

# Basi di dati

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

---

**Oracle MySQL**  
A.A. 2017-2018

1  
Ing. Francesco Pistolesi  
*Postdoctoral Researcher*  
Data Science and Engineering Lab  
Dipartimento di Ingegneria dell'Informazione  
[francesco.pistolesi@iet.unipi.it](mailto:francesco.pistolesi@iet.unipi.it)

# Informazioni utili

---

## Ing. Francesco Pistoletti

Postdoctoral Researcher

Data Science and Engineering Lab

Dipartimento di Ingegneria dell'Informazione, Università di Pisa



<http://www.iet.unipi.it/f.pistolesi>



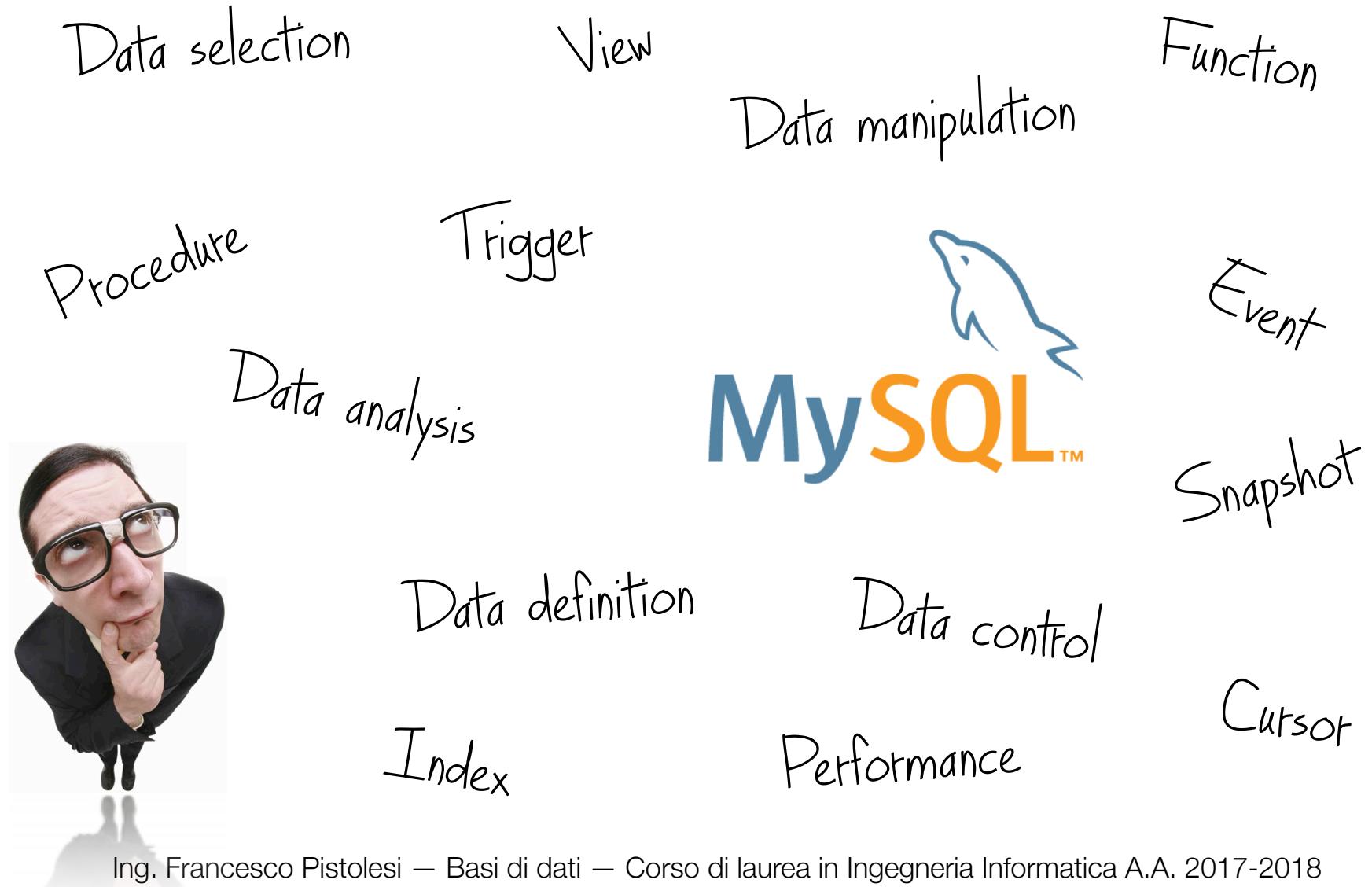
[francesco.pistolesi@iet.unipi.it](mailto:francesco.pistolesi@iet.unipi.it)



ricevimento studenti giovedì dalle 17.00 alle 19:00  
(inviare una email con un anticipo minimo di 48 ore)  
presso il Dipartimento di Ingegneria dell'Informazione, stanza 117  
largo L. Lazzarino 1 (già via Diotisalvi, 2), edificio A, secondo piano.  
Si riceve risposta non appena c'è disponibilità di appuntamenti.

# Cosa imparerò stavolta?

---



# Propedeuticità

---

Fondamenti di  
programmazione



Basi di dati

# Esame

---

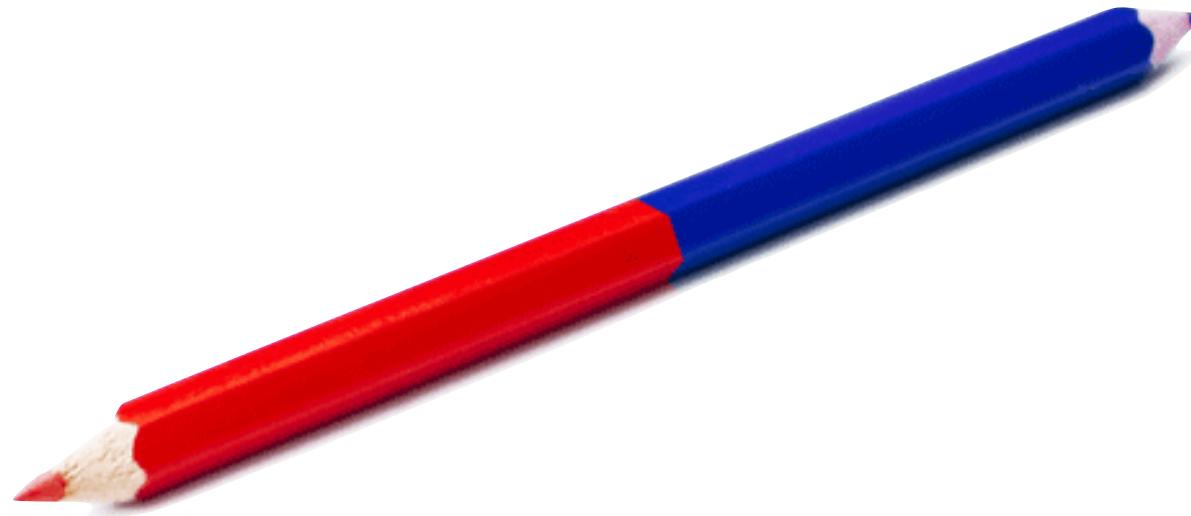
Implementazione al calcolatore, in ambiente MySQL, di:

- **Query complesse** (data analysis, reporting, business intelligence...)
- **Stored program** (stored procedure e function)
- **Trigger ed event**
- **Query optimization** (index, performance)



# Attenzione!

---





*Insegnare i dettagli  
significa portare confusione;  
stabilire i rapporti tra le cose  
significa dare conoscenza.*

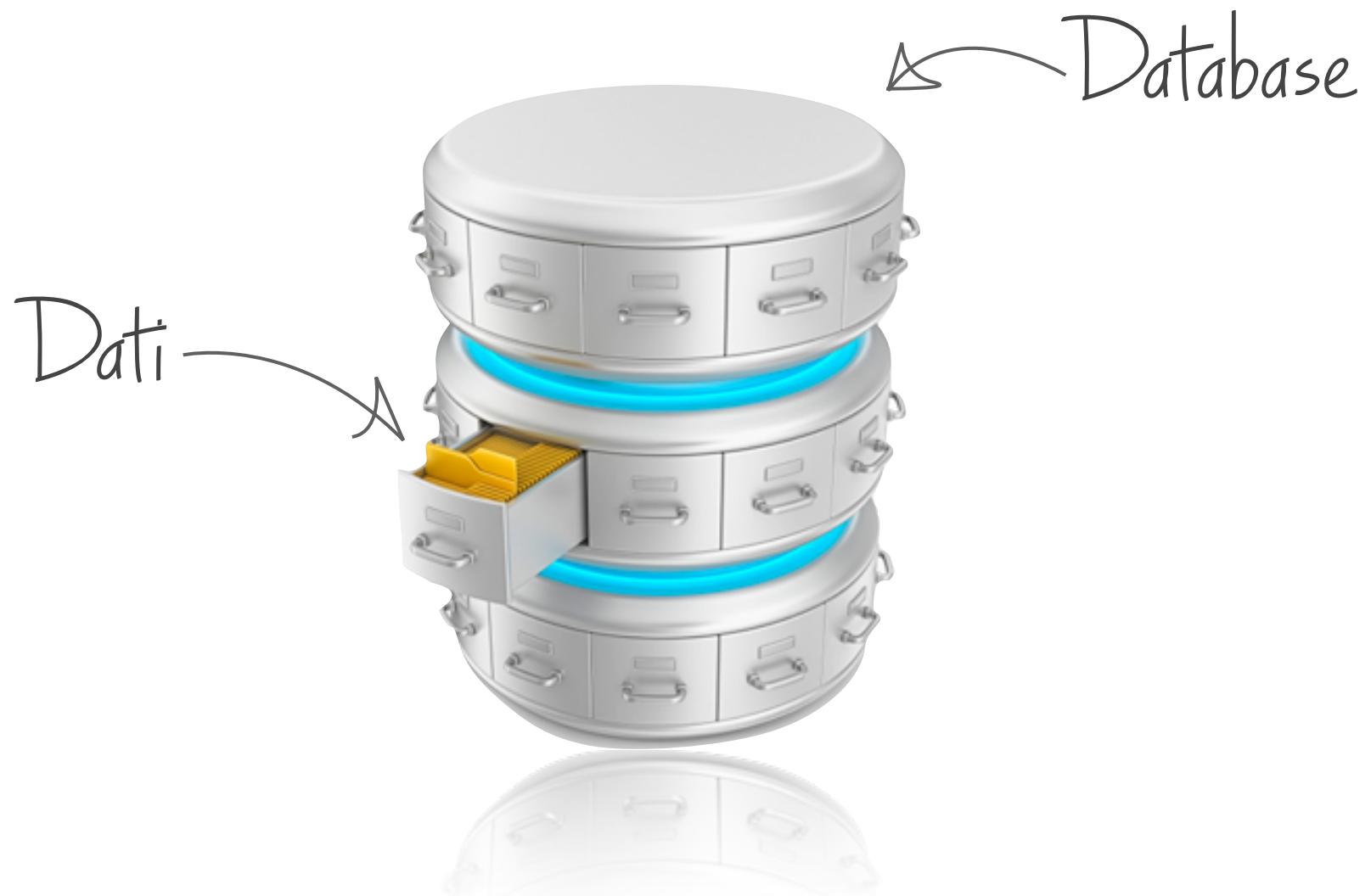
*Maria Montessori*

# Cos'è un database?



# Cos'è un database?

---



# Cos'è un database?

---

si memorizzano i dati con criterio...

