SQL: query annidate

VIDEOCONFERENZA 06/05/2020: **QUERY ANNIDATE COMPLESSE, VISTE, VINCOLI DI INTEGRITA' GENERICI, OPERAZIONI DI AGGIORNAMENTO**

Docente: CHIARA DAMIANI chiara.damiani@unimib.it

Materiale propedeutico da e-learning

- 8.11_regole di visibilità_nelle espressioni nidificate
- 8.12 operazioni di aggiornamento e vincoli di integrità
- 8.13 metodo generale applicabile per la costruzione di interrogazioni

QUERY ANNIDATE

Una subquery si può trovare nei seguenti punti dei seguenti statement:

- Nelle clausole **WHERE** e **HAVING** dello statement **SELECT**
- Nella clausola **FROM** dell'istruzione **SELECT**
- Nella clausola **WHERE** di un **INSERT INTO, DELETE FROM, UPDATE**
- Nella clausola **SET** di un comando **UPDATE**
- Nella clausola **CHECK** di un comando **DDL**

Una subquery puo' fare uso a sua volta di altre subquery.

Dove eravamo rimasti

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 16: trovare lo studente (matricola) con la media ponderata più alta

```
SELECT  Matricola_studente,
sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
HAVING media_ponderata >=  ALL (
            SELECT
            sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
            media_ponderata
            FROM Esame E, Corso C
            WHERE E.Codice_corso=C.Codice_c
            GROUP BY Matricola_studente )
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 20: trovare la media ponderata massima

Poll 5: La seguente soluzione è corretta?

```
SELECT
MAX(sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione)) AS
MaxMedia
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
```

NO. Ci sono due livelli di aggregazione. La media ponderata deve essere calcolata sui gruppi (il singolo studente), mentre il massimo deve essere calcolato su tutte le righe (le medie degli studenti).

Quello che vorremmo fare

Voti e CFU esami

Matr	Voto	CFU
1497001	19	8
1497001	26	3
1497001	30	5
1497001	30	6
1515801	30	5
1515801	24	3
1515801	29	6
1515801	20	6
1515801	19	8
1524501	19	8
1524501	29	6
1524501	31	8
1524501	29	6

Media di ogni studente

media_ponderata
26.7143
23.8571
25.4545

Massimo delle medie

media_ponderata
26.7143

Quello che stiamo facendo

Voti e CFU esami

Matr	Voto	CFU
1497001	19	8
1497001	26	3
1497001	30	5
1497001	30	6
1515801	30	5
1515801	24	3
1515801	29	6
1515801	20	6
1515801	19	8
1524501	19	8
1524501	29	6
1524501	31	8
1524501	29	6

Massimo della Media

Max(Sum(Voto*CFU))
?

Di ogni studente?

```
SELECT
MAX(sum(Voto*(Crediti_lezione+Crediti_esercitazione))/su
m(Crediti_lezione+Crediti_esercitazione)) AS MaxMedia
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
```

Una possibile soluzione

Voti e CFU esami

Matr	Voto	CFU
1497001	19	8
1497001	26	3
1497001	30	5
1497001	30	6
1515801	30	5
1515801	24	3
1515801	29	6
1515801	20	6
1515801	19	8
1524501	19	8
1524501	29	6
1524501	31	8
1524501	29	6

Media di ogni studente

media_ponderata
26.7143
23.8571
25.4545

>= ALL

Media di ogni studente

media_ponderata
26.7143
23.8571
25.4545

Massimo della Media

media_ponderata
26.7143

Voti e CFU esami

Matr	Voto	CFU
1497001	19	8
1497001	26	3
1497001	30	5
1497001	30	6
1515801	30	5
1515801	24	3
1515801	29	6
1515801	20	6
1515801	19	8
1524501	19	8
1524501	29	6
1524501	31	8
1524501	29	6

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 20: trovare la media ponderata massima

Una possibile soluzione

```
SELECT  sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
HAVING media_ponderata >=ALL (
        SELECT
        sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
        media_ponderata
        FROM Esame E, Corso C
        WHERE E.Codice_corso=C.Codice_c
        GROUP BY Matricola_studente)
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 20: trovare la media ponderata massima

