Università degli Studi di Milano Bicocca
Corso di Laurea in Informatica

MySQL Select Interrogazioni complesse

LABORATORIO DI BASI DI DATI
A.A. 2019/2020

Dott. Marco Savi

Contenuti riadattati a partire da slide gentilmente concesse
dai **Dott. Paolo Napoletano** e **Claudio Venturini**

Riepilogo

- Nello scorso laboratorio:
  - Interrogazioni su una singola tabella
  - Interrogazioni con join di più tabelle
  - Operatori e funzioni

- Nel laboratorio di oggi:
  - Ordinamenti
  - Aggregazioni
  - Raggruppamenti
  - Interrogazioni nidificate (sotto-interrogazioni)
  - Interrogazioni insiemistiche

MySQL Select – Ordinamento

- L'insieme dei risultati dell'interrogazione può essere ordinato rispetto ad una o più colonne utilizzando la clausola `ORDER BY`

- *Selezionare la matricola, il cognome e il nome degli studenti iscritti dal 2005, ordinati alfabeticamente per cognome e nome*

```
> SELECT matricola, cognome, nome FROM studente
        WHERE YEAR(data_iscrizione) >= 2005
        ORDER BY cognome ASC, nome ASC;
```

Colonne di ordinamento:
- `ASC`: ordine crescente
- `DESC`: ordine decrescente

studente

matricola	cognome	nome	data_iscrizione
111	Rossi	Mario	03/09/2005
112	Verdi	Paolo	15/09/2006
113	Esposito	Maria	15/09/2006
114	Brambilla	Giovanni	20/09/2004
115	Ferrari	Alice	01/10/2005

matricola	cognome	nome
113	Esposito	Maria
115	Ferrari	Alice
111	Rossi	Mario
112	Verdi	Paolo

MySQL Select – Ordinamento

- L'ordinamento può essere effettuato anche rispetto a colonne di tabelle diverse o rispetto a valori calcolati tramite operazioni e funzioni

- *Selezionare gli studenti, con matricola e comune di residenza, ordinati in modo decrescente per numero di abitanti del comune di residenza, e per anno di iscrizione crescente*

```
> SELECT s.matricola, s.comune
        FROM studente AS s INNER JOIN comune AS c ON s.comune = c.nome
        ORDER BY c.abitanti DESC, YEAR(s.data_iscrizione) ASC;
```

studente

matricola	comune	data_iscrizione
111	Milano	03/09/2005
112	Bergamo	15/09/2006
113	Milano	15/09/2006
114	Milano	20/09/2004
115	Bergamo	01/10/2005

comune

nome	abitanti
Milano	1336364
Bergamo	118756

matricola	comune
114	Milano
111	Milano
113	Milano
115	Milano
112	Bergamo

MySQL Select – Aggregazione

- Gli operatori di aggregazione sono funzioni applicabili all'insieme di tuple risultanti da una interrogazione
  - Poiché si applicano all'intero set risultante, quando vengono utilizzati **la SELECT non può selezionare valori diversi da quelli calcolati con gli operatori di aggregazione** (se non si ha anche raggruppamento)

- Valutano l'espressione data per ogni tupla ottenuta dall'interrogazione e, allo stesso tempo, aggregano i risultati
  - `COUNT(expr)`: conta il numero di valori non NULL
    - **`COUNT(*)` è un'eccezione!** Conta il numero di risultati anche se NULL
  - `SUM(expr)`: calcola la somma dei valori
  - `AVG(expr)`: calcola la media dei valori
  - `MIN(expr)`: calcola il minimo dei valori
  - `MAX(expr)`: calcola il massimo dei valori
  - … e altri https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html

MySQL Select – Aggregazione

- *Calcolare il numero di studenti iscrittisi prima del 2006 e la media del voto di laurea*