Archivio di dati contenente informazioni ben strutturate,  
organizzate secondo **un modello logico**.

se relazionale, l'informazione è un record di una tabella

# Modello logico relazionale



## COMPITI

Codice	Descrizione
01	Fare il bucato
02	Comprare il pane
03	Far uscire il cane



## SCADENZE

Cod	Chi	Cosa	Quando
S1	A	01	oggi
S2	B	03	oggi
S3	A	02	domani
S4	C	01	domani
S5	C	03	oggi

# Modello logico relazionale

---

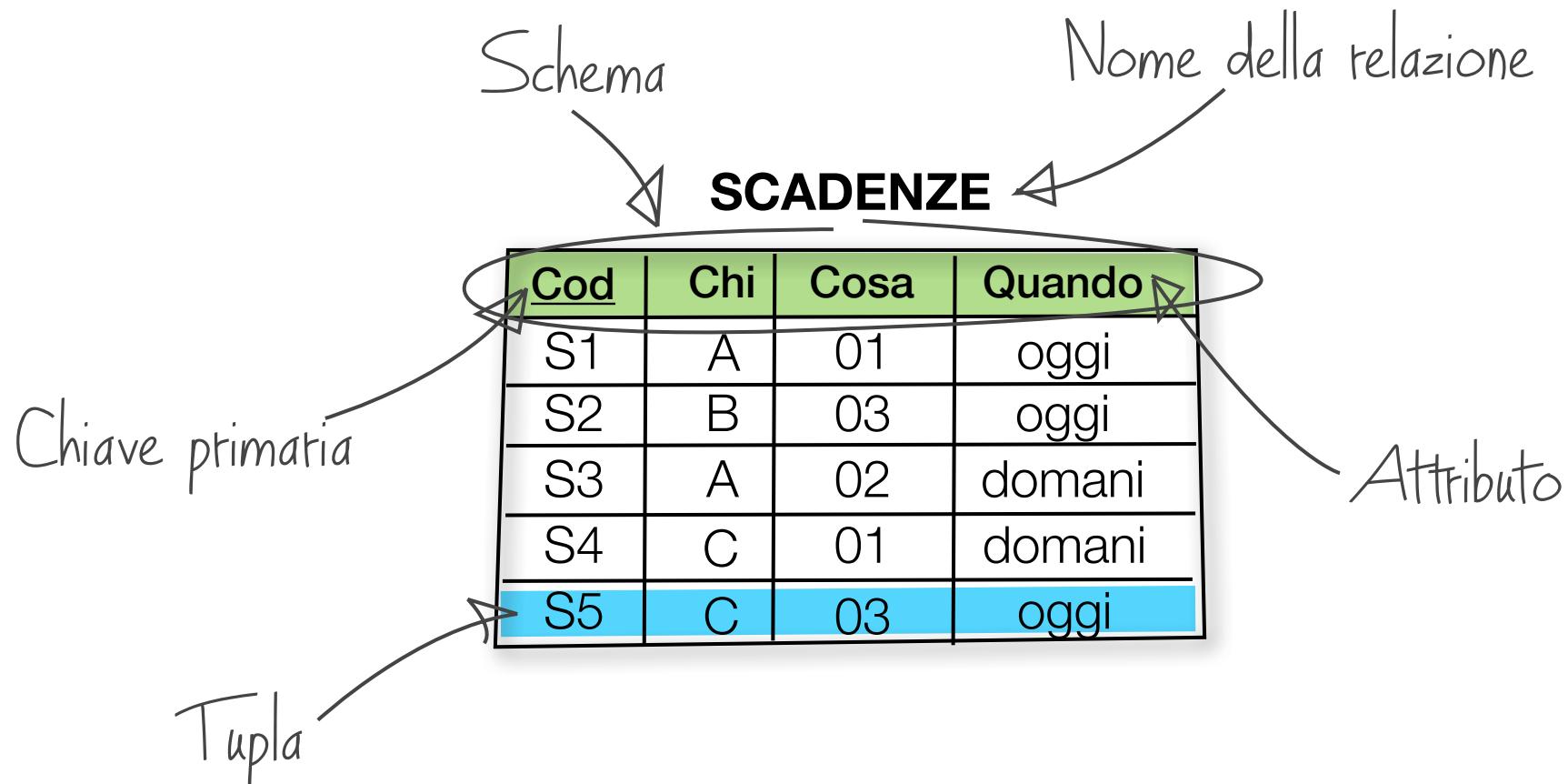
L'informazione è una **riga** di una tabella

o record

in parole, è "una tupla di una relazione"

# Tabelle: i paroloni

---



# Chiave

---

Insieme di attributi dello schema i cui valori non sono **mai replicati** in tutta la tabella

avente cardinalità minima

identificano univocamente un record



# Tabelle vs. post-it: 0-1

---



# Interrogazione

---

Per estrarre le informazioni d'interesse, si deve  
formulare un'**interrogazione**



per esempio in linguaggio SQL

# SQL [Structured Query Language]

---

dichiara le proprietà del risultato,  
non come lo si ottiene

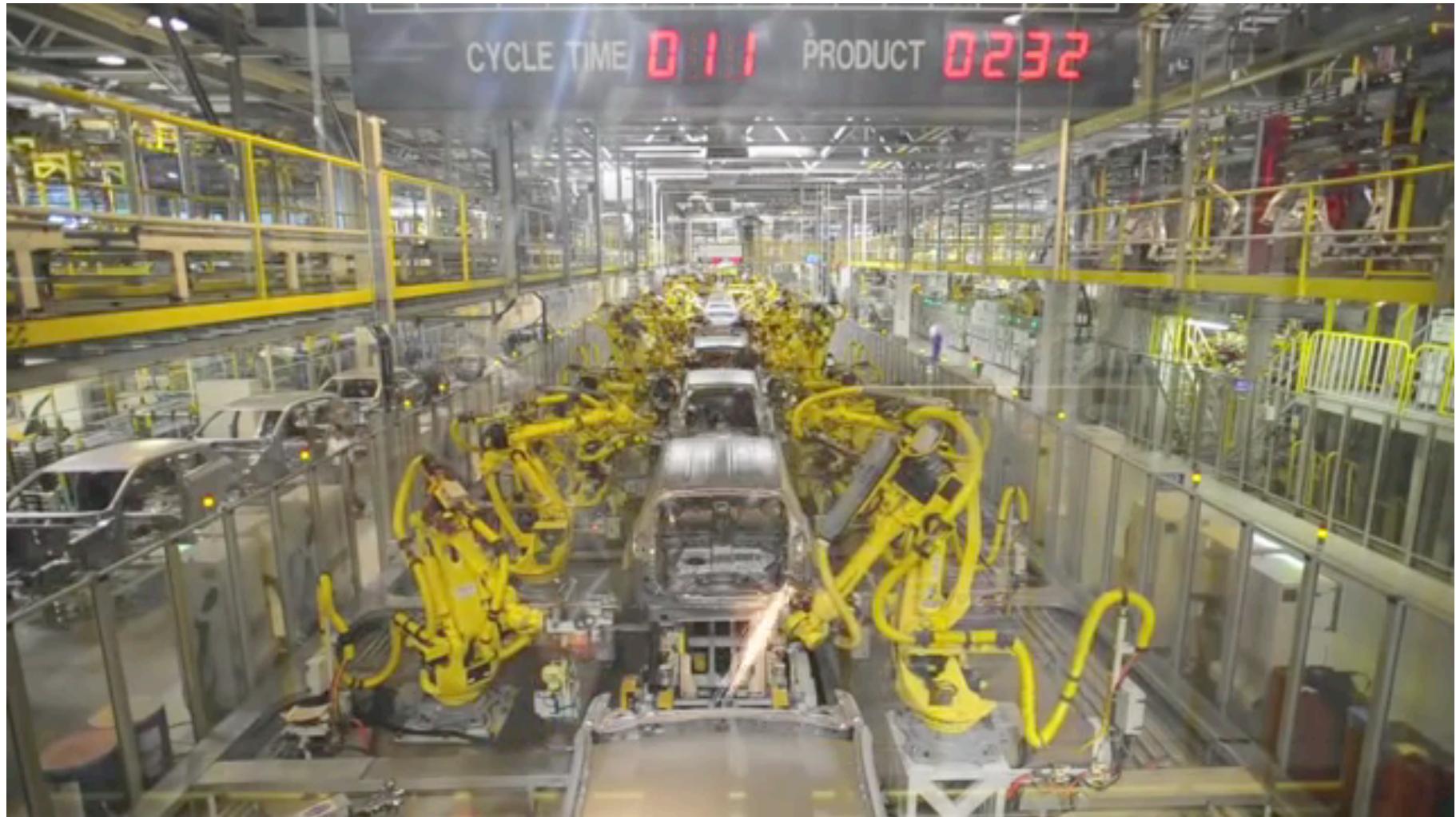
Linguaggio **dichiarativo**  
per interagire con database relazionali



interrogate, modificate, create, amministrate

# Linguaggio procedurale

---



# Linguaggio dichiarativo

---



Figo!

# Linguaggio dichiarativo vs. linguaggio procedurale

---

Voglio conoscere i compiti  
assegnati oggi ad Antonella

appuccio dichiarativo



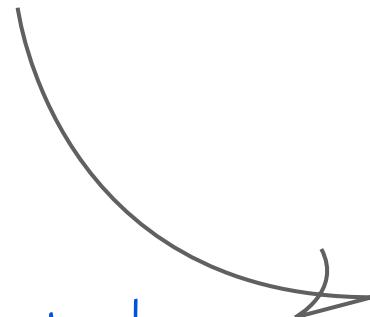
Compiti che hanno la data  
uguale a oggi e la persona è  
Antonella

# Linguaggio dichiarativo vs. linguaggio procedurale

---

Voglio conoscere i compiti  
assegnati oggi ad Antonella

appuccio procedurale



## Compito

cod	chi	cosa	quando	next
-----	-----	------	--------	------

Carica i compiti da file come una lista di oggetti  
struct Compito; considerare il puntatore di  
testa compiti, finché compiti->next è  
diverso da 0 allora considera un compito,  
confronta il campo quando con la data odierna,  
se sono diversi scorri il puntatore e carica un'altro  
compito, se sono uguali allora confronta il campo  
chi con il parametro attuale 'Antonella', se  
anche questi sono uguali allora crea una struttura  
dati (ad esempio una lista di oggetti Compito),  
inserisci nella lista il compito trovato, slaccia e  
riallaccia i puntatori come Dio comanda, e ripeti  
tutto finché ci sono elementi nella lista di compiti.  
Infine, restituisci il puntatore di testa della lista  
risultato al chiamante.

# Linguaggio dichiarativo vs. linguaggio procedurale

```
struct Compito{  
    string cod;  
    string chi;  
    string cosa;  
    string data; //formato 'gg-mm-aa'  
    Compito* next;  
};
```