Una soluzione più leggibile

```
SELECT max(media_ponderata)
FROM (   SELECT
         sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
         media_ponderata
         FROM Esame E, Corso C
         WHERE E.Codice_corso=C.Codice_c
         GROUP BY Matricola_studente) AS media_voti
```

Per potere usare l'operatore aggregato **MAX** ci serve una sottoquery nella clausola **FROM**

La ridenominazione è necessaria

Una soluzione più leggibile

Voti e CFU esami

Matr	Voto	CFU
1497001	19	8
1497001	26	3
1497001	30	5
1497001	30	6
1515801	30	5
1515801	24	3
1515801	29	6
1515801	20	6
1515801	19	8
1524501	19	8
1524501	29	6
1524501	31	8
1524501	29	6

Media di ogni studente

media_ponderata
26.7143
23.8571
25.4545

```
SELECT
sum(Voto*(Crediti_lezione+Crediti_esercitazione))
/sum(Crediti_lezione+Crediti_esercitazione)  AS media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
```

Massimo delle medie

media_ponderata
26.7143

```
SELECT max(media_ponderata)
FROM () AS media_voti
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 20: trovare la media ponderata massima

```
CREATE VIEW media_voti AS (
        SELECT
        sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
        media_ponderata
        FROM Esame E, Corso C
        WHERE E.Codice_corso=C.Codice_c
        GROUP BY Matricola_studente)


SELECT max(media_ponderata)
FROM media_voti
```

La sottoquery nella clausola **FROM** permette la stessa espressività delle **VISTE** senza modificare lo schema della BD.

Dove eravamo rimasti

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 16: trovare lo studente (matricola) con la media ponderata più alta

```
SELECT  Matricola_studente,
sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
HAVING media_ponderata >=  ALL (
            SELECT    sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione)
            AS        media_ponderata
            FROM Esame E, Corso C
            WHERE E.Codice_corso=C.Codice_c
            GROUP BY Matricola_studente )
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 16: trovare lo studente (matricola) con la media ponderata più alta

Una soluzione forse più leggibile

```
SELECT  Matricola_studente,
sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
HAVING media_ponderata = (SELECT max(media_ponderata)
        FROM (    SELECT
        sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS
        media_ponderata
        FROM Esame E, Corso C
        WHERE E.Codice_corso=C.Codice_c
        GROUP BY Matricola_studente) AS media_voti)
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 16: trovare lo studente (matricola) con la media ponderata più alta

Si valuta prima la query più interna e poi quella più esterna

```
SELECT  Matricola_studente,
sum(Voto*(Crediti_lezione+Crediti_esercitazione))/sum(Crediti_lezione+Crediti_esercitazione) AS media_ponderata
FROM Esame E, Corso C
WHERE E.Codice_corso=C.Codice_c
GROUP BY Matricola_studente
HAVING media_ponderata = 26.714
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno preso più di 25 in tutti gli esami sostenuti

Ovvero studenti che non hanno mai preso meno di 25

```
SELECT DISTINCT Matricola_St, Nome, Cognome
FROM Studente S JOIN  Esame E ON E.Matricola_studente=S.Matricola_st
WHERE Matricola_St NOT IN
        (SELECT Matricola_Studente
        FROM Esame
        WHERE Voto < 25)
```

Si valuta prima la query più interna e poi quella più esterna

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno preso più di 25 in tutti gli esami sostenuti

Ovvero studenti che non hanno mai preso meno di 25

oppure

```
SELECT DISTINCT Matricola_St, Nome, Cognome
FROM Studente S JOIN  Esame E ON E.Matricola_studente=S.Matricola_st
WHERE NOT EXISTS (
      SELECT *
      FROM Esame
      WHERE  Matricola_St= Matricola_Studente and Voto < 25)
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno sempre preso più di 25 in un esame

Ovvero studenti che non hanno mai preso meno di 25

oppure