Il numero di laureati è il numero totale di tuple con voto laurea non NULL

Media dei voti di laurea

```
> SELECT COUNT(voto_laurea) AS laureati, AVG(voto_laurea) AS media
       FROM studente
       WHERE YEAR(data_iscrizione) < 2006;
```

studente

matricola	cognome	nome	data_iscrizione	voto_laurea
111	Rossi	Mario	03/09/2005	102
112	Verdi	Paolo	15/09/2006	110
113	Esposito	Maria	15/09/2005	95
114	Brambilla	Giovanni	20/09/2004	110
115	Ferrari	Alice	01/10/2005	NULL

laureati	media
3	102.333333

Ignorato da `COUNT()` e `AVG()` poiché `voto_laurea` è NULL

MySQL Select – Aggregazione

- *Calcolare la percentuale di studenti iscrittisi prima del 2006 che si sono laureati e la media del voto di laurea*

```
> SELECT
      (COUNT(voto_laurea) / COUNT(*)) * 100 AS perc_laureati,
      AVG(voto_laurea) AS media
  FROM studente
  WHERE YEAR(data_iscrizione) < 2006;
```

`COUNT(voto_laurea)` ignora le tuple con voto_laurea NULL, mentre `COUNT(*)` **le considera**!

studente

matricola	cognome	nome	data_iscrizione	voto_laurea
111	Rossi	Mario	03/09/2005	102
112	Verdi	Paolo	15/09/2006	110
113	Esposito	Maria	15/09/2005	95
114	Brambilla	Giovanni	20/09/2004	110
115	Ferrari	Alice	01/10/2005	NULL

perc_laureati	media
75	102.333333

Considerato solo da `COUNT(*)`!

MySQL Select – Aggregazione

- Applicando la keyword `DISTINCT` all'operatore di aggregazione è possibile calcolare il valore aggregato considerando i soli valori diversi fra loro

- *Contare il numero di comuni in cui risiedono gli studenti iscrittisi prima del 2006*

```
> SELECT COUNT(DISTINCT comune) AS n_comuni
      FROM studente
      WHERE YEAR(data_iscrizione) < 2006;
```

studente

matricola	cognome	nome	data_iscrizione	comune
111	Rossi	Mario	03/09/2005	Milano
112	Verdi	Paolo	15/09/2006	Bergamo
113	Esposito	Maria	15/09/2006	Napoli
114	Brambilla	Giovanni	20/09/2004	Milano
115	Ferrari	Alice	01/10/2005	Bergamo

n_comuni
2

Milano compare 2 volte ma viene **contato una sola volta**

MySQL Select – Raggruppamento

- La clausola `GROUP BY` consente di partizionare l'insieme di tuple risultanti dall'interrogazione in gruppi omogenei di tuple
- Il raggruppamento può avvenire in base a valori uguali:
  - di una o più colonne
  - di un'espressione calcolata su ogni tupla

- Quando si effettua un raggruppamento gli operatori di aggregazione vengono valutati per ogni gruppo
- I valori selezionabili dalla `SELECT` possono essere solamente:
  - valori calcolati dagli operatori di aggregazione (se utilizzati)
  - valori delle colonne o delle espressioni utilizzate come criterio di raggruppamento

MySQL Select – Raggruppamento

- *Calcolare il numero di studenti iscrittisi prima del 2006, per comune di residenza, con il voto medio di laurea, ordinando alfabeticamente per comune*

studente

matricola	data_iscrizione	comune	voto_laurea
111	03/09/2005	Milano	102
112	15/09/2006	Bergamo	110
113	15/09/2005	Napoli	95
114	20/09/2004	Milano	110
115	01/10/2005	Bergamo	98
116	13/09/2004	Napoli	103
117	08/10/2005	Milano	108

I valori selezionati sono solo le colonne di raggruppamento e gli aggregati

```
> SELECT
      comune,
      COUNT(*) AS studenti,
      AVG(voto_laurea) AS voto_medio
  FROM studente
  WHERE YEAR(data_iscrizione) < 2006
  GROUP BY comune
  ORDER BY comune ASC;
```

