SQL :Structured **Q**uery **L**anguage: SELECT (II)

Prof. Alfredo Pulvirenti
Prof. Salvatore Alaimo

Sintassi, riassumiamo

SelectSQL ::=

`select` ListaAttributiOEspressioni

`from` ListaTabelle

[`where` CondizioniSemplici]

[`group by` ListaAttributiDiRaggruppamento]

[`having` CondizioniAggregate]

[`order by` ListaAttributiDiOrdinamento]

Esempio

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 1	222222
Ugo Bianchi	234567	Via Roma 2	333333
Teo Verdi	345678	Via Enna 3	444444

Corso	Professore
Programmazione	Ferro
Architettura	Pappalardo
Matematica Discreta	Lizzio

Corso	Matricola	Voto
Programmazione	345678	27
Architettura	123456	30
Programmazione	234567	18
Matematica Discreta	345678	22
Architettura	345678	30

Creare una query che restiuisca:
nome, matricola, voto minimo,voto massimo, voto medio
per gli studenti che hanno dato più di 8 materie

Esempio

```
SELECT Nome, Matricola, MIN(Voto), MAX(Voto),AVG(Voto)
FROM   Esami, Studenti
WHERE  Esami.Matricola = Studenti.Matricola
GROUP  BY Nome,Matricola
HAVING COUNT(*) > 8
```

Creare una query che restiuisca:
nome, matricola, voto minimo,voto massimo, voto medio
per gli studenti che hanno dato più di 8 materie

- La clausola HAVING ammette come argomento un’espressione booleana su predicati semplici.
- In questo caso sono i risultati di un confronto tra la valutazione di un operatore aggregato e una generica espressione.
- Regola generale:
  - Solo i predicati in cui compaiono operatori aggregati devono essere argomento della clausola HAVING

Esercitazione

- Considera il seguente schema relazionale:
  - `Quadri( Pittore, Titolo, DataCreazione, NomeMuseo)`
  - `Musei( NomeMuseo, Citta, Curatore)`
  - `Costo( Titolo, Prezzo)`

- Scrivere le seguenti query in SQL:
  - Elenco dei musei e del valore medio dei quadri posseduti.
  - Il numero di quadri per ogni pittore fra tutti i musei parigini.

Elenco dei musei e del valore medio dei quadri posseduti.

- Quadri(Pittore, Titolo, DataCreazione, NomeMuseo)
- Musei(NomeMuseo, Citta, Curatore)
- Costo(Titolo, Prezzo)

```
SELECT avg(c.prezzo)‘prezzo medio’, q.nomeMuseo
FROM   quadri q, costo c
WHERE   q.Titolo = c.Titolo
GROUP BY q.nomeMuseo
```

Elenco dei musei e del valore medio dei quadri posseduti.

- Quadri(Pittore, Titolo, DataCreazione, NomeMuseo)
- Musei(NomeMuseo, Citta, Curatore)
- Costo(Titolo, Prezzo)

```
SELECT avg(c.prezzo) ‘prezzo medio’, q.nomeMuseo
FROM  quadri q
JOIN  costo c ON q.Titolo = c.Titolo
GROUP BY q.nomeMuseo
```

Il numero di quadri per ogni pittore fra tutti i musei parigini.

- Quadri(Pittore, Titolo, DataCreazione, NomeMuseo)
- Musei(NomeMuseo, Citta, Curatore)
- Costo(Titolo, Prezzo)

```
SELECT count(*)‘numero quadri’, pittore
FROM musei m, quadri q
WHERE m.Citta = ‘Parigi’ AND m.nomeMuseo = q.nomeMuseo
GROUP BY pittore
```

Interrogazioni nidificate

```
SELECT      [DISTINCT] {*| colonna [alias], ...}
FROM        tabella
[WHERE      condizione(i)];
```

- Un primo modo e' scrivere una condizione del tipo:

  Attributo op (Sottoselect)

  - op in {=, <>, >, >=, <, <=}
  - SottoSelect deve dare come risultato una tabella **con un solo elemento o vuota** (vedremo alcuni esempi)

