Terza Esercitazione di Basi di Dati

Giorgia Nadizar

3 maggio 2022

Parte 1

Riepilogo

Database dell'Università (completo)

Studenti

Matricola	Nome	Cognome	Codice Fiscale

- Nome, Cognome e Codice Fiscale non devono essere `NULL`
- Codice Fiscale dev'essere `UNIQUE`

Professori

Matricola	Nome	Cognome	Codice Fiscale	Settore

- Nome, Cognome, Codice Fiscale e Settore non devono essere `NULL`
- Codice Fiscale dev'essere `UNIQUE`

Corsi

Codice	Nome	CFU	Professore [Matricola]

- Nome e CFU non devono essere `NULL`

Esami

Corso [Codice]	Studente [Matricola]	Data	Voto	Lode

- Voto dev'essere tra 18 e 30
- Lode non può essere `TRUE` se il voto non è 30

Tabelle nel DB

studenti(matricola, nome, cognome, cf)

professori(matricola, nome, cognome, cf, settore)

corsi(codice, nome, cfu, professore)

esami(corso, studente, data, voto, lode)

Tabelle nel DB

studenti(matricola, nome, cognome, cf)

professori(matricola, nome, cognome, cf, settore)

corsi(codice, nome, cfu, professore)

esami(corso, studente, data, voto, lode)

Abbiamo già il DB pronto dalle esercitazioni precedenti, ma ricordiamoci di *usarlo*:

```
USE uni_db;
```

Parte 2

Soluzioni degli esercizi per casa

Query

1. Quali prof. hanno una media voti più bassa del normale agli esami?
2. Quanti esami sono stati svolti per ciascun anno? E per ciascun mese dell'anno?
3. Ci sono casi di omonimia (nome e cognome uguale) tra studenti e/o professori?
4. Quanti studenti e professori ci sono con lo stesso nome? (es. 9 persone si chiamano Luca, 11 si chiamano Matteo, ...)

Prepared Statement

1. Creare uno statement che mostri tutti gli esami (con voto e nome del corso) di uno studente passato come parametro (come parametro ci aspettiamo la matricola dello studente)
2. Creare uno statement che restituisca la media aritmetica e la media ponderata degli esami sostenuti in un determinato mese di un determinato anno (mese ed anno sono passati come parametri)
3. Creare uno statement che restituisca tutti gli studenti (nome, cognome e matricola) che hanno sostenuto l'esame di un dato corso (passato come parametro, tramite codice)

Query Complesse (da risolvere tramite Viste)

1. Quali sono i voti preferiti di ogni professore?
2. Quali sono gli studenti più bravi di ogni corso di laurea?
3. Quali studenti hanno migliorato almeno una volta la loro media nel corso della loro carriera universitaria? (es. Gianni nel 2016 ha la media del 30, nel 2017 la media del 22, e nel 2018 la media del 25: Gianni è migliorato almeno una volta. Supponiamo per semplicità che la media nell'anno x sia data dai soli esami svolti quell'anno.)

Query 1

Quali prof. hanno una media voti più bassa del normale agli esami?

Query 1

Quali prof. hanno una media voti più bassa del normale agli esami?

```
SELECT c.professore, AVG(e.voto) as media
FROM esami e
INNER JOIN corsi c
ON e.corso = c.codice
GROUP BY c.professore
HAVING media < (
SELECT AVG(voto)
FROM esami);
```

Query 2

Quanti esami sono stati svolti per ciascun anno? E per ciascun mese dell'anno?

Query 2

Quanti esami sono stati svolti per ciascun anno? E per ciascun mese dell'anno?

```
SELECT YEAR(e.data) as anno, COUNT(*)
FROM esami e
GROUP BY anno
ORDER BY anno;
```

Query 2

Quanti esami sono stati svolti per ciascun anno? E per ciascun mese dell'anno?

```
SELECT YEAR(e.data) as anno, COUNT(*)
FROM esami e
GROUP BY anno
ORDER BY anno;
```

```
SELECT MONTH(e.data) as mese, COUNT(*)
FROM esami e
GROUP BY mese
ORDER BY mese;
```

Query 3

Ci sono casi di omonimia (nome e cognome uguale) tra studenti e/o professori?