Struttura dati

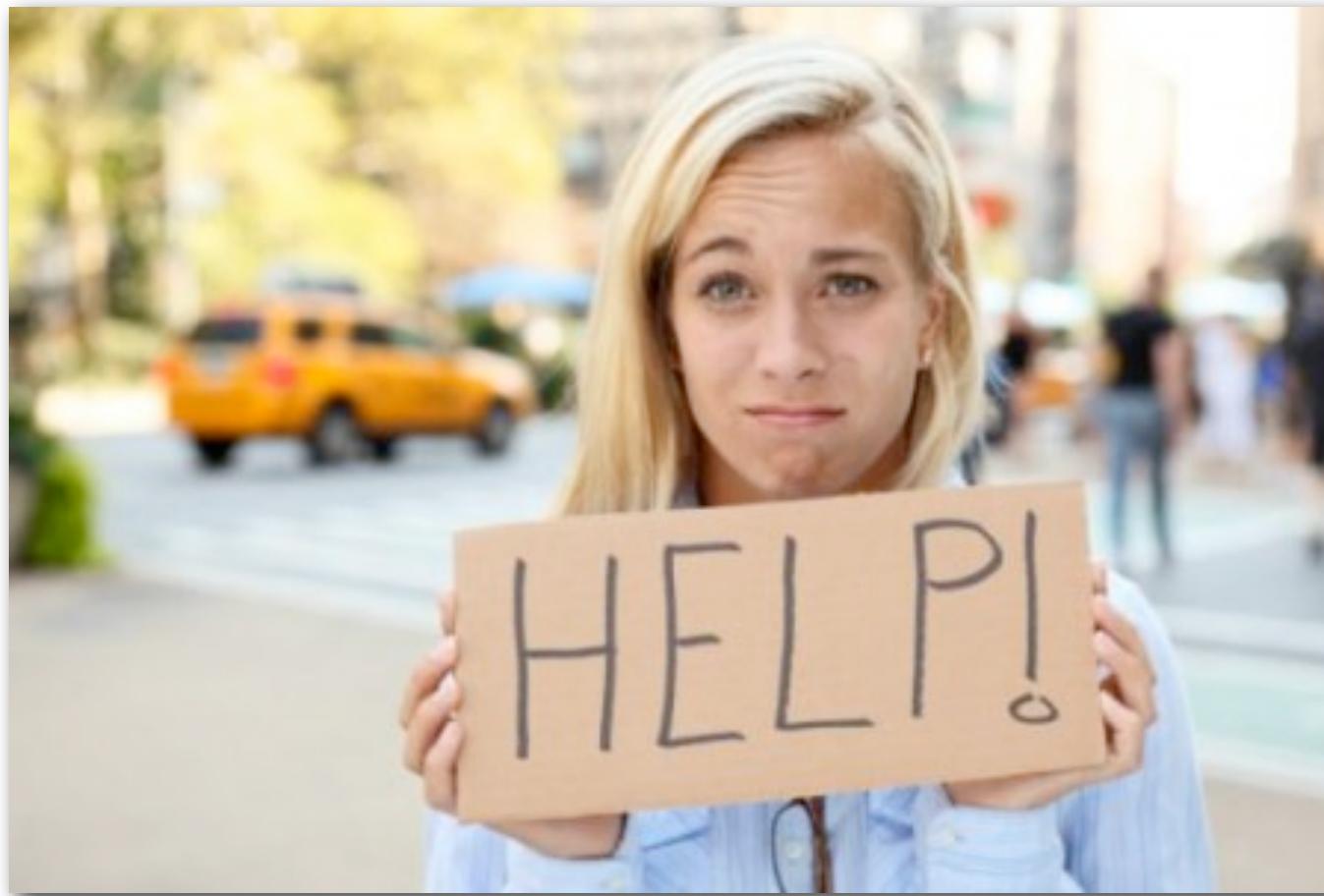
```
Compito* trovaCompitiPersona(Compito* compiti,  
                             string persona, string data) {  
    Compito* compitiPersona = 0;  
    while (compieti->next != 0) {  
        if (compieti->chi == chi && compiti->data == data) {  
            Compito* compito = new Compito;  
            compito->cod = compiti->cod;  
            compito->chi = compiti->chi;  
            compito->cosa = compiti->cosa;  
            compito->data = compiti->data;  
            compito->next = compitiPersona;  
            compitiPersona = compito;  
        }  
        compiti = compiti->next;  
    }  
    return compitiPersona;  
}
```

Funzione

```
int main(){  
    Compito* risultato = trovaCompitiPersona("Antonella", "12-03-15");  
}
```

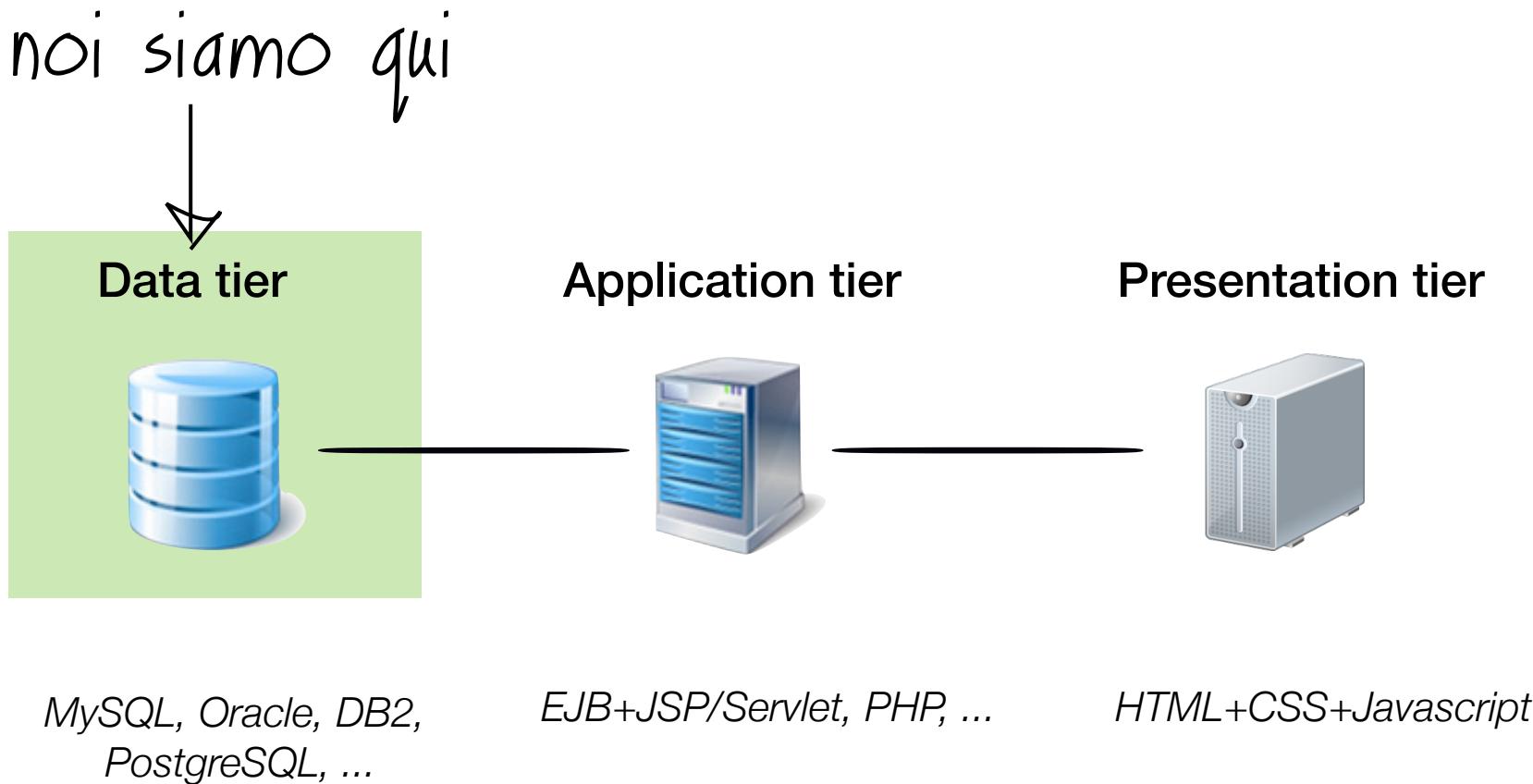
# SQL salvaci...

---



# Architettura multi-tier

---



# Data tier

---

database



DBMS

Contiene sia l'**insieme dei dati** che l'**applicazione** che ne gestisce  
l'accesso e le politiche di gestione

Data tier



# Cos'è MySQL?

---

MySQL è un **DBMS relazionale** composto da un **server** e un client

lavora con tabelle

sono su host diversi

processo che svolge i comandi inviati dal client

**Data tier**



# Un semplice database: una sola tabella

---

**PERSONA**



<u>CodFiscale</u>	Cognome	Nome	Età
BVEMDL	Bove	Maddalena	45
GTTTMO	Gatti	Tommaso	37
LPRDOD	Lepre	Edoardo	39
NTRLRL	Nutrie	Lorella	65
LPRNRN	Lepre	Norina	83
RTTBRT	Ratto	Umberto	54

# La nostra prima query

---

attributi d'interesse ←  
il "sacco" da cui si pesca  
Indicare cognome e nome delle persone  
di età inferiore a 40 anni  
condizione

# La nostra prima query

Indicare cognome e nome delle persone di età inferiore a 40 anni

**PERSONA**

<u>CodFiscale</u>	Cognome	Nome	Età
D'AVEN	Rave	Maria Grazia	45
GTTTMO	Gatti	Tommaso	37
LPRDOD	Lepre	Edoardo	39
MOLPE	Maria	Corilla	65
LEPORN	Lepro	Morina	83
PTEPPI	Fatto	Umberto	54

# La nostra prima query

---

Indicare **cognome e nome** delle persone di età inferiore a 40 anni



CodFiscale	Cognome	Nome	Età
GTTTMO	Gatti	Tommaso	37
LPRDOD	Lepre	Edoardo	39

# La nostra prima query

---

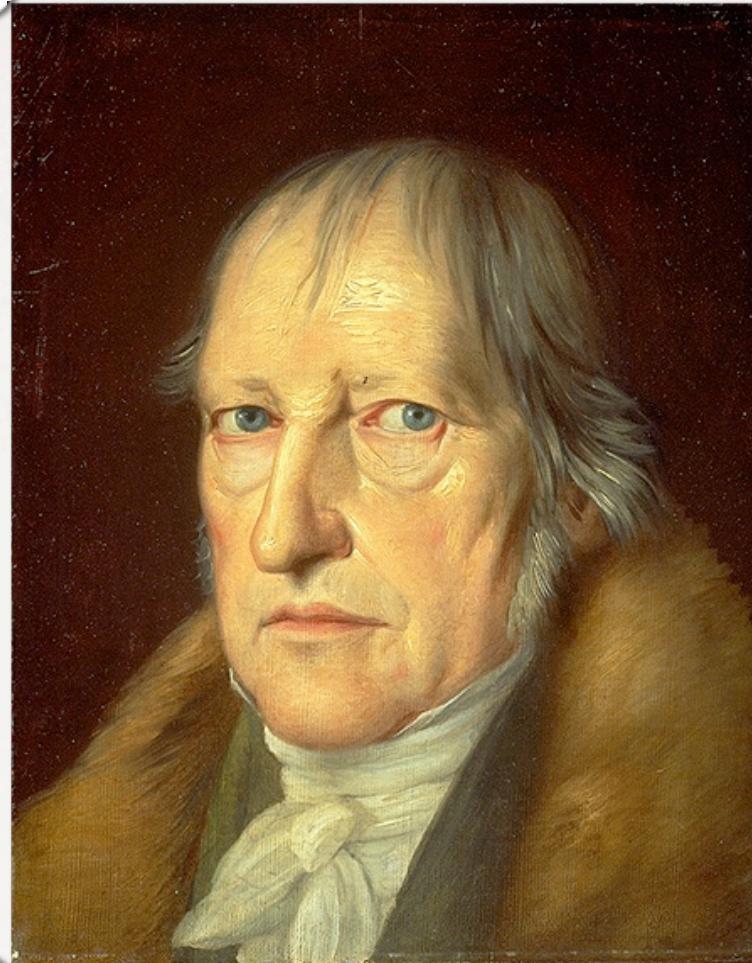
Indicare cognome e nome delle persone di età inferiore a 40 anni

Il risultato contiene gli attributi  
e le righe desiderate

Cognome	Nome
Gatti	Tommaso
Lepre	Edoardo

# SQL

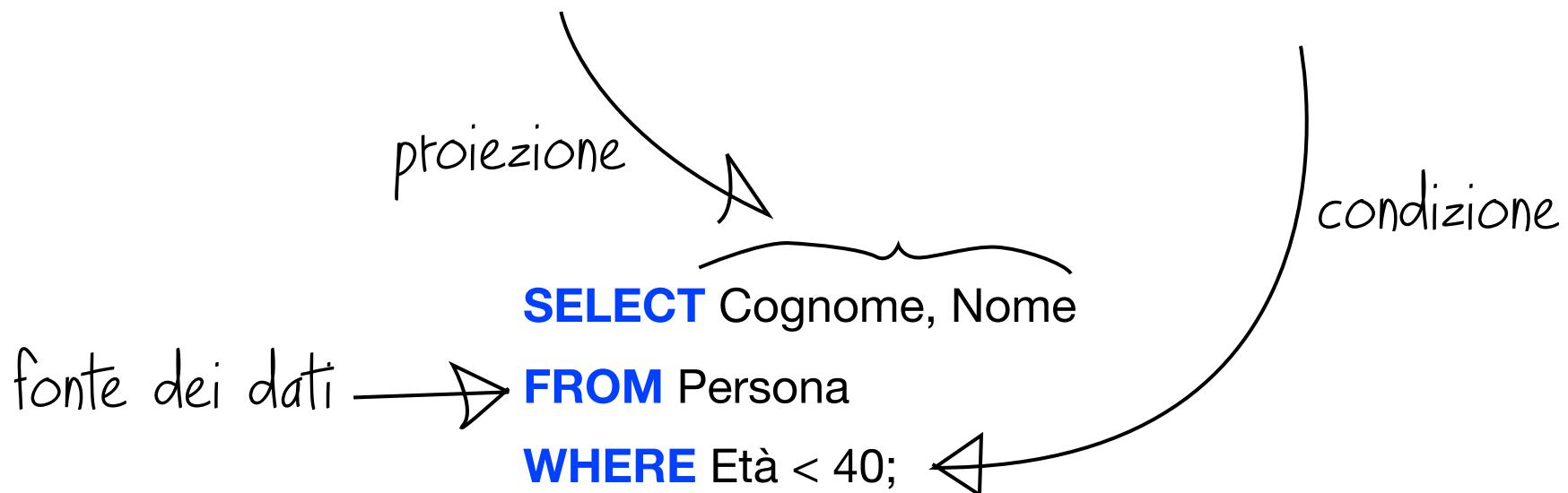
---



# La nostra prima query in SQL

---

Indicare cognome e nome delle persone di età inferiore a 40 anni



# Lo studente zelante scrive...

---

Indicare cognome e nome delle persone di età inferiore a 40 anni

```
SELECT Nome, Cognome  
FROM Persona  
WHERE Età < 40;
```



ordine della proiezione incoerente!