Raggruppamento per valori uguali di una sola colonna

comune	studenti	voto_medio
Bergamo	1	98
Milano	3	106.666667
Napoli	2	99

MySQL Select – Raggruppamento

- *Calcolare il numero di studenti iscrittisi prima del 2006, per anno di iscrizione e comune di residenza, con il voto medio di laurea, ordinando per anno e alfabeticamente per comune*

```
SELECT
      YEAR(data_iscrizione) AS anno,
      comune,
      COUNT(*) AS studenti,
      AVG(voto_laurea) AS voto_medio
FROM studente
WHERE YEAR(data_iscrizione) < 2006
GROUP BY anno, comune
ORDER BY anno ASC, comune ASC;
```

Raggruppamento per valori uguali di più colonne o espressioni calcolate

… con gli stessi dati dell'esempio precedente

anno	comune	studenti	voto_medio
2004	Milano	1	110
2004	Napoli	1	103
2005	Bergamo	1	98
2005	Milano	2	105
2005	Napoli	1	95

MySQL Select – Raggruppamento

- Attraverso la clausola `HAVING` è possibile filtrare il risultato finale
  - Equivalente alla clausola `WHERE`, ma **valutata solo dopo il raggruppamento**
  - Permette di applicare anche **condizioni sui risultati degli operatori di aggregazione**

- Riprendendo l'esempio precedente:
  Calcolare il numero di studenti iscrittisi prima del 2006, per anno di iscrizione e comune di residenza, con il voto medio di laurea, solamente per gli anni e i comuni che hanno almeno 2 iscritti

```
SELECT
      YEAR(data_iscrizione) AS anno,
      comune,
      COUNT(*) AS studenti,
      AVG(voto_laurea) AS voto_medio
FROM studente
WHERE anno < 2006
GROUP BY anno, comune
HAVING studenti >= 2;
```

… con gli stessi dati dell'esempio precedente

anno	comune	studenti	voto_medio
~~2004~~	~~Milano~~	~~1~~	~~110~~
~~2004~~	~~Napoli~~	~~1~~	~~103~~
~~2005~~	~~Bergamo~~	~~1~~	~~98~~
2005	Milano	2	105
~~2005~~	~~Napoli~~	~~1~~	~~95~~

MySQL Select – Interrogazioni nidificate

- Consentono di **costruire interrogazioni complesse**, il cui risultato dipende dal risultato di altre sotto-interrogazioni (o *subquery*)

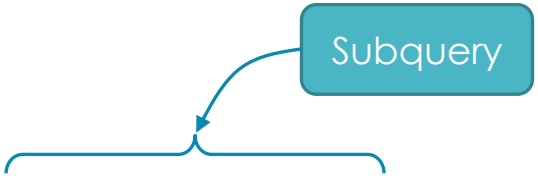
- Una subquery può essere utilizzata:
  - Nella clausola `WHERE`: per **filtrare i record** selezionati dalla query principale sulla base dell'esito della subquery
  - Nella clausola `FROM`: per **arricchire l'insieme di tuple** su cui viene effettuata l'interrogazione principale con un set di tuple fornito dalla subquery

- Non c'è limite al numero di subquery e di livelli di nidificazione
- Aumentano la potenza espressiva del linguaggio
  - Molte query semplici possono essere riscritte utilizzando query nidificate
    - A volte migliora la leggibilità (ma spesso a scapito delle performance)
  - **Attenzione**: non è vero il contrario! **Alcune interrogazioni sono esprimibili solo utilizzando query nidificate**.

MySQL Select – Interrogazioni nidificate

- Subquery nella clausola `WHERE`

```
> SELECT …
  FROM …
  WHERE expr operator (SELECT … FROM …)
  …;
```



  - Per ogni tupla della query esterna valuta l'espressione `expr`, e ne confronta il risultato, tramite l'operatore specificato, con il risultato della sub-query
  - Operatori:
    - `IN`: produce `true` se il valore è contento nel risultato della subquery
    - `NOT IN`: contrario di `IN`
    - `=`, `>`, `<`, `<=`, `>=`, `<>`: operatori di confronto