Esempio

- nome e reddito del padre di Franco

```
SELECT Nome, Reddito
FROM   Persone, Paternita
WHERE  Nome = Padre AND Figlio = 'Franco'


SELECT Nome, Reddito
FROM Persone
WHERE Nome = (   SELECT Padre
        FROM Paternita
                WHERE Figlio = 'Franco')
```

- La query nella clausola WHERE è la query nidificata

Interrogazioni nidificate

- le condizioni in SQL permettono anche il confronto fra un attributo e il risultato di una sottoquery
  - Attributo op (ANY | ALL) (Sottoselect)
    - `ANY`: il predicato e' vero se almeno uno dei valori restituiti da Query soddisfano la condizione
    - `ALL`: il predicato e' vero se tutti i valori restituiti dalla Query soddisfano la condizione
  - quantificatore esistenziale
    - [NOT] EXISTS (Sottoselect)
    - Il predicato e' vero se la SelectQuery restituisce almeno una tupla
  - Attributo [NOT] IN (Sottoselect)

```
SELECT P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre AND Figlio = F.Nome
 AND F.Reddito > 20

SELECT Nome, Reddito
FROM Persone
WHERE Nome in (SELECT Padre
    FROM Paternita
    WHERE Figlio = any (SELECT Nome
            FROM Persone
            WHERE Reddito > 20))
```

```
SELECT  P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre AND Figlio = F.Nome
 AND F.Reddito > 20

SELECT Nome, Reddito
FROM Persone
WHERE Nome in (SELECT Padre
    FROM Paternita, Persone
    WHERE Figlio = Nome
    AND Reddito > 20)
```

Interrogazioni nidificate, commenti

- La forma nidificata è “meno dichiarativa”, ma talvolta più leggibile (richiede meno variabili).
- La forma piana e quella nidificata possono essere combinate.
- Le sottointerrogazioni non possono contenere operatori insiemistici (“l’unione si fa solo al livello esterno”).

Negazione con le query nidificate

- Trovare quei dipartimenti dove non c'è nessuno che si chiama 'Brown':

```
select DeptName
from Department
where DeptName <> all (select Dept
                       from Employee
                       where Surname = 'Brown')
```

- Oppure:

```
select DeptName
from Department
   except
select Dept
from Employee
where Surname = 'Brown'
```

Operatori IN e NOT IN

- IN e' sinonimo di: =ANY

```
select FirstName, Surname
from Employee
where Dept in (select DeptName
               from Department
               where City = 'London')
```

- NOT IN e' sinonimo di: <>ALL

```
select DeptName
from Department
where DeptName not in (select Dept
                       from Employee
                       where Surname = 'Brown')
```

MAX e MIN con le query nidificate

- Esempio: Il dipartimento(i) dove lavora colui con lo stipendio più alto di tutta l'azienda:

```
select Dept
from Employee
where Salary in (select max(Salary)
                 from Employee
```

- Oppure:

```
select Dept
from Employee
where Salary >= all (select Salary
                     from Employee
```

MAX e MIN con le query nidificate

- I dipartimenti che hanno una somma di salari maggiore rispetto alla somma media dei salari dell'azienda

```
SELECT Dept
FROM Emp
GROUP BY Dept
HAVING SUM(Salary) > (
     SELECT AVG(Totale.SalTot)
     FROM (SELECT SUM(Salary) 'SalTot'
                FROM Emp
                GROUP BY Dept) AS Totale)
```

Interrogazioni nidificate, commenti

- regole di visibilità:
  - non è possibile fare riferimenti a variabili definite in blocchi più interni
  - se un nome di variabile è omesso, si assume riferimento alla variabile più “vicina”
- in un blocco si può fare riferimento a variabili definite in blocchi più esterni

Quantificazione esistenziale

- Ulteriore tipo di condizione
  - EXISTS (Sottoespressione)
- Le persone che hanno almeno un figlio (viene valutata l'interrogazione esterna e per ogni riga quella interna)

```
SELECT *
FROM Persone p
WHERE EXISTS (  SELECT *
          FROM Paternita
          WHERE Padre = p.Nome)
      OR
      EXISTS (  SELECT *
          FROM Maternita
          WHERE Madre = p.Nome)
```