```
SELECT DISTINCT Matricola_St, Nome, Cognome
FROM Studente S JOIN  Esame E ON E.Matricola_studente=S.Matricola_st
WHERE NOT EXISTS (
      SELECT *
      FROM Esame
      WHERE  S.Matricola_St= Matricola_Studente and Voto < 25)
```

PASSAGGIO DI BINDING: Per ogni tupla del blocco esterno, considera il valore di **S.Matricola_St** e risolvi la query innestata

L'operatore esistenziale EXISTS/NOT EXISTS ha senso solo nel caso in cui la query interna faccia riferimento alla query esterna

Se la subquery fa riferimento al blocco esterno, allora si dice correlata

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno sempre preso più di 25 in un esame

Ovvero studenti che non hanno mai preso meno di 25

oppure

```
SELECT DISTINCT Matricola_St, Nome, Cognome
FROM Studente S JOIN Esame E ON E.Matricola_studente=S.Matricola_st
WHERE NOT EXISTS (
      SELECT *
      FROM Esame
      WHERE S.Matricola_St= Matricola_Studente and Voto < 25)
```

PASSAGGIO DI BINDING: Per ogni tupla del blocco esterno, considera il valore di **S.Matricola_St** e risolvi la query innestata

Si valuta la query interna per ogni tupla della query esterna

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno preso più di 25 in tutti gli esami sostenuti

Possiamo ottenere lo stesso risultato usando invece l'operatore ALL

```
SELECT distinct Matricola_St, Nome, Cognome
FROM Studente S
WHERE 25 <= ALL
        (SELECT voto
        FROM Esame E
        WHERE S.Matricola_St =E.Matricola_studente)
```

Si valuta la query interna per ogni tupla della query esterna

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 21: studenti che hanno preso più di 25 in tutti gli esami sostenuti. Nel caso in cui l'attributo voto possa assumere valori NULL

oppure

```
SELECT DISTINCT Matricola_St, Nome, Cognome
FROM Studente S JOIN  Esame E ON E.Matricola_studente=S.Matricola_st
WHERE Matricola_St NOT IN
        (Select Matricola_Studente
        FROM Esame
        WHERE Voto < 25 OR Voto IS NULL)
```

Se non eliminiamo le tuple con i valori nulli non possiamo escludere che il voto sia maggiore di 25

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno sostenuto tutti gli esami previsti dal loro piano di studi

AVETE PROVATO A FARLO A CASA?

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

Poll 6: La soluzione proposta da Pippo è corretta?

```
SELECT S.Matricola_St, Nome, Cognome
FROM Studente S, Esame E, Piano_Di_Studio PS
WHERE E.Matricola_Studente = S.Matricola_St AND PS.Matricola_St = S.Matricola_St AND PS.Codice_Corso
= ALL (
        SELECT Codice_Corso
        FROM Esame)
```

NO non è corretta

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

Poll 6: La soluzione proposta da Pippo è corretta?

```
SELECT S.Matricola_St, Nome, Cognome
FROM Studente S, Esame E, Piano_Di_Studio PS
WHERE E.Matricola_Studente = S.Matricola_St AND PS.Matricola_St = S.Matricola_St AND PS.Codice_Corso
= ALL (
        SELECT Codice_Corso
        FROM Esame)
```

Nella query interna non c'è nessuna selezione, quindi restituisce TUTTE le tuple della tabella Esame

Il predicato **= ALL** è vero solo se una tupla della query esterna è uguale a **TUTTE** le tuple della query interna.
Perché ciò accada tutte le tuple della query interna devono essere identiche.

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

Poll 6: La soluzione proposta da Pippo è corretta?

```
SELECT S.Matricola_St, Nome, Cognome
FROM Studente S, Esame E, Piano_Di_Studio PS
WHERE E.Matricola_Studente = S.Matricola_St AND PS.Matricola_St = S.Matricola_St AND PS.Codice_Corso
= ALL (
        SELECT Codice_Corso
        FROM Esame)
```

Il theta-join restituisce gli esami che appartengono ai piani di studio degli studenti

Nella query interna non c'è nessuna selezione, quindi restituisce TUTTE le tuple della tabella Esame

Il predicato **= ALL** è vero solo se una tupla della query esterna è uguale a **TUTTE** le tuple della query interna.
Perché ciò accada tutte le tuple della query interna devono essere identiche.

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

Equivale a: Quali sono gli studenti che verificano**:**
Per ogni corso nel piano di studi, tale corso è presente negli esami sostenuti?

In SQL non esiste l'operatore di **quantificazione universale ∀**

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

Possiamo riformulare come: Quali sono gli studenti che verificano**:**
Non esiste un corso nel piano di studi, **non presente** negli esami sostenuti?

Le subquery permettono di esprimere la quantificazione universale per mezzo di una **doppia negazione.**

Per ogni x vale la proprietà P equivale a dire: Non esiste alcun x per cui non vale la proprietà P

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

```
SELECT DISTINCT Matricola_st
FROM Piano_di_studio P1
WHERE NOT EXISTS (
        SELECT *
        FROM Piano_di_studio P2
        WHERE P1.Matricola_St=P2.Matricola_St and P2.Codice_corso NOT IN (
                SELECT Codice_e
                FROM Esame E2
                WHERE E2.Matricola_studente=P1.Matricola_st))
```

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 19: studenti che hanno dato tutti gli esami previsti dal loro piano di studi

```
SELECT DISTINCT Matricola_st
FROM Piano_di_studio P1
WHERE NOT EXISTS (
        SELECT *
        FROM Piano_di_studio P2
        WHERE P1.Matricola_St=P2.Matricola_St and NOT EXISTS (
                SELECT *
                FROM Esame E
                WHERE E.Matricola_studente=P1.Matricola_st and E.Codice_e=P2.Codice_corso))
```

Semantica equivalente.

Dato il DDL della tabella Esame

```
CREATE TABLE `Esame` (
  `Codice_e` int NOT NULL,
  `Matricola_studente` int NOT NULL,
  `Codice_corso` varchar(10) NOT NULL,
  `Data` datetime DEFAULT NULL,
  `Voto` int NOT NULL,
  PRIMARY KEY (`Codice_e`),
  FOREIGN KEY (`Codice_corso`) REFERENCES `Corso` (`Codice_c`) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (`Matricola_studente`) REFERENCES `Studente` (`Matricola_st`) ON DELETE CASCADE ON UPDATE
CASCADE,
....
)
```

Esercizio 21: inserire il vincolo che il voto esame deve essere compreso tra 18 e 31

CHECK (voto <31 **&** voto >18)

Nel DDL della tabella Studente

```
CREATE TABLE `Studente` (
  `Matricola_st` int NOT NULL,
  `Cognome` varchar(45) NOT NULL,
  `Nome` varchar(45) NOT NULL,
  `Corso_di_Laurea` varchar(6) NOT NULL,
  PRIMARY KEY (`Matricola_st`),
....
)
```

Esercizio 22: inserire il vincolo che il numero di matricola non deve essere già stato assegnato ad un docente

```
CHECK (matricola_st NOT IN (
        SELECT matricola_d
        FROM Personale_docente)
```

Le subquery nella clausola check sono previste dallo standard, ma non supportate dai vari DBMS e in particolare da MYSQL

Studente(matricola_st,Cognome,Nome, corso_di_Laurea)
Personale_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Personale_non_docente(matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)

Esercizio 22: inserire il vincolo che il numero di matricola deve essere unico per le varie persone (siano esse studenti, personale_docente, personale_non_docente

```
CREATE ASSERTION MatricolaUnica CHECK (
        NOT EXISTS (
                SELECT Matricola_st FROM Studente
                INTERSECT
                SELECT Matricola_d FROM Personale_docente
                INTERSECT
                SELECT Matricola_d FROM Personale_non_docente  ))
```

Le asserzioni sono previste dallo standard,
ma supportate dai pochi DBMS e in particolare NON da MYSQL

TRIGGER

```
create trigger NomeTrigger
  { before | after }
  { insert | delete | update [of Column] } on TabellaTarget
  [referencing
        {[old table [as] VarTuplaOld]
         [new table [as] VarTuplaNew] } |
        {[old [row] [as] VarTabellaOld]
         [new [row] [as] VarTabellaNew] }]
  [for each { row | statement }]
  [when Condizione]
  StatementProceduraleSQL
```

Nella realtà dei fatti dovremo specificare questo tipi di vincolo utilizzando i trigger, che tuttavia non affronteremo in questo corso

Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(Matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Stipendio(Classe,Valore)

Esercizio 23: eliminare i docenti che non hanno corsi