MySQL Select – Interrogazioni nidificate

- *Selezionare la matricola e il comune di residenza degli studenti che risiedono in comuni di almeno 100.000 abitanti, ordinati per matricola*

```
SELECT s.matricola, c1.nome AS comune
FROM studente AS s INNER JOIN comune AS c1 ON s.comune = c1.codice
WHERE c1.codice IN (
      SELECT c2.codice FROM comune AS c2 WHERE c2.abitanti >= 100000
)
ORDER BY s.matricola ASC;
```

studente

matricola	cognome	nome	comune
111	Rossi	Mario	1
112	Verdi	Paolo	2
113	Brambilla	Giovanni	1
114	Ferrari	Alice	3
115	Bianchi	Marco	2
116	Ronchi	Alice	1

comune

codice	nome	abitanti
1	Milano	1336364
2	Pisa	89620
3	Bergamo	118756

matricola	comune
111	Milano
113	Bergamo
114	Bergamo
116	Milano

MySQL Select – Interrogazioni nidificate

- *Selezionare la matricola e il voto di laurea dei laureati con un voto superiore alla media dei voti di laurea dei laureati dello stesso anno*

```
> SELECT s1.matricola, s1.voto_laurea
  FROM studente AS s1
  WHERE s1.voto_laurea >= (
        SELECT AVG(s2.voto_laurea) FROM studente AS s2
        WHERE s2.anno_laurea = s1.anno_laurea
  );
```

Per usare i normali operatori di confronto **la subquery deve produrre un unico valore**

La subquery si dice **correlata** poiché fa riferimento a colonne della query esterna

studente

matricola	cognome	nome	anno_laurea	voto_laurea
111	Rossi	Mario	2010	101
112	Verdi	Paolo	2011	103
113	Brambilla	Giovanni	2011	98
114	Ferrari	Alice	2012	102
115	Bianchi	Marco	2010	99
116	Ronchi	Alice	2011	110

matricola	voto_laurea
111	101
114	102
116	110

MySQL Select – Interrogazioni nidificate

- Con gli operatori di confronto è possibile applicare dei quantificatori:
  - `ANY` (oppure `SOME`): **quantificatore esistenziale**
    Produce `true` se l'operatore restituisce `true` nel confronto con **almeno una delle tuple** risultanti dalla subquery
  - `ALL`: **quantificatore universale**
    Produce `true` se l'operatore restituisce `true` nel confronto con **tutte le tuple** risultanti dalla subquery
- I quantificatori consentono di applicare gli operatori di confronto anche se la subquery produce più di un valore
- Esempi:
  - `expr > ALL (SELECT …)` è `true` se `expr` è maggiore di ogni risultato della subquery
  - `expr = ANY (SELECT …)` è `true` se `expr` è uguale ad un qualsiasi risultato della subquery

MySQL Select – Interrogazioni nidificate

- *Selezionare i comuni presso cui non risiede nessuna studentessa con nome "Alice"*

```
> SELECT nome
  FROM comune
  WHERE codice <> ALL (
        SELECT comune FROM studente WHERE nome = "Alice"
  );
```

Il codice del comune deve essere **diverso da quelli di tutti i comuni** presso cui risiedono studentesse che si chiamano "Alice"

studente

matricola	cognome	nome	comune
111	Rossi	Mario	1
112	Verdi	Paolo	2
113	Brambilla	Giovanni	4
114	Ferrari	Alice	1
115	Bianchi	Marco	4
116	Ronchi	Alice	3

comune

codice	nome
1	Milano
2	Pisa
3	Bergamo
4	Napoli

comune
Napoli
Pisa

MySQL Select – Interrogazioni nidificate

- SQL consente di utilizzare il quantificatore esistenziale anche per verificare l'esistenza di almeno un record nel risultato della subquery
  - `EXISTS`: `true` se la subquery produce almeno un risultato
  - `NOT EXISTS`: `true` se la subquery non produce risultati
- In questo caso non si confronta il valore di un'espressione con il risultato della subquery, ma **si verifica solo il numero di risultati** prodotti da essa
- Selezionare i comuni presso cui non risiede nessuno degli studenti