Quantificazione esistenziale, 2

- I padri i cui figli guadagnano tutti più di venti

Quantificazione esistenziale, 2

- I padri i cui figli guadagnano tutti più di venti

```
SELECT distinct Padre
FROM Paternita Z
WHERE NOT EXISTS (
        SELECT *
      FROM    Paternita W, Persone
      WHERE   W.Padre = Z.Padre
      AND W.Figlio = Nome
      AND Reddito <= 20)
```

Semantica delle espressioni “correlate”

- La query più interna può usare variabili della query esterna
- L’interrogazione interna viene eseguita una volta per ciascuna ennupla dell’interrogazione esterna
- Esempio, trovare tutti gli studenti che hanno un omonimo:

```
SELECT *
FROM Student S
WHERE EXISTS (SELECT *
              FROM Student S2
              WHERE S2.Nome = S.Nome
              AND   S2.Cognome = S.Cognome
              AND   S2.Matricola <> S.Matricola)
```

Semantica delle espressioni “correlate”, 2

- Esempio, trovare tutti gli studenti che NON hanno un omonimo:

```
SELECT *
FROM Student S
WHERE NOT EXISTS (SELECT *
                  FROM   Student S2
                  WHERE  S2.Nome = S.Nome
                  AND    S2.Cognome = S.Cognome
                  AND    S2.Matricola <> S.Matricola)
```

Visibilità

- Scorretta:

```
SELECT *
FROM Impiegato
WHERE Dipart in (SELECT Nome
                 FROM Dipartimento D1
                 WHERE Nome = 'Produzione') OR
      Dipart in (SELECT Nome
                 FROM Dipartimento D2
                 WHERE D2.Citta = D1.Citta)
```

  - D1 non e' visibile nella seconda query nidificata in quanto le due sottoquery sono allo stesso livello

Confronto su più attributi

- Il confronto con il risultato di una query nidificata può essere basato su più attributi
- Stessa query di prima, trovare tutti gli studenti che hanno un omonimo:

```
SELECT *
FROM   Student S
WHERE  (Nome, Cognome) IN
               (SELECT Nome, Cognome
                FROM   Student S2
                WHERE  S2.Matricola <> S.Matricola)
```

Esempio: ancora il quantificatore Universale

Agenti(CodiceAgente,Nome,Zona Supervisore,Commissione)

Clienti(CodiceCliente,Nome,Citta',Sconto)

Ordini(CodiceOrdine,CodiceCliente,CodiceAgente,Articolo,Data,Ammontare)

Esempio: ancora il quantificatore Universale

- Supponiamo di voler trovare i codici di quei clienti che hanno fatto ordini a TUTTI gli agenti di Catania.

- Per ogni agente z di Catania esiste un ordine y del nostro cliente x a z.

$$\forall\, z\, \exists\, y\; y(n, x, z, p, d, a)$$
$$\Leftrightarrow \nexists z \nexists y\; y(n, x, z, p, d, a)$$

Tradotta in SQL

```
SELECT C.CodiceCliente
FROM     Clienti C
WHERE NOT EXISTS
   (SELECT *
    FROM   Agenti A
    WHERE A.Zona = 'Catania'
    AND NOT EXISTS
    (SELECT *
     FROM Ordini V
     WHERE V.CodiceCliente = C.CodiceCliente
     AND   V.CodiceAgente =  A.CodiceAgente) )
```

City(id,city,country,district,population)

Cities_Stores(city,store_type,address)

Stores(store_type,description)

Esempi

- Che tipi di negozi sono presenti in una o più città?

```
SELECT DISTINCT store_type
FROM Stores WHERE EXISTS
                  (SELECT * FROM Cities_Stores WHERE
                   Cities_Stores.store_type
                           = Stores.store_type)
```

- Quali tipi di negozi non sono presenti nelle città?

```
SELECT DISTINCT store_type
FROM Stores WHERE NOT EXISTS
                        (SELECT * FROM Cities_Stores
                         WHERE Cities_Stores.store_type
                               = Stores.store_type)
```

Esempi

- Quali ***tipi di negozi*** sono presenti in TUTTE le città?

```
SELECT DISTINCT store_type FROM Stores
WHERE NOT EXISTS (
  SELECT * FROM Cities WHERE
  NOT EXISTS (
     SELECT * FROM Cities_Stores
     WHERE Cities_Stores.city = Cities.city AND
     Cities_Stores.store_type = Stores.store_type))
```