Query 3

Ci sono casi di omonimia (nome e cognome uguale) 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;
```

Query 4

Quanti studenti e professori ci sono con lo stesso nome? (es. 9 persone si chiamano Luca, 11 si chiamano Matteo, ...)

Query 4

Quanti studenti e professori ci sono con lo stesso nome? (es. 9 persone si chiamano Luca, 11 si chiamano Matteo, ...)

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

Prepared Statement 1

Creare uno statement che mostri tutti gli esami (con voto e nome del corso) di uno studente passato come parametro (come parametro ci aspettiamo la matricola dello studente)

Prepared Statement 1

Creare uno statement che mostri tutti gli esami (con voto e nome del corso) di uno studente passato come parametro (come parametro ci aspettiamo la matricola dello studente)

```
PREPARE esami_con_voto FROM
"SELECT e.corso, c.nome, e.data, e.voto, e.lode
FROM esami e
INNER JOIN corsi c
ON e.corso = c.codice
WHERE e.studente = ?";
```

Prepared Statement 2

Creare uno statement che restituisca la media aritmetica e la media ponderata degli esami sostenuti in un determinato mese di un determinato anno (mese ed anno sono passati come parametri)

Prepared Statement 2

Creare uno statement che restituisca la media aritmetica e la media ponderata degli esami sostenuti in un determinato mese di un determinato anno (mese ed anno sono passati come parametri)

```
PREPARE media_esami_mese_anno FROM
"SELECT SUM(e.voto*c.cfu)/SUM(c.cfu) as mp,
AVG(e.voto) as ma
FROM esami e
INNER JOIN corsi c ON e.corso = c.codice
WHERE MONTH(e.data)=? AND YEAR(e.data)=?";
```

Prepared Statement 3

Creare uno statement che restituisca tutti gli studenti (nome, cognome e matricola) che hanno sostenuto l'esame di un dato corso (passato come parametro, tramite codice)

Prepared Statement 3

Creare uno statement che restituisca tutti gli studenti (nome, cognome e matricola) che hanno sostenuto l'esame di un dato corso (passato come parametro, tramite codice)

```
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=?";
```

Query Complesse 1

Quali sono i voti preferiti di ogni professore?

Query Complesse 1

Quali sono i voti preferiti di ogni professore?

- *Hint 1:* spezziamo la query in due parti

Query Complesse 1

Quali sono i voti preferiti di ogni professore?

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando la distribuzione dei voti per ogni docente

Query Complesse 1

Quali sono i voti preferiti di ogni professore?

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando 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;
```

Query Complesse 1

- *Hint 3:* a questo punto scegliamo il voto più frequente per ogni docente

Query Complesse 1

- *Hint 3:* 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
);
```

Query Complesse 2

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

Query Complesse 2

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

- *Hint 1:* spezziamo la query in due parti

Query Complesse 2

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

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando la *bravura* di ogni studente (es. somma di crediti moltiplicati cfu, ma potremmo usare altre metriche...)

Query Complesse 2

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

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando la *bravura* di ogni studente (es. somma di crediti moltiplicati cfu, ma potremmo usare altre metriche...)

```
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;
```

Query Complesse 2

- *Hint 3:* a questo punto scegliamo il più bravo di ciascun corso

Query Complesse 2

- *Hint 3:* 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
);
```

Query Complesse 3

Quali studenti hanno migliorato almeno una volta la loro media nel corso della loro carriera universitaria?

Query Complesse 3

Quali studenti hanno migliorato almeno una volta la loro media nel corso della loro carriera universitaria?

- *Hint 1:* spezziamo la query in due parti

Query Complesse 3

Quali studenti hanno migliorato almeno una volta la loro media nel corso della loro carriera universitaria?

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando la media ponderata degli studenti per ogni anno

Query Complesse 3

Quali studenti hanno migliorato almeno una volta la loro media nel corso della loro carriera universitaria?

- *Hint 1:* spezziamo la query in due parti
- *Hint 2:* iniziamo calcolando la media ponderata degli studenti per ogni anno

```
CREATE VIEW media_per_anno AS
SELECT s.matricola, s.nome, s.cognome,
SUM(e.voto*c.cfu)/SUM(c.cfu) as media, YEAR(e.
   data) as anno
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, anno;
```

Query Complesse 3

- *Hint 3:* a questo punto scegliamo tutti quelli la cui media è salita almeno una volta

Query Complesse 3

- *Hint 3:* a questo punto scegliamo tutti quelli la cui media è salita almeno una volta

```
SELECT DISTINCT matricola, nome, cognome
FROM media_per_anno m1
WHERE m1.media > (
  SELECT MIN(media)
    FROM media_per_anno m2
    WHERE m1.matricola=m2.matricola AND
    m1.anno > m2.anno
);
```

Parte 3

Esercizi in aula

Transazione

Scrivere una transazione che assegni al prof. meno impegnato l'unico corso scoperto

Transazione

Scrivere una transazione che assegni al prof. meno impegnato l'unico corso scoperto

- *Hint 1:* svolgiamo prima le query di prova

Transazione

Scrivere una transazione che assegni al prof. meno impegnato l'unico corso scoperto

- *Hint 1:* svolgiamo prima le query di prova
- *Hint 2:* qual è il corso scoperto?