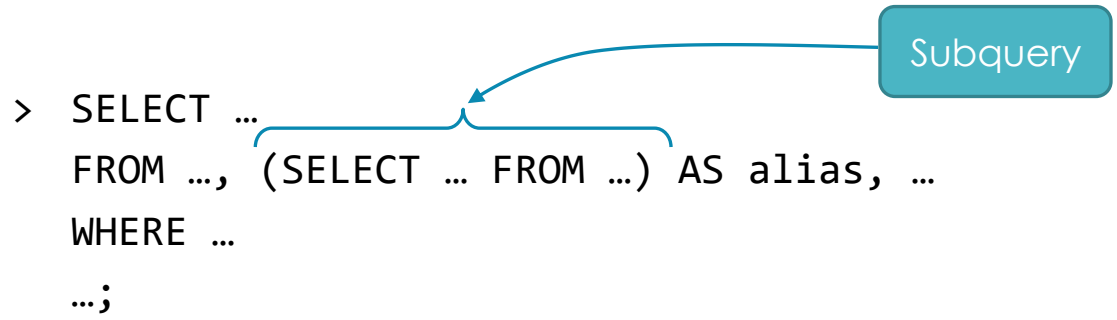
```
> SELECT c.nome
  FROM comune AS c
  WHERE NOT EXISTS (
        SELECT s.matricola FROM studente AS s WHERE s.comune = c.codice
  );
```

Seleziona il comune **solo se non esistono** studenti che vi risiedono

Quali siano i dati selezionati dalla subquery in questo caso non ha alcuna rilevanza

MySQL Select – Interrogazioni nidificate

- Subquery nella clausola `FROM`



  - Una subquery può comparire come qualsiasi altra tabella, **sia in join implicito sia in join esplicito**
  - Il risultato della subquery è utilizzabile nella query esterna come se fosse una qualsiasi tabella in join
  - È buona norma assegnare un nome (*alias*) al risultato, tramite la keyword `AS`
    - Consente alla query esterna di fare riferimento ai valori selezionati dalla subquery

MySQL Select – Interrogazioni nidificate

- Riscriviamo una query di un esempio precedente…

  Selezionare la matricola e l'anno e il voto di laurea dei laureati con un voto superiore alla media dei voti di laurea dei laureati dello stesso anno

```
> SELECT s1.matricola, s1.voto_laurea
  FROM studente AS s1 INNER JOIN (
      SELECT s2.anno_laurea AS anno, AVG(s2.voto_laurea) AS media
      FROM studente AS s2
      GROUP BY s2.anno_laurea
  ) AS voti_medi ON s1.anno_laurea = voti_medi.anno
  WHERE s1.voto_laurea >= voti_medi.media;
```

Assegna l'alias `voti_medi` al risultato della subquery

La subquery calcola la **media dei voti di laurea per tutti gli anni**

MySQL Select – Interrogazioni insiemistiche

- Consentono di effettuare operazioni su insiemi di risultati di interrogazioni differenti, ossia sui risultati di due SELECT
- I due **insiemi devono essere omogenei**: devono avere colonne in **uguale numero e tipo** di dati
- Operatori insiemistici:
  - `UNION`: produce l'**unione** dei due insiemi di risultati
  - `INTERSECT`: produce l'**intersezione** dei due insiemi di risultati
  - `EXCEPT`: produce la **differenza** tra i due insiemi, restituendo tutte e sole le tuple del primo insieme che non sono presenti nel secondo insieme

Non supportati da MYSQL!
Emulabili con `LEFT/RIGHT JOIN` e/o subquery

- Il risultato è un insieme di risultati **privo di duplicati**
  - La definizione matematica di insieme impone che contenga solo elementi distinti
  - Per mantenere i duplicati è possibile specificare anche la keyword `ALL`

MySQL Select – Interrogazioni insiemistiche

- *Selezionare i nomi degli studenti laureatisi nel 2011 oppure nel 2012*

```
> (SELECT nome FROM studente WHERE anno_laurea = 2011)
UNION
(SELECT nome FROM studente WHERE anno_laurea = 2012);
```

Insieme dei nomi degli studenti laureatisi nel 2011

Insieme dei nomi degli studenti laureatisi nel 2012

Unione tra i due insiemi di nomi

studente

matricola	cognome	nome	anno_laurea
111	Rossi	Mario	2010
112	Verdi	Paolo	2011
113	Brambilla	Giovanni	2011
114	Ferrari	Alice	2012
115	Bianchi	Marco	2010
116	Ronchi	Alice	2011

nome
Paolo
Giovanni
Alice

UNION ALL

nome
Paolo
Giovanni
Alice
Alice

Alice compariva due volte!