# Sintassi di una query MySQL

---

separati da virgola

**SELECT** Lista\_Attributi  
**FROM** Insieme\_Tabelle  
**WHERE** Lista\_Condizioni;

generato mediante "mischugli" che vedremo prossimamente...

si termina sempre con ;

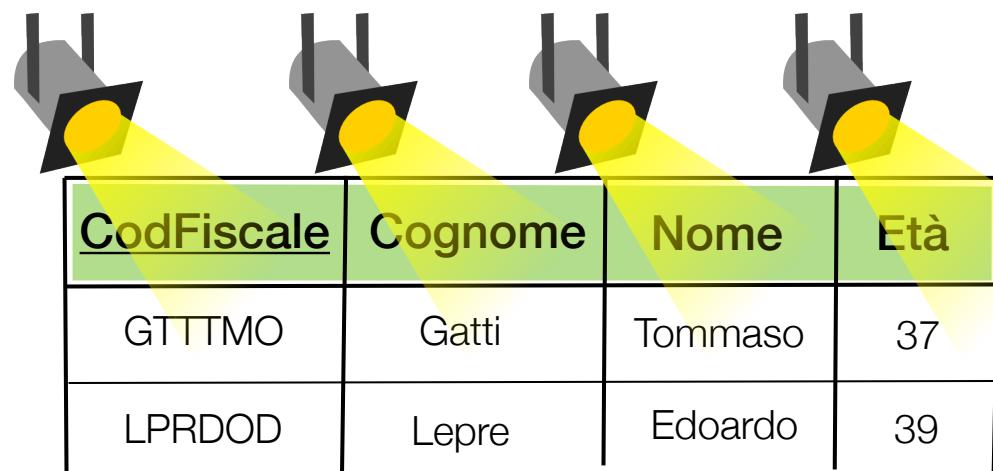
condizioni (anche molto articolate)

# Tutti gli attributi

---

Indicare **tutte le informazioni** delle persone di età inferiore a 40 anni

**SELECT \***  
**FROM** Persona  
**WHERE** Età < 40;



CodFiscale	Cognome	Nome	Età
GTTTMO	Gatti	Tommaso	37
LPRDOD	Lepre	Edoardo	39

Non interessa un particolare sottoinsieme di attributi

# Lista tabelle

---

un database ne ha più di una

Le tabelle possono essere **combinate** per formare la lista tabelle

...ma per ora sotvoliamo...

# Lista condizioni

---

Condizioni più espressive si ottengono mediante **operatori logici**



AND  
NOT  
OR  
NOT

# Lista condizioni: esempio

Indicare il codice fiscale delle persone  
di età inferiore a 40 anni il cui cognome è Lepre

```
SELECT CodFiscale  
FROM Persona  
WHERE Età < 40  
      AND Cognome = 'Lepre';
```

contemporaneamente!

CodFiscale	Cognome	Nome	Età
RVEMDI_	Rova	Maddalena	45
GTITMO	Catti	Tancredo	37
LPRDOD	Lepre	Edoardo	39
NTRBLI_	Mario	Luigi	65
LEPEN	Lepre	No ina	83
PTEPAT	Della	Ursula	54

# Lista condizioni: altro esempio

Indicare codice fiscale ed età delle persone il cui cognome è Nutrie o il cui nome è Maddalena

```
SELECT CodFiscale, Età  
FROM Persona  
WHERE Cognome = 'Nutrie'  
      OR Nome = 'Maddalena';
```

oppure!

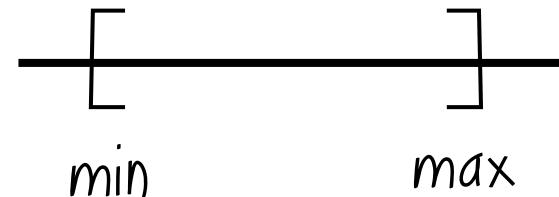
CodFiscale	Cognome	Nome	Età
BVEMDL	Bove	Maddalena	45
GTETMO	Catti	Tancredo	37
IPPPON	Lanza	Edoardo	30
NTRLRL	Nutrie	Lorella	65
IEEEN	Lepre	Norma	83
PTEON	Fazio	Umberto	54

# Intervalli di valori

---

→ numerico, data o timestamp

Per verificare se il valore di un attributo appartiene a un certo intervallo si usa la direttiva **BETWEEN** oppure una congiunzione logica di  **$\geq$**  e  **$\leq$**



# Intervalli di valori: esempio

Indicare cognome e nome delle persone di età compresa fra 45 e 60 anni

```
SELECT Cognome, Nome  
FROM Persona  
WHERE Età BETWEEN 45 AND 60;
```

oppure

```
SELECT Cognome, Nome  
FROM Persona  
WHERE Età >= 45  
      AND Età <= 60;
```

CodFiscale	Cognome	Nome	Età
BVEMDL	Bove	Maddalena	45
GTITMO	Gatti	Tommaso	37
LPRDOD	Lepre	Edoardo	39
NTEBLO	Milne	Sorcella	65
LPZLBN	Lepre	Ivana	83
RTTBRT	Ratto	Umberto	54

# Duplicati

---



# Duplicati: il concetto

---

I valori degli attributi non chiave **possono ripetersi** in record diversi

# Problemi?!

---

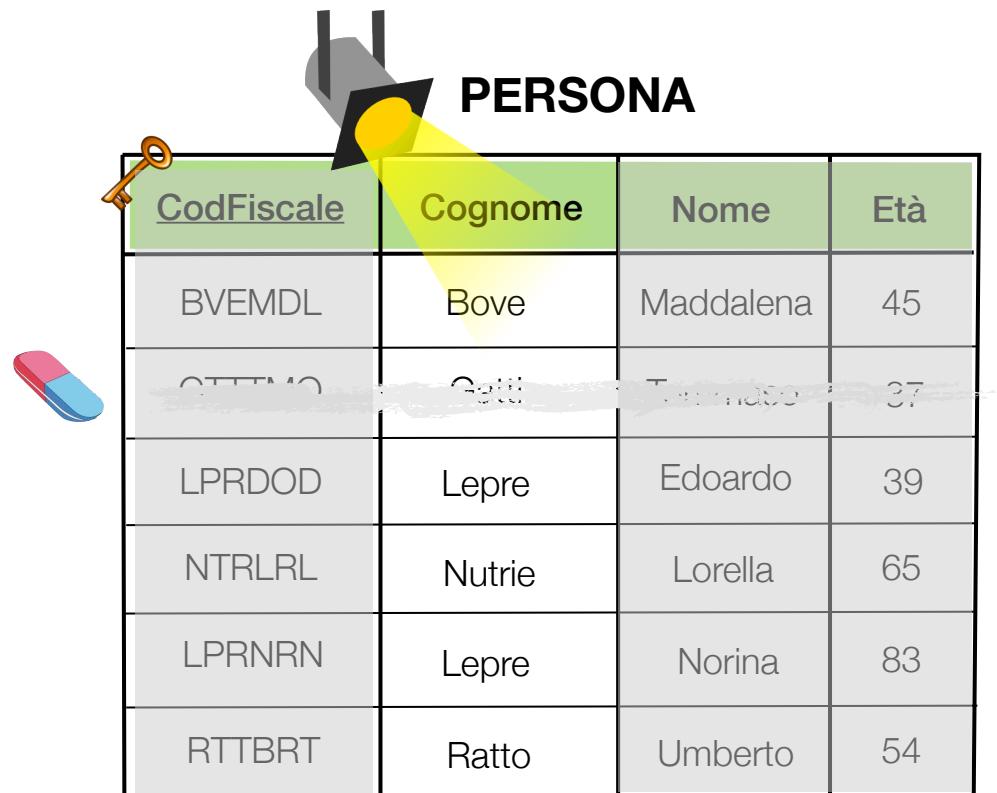


# Esempio

---

Indicare i cognomi delle persone di età almeno pari a 38 anni

```
SELECT Cognome  
FROM Persona  
WHERE Età >= 38;
```



CodFiscale	Cognome	Nome	Età
BVEMDL	Bove	Maddalena	45
OTTEMIO	Getti	Tanimaco	37
LPRDOD	Lepre	Edoardo	39
NTRLRL	Nutrie	Lorella	65
LPRNRN	Lepre	Norina	83
RTTBRT	Ratto	Umberto	54

# Esempio

---

Indicare i cognomi delle persone di età almeno pari a 38 anni



```
SELECT Cognome  
FROM Persona  
WHERE Età >= 38;
```

Cognome
Bove
Lepre
Nutrie
Lepre
Ratto



# Eliminazione duplicati in MySQL

---

Indicare i cognomi delle persone di età almeno pari a 38 anni

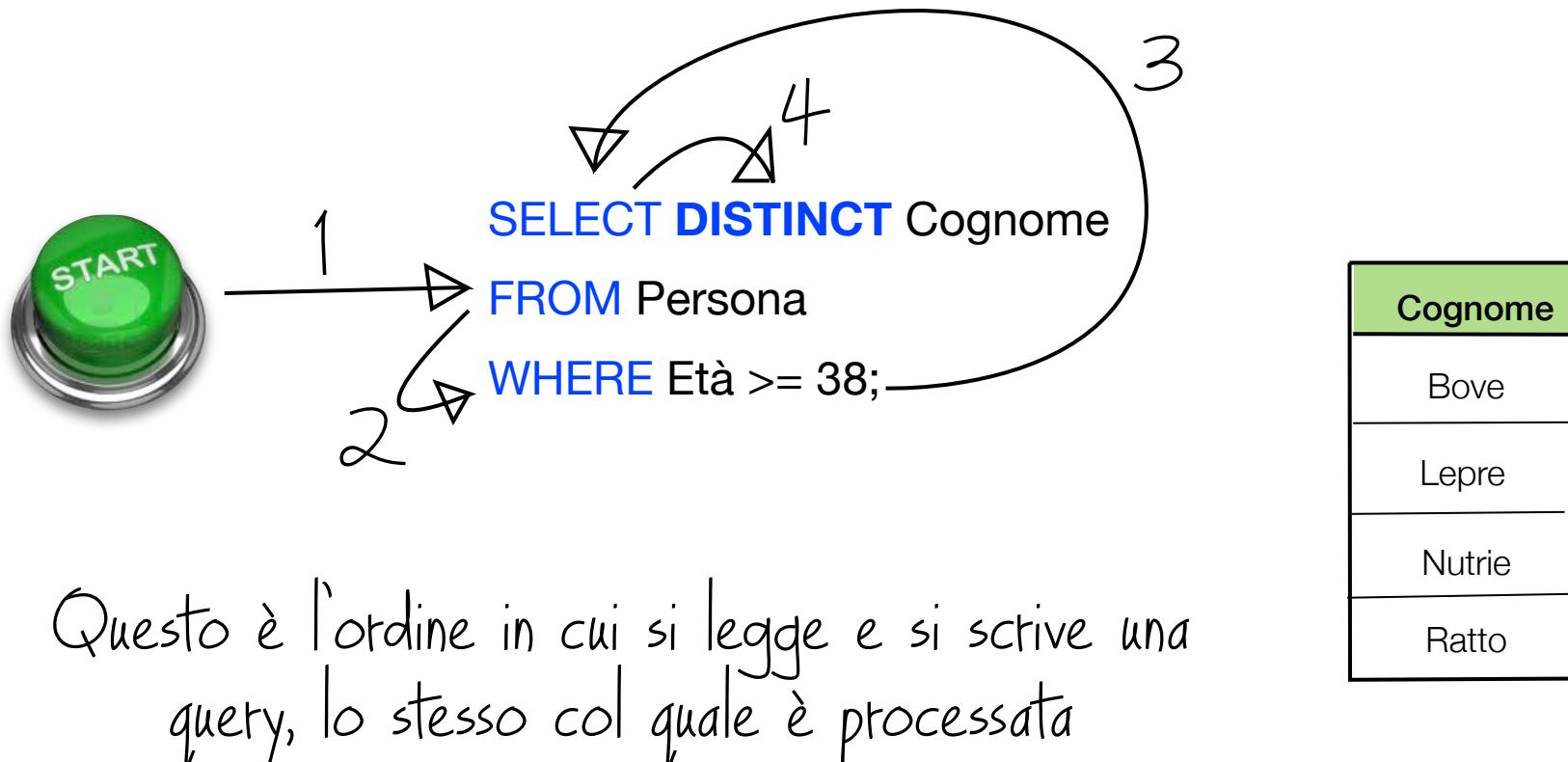
```
SELECT DISTINCT Cognome  
FROM Persona  
WHERE Età >= 38;
```