```
DELETE FROM  Personale_Docente
WHERE Matricola_d
NOT IN (
        SELECT Docente
        FROM Corso)
```

Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(Matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Stipendio(Classe,Valore)

Esercizio 24: assegnare la classe di stipendio massima ai docenti titolari di più di 4 corsi

Identifichiamo prima l'insieme dei docenti titolari di più di 4 corsi

```
SELECT Docente
FROM Corso
GROUP BY Docente
HAVING count(*)>4
```

Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(Matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Stipendio(Classe,Valore)

Esercizio 24: assegnare la classe di stipendio massima ai docenti titolari di più di 4 corsi

Poi andiamo a modificare lo stipendio dei docenti che vi appartengono assumendo di sapere che la classe di stipendio massima è la 15

```
UPDATE Personale_docente
SET Classe_Stipendio=15
WHERE Matricola_d IN (
        SELECT Docente
        FROM Corso
        GROUP BY Docente
        HAVING count(*)>4)
```

Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(Matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Stipendio(Classe,Valore)

Esercizio 24: assegnare la classe di stipendio massima ai docenti titolari di più di 4 corsi

Cerchiamo ora la classe massima. Ricordiamoci che il dato «classe» è di tipo alfanumerico quindi dobbiamo associarla al valore corrispondente.

```
SELECT Classe
FROM Stipendio
WHERE valore = (
        SELECT max(Valore)
        FROM Stipendio))
```

Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Personale_docente(Matricola_d,Cognome,Nome, Ruolo, Classe_stipendio)
Stipendio(Classe,Valore)

Esercizio 24: assegnare la classe di stipendio massima ai docenti titolari di più di 4 corsi

Infine scriviamo l'istruzione di aggiornamento completa

```
UPDATE Personale_docente
SET Classe_Stipendio=(
        SELECT Classe
        FROM Stipendio
        WHERE valore = (
                SELECT max(Valore)
                FROM Stipendio))
WHERE Matricola_d IN (
        SELECT Docente
        FROM Corso
        GROUP BY Docente
        HAVING count(*)>4)
```

Studente(Matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Fuori_Corso (Matricola_st,Cognome,Nome, corso_di_Laurea))

Esercizio 25: inserire gli studenti che non danno esami dal 98 nella tabella Fuori_Corso (matricola_st,Cognome,Nome, corso_di_Laurea))

INSERT INTO Fuori_Corso (SELECT ….)

Studente(Matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Fuori_Corso (Matricola_st,Cognome,Nome, corso_di_Laurea))

Esercizio 25: inserire gli studenti che non danno esami dal 98 nella tabella Fuori_Corso (matricola_st,Cognome,Nome, corso_di_Laurea))

```
INSERT INTO Fuori_Corso (
SELECT *
FROM Studente S JOIN Esame E on S.matricola_st=E.matricola_studente
WHERE NOT EXISTS
        (SELECT *
        FROM Esame E
        WHERE E.Matricola_studente=S.Matricola_st and Data>1998-01-01))
```

Studente(Matricola_st,Cognome,Nome, corso_di_Laurea)
Esame (Codice_e, Matricola_studente, Codice_corso, Data, Voto)
Piano_di_studio(Matricola_st,Codice_corso)
Corso(Codice_c, Nome, Ore_lezione, Ore_esercitazione, Crediti_lezione, Crediti_esercitazione, Docente)
Fuori_Corso (Matricola_st,Cognome,Nome, corso_di_Laurea))

Esercizio 25: inserire gli studenti che non danno esami dal 98 nella tabella Fuori_Corso (matricola_st,Cognome,Nome, corso_di_Laurea))

```
INSERT INTO Fuori_Corso (
SELECT *
FROM Studente S JOIN Esame E on S.matricola_st=E.matricola_studente
WHERE NOT EXISTS
        (SELECT *
        FROM Esame E
        WHERE E.Matricola_studente=S.Matricola_st and  year(Data)>1998))
```

Possiamo usare anche la funzione **year**