1. Date le relazioni:
   `Prodotto(marca, modello)`
   `PC(modello, velocità, ram)`

Scrivere una query SQL che trovi le marche di PC che hanno il processore più veloce tra tutti i PC che hanno la minor quantità di RAM.

```
SELECT Prodotto.marca
FROM   PC p1, Prodotto
WHERE NOT EXISTS  (SELECT *
                   FROM PC
                   WHERE PC.ram < p1.ram)
      AND p1.speed  >= ALL (SELECT speed
                            FROM PC
                            WHERE PC.ram = p1.ram)
      AND p1.modello = Prodotto.modello
```

Commenti finali sulle query nidificate

- Query nidificate possono essere “meno dichiarative” in un certo senso ma spesso sono piu’ facilmente interpretabili
  - Suddivisibili in blocchi più semplici da interpretare
- L’utilizzo di variabili deve rispettare le regole di visibilità
  - Cioè, una variabile può essere usata solo all’interno dello stesso blocco e in un blocco piu’ interno
- Query nidificate complesse possono essere di difficile comprensione
  - Soprattutto quando si usano molte variabili comuni tra blocchi diversi

Note sulle sottoselect e valori nulli

- Espr op (Sottoselect)
  - Vale unknown
    - se espr è NULL oppure
    - se la Sottoselect produce una tabella vuota

- Espr IN (Sottoselect)
  Espr op ANY (Sottoselect)
  - Vale unknown
    - se espr e' NULL oppure
    - se NULL è fra I valori della SottoSelect e il predicato e' falso per gli altri valori non NULL

- Espr op ALL (Sottoselect)
  - Vale unknown
    - se espr è NULL oppure
    - se NULL è fra i valori della SottoSelect

MADRE	FIGLIO	ANNI
Maria	Luigi	30
Maria	Agata	NULL
Carla	Francesco	NULL

```
SELECT madre FROM madri
WHERE anni > 30  -- risultato tab vuota

SELECT madre FROM madri
WHERE madre =‘Carla’ and  anni > 30  -- risultato tab vuota

SELECT madre FROM madri
WHERE madre =‘Carla’ or  anni > 30  -- risultato tupla riferita a Carla

SELECT madre FROM madri
WHERE anni >= ALL (select anni from madri);
```

Sintassi Completa del SELECT

```
Select ::= Sottoselect
{(UNION|EXCEPT) Sottoselect}
[ORDER BY Attributo[DESC]
{, Attributo[DESC]} ]
```

Sottoselect

Sottoselect ::= SELECT [DISTINCT]
(* | Espr[[AS] NewName] {,Espr [[AS] NewName]})
FROM Tabella [Ide]{,Tabella[Ide]}
[WHERE Condizione]
[GROUP BY Attributo {,Attributo}]
[HAVING Condizione]

Condizione

```
Condizione ::=  Predicato |
                “(“Condizione “)” |
              NOT Condizione    |
                 Condizione (AND | OR) Condizione
```

Predicato

```
Predicato::= Espr [NOT] IN “(“ SottoSelect “)” |
             Espr [NOT] IN “(“ Valore {,Valore} “)” |
         Espr opc (Espr | “(“ SottoSelect “)” )  |
         Espr IS [NOT] NULL |
         Espr opc (ANY | ALL) “(“ SottoSelect “)”   |
             [NOT] EXISTS “(“ SottoSelect “)”   |
         Espr [NOT] BETWEEN Espr AND Espr |
         Espr [NOT] LIKE Stringa
```

opc ::= $<$ / $>$ / $\leq$ / $\geq$ / $=$ / $\neq$

Espressioni

```
Espr ::= [Ide.] Attributo |
              Costante |
               “(” Espr “)” |
               [-] Espr [ opa Espr] |
               (SUM | COUNT | AVG | MAX | MIN)
                   “(” [DISTINCT] [Ide.] Attributo“)” |
                COUNT “(” * “)”

opa ::= (+ |  -  |  *  |  /  )
```

Tabelle

```
Tabella ::= Ide  |
                  Tabella opins Tabella  |
                   Tabella giunzione Tabella
  [USING “(“ Attributo {, Attributo } “)” | ON Condizione]

giunzione ::= [CