UNIVERSITÀ DEGLI STUDI
DI TRIESTE

ARTIFICIAL INTELLIGENCE
& DATA ANALYTICS

Seconda esercitazione di Basi di Dati

Università degli studi di Trieste

Giovanni **Pinna**

22 aprile 2024

Contatti

GIOVANNI.PINNA@phd.units.it

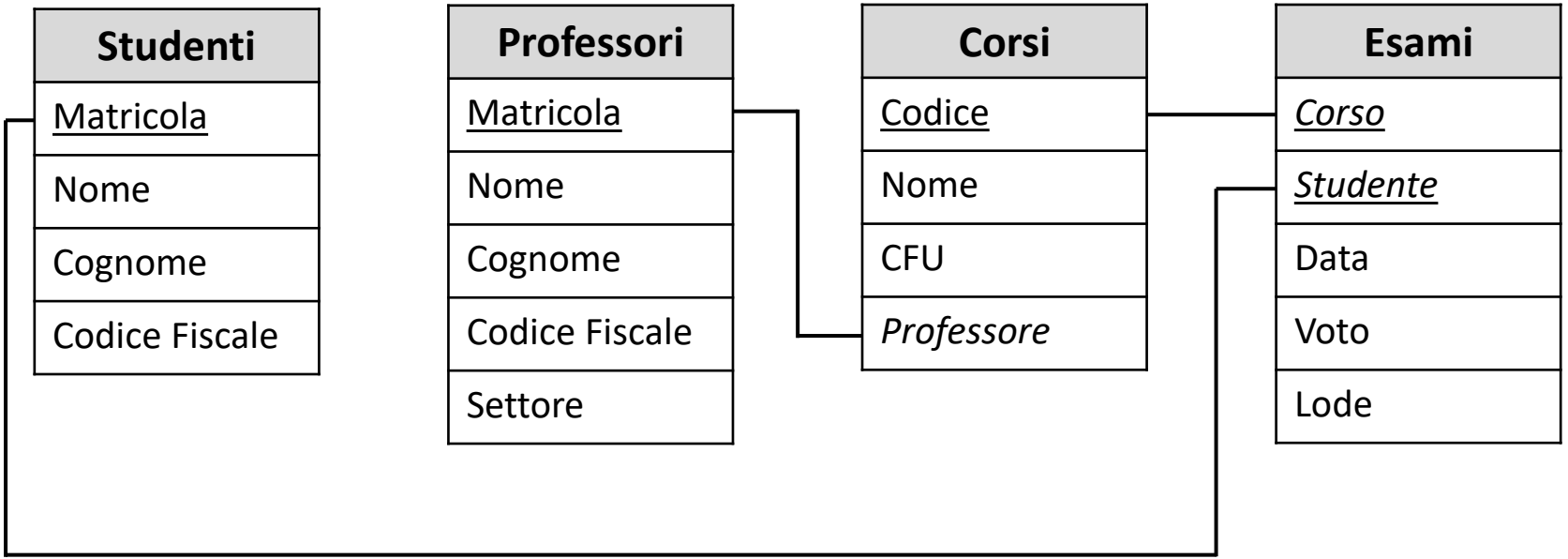
Edificio: C3, Ufficio: C3.232
(contattatemi prima via mail)

https://github.com/giovannipinna96/DB_2024_Exercises

Sommario

1. Riassunto della prima esercitazione
2. Altre query
3. Prepared statement
4. Query su viste

Database (molto semplice) dell'Università



Riassunto della prima esercitazione

Creazione e popolamento del DB

Query sul DB creato:

Query 1: elencare tutti gli studenti iscritti ad ingegneria

- LIKE

Query 2: elencare tutte le studentesse iscritte ad ingegneria

- LIKE, BETWEEN, IN, SUBSTRING
- Aggiunta di una colonna «*Genere*» per facilitare le operazioni future

Query 3: quanti studenti hanno preso una lode con il prof. De Lorenzo?

- DISTINCT, COUNT, INNER JOIN

Query 4: elencare quanti 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
- Crea un nuovo DB
- Lo popola

File ***query_prima_esercitazione.sql**:*

- Contiene il codice per modificare il DB (colonna *«Genere»*)
- Contiene le query della prima esercitazione

Se ti sei perso l'esercitazione precedente?

- esegui **uni_db.sql** e **query_prima_esercitazione.sql**

Se hai già creato il DB la volta scorsa

- USE uni_db;

Per scrivere codice SQL

Per chi non ha MySQL o MySQL Workbench…
Esistono questi editor online:

- https://onecompiler.com/studio
- 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 5

Calcolare la media ponderata per lo studente IN0500273.

$$Media\ ponderata = \frac{\sum_{i=1}^{\#esami} voto_i * CFU_i}{\sum_{i=1}^{\#esami} CFU_i}$$

Query 5

```sql
--Query 5: media ponderata per uno specifico studente
SELECT
    e.studente,
    SUM(e.voto * c.cfu) / SUM(c.cfu) AS media_ponderata
FROM esami e
    INNER JOIN corsi c ON e.corso = c.codice
WHERE e.studente = 'IN0500273'
GROUP BY e.studente;
```

Query 5.1

Mostrare la media ponderata di ogni studente (mostrate matricola, nome e cognome di ogni studente)

Query 5.1

Mostrare la media ponderata di ogni studente (mostrate matricola, nome e cognome di ogni studente)

```sql
--Query 5.1: mostrare la media ponderata di ogni studente
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 6

Quali studenti non hanno mai preso una lode?

Query 6

Quali studenti non hanno mai preso una lode?

Suggerimento: sono quegli studenti per cui non esiste un esame con la lode

Query 6

Quali studenti non hanno mai preso una lode?

Suggerimento: sono quegli studenti per cui non esiste un esame con la lode

```sql
--Query 6: quanti studenti non hanno mai preso una lode?
SELECT *
FROM studenti s
WHERE
    NOT EXISTS (
        SELECT *
        FROM esami e
        WHERE e.lode = TRUE AND e.studente = s.matricola
    );
```

Query 6 – alternativa

Quali studenti non hanno mai preso una lode?

Suggerimento: sono quegli studenti per cui non esiste un esame con la lode

```sql
--Query 6 alternativa
SELECT *
FROM studenti s
WHERE s.matricola NOT IN(
    SELECT DISTINCT studente
    FROM esami e
    WHERE e.lode = TRUE
);
```

Query 7

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

Query 7

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

Suggerimento: un CFU corrispondono a 8 ore di lezione

Query 7

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

Suggerimento: un CFU corrispondono a 8 ore di lezione

```sql
-- Query 7: Quali docenti svolgono un monte ore minore di 120 ore?
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 8

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

Query 8

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

```sql
--Query 8: Verificare 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  PREPARE nomeStatement FROM
2  "la mia query passata come stringa
3  dove il '?' corrisponde al parametro che verrà comunicato
4  in sede di esecuzione";
```

```
1  EXECUTE nomeStatement
2      USING @var1, @var2 ... ;
```

Prepared statement 1

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

Prepared statement 1

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

Suggerimento: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente

Prepared statement 1

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

Suggerimento: il corso di laurea corrisponde ai primi 4 caratteri della matricola dello studente

```
1  --Prepared statement 1:
2  --Creare un prepared statement che mostri tutti gli studenti appartenenti
   ad un corso di laurea passato come parametro
3  PREPARE studenti_cdl FROM
4      "SELECT *
5      FROM studenti
6      WHERE matricola LIKE CONCAT(?, '%')";
```

Prepared statement 1

Usiamo lo *statement* per mostrare gli studenti del corso di ingegneria informatica della triennale (IN05):

```
1  --usiamo lo statement per mostrare gli ingegneri informatici della triennale
2  SET @cdl = "IN05";
3  EXECUTE studenti_cdl USING @cdl;
```

Prepared statement 2

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

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.

```
1  --PREPARED statement 2
2  --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
3  PREPARE studenti_superato_corso FROM
4      "SELECT s.nome, s.cognome, s.matricola
5      FROM studenti s
6      INNER JOIN esami e
7      ON s.matricola = e.studente
8      WHERE e.corso = ?";
```

Prepared statement 2

Usiamo lo *statement* per mostrare gli studenti che hanno passato il corso di basi di dati (079IN):

```
1  -- usiamo lo statement per mostrare gli studenti
2  -- che hanno passato il corso di basi di dati (079IN)
3  SET @codice_corso = "079IN";
4  EXECUTE studenti_superato_corso USING @codice_corso;
```

Vista

```
1   CREATE VIEW nomeView AS
2       SELECT ...
3       FROM ...
4       ... ;
```

```
1   SELECT * FROM nomeView;
```

Vista 1

Quali sono i voti preferiti di ogni professore?

Vista 1

Quali sono i voti preferiti di ogni professore?

Suggerimento:

1. creare prima una vista che mostri la distribuzione dei voti per ogni docente.

2. Poi scegliamo il voto più frequente per ogni docente

Vista 1

Quali sono i voti preferiti di ogni professore?

```sql
--Vista 1: Quali sono i voti preferti di ogni professore?
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

Scegliamo il voto più frequente per ogni studente

```sql
--ora scgliamo 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

Quali sono i gli studenti più bravi per ogni corso di laurea?

Suggerimento:

1. possiamo definire la «bravura» come somma dei voti moltiplicati per i rispettivi CFU, ma anche altre metriche sono possibili.

Vista 2

Quali sono i gli studenti più bravi per ogni corso di laurea?

Suggerimento:

1. possiamo definire la «bravura» come somma dei voti moltiplicati per i rispettivi CFU, ma anche altre metriche sono possibili.
2. poi possiamo scegliere lo studente più bravo di ciascun corso

Vista 2

Quali sono i gli studenti più bravi per ogni corso di laurea?

```sql
--Vista 2: quali sono gli studenti più "bravi" di ogni corso di laurea?
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

Scegliamo il più bravo di ciascun corso

```sql
--scgliamo 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
);
```

Grazie !