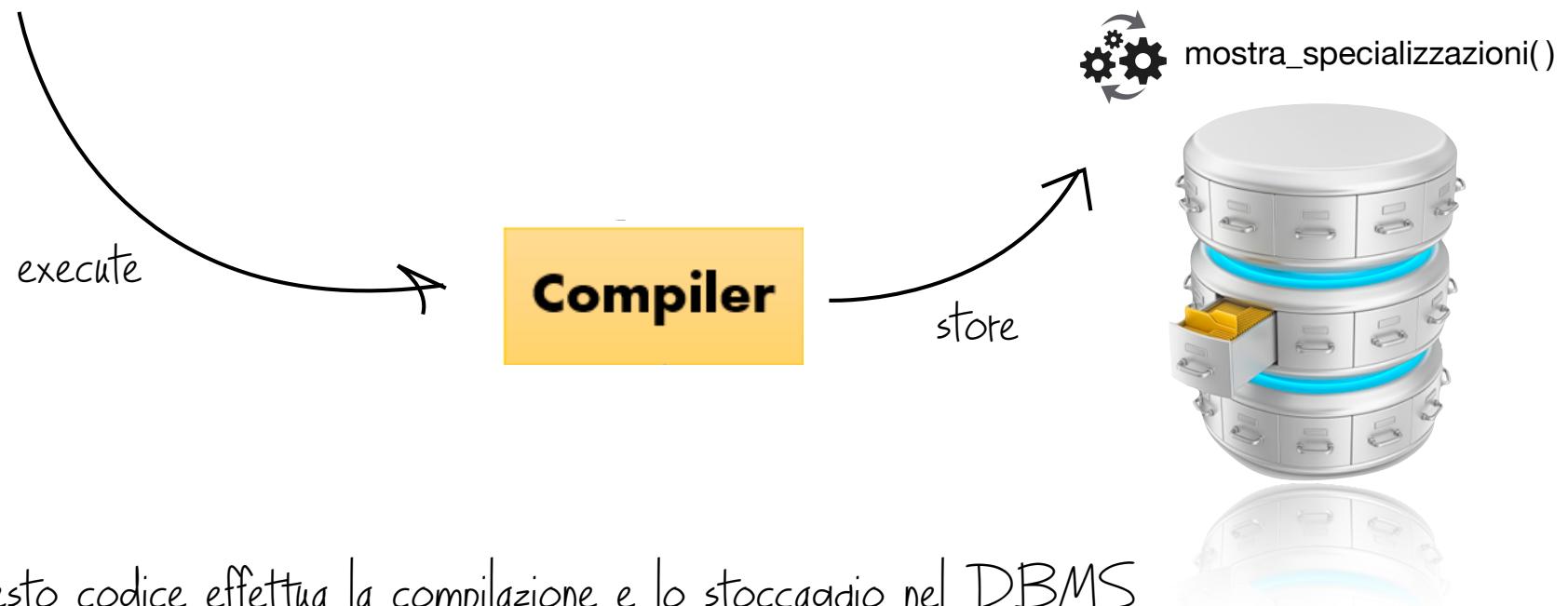
Esecuzione

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** Untitled @Clinica (localhost)
- Toolbar:** Run, Stop, Export Wizard, New, Load, Save, Save As, Grid View, Form View, Image, Text, Hex.
- Tab Bar:** Ne... (disabled), Untitled @ClinicaNe..., Untitled @ClinicaNe..., Untitled @ClinicaNe..., Untitled @Clinica (lo..., Untitled @Clinica (lo..., Untitled @Clinica (lo... (disabled).
- Query Editor Tab:** Query Builder (disabled), Query Editor (selected).
- Query Text:** (Numbered lines 1-9) DROP PROCEDURE IF EXISTS mostra_specializzazioni;
DELIMITER \$\$
CREATE PROCEDURE mostra_specializzazioni()
BEGIN
 SELECT DISTINCT Specializzazione
 FROM Medico;
END \$\$
- Status Bar:** 9:7
- Message Panel:** Message
- | Query | Message |
|--|---------|
| DROP PROCEDURE IF EXISTS mostra_specializzazioni | OK |
| CREATE PROCEDURE mostra_specializzazioni() | OK |

Storing...

```
DROP PROCEDURE IF EXISTS mostra_specializzazioni;  
DELIMITER $$  
CREATE PROCEDURE mostra_specializzazioni()  
BEGIN  
    SELECT DISTINCT Specializzazione  
    FROM Medico;  
END $$  
DELIMITER ;
```



Eseguire questo codice effettua la compilazione e lo stoccaggio nel DBMS

Chiamata

Scrivere una stored procedure che restituisca le specializzazioni mediche offerte dalla clinica

CALL mostra_specializzazioni();

La chiamata esegue la stored procedure e ottiene il risultato restituito dall'esecuzione del body

Esecuzione della chiamata

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** Untitled @db_root (localhost)
- Toolbar:** Run, Stop, Export Wizard, New, Load, Save, Save As, Grid View, Form View, Image, Text, Hex.
- Tab Bar:** Ne..., Untitled @ClinicaNe..., Untitled @Clinica (lo..., Untitled @Clinica (lo..., medico @db_root (lo..., Untitled @db_root (l...).
- Query Editor Tab:** Query Builder (disabled), Query Editor (selected).
- Query Editor Content:**

```
1 CALL mostra_specializzazioni();
```
- Result Window:** Shows the output of the query:

Specializzazione
Cardiologia
Gastroenterologia
Medicina generale
Nefrologia
Neurologia
Oculistica
Ortopedia
Otorinolaringoiatria
Psichiatria
- Bottom Status Bar:** CALL mostra_specializzazioni();, 0.01 sec elapsed, 9 records.

Variabili locali

Sono usate all'interno di una stored procedure,
per **memorizzare informazioni intermedie** di ausilio



devono essere dichiarate tutte insieme all'inizio del body!

Variabili locali: sintassi

int, double, char, varchar,
date, datetime...

(tipizzazione forte, non si possono
dichiarare senza specificare il tipo)

senza default value
il valore iniziale è NULL

DECLARE nome_variabile tipo(size) DEFAULT valore_default;

→ capacità della variabile
(se non settata, è il valore default del tipo di dato)

Assegnamento

È possibile **assegnare un valore a una variabile** in due modalità:
istruzione **SET** oppure **SELECT+INTO**

i due modi sono **alternativi**, dipende dai gusti



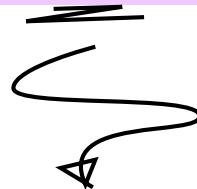
Assegnamento statico con SET

Supporre di essere nel body di una stored procedure e creare una variabile contenente il minimo numero di visite mensili da effettuare, impostato a 20.

```
DECLARE min_visite_mensili INT DEFAULT 0;
```

```
:
```

```
SET min_visite_mensili = 20;
```



la sintassi imporrebbe l'uso di `:=`, ma i due punti si
possono omessi se si usa l'istruzione `SET`

Assegnamento calcolato con SELECT+INTO

Supporre di essere nel body di una stored procedure e creare una variabile contenente il numero di visite effettuate nel mese in corso

```
DECLARE visite_mese_attuale INT DEFAULT 0;  
:  
SELECT COUNT(*) INTO visite_mese_attuale  
FROM Visita V  
WHERE MONTH(V.Data) = MONTH(CURRENT_DATE)  
AND YEAR(V.Data) = YEAR(CURRENT_DATE);
```

oppure

```
SELECT COUNT(*)  
FROM Visita V  
WHERE MONTH(V.Data) = MONTH(CURRENT_DATE)  
AND YEAR(V.Data) = YEAR(CURRENT_DATE)  
INTO visite_mese_attuale;
```

Assegnamento calcolato con SET

Creare una variabile contenente il numero di visite effettuate nel mese in corso

```
DECLARE visite_mese_attuale INT DEFAULT 0;
```

```
SET visite_mese_attuale =
(
    SELECT COUNT(*)
    FROM Visita V
    WHERE MONTH(V.Data) = MONTH(CURRENT_DATE)
        AND YEAR(V.Data) = YEAR(CURRENT_DATE)
);
```

Utilizzo errato delle variabili

```
DECLARE visite_mese_attuale VARCHAR(255);  
SELECT * INTO visite_mese_attuale  
FROM Visita  
WHERE MONTH(Data) = MONTH(CURRENT_DATE)  
AND YEAR(V.Data) = YEAR(CURRENT_DATE);
```

una variabile non può contenere un result set

Variabili user-defined

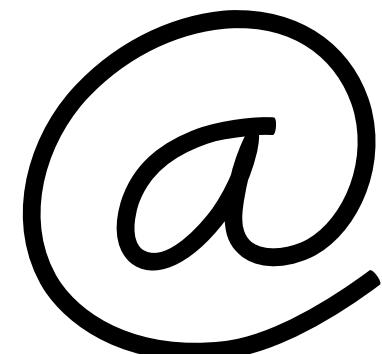
tipizzazione debole

(non si specifica il tipo della variabile, può contenere qualsiasi tipo di dato, e tipi di dato diversi in istanti diversi)

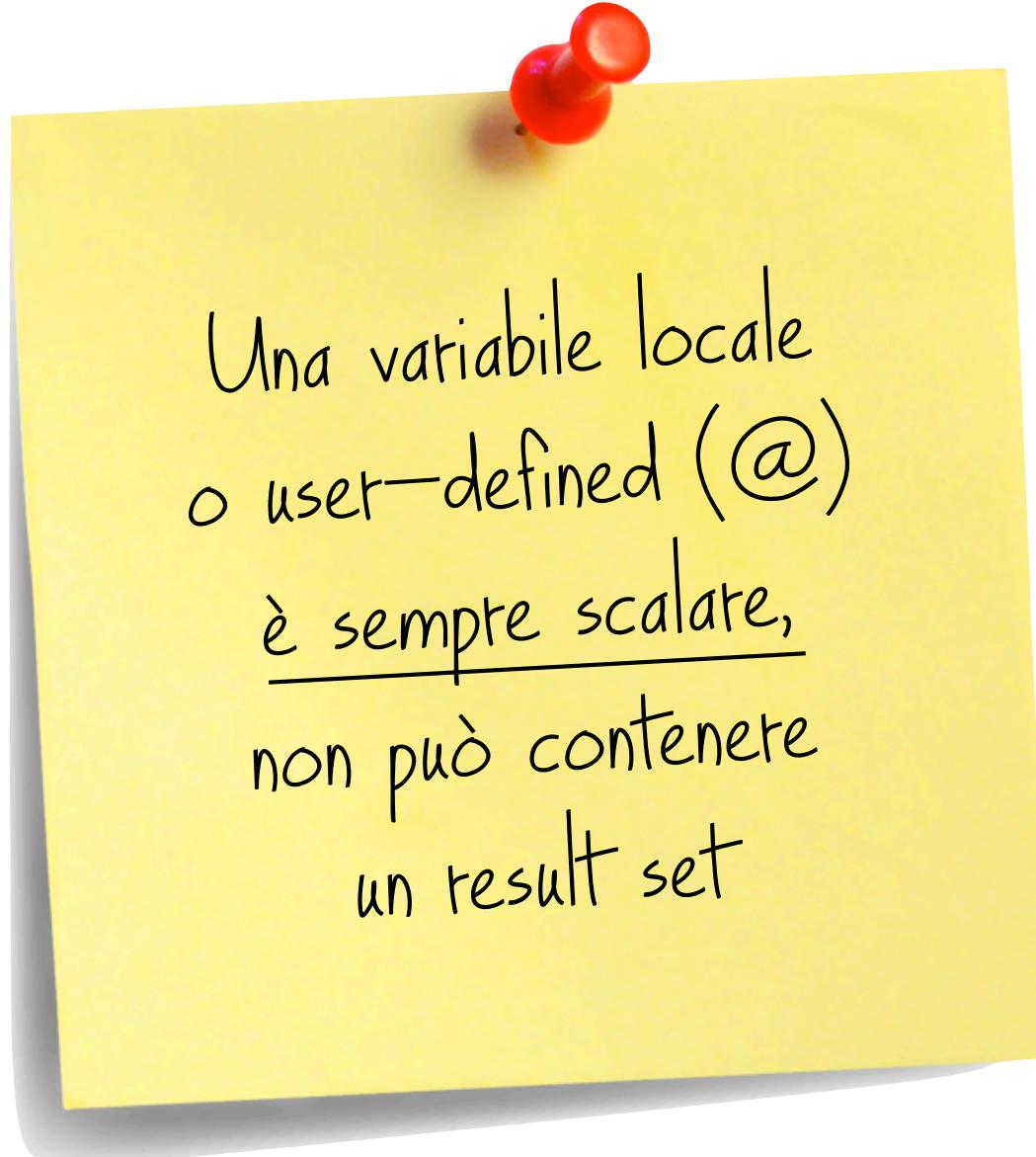


Sono inizializzate dall'utente **senza necessità di dichiarazione**, e il loro ciclo di vita equivale alla durata della connection a MySQL server

il contenuto è visibile ovunque, ma solo all'utente che le ha inizializzate, sono case insensitive e il loro identificatore deve iniziare con '@'



Attenzione



Per casa

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. 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.
4. 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.