MySQL Select – Interrogazioni insiemistiche

- *Selezionare i nomi di tutti gli studenti, tranne quelli di studenti laureati prima del 2011*

```
> (SELECT nome FROM studente)
  EXCEPT
  (SELECT nome FROM studente WHERE anno_laurea < 2011);
```

EXCEPT **non è supportata** da MySQL

  - Emuliamo lo stesso effetto riscrivendo la query utilizzando solo costrutti supportati da MySQL

```
> SELECT DISTINCT nome
  FROM studente
  WHERE nome NOT IN (
        SELECT nome FROM studente WHERE anno_laurea < 2011
  );
```

`DISTINCT` rimuove eventuali duplicati, emulando l'effetto degli operatori insiemistici.
Per emulare `EXCEPT ALL` è sufficiente rimuovere `DISTINCT`

Esercizi

Esercizio 9 – Query complesse

- Utilizzare, come per l'esercizio 7, il database creato dallo script `universita.sql`
- Scrivere ed eseguire le seguenti query **utilizzando l'operatore di join esplicito, le clausole di ordinamento, gli operatori di aggregazione, i raggruppamenti, le interrogazioni nidificate e gli operatori insiemistici** (viene fornito il numero di righe per la soluzione corretta):
  1. Stilare una classifica degli studenti (con cognome, nome, matricola e voto) che hanno sostenuto l'esame con codice E3101Q117 con un voto maggiore di 23, ordinandola per voto decrescente [112 righe]
  2. Contare il numero di studenti che hanno superato l'esame con codice E3101Q117 con un voto maggiore di 23 [1 riga, 112 studenti]
  3. Selezionare gli studenti (con nome, cognome e matricola) con la media dei voti degli esami, e ordinarli secondo la media in ordine decrescente [700 righe]
  4. Selezionare gli studenti (con nome, cognome e matricola) che iniziano con la lettera B e il numero di esami svolti [14 righe]
  5. Selezionare gli studenti (con nome, cognome e matricola) che hanno svolto l'esame con codice E3101Q020 e/o con codice E3101Q117 [298 righe]
  6. Selezionare gli studenti (con nome, cognome e matricola) che hanno svolto l'esame con codice E3101Q020 e con codice E3101Q117, usando interrogazioni nidificate[41 righe]

Esercizio 9 – Query complesse

7. Visualizzare i corsi (con codice e nome) di cui lo studente con matricola 1492601 non ha ancora sostenuto l'esame [14 righe]
8. Visualizzare i corsi (con codice e nome) il cui esame deve ancora essere sostenuto da almeno uno studente (rispetto al relativo piano di studi) e il numero di studenti che li devono sostenere. Si considerino solo i corsi della laurea triennale e si ordinino i risultati in ordine decrescente per numero di studenti [31 righe]
9. Visualizzare i corsi (con codice e nome) con la media dei voti conseguiti dagli studenti nei relativi esami, e il numero di studenti che li hanno sostenuti. Selezionare solo i corsi della laurea magistrale il cui esame è stato superato da almeno 60 studenti. Si ordinino i risultati in ordine decrescente rispetto alla media dei voti [7 righe]
10. Selezionare il miglior studente in termini di media dei voti (con cognome, nome, matricola, media e numero esami sostenuti) della laurea triennale [1 riga, studente: Accarino Nicola]
11. Visualizzare le città di residenza (con nome e regione) e il numero di studenti che provengono da tali città. Ordinare i risultati in ordine decrescente rispetto al numero di studenti [298 righe]
12. Visualizzare gli studenti (con cognome, nome, matricola) della triennale e il numero di crediti che hanno finora ottenuto, in ordine decrescente per numero di crediti ottenuti [531 righe]