Transazione

Scrivere una transazione che assegni al prof. meno impegnato l'unico corso scoperto

- *Hint 1:* svolgiamo prima le query di prova
- *Hint 2:* qual è il corso scoperto?

```
SELECT *
FROM corsi c
WHERE professore IS NULL;
```

Transazione

- *Hint 3:* qual è il prof. meno impegnato?

Transazione

- *Hint 3:* qual è il prof. meno impegnato?

```
SELECT matricola, sum(cfu) as cfu_tot
FROM professori p
INNER JOIN corsi c
ON c.professore=p.matricola
GROUP BY professore
ORDER BY cfu_tot asc
LIMIT 1;
```

Transazione

```
START TRANSACTION;

SELECT @prof := matricola, sum(cfu) as cfu_tot
FROM professori p
INNER JOIN corsi c
ON c.professore=p.matricola
GROUP BY professore
ORDER BY cfu_tot asc
LIMIT 1;

UPDATE corsi
SET professore = @prof
WHERE professore IS NULL;

COMMIT;
```

Stored Procedure 1

Scrivere una stored procedure che restituisca le medie ponderate ed aritmetiche di tutti gli studenti

Stored Procedure 1

Scrivere una stored procedure che restituisca le medie ponderate ed aritmetiche di tutti gli studenti

```
DELIMITER $$
CREATE PROCEDURE CalcoloMedie()
BEGIN
  SELECT s.matricola, s.nome, s.cognome,
  SUM(e.voto*c.cfu)/SUM(c.cfu) as mp,
  AVG(e.voto) as ma
  FROM studenti s INNER JOIN esami e
  ON s.matricola = e.studente
    INNER JOIN corsi c ON e.corso = c.codice
  GROUP BY e.studente;
END $$
DELIMITER ;
```

Stored Procedure 2

Scrivere una stored procedure che restituisca gli studenti con la media superiore ad un dato voto

Stored Procedure 2

Scrivere una stored procedure che restituisca gli studenti con la media superiore ad un dato voto

```
DELIMITER $$
CREATE PROCEDURE StudentiConMediaSopraSoglia(IN
    soglia INT)
BEGIN
  SELECT s.matricola, s.nome, s.cognome,
  SUM(e.voto*c.cfu)/SUM(c.cfu) as mp
  FROM studenti s
  INNER JOIN esami e ON s.matricola = e.studente
  INNER JOIN corsi c ON e.corso = c.codice
  GROUP BY e.studente HAVING mp >= soglia;
END $$
DELIMITER ;
```

Stored Procedure 3

Scrivere una stored procedure che restituisca in una variabile passata il numero di studenti che hanno sostenuto almeno un esame

Stored Procedure 3

Scrivere una stored procedure che restituisca in una variabile passata il numero di studenti che hanno sostenuto almeno un esame

```
DELIMITER $$
CREATE PROCEDURE NStudentiConEsami(OUT numero INT
    )
BEGIN
  SELECT COUNT(DISTINCT matricola)
    INTO numero
    FROM studenti s
    INNER JOIN esami e
    ON s.matricola = e.studente;
END $$
DELIMITER ;
```

Stored Procedure 3 (esecuzione)

```
set @numero = 0;

CALL NStudentiConEsami(@numero);

select @numero;
```

Stored Procedure 4

Scrivere una stored procedure che restituisca in una variabile passata il monte di ore di un dato docente (se il docente non esiste bisogna lanciare un errore)

Stored Procedure 4

```
DELIMITER $$
CREATE PROCEDURE MonteOre(IN docente INT, OUT ore
    INT)
BEGIN
  SELECT SUM(cfu*8)
    INTO ore
    FROM corsi c
    WHERE professore=docente;
    IF ore IS NULL THEN
        SIGNAL SQLSTATE "02000"
        SET MESSAGE_TEXT = "Docente not found!";
    END IF;
END $$
DELIMITER ;
```

Riassunto

Transazione

1. Scrivere una transazione che assegni al prof. meno impegnato l'unico corso scoperto

Stored Procedures

1. Scrivere una SP che restituisca le medie aritmetiche e ponderate di tutti gli studenti
2. Scrivere una SP che restituisca gli studenti con la media superiore ad un dato voto
3. Scrivere una SP che restituisca in una variabile passata il numero di studenti che hanno sostenuto almeno un esame
4. Scrivere una SP che restituisca in una variabile passata il monte di ore di un dato docente (se il docente non esiste bisogna lanciare un errore)