Cognome
Bove
Lepre
Nutrie
Ratto

Distinct elimina i duplicati sugli attributi proiettati

# Ordine di processazione in MySQL

Indicare i cognomi delle persone di età almeno pari a 38 anni



# DISTINCT con più attributi

---

Con proiezione multiattributo, **DISTINCT** agisce su **tutti gli attributi proiettati**



Un record proiettato è un duplicato se esiste un altro record nel risultato con uguali valori sugli stessi attributi

# Valori NULL

---



# Valori NULL

---

È indispensabile che ci siano?

Cosa significa?

Gli attributi non chiave possono assumere **NULL** come valore

Quanto vale **NULL**?



# Valori NULL: significato

---

Valore **mancante** (sconosciuto, perduto, indeterminato...)

Non si conosce l'età  
di Tommaso Gatti,  
né quella di Lotella Nutrie

**PERSONA**

<u>CodFiscale</u>	Cognome	Nome	Età
BVEMDL	Bove	Maddalena	45
GTTTMO	Gatti	Tommaso	<b>NULL</b>
LPRDOD	Lepre	Edoardo	39
NTRLRL	Nutrie	Lorella	<b>NULL</b>
LPRNRN	Lepre	Norina	83
RTTBRT	Ratto	Umberto	54

# Valori NULL: significato

---

Valore mancante ma **significato logicamente definito**

Titolo di studio non  
ancora conseguito per  
Verdi, Rossi e Bianchi

**STUDENTE**

<u>Matricola</u>	Cognome	Datalscrizione	DataLaurea
3893	Neri	1950-09-20	1957-10-03
6288	Verdi	2005-09-15	<b>NULL</b>
8097	Gialli	1947-08-02	1953-07-15
1282	Verdi	2001-09-10	2010-04-30
4823	Rossi	2009-09-15	<b>NULL</b>
8502	Bianchi	2010-09-15	<b>NULL</b>

# Esempio

---

Indicare matricola e data di laurea degli studenti immatricolati  
fra l'anno 2001 e l'anno 2005

```
SELECT Matricola, DataLaurea  
FROM Studente  
WHERE Datascrizione BETWEEN '2001-01-01' AND '2005-12-31';
```

Ha senso?

 Matricola	DataLaurea
1282	2010-04-30
6288	NULL

## Esempio (cont)

---

Indicare matricola e data di laurea degli studenti **laureati** immatricolati  
fra l'anno 2001 e l'anno 2005



Richiesta più chiara

```
SELECT Matricola, DataLaurea  
FROM Studente  
WHERE DataLaurea IS NOT NULL  
      AND Datascrizione BETWEEN '2001-01-01' AND '2005-12-31';
```

Soluzione più sensata

 Matricola	DataLaurea
1282	2010-04-30

# Altro esempio

---

Indicare la matricola degli studenti **non ancora laureati**

```
SELECT Matricola  
FROM Studente  
WHERE DataLaurea IS NULL;
```

Significa che la data di laurea  
è **NULL**

Matricola
6288
4823
8502

# Ricapitolando...

---

La condizione **IS NOT NULL** su un attributo elimina tutti i record che assumono valore NULL su tale attributo.

La condizione **IS NULL** su un attributo elimina tutti i record che assumono valore diverso da NULL su tale attributo.

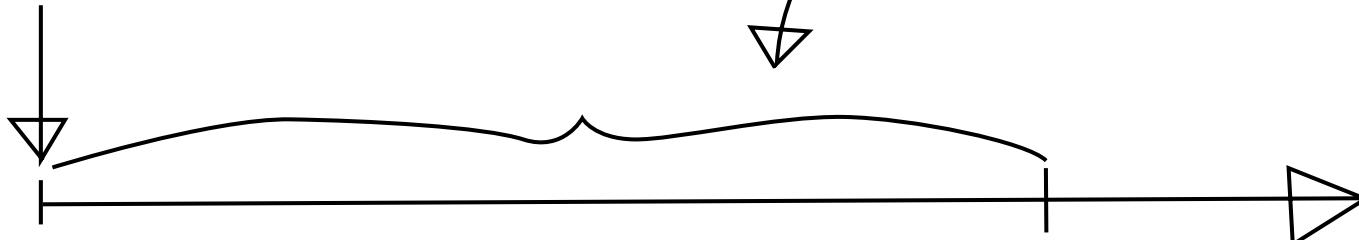
# Gestione delle date

---



# Cos'è una data in MySQL

---



**Big Bang**

**2018-03-08**

# Tipi di dato data :-)

---

Tipo	Formato
DATE	YYYY-MM-DD
TIMESTAMP	YYYY-MM-DD HH:MM:SS

# Formattazione delle date in MySQL

---

In MySQL, è possibile “**manipolare**” una data restituita da una query mediante la funzione **DATE\_FORMAT**

utile per usare alcune funzioni di utilità, o per effettuare parsing

# Alcuni formati utili

---

Formato	Descrizione
%Y	anno (4 cifre)
%y	anno (2 cifre)
%M	nome del mese
%m	mese (2 cifre)
%d	giorno del mese (00-31)
%W	nome del giorno
%w	giorno della settimana {0,...,6}
%T	orario (hh:mm:ss)

# Esempio di utilizzo di DATE\_FORMAT

---

Indicare matricola e data di laurea (nel formato ‘dd | mm | yyyy, nome\_giorno’) degli studenti iscritti prima del 2005

```
SELECT Matricola, DATE_FORMAT(DataLaurea, '%d|%m|%Y, %W')
FROM Studente
WHERE Datascrizione < '2005-01-01';
```

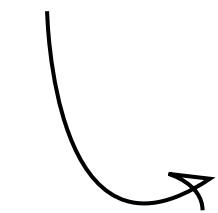
 Matricola	DATE_FORMAT(...)
1282	30 04 2010, Friday
3893	03 10 1957, Monday
8097	15 07 1953, Thursday

# Altro esempio

---

Indicare la matricola degli studenti che si sono laureati di mercoledì

```
SELECT Matricola  
FROM Studente  
WHERE DATE_FORMAT(DataLaurea, '%w') = 3;
```



estrae il giorno da DataLaurea  
come un intero in  $\{0, \dots, 6\}$



# Condizioni con le date: confronti

---

Si possono usare gli operatori **maggiore**, **minore**, **uguale**, combinazioni di essi, o BETWEEN



# Estrazione di giorno, mese, anno

---

i loro risultati si possono confrontare  
con maggiore, minore, etc.

Le funzioni **DAY**, **MONTH** e **YEAR** prendono come argomento una data e ne restituiscono, rispettivamente, giorno, mese e anno.

espressi come numeri interi

# Esempio

---

Indicare matricola e mese di laurea degli studenti immatricolati **dopo il 2000**

```
SELECT Matricola, MONTH(DataLaurea)
FROM Studente
WHERE DataLaurea IS NOT NULL
AND YEAR(Datalscrizione) > 2000;
```

 Matricola	MONTH(DataLaurea)
1282	04

# Complichiamo leggermente...

---

Indicare il cognome degli studenti che si sono laureati **cinque anni fa**

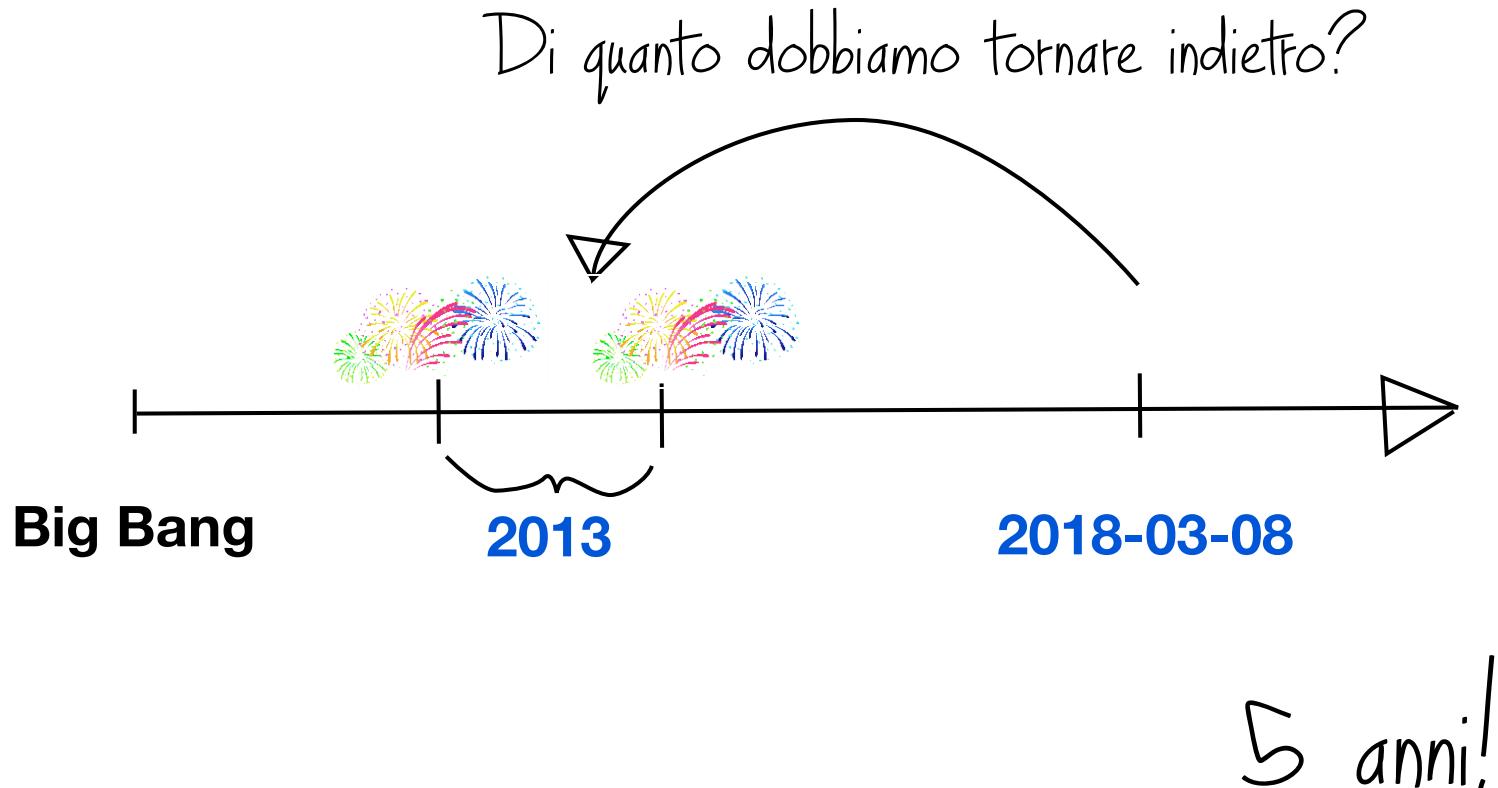
Attenzione: la query deve esprimere  
la condizione «**cinque anni fa**»,  
indipendentemente da quando viene eseguita.



# Balzo nel passato

---

Indicare il cognome degli studenti che si sono laureati **cinque anni fa**



# Balzo nel passato

---

Indicare il cognome degli studenti che si sono laureati **cinque anni fa**

```
SELECT DISTINCT Cognome  
FROM Studente  
WHERE DataLaurea IS NOT NULL  
      AND YEAR(DataLaurea) = YEAR(CURRENT_DATE) - 5;
```



Variabile di sistema che esprime la data odierna

# Lassi di tempo

---

Numero di mesi che separano due date  
espresso nel formato 'yyyymm' con date\_format()

Le date in MySQL **non possono essere sottratte**  
tramite l'operatore classico '-', ma solo con **DATEDIFF** o **PERIOD\_DIFF**

Numero di giorni che separano due date

# Lassi di tempo in giorni: esempio

---

Indicare matricola e **da quanti giorni risultavano iscritti** gli studenti, ad oggi laureati, che non si erano ancora laureati il 15 Luglio 2005

```
SELECT Matricola, DATEDIFF('2005-07-15', Datascrizione)  
FROM Studente  
WHERE Datascrizione < '2005-07-15'  
      AND DataLaurea > '2005-07-15';
```

datediff(data1, data2) esprime di quanti giorni  
data1 è più recente di data2

# Lassi di tempo in mesi: esempio

---

Indicare matricola e **da quanti mesi risultavano iscritti** gli studenti, ad oggi laureati, che non si erano ancora laureati il 15 Luglio 2005

```
SELECT Matricola, PERIOD_DIFF(  
    DATE_FORMAT(DataLaurea, '%Y%m'),  
    DATE_FORMAT(Datalscrizione, '%Y%m')  
)  
FROM Studente  
WHERE Datalscrizione < '2005-07-15'  
    AND DataLaurea > '2005-07-15';
```

la data più recente è il primo argomento  
della funzione period\_diff()

# Sommare/sottrarre intervalli di tempo

---

Le funzioni **DATE\_ADD** e **DATE\_SUB** permettono di sommare o sottrarre intervalli di tempo a una data

Il risultato è una nuova data

Sono espressi col la keyword **INTERVAL**

# INTERVAL, sintassi

---

Esprime un **intervallo di tempo** espresso in giorni, mesi o anni

**INTERVAL** Numerointero [ **YEAR** | **MONTH** | **DAY** ]

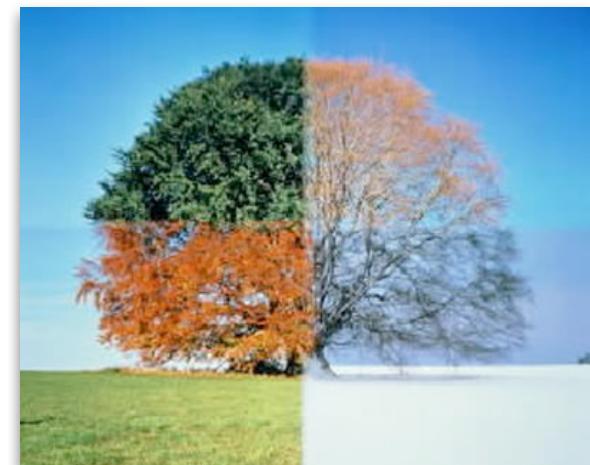
3 year

2 day

1 month

6 month

10 year



# Esempio

---

Indicare la matricola e il mese di iscrizione degli studenti che si sono laureati  
**dopo cinque anni esatti** dal giorno dell'iscrizione

```
SELECT Matricola, MONTH(Datalscrizione)
FROM Studente
WHERE DataLaurea IS NOT NULL
AND DataLaurea = DATE_ADD(Datalscrizione, INTERVAL 5 YEAR);
```

# Somma e sottrazione dirette

---

Indicare la matricola e il mese di iscrizione degli studenti che si sono laureati  
**dopo cinque anni esatti** dal giorno dell'iscrizione

```
SELECT Matricola, MONTH(Datalscrizione)
FROM Studente
WHERE DataLaurea IS NOT NULL
AND DataLaurea = Datalscrizione + INTERVAL 5 YEAR;
```

fa la stessa cosa della slide precedente



La somma e la sottrazione si fanno usando  
sempre INTERVAL, mai fra date

# Altre funzioni di utilità sulle date

---

Funzione	Risultato
dayname ()	nome del giorno
monthname ()	nome del mese
dayofweek ()	giorno della settimana in {1,...,7}
weekday ()	giorno della settimana in {0,...,6}
last_day ()	ultimo giorno del mese della data
dayofyear ()	messe (2 cifre)
weekofyear ()	numero della settimana {0,...,53}
yearweek ()	anno e settimana

altre funzioni sul manuale

# Operatori di aggregazione

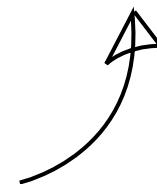
---



# Operatori di aggregazione (cont.)

---

conteggio, somma, minimo/massimo, media



Permettono di eseguire **calcoli** i cui operandi sono i valori  
assunti da un attributo, in un insieme di record



collassa in un solo record formato da  
un solo attributo numerico



# Conteggio

---

**Conta le righe** di una tabella, di un suo sottoinsieme,  
o di una combinazione di tabelle



# Conteggio: esempio

Indicare il **numero di visite** effettuate in data 1° Marzo 2013

SELECT COUNT(\*) AS VisitePrimoMarzo  
FROM Visita  
WHERE Data = '2013-03-01';

VisitePrimoMarzo
4

Conta le righe del risultato

Medico	Paziente	Data
35512	GTTFBL	2013-03-01
29858	MNZMBT	2012-11-30
18339	CPRLND	2013-03-01
35512	GTTFBL	2013-02-20
16220	MNZMBT	2013-01-25
35512	LPRNTA	2013-03-01
35512	CPRLND	2013-03-01

semplice realtà medica...

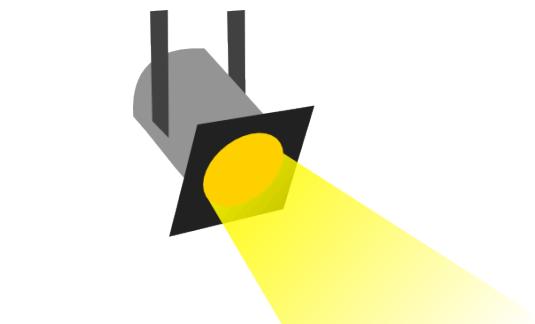
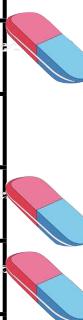
# Conteggio: come funziona?

---

Indicare il **numero di visite** effettuate in data 1° Marzo 2013

**VISITA**

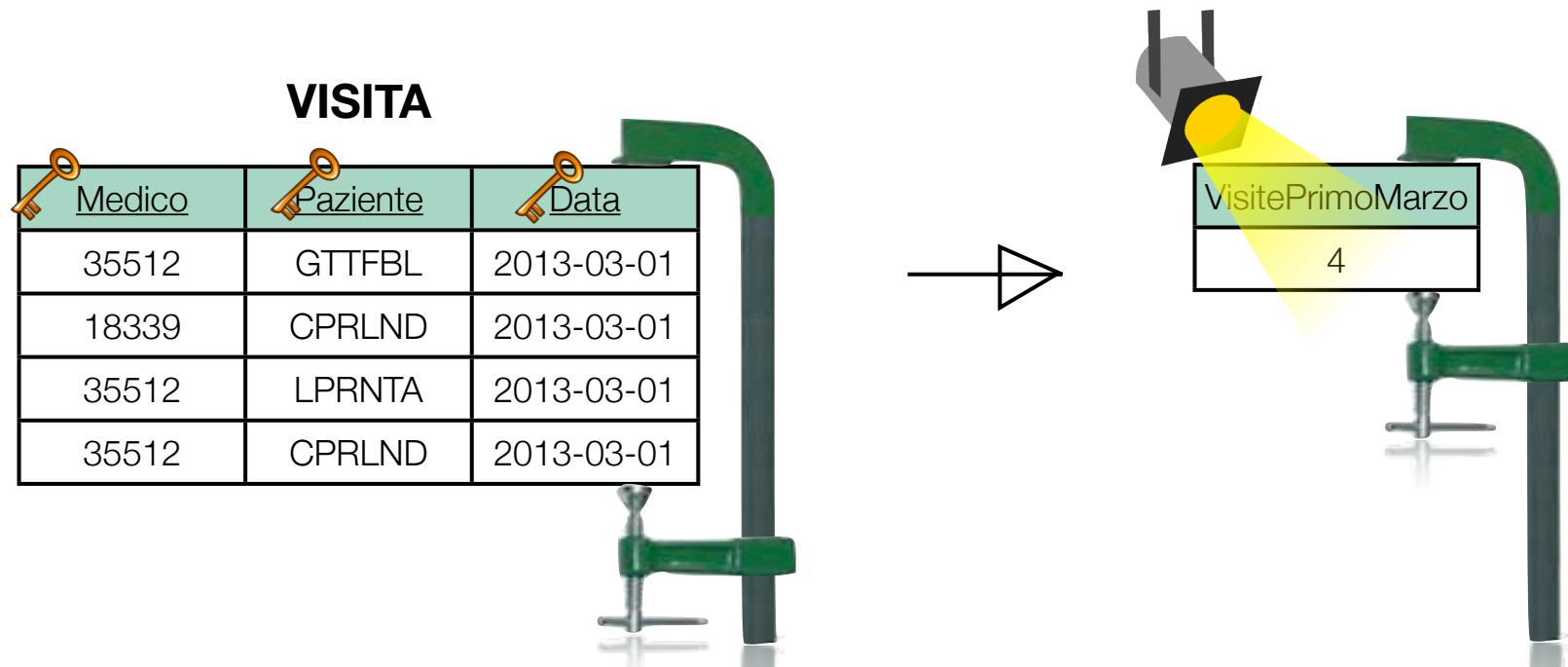
 Medico	 Paziente	 Data
35512	GTTFBL	2013-03-01
12358	N. P. V. I.	2012-11-30
18339	CPRLND	2013-03-01
35512	CPRLND	2013-02-20
1220	N. P. V. I.	2013-01-25
35512	LPRNTA	2013-03-01
35512	CPRLND	2013-03-01



Ma cosa si proietta?

# Conteggio: come funziona?

Indicare il **numero di visite** effettuate in data 1° Marzo 2013



Un solo record nel risultato!

# Conteggio su attributo/i

---

Permette di **contare i valori diversi** assunti da un attributo  
(o più attributi) in un insieme di record

Invece il conteggio classico conta le righe!



# Conteggio su attributo/i

Indicare il **numero di pazienti** visitati nel mese di Marzo 2013

VISITA		
Medico	Paziente	Data
35512	GTTFBL	2013-03-01
35512	CPRLND	2013-03-01
18339	CPRLND	2013-03-01
35512	CPRLND	2013-02-20
35512	CPRLND	2013-01-25
35512	LPRNTA	2013-03-01
35512	CPRLND	2013-03-01

*(Annotations: A callout bubble points to the row with Medico '35512' and Paziente 'CPRLND' on 2013-03-01, with the text 'Paziente visitato due volte!'. There are three red and blue capsule icons placed near the bottom right of the table.)*

Non si può usare `count(*)` che restituirebbe 4

# Conteggio su attributo/i

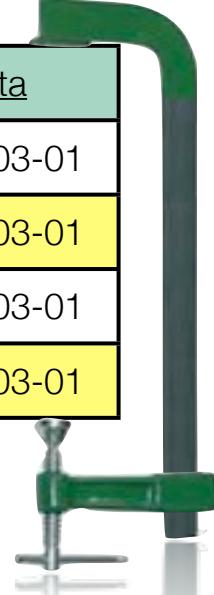
Indicare il **numero di pazienti** visitati nel mese di Marzo 2013

```
SELECT COUNT(DISTINCT Paziente)  
FROM Visita  
WHERE MONTH(Data) = '03'  
      AND YEAR(Data) = '2013';
```

COUNT(DISTINCT Paziente)
3

**VISITA**

Medico	Paziente	Data
35512	GTTFBL	2013-03-01
18339	CPRLND	2013-03-01
35512	LPRNTA	2013-03-01
35512	CPRLND	2013-03-01



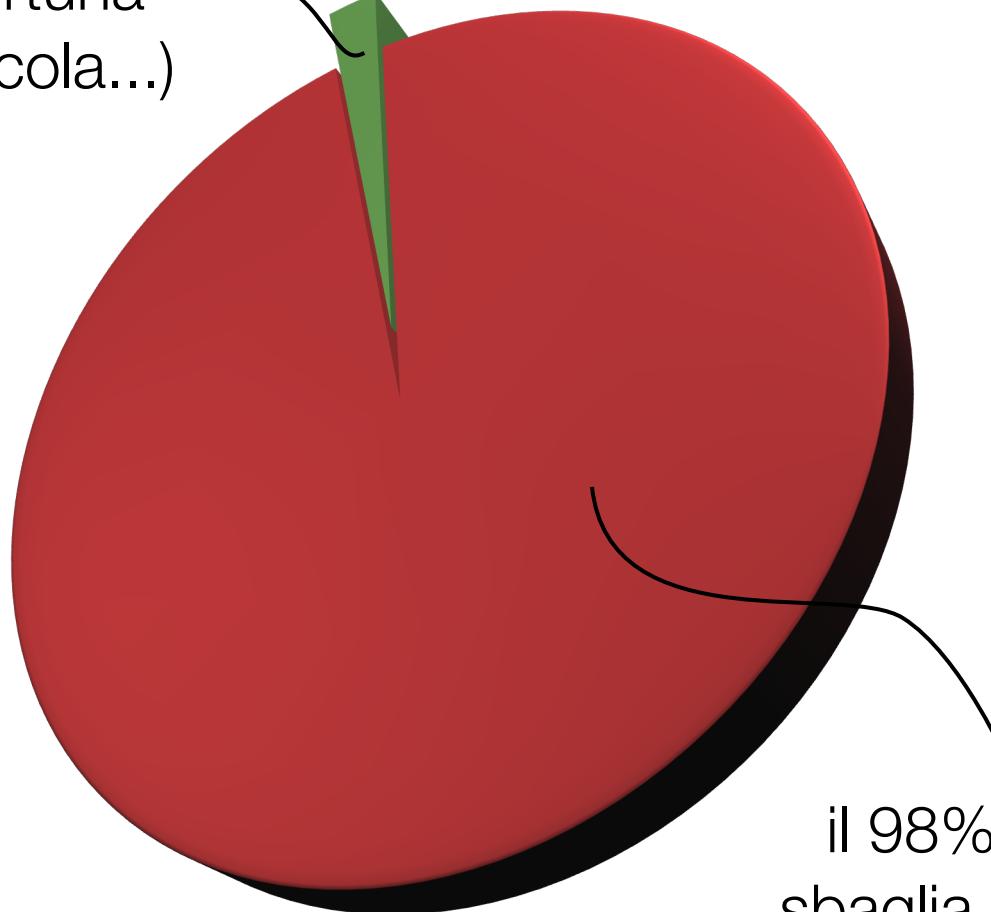
Conta le righe togliendo i duplicati su Paziente

# Statistiche (allarmanti) sul COUNT

---

il 2% ci azzecca  
(spesso per fortuna  
con la C maiuscola...)

● ERRORE ● GIUSTO



il 98% degli studenti  
sbaglia a usare COUNT

# Somma e media

---

**Somma o calcola la media aritmetica dei valori assunti  
da un attributo in un insieme di record**



# Somma e media

---

## PAZIENTE

CodFiscale	Cognome	Nome	DataNascita	Reddito
LPRNTA	Lepre	Antonella	1958-05-03	1200
GTTFBL	Gatto	Fabiola	1983-10-30	1400
MNZMBT	Manzi	Umberto	1949-07-12	1350
CPRLND	Capra	Leonardo	1967-09-20	1600

Semplice tabella per esercitarcisi sui calcoli...

# Somma: esempio

Supponendo che Umberto Manzi sia il marito di Antonella Lepre,  
calcolare il **reddito totale** della famiglia Manzi.

```
SELECT SUM(Reddotto) AS ReddottoTotale  
FROM Paziente  
WHERE (Cognome = 'Lepre'  
       AND Nome = 'Antonella')  
      OR  
(Cognome = 'Manzi'  
       AND Nome = 'Umberto');
```

PAZIENTE				
CodFiscale	Cognome	Nome	DataNascita	Reddotto
LPRNTA	Lepre	Antonella	1958-05-03	1200
GTTFE'L	Cattaneo	Giobio	1963-10-26	1100
MNZMBT	Manzi	Umberto	1949-07-12	1350
CPRLND	Papetti	Leonardo	1961-09-22	950

ReddottoTotale
2550

# Perché OR?!

---

Supponendo che Umberto Manzi sia il marito di Antonella Lepre,  
calcolare il **reddito totale** della famiglia Manzi.



La processazione della tabella  
avviene **record per record**.  
Non ha senso imporre che  
una persona si chiami  
contemporaneamente (quindi  
con AND) Umberto Manzi  
e Antonella Lepre!

**PAZIENTE**

CodFiscale	Cognome	Nome	DataNascita	Reddito
LPRNTA	Lepre	Antonella	1958-05-03	1200
GTTFBL	Gatto	Fabiola	1983-10-30	1400
MNZMBT	Manzi	Umberto	1949-07-12	1350
CPRLND	Capra	Leonardo	1967-09-20	1600

# Il dramma

---

Supponendo che Umberto Manzi sia il marito di Antonella Lepre,  
calcolare il **reddito totale** della famiglia Manzi.



```
SELECT SUM(Reddito) AS RedditoTotale  
FROM Paziente  
WHERE (Cognome = 'Lepre'  
       AND Nome = 'Antonella')  
      AND  
      (Cognome = 'Manzi'  
       AND Nome = 'Umberto');
```

*Scambio del connettivo logico*

# Media: esempio

Calcolare il **reddito medio** dei pazienti nati dopo il 1950

```
SELECT AVG(Reddotto) AS ReddottoMedio
```

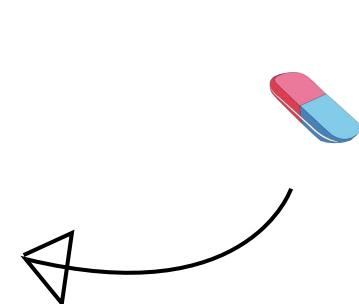
```
FROM Paziente
```

```
WHERE DataNascita > '1950-12-31';
```

PAZIENTE

CodiceFiscale	Cognome	Nome	DataNascita	Reddotto
LPRNTA	Lepre	Antonella	1958-05-03	1200
GTTFBL	Gatto	Fabiola	1983-10-30	1400
MNZMBT	Menzel	Alberto	1949-07-12	300
CPRLND	Capra	Leonardo	1967-09-20	1600

ReddottoMedio
1400



# Minimo e massimo

---

Individuano il **valore minimo** o il **valore massimo**  
fra i valori assunti da un attributo in un insieme di record



# Minimo e massimo

---

Ricavare il **reddito massimo/minimo** fra quelli di tutti i pazienti

**SELECT MAX(Reddotto)**

**FROM Paziente;**

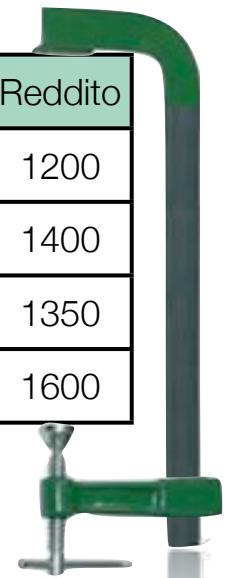
MAX(Reddotto)
1600

**SELECT MIN(Reddotto)**

**FROM Paziente;**

MIN(Reddotto)
1200

PAZIENTE				
CodiceFiscale	Cognome	Nome	DataNascita	Reddotto
LPRNTA	Lepre	Antonella	1958-05-03	1200
GTTFBL	Gatto	Fabiola	1983-10-30	1400
MNZMBT	Manzi	Umberto	1949-07-12	1350
CPRLND	Capra	Leonardo	1967-09-20	1600



# ...e se volessimo

---

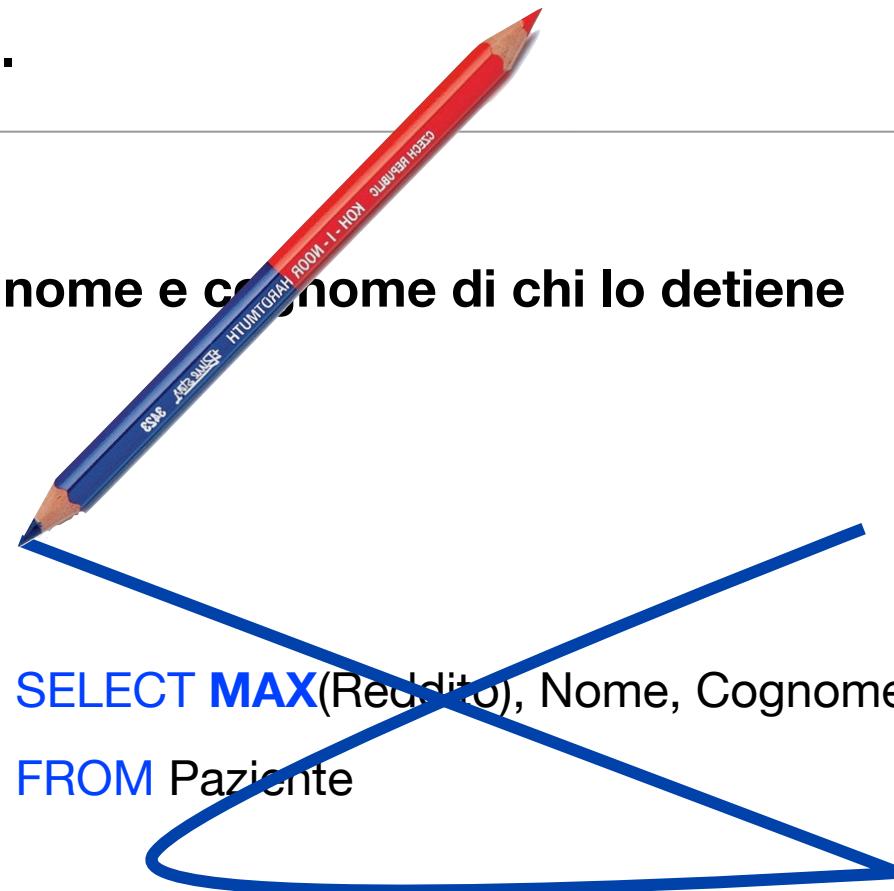
Indicare qual è il reddito massimo, **e il nome e cognome di chi lo detiene**

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



# Una tentazione malefica...

---



...scopriremo come sconfiggerla  
nella prossima lezione...

# Prossima lezione

---

La prossima lezione di Basi di dati relativa a MySQL si terrà **giovedì 15 Marzo** dalle ore **10:30** alle ore **12:30** in **aula F9**.

Durante le due ore di venerdì prossimo **risolverò alla lavagna** gli esercizi assegnati, spiegando il ragionamento risolutivo passo passo. Chiariremo i dubbi incontrati a casa nello svolgimento degli esercizi mediante un dibattito aperto a tutti.

Le **istruzioni per l'installazione di MySQL Server e MySQL Workbench** (client) sono riassunte nelle slide successive. State molto attenti durante l'installazione e non siate impazienti: lasciate che l'installazione termini senza spazientirvi anche se sembra che tutto sia bloccato. In alcuni punti il processo di installazione può risultare molto lento.

Occorrente per la prossima lezione: **matita verde** e **matita arancione**. Potete portare anche il vostro notebook per farmi vedere eventuali problemi incontrati.

# Esercizi per casa

---

Sia dato il seguente schema:

STUDENTE (Matricola, Cognome, Nome, DataNascita, DataIscrizione, DataLaurea,  
NumeroEsamiSostenuti, Facolta),

esprimere le seguenti richieste in linguaggio MySQL:

- Indicare matricola e da quanti giorni risultavano iscritti gli studenti, che non si erano ancora laureati il 15 Luglio 2005.
- Indicare matricola e cognome degli studenti il cui percorso di studi è durato (o dura da) oltre sei anni. Risolvere con e senza l'uso di `INTERVAL`.
- Indicare il numero medio di esami sostenuti dagli studenti nati nel 1987, iscritti con un anno di anticipo all'università, ad oggi non ancora laureati.
- Indicare nome, cognome ed età degli studenti laureati quest'anno in Lettere (durata standard 5 anni a ciclo unico), non fuori corso e come minimo con un anticipo di sei mesi rispetto alla durata standard.
- Indicare il numero di studenti che si sono laureati nel 2005, dopo il compimento del ventisettesimo anno d'età, e la loro età media al momento della laurea. Risolvere l'esercizio con e senza l'uso di `INTERVAL`.
- Indicare il numero di mesi impiegati per laurearsi dallo studente più veloce a laurearsi della facoltà di Ingegneria Meccanica, fra quelli laureati in pari, iscritti nel 2001. (*Laureati in pari* significa non oltre il mese di Aprile del 6° anno dall'iscrizione.)
- Indicare matricola e durata in mesi del percorso di studi degli studenti laureati fuori corso, cioè oltre il mese di Aprile del 6° anno, nell'anno accademico 2009-2010.

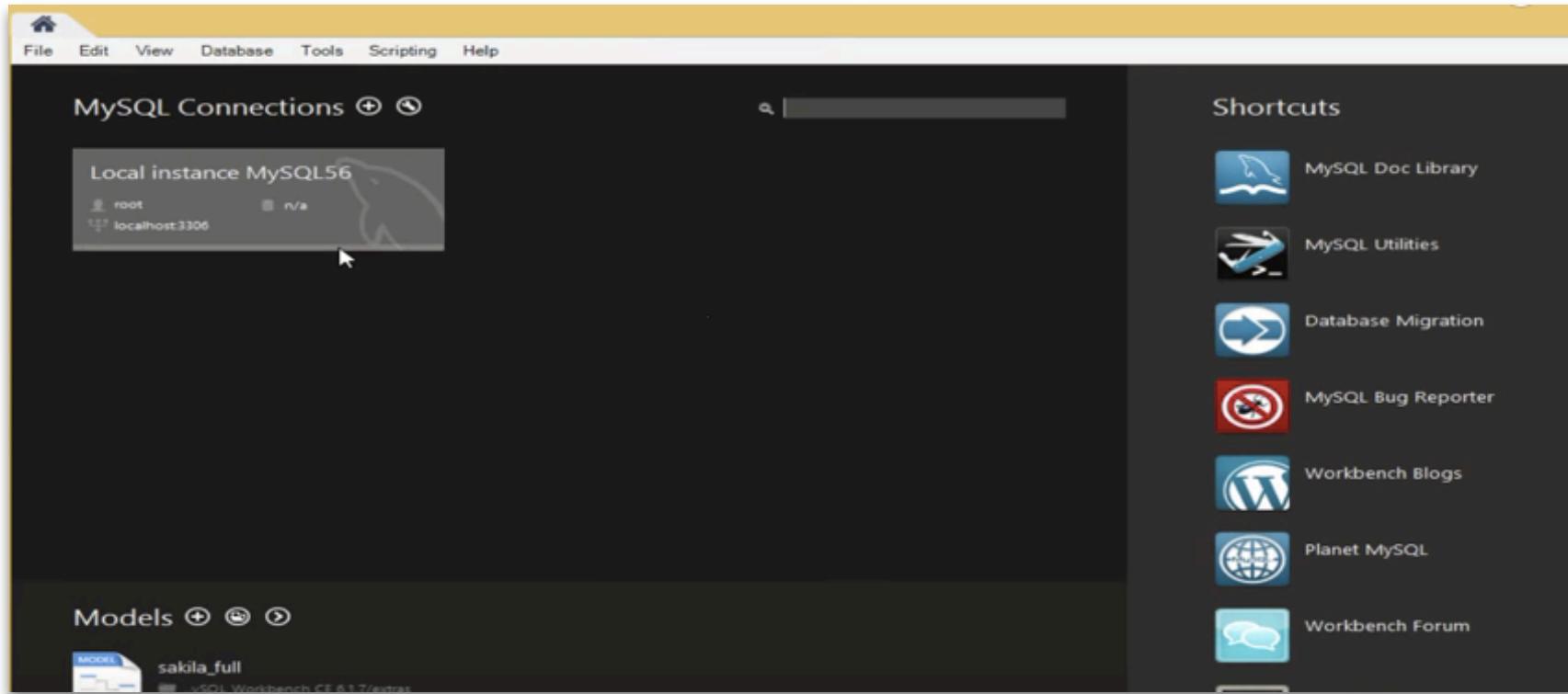
# Installazione MySQL Server e Workbench

---

## Microsoft Windows

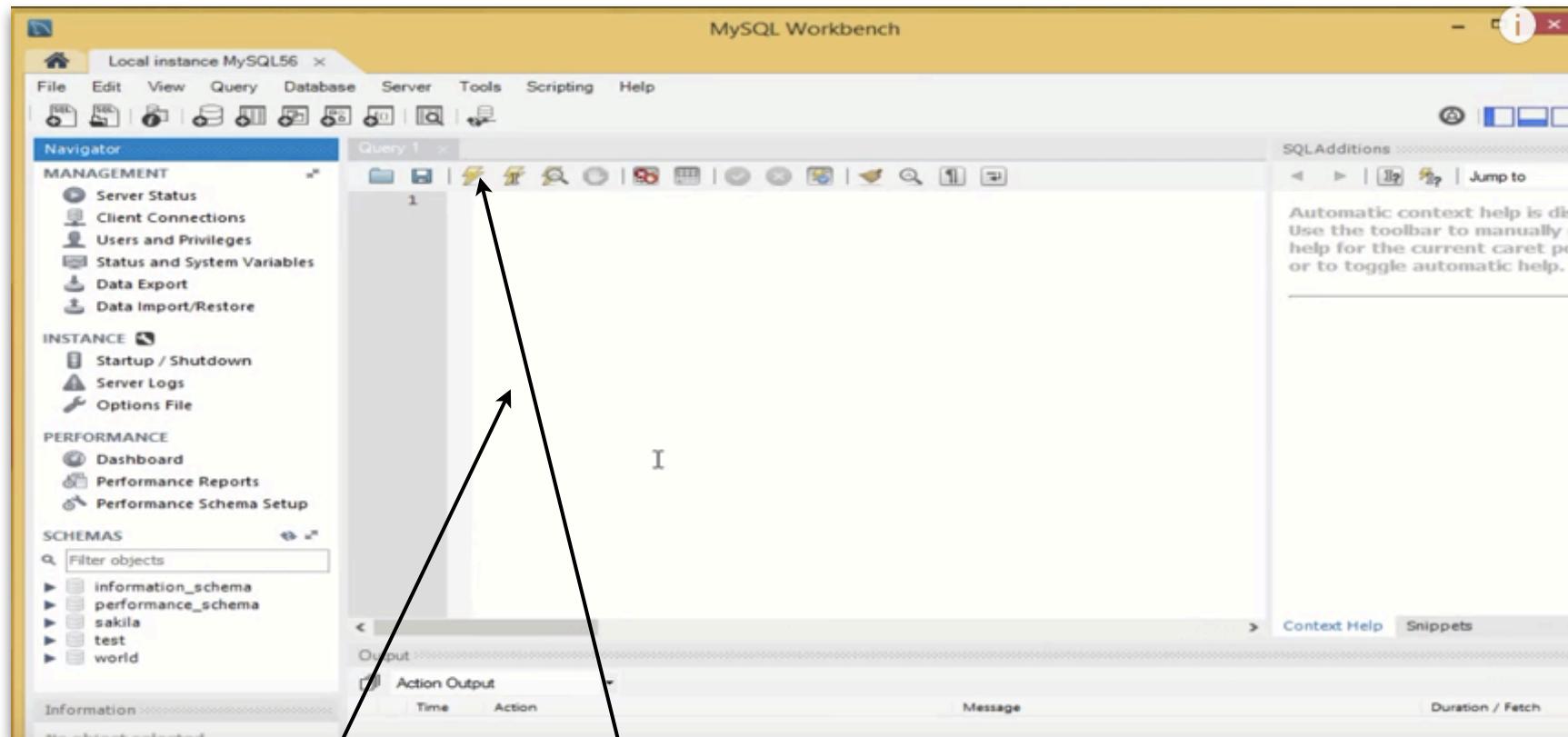
1. Sul sito di MySQL è presente un pacchetto installer che dovrebbe fare tutto da solo (e bene). Andare a questo indirizzo <http://dev.mysql.com/downloads/installer/> e fare clic su Download nella seconda riga (il file che pesa di più)
2. Nella finestra che si apre, clic in basso su “No thanks, just start my download”
3. Al termine del download chiudere tutte le applicazioni e lanciare l’installer
4. Al termine di tutto il processo di installazione lanciare MySQL Workbench si apre la finestra principale (vedi slide successiva)

# Installazione MySQL Server e Workbench



5. Fare clic sul rettangolo grigio col delfino dentro, in alto a sinistra
6. Digitare la password di root richiesta (quella scelta durante l'installazione) e clic su OK
7. Si apre il programma (vedi slide successiva)

# Installazione MySQL Server e Workbench



5. Copiare qui IL CONTENUTO del file Laboratorio.sql ricevuto per email
6. Premere il tasto a forma di fulmine per eseguire

# Installazione MySQL Server e Workbench

---

7. Una volta eseguito il codice viene creato il database e lo trovate in basso a sinistra nella sezione SCHEMAS (se non c'è, tasto destro su SCHEMAS e clic su REFRESH);
8. Fate doppio clic sul database creato (nella sezione SCHEMAS). Il database "Laboratorio" deve divenire in grassetto;
9. Iniziate a scrivere le query nell'area nella quale avete copiato precedentemente il codice dello script (cancellandolo prima ovviamente) ed eseguitele mediante il tasto a forma di fulmine;
10. In caso di errori, il compilatore in basso vi dice cosa c'è che non va.

# Installazione MySQL Server e Workbench

## Mac OS X (e altri sistemi UNIX)

1. Sul sito di MySQL scaricare il server da qui <http://dev.mysql.com/downloads/mysql/> selezionando un file di tipo DMG Archive compatibile con la versione di OS X (10.10, 10.9). Se usate versioni meno recenti, fate clic su “Looking for previous GA versions” e selezionate un DMG Archive compatibile nella finestra che si aprirà.
2. Nella finestra che si apre, fare clic sul testo in basso “No thanks, just start my download.”
3. Installare il server (doppio clic sul file .dmg scaricato) seguendo le istruzioni
4. Andare qui <http://dev.mysql.com/downloads/workbench/>
5. Scaricare e installare il software necessario indicato come prerequisito facendo clic sui seguenti due link e installando ciò che trovate

MySQL Workbench Prerequisites:

To be able to install and run MySQL Workbench your System needs to have libraries listed below installed. The listed items are provided as links to the corresponding download pages where you can fetch the necessary files.

- [Microsoft .NET Framework 4 Client Profile](#)
- [Visual C++ Redistributable for Visual Studio 2013](#)

# Installazione MySQL Server e Workbench

---

5. A questo punto (sempre dalla pagina <http://dev.mysql.com/downloads/workbench/>) scaricare il DMG di MySQL Workbench (clic su Download)
6. Nella finestra che si apre, fare clic sul testo in basso “No thanks, just start my download.”
7. Al termine del download, lanciare il DMG e installare MySQL Workbench
8. Al termine aprire l'applicazione MySQL Workbench e proseguire, mutatis mutandis, dalla slide 103 alla slide 105.  
Dovrebbe essere abbastanza intuitivo, a parte l'interfaccia grafica diversa.