Seconda Esercitazione di Basi di Dati

Lorenzo Bonin
Università degli Studi di Trieste

14 aprile 2023

Overview

1. Riassunto dell'esercitazione precedente
2. Altre query
3. Prepared statements
4. Query su viste

Database dell'Università

Studenti

Matricola	Nome	Cognome	Codice Fiscale

Professori

Matricola	Nome	Cognome	Codice Fiscale	Settore

Corsi

Codice	Nome	CFU	Professore [Matricola]

Esami

Corso [Codice]	Studente [Matricola]	Data	Voto	Lode

Cosa abbiamo visto fino ad ora

- Creazione e popolamento del DB
- Query sul DB creato
  - **Query 1**: *elencare tutte le ragazze iscritte ad ingegneria*
    → `LIKE`, `BETWEEN`, `IN`, `SUBSTRING`
    → `aggiunta di una colonna "Genere" per facilitare operazioni future`
  - **Query 2**: *quanti studenti hanno preso una lode negli esami del prof. De Lorenzo?*
    → `DISTINCT`, `COUNT`, `INNER JOIN`
  - **Query 3**: *elencare quali studenti hanno preso più di una lode con il prof. De Lorenzo*
    → `DISTINCT`, `GROUP BY`, `HAVING`, `INNER JOIN`

Per partire tutti dallo stesso punto...

File **uni_db.sql**

- Elimina il DB se esiste già
- Ne crea uno nuovo
- Lo popola

File **query1.sql**

- Contiene le query della prima esercitazione
- Contiene il codice per modificare il DB

Ti sei perso l'esercitazione precedente? → esegui **uni_db.sql** e **query1.sql**.
Hai già creato il DB la scorsa volta? → `USE uni_db;`

Esecuzione di un file SQL da MySQL Workbench

- File $\rightarrow$ Open SQLScript $\rightarrow$ ...
- Selezionare la porzione di codice da eseguire (in questo caso tutto)
- Selezionare il pulsante con il fulmine

Per chi non ha MySQL e MySQL Workbench...

Esiste questo editor online: https://www.jdoodle.com/execute-sql-online/. Svantaggi:

- potrebbe non supportare alcune operazioni
- è volatile, ovvero non ricorda quanto eseguito in precedenza (è necessario creare, popolare ed interrogare in un'unica esecuzione!)

Morale: può andare bene come soluzione temporanea.

Query 4

Task

Quali studenti non hanno mai preso una lode?

Query 4

Task

Quali studenti non hanno mai preso una lode?

- *Hint*: quelli per cui *non esiste* un esame con la lode.

Query 4

Task

Quali studenti non hanno mai preso una lode?

- *Hint*: quelli per cui *non esiste* un esame con la lode.

```
SELECT *
FROM studenti s
WHERE NOT EXISTS(
    SELECT *
    FROM esami e
    WHERE e.lode = TRUE AND
    e.studente = s.matricola
);
```

Query 4 - alternativa

Task

Quali studenti non hanno mai preso una lode?

- *Hint*: quelli che *non appartengono* all'insieme degli studenti che hanno preso una lode.

Query 4 - alternativa

Task

Quali studenti non hanno mai preso una lode?

- *Hint*: quelli che *non appartengono* all'insieme degli studenti che hanno preso una lode.

```
SELECT *
FROM studenti s
WHERE s.matricola NOT IN(
    SELECT DISTINCT studente
    FROM esami e
    WHERE e.lode = TRUE
);
```

Query 5

Task

Quali docenti svolgono un monte ore annuo minore di 120 ore?

Query 5

Task

Quali docenti svolgono un monte ore annuo minore di 120 ore?

- *Hint*: un CFU corrisponde a 8 ore di lezione

Query 5

Task

Quali docenti svolgono un monte ore annuo minore di 120 ore?

- *Hint*: un CFU corrisponde a 8 ore di lezione

```
SELECT p.nome, p.cognome, SUM(8 * c.cfu) as
    monte_ore
FROM professori p
INNER JOIN corsi c
ON p.matricola = c.professore
GROUP BY c.professore
HAVING monte_ore < 120;
```

Query 6

Task

Mostrare la media ponderata di ogni studente.

Query 6

Task

Mostrare la media ponderata di ogni studente.

- *Hint*: la media ponderata si calcola moltiplicando il voto dell'esame per il suo peso in CFU, sommando, e poi dividendo per il numero di CFU.

Query 6

Task

Mostrare la media ponderata di ogni studente.

- *Hint*: la media ponderata si calcola moltiplicando il voto dell'esame per il suo peso in CFU, sommando, e poi dividendo per il numero di CFU.

```
SELECT s.matricola, s.nome, s.cognome,
SUM(e.voto*c.cfu)/SUM(c.cfu) as media
FROM studenti s
INNER JOIN esami e
ON s.matricola = e.studente
    INNER JOIN corsi c
    ON e.corso = c.codice
GROUP BY e.studente;
```

Query 7

Task

Verificare se ci sono casi di omonimia tra studenti e/o professori

Query 7

Task

Verificare se ci sono casi di omonimia tra studenti e/o professori

```
SELECT nome, cognome, COUNT(*) AS c
FROM(
    SELECT nome , cognome
    FROM studenti
    UNION ALL
    SELECT nome, cognome
    FROM professori) AS t
GROUP BY nome, cognome
ORDER BY c DESC;
```

Prepared statement 1

Task

Creare un prepared statement che mostri tutti gli studenti appartenenti ad un corso di laurea passato come parametro.

Prepared statement 1

Task

Creare un prepared statement che mostri tutti gli studenti appartenenti ad un corso di laurea passato come parametro.

- *Hint*: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente.

Prepared statement 1

Task

Creare un prepared statement che mostri tutti gli studenti appartenenti ad un corso di laurea passato come parametro.

- *Hint*: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente.

```
PREPARE studenti_cdl FROM
    "SELECT *
    FROM studenti
    WHERE matricola LIKE CONCAT(?, '%')";
```

Prepared statement 1

Task

Creare un prepared statement che mostri tutti gli studenti appartenenti ad un corso di laurea passato come parametro.

- *Hint*: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente.

```
PREPARE studenti_cdl FROM
    "SELECT *
    FROM studenti
    WHERE matricola LIKE CONCAT(?, '%')";
```

Usiamo lo statement per mostrare gli ingegneri informatici della triennale:

Prepared statement 1

Task

Creare un prepared statement che mostri tutti gli studenti appartenenti ad un corso di laurea passato come parametro.

- *Hint*: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente.

```
PREPARE studenti_cdl FROM
    "SELECT *
    FROM studenti
    WHERE matricola LIKE CONCAT(?, '%')";
```

Usiamo lo statement per mostrare gli ingegneri informatici della triennale:

```
SET @cdl = "IN05";
EXECUTE studenti_cdl USING @cdl;
```

Prepared statement 2

Task

Creare un prepared statement che mostri tutti gli studenti che hanno superato l'esame di un dato corso, il cui codice è passato come parametro.

Prepared statement 2

Task

Creare un prepared statement che mostri tutti gli studenti che hanno superato l'esame di un dato corso, il cui codice è passato come parametro.

```
PREPARE studenti_superato_corso FROM
    "SELECT s.nome, s.cognome, s.matricola
    FROM studenti s
    INNER JOIN esami e
    ON s.matricola = e.studente
    WHERE e.corso = ?";
```

Vista 1

Task

Quali sono i voti preferiti di ogni professore?

Vista 1

Task

Quali sono i voti preferiti di ogni professore?

- *Hint*: creare prima una vista che mostri la distribuzione dei voti per ogni docente.

Vista 1

Task

Quali sono i voti preferiti di ogni professore?

- *Hint*: creare prima una vista che mostri la distribuzione dei voti per ogni docente.

```
CREATE VIEW dist_voti AS
SELECT p.matricola, p.nome, p.cognome, e.voto,
    COUNT(e.voto) as n_voti
FROM professori p
INNER JOIN corsi c ON p.matricola = c.professore
    INNER JOIN esami e ON c.codice = e.corso
    GROUP BY p.matricola, e.voto;
```

Vista 1

A questo punto scegliamo il voto più frequente per ogni docente:

Vista 1

A questo punto scegliamo il voto più frequente per ogni docente:

```
SELECT DISTINCT matricola, nome, cognome, voto
FROM dist_voti d1
WHERE n_voti = (
    SELECT MAX(n_voti)
    FROM dist_voti d2
    WHERE d1.matricola = d2.matricola
);
```

Vista 2

Task

Quali sono gli studenti più "bravi" di ogni corso di laurea?

Vista 2

Task

Quali sono gli studenti più "bravi" di ogni corso di laurea?

- *Hint*: possiamo definire la "bravura" come somma dei voti moltiplicati per i rispettivi CFU, ma anche altre metriche sono possibili.

Vista 2

Task

Quali sono gli studenti più "bravi" di ogni corso di laurea?

- *Hint*: possiamo definire la "bravura" come somma dei voti moltiplicati per i rispettivi CFU, ma anche altre metriche sono possibili.

```
CREATE VIEW bravura_per_cdl AS
SELECT s.matricola, s.nome, s.cognome,
    SUBSTRING(s.matricola, 1, 4) as cdl,
    SUM(e.voto * c.cfu) as bravura
FROM studenti s
INNER JOIN esami e ON s.matricola = e.studente
    INNER JOIN corsi c ON e.corso = c.codice
GROUP BY s.matricola ;
```

Vista 2

A questo punto scegliamo il più bravo di ciascun corso:

Vista 2

A questo punto scegliamo il più bravo di ciascun corso:

```
SELECT DISTINCT matricola , nome , cognome , cdl
FROM bravura_per_cdl b1
WHERE bravura = (
    SELECT MAX(bravura)
    FROM bravura_per_cdl b2
    WHERE b1.cdl=b2.cdl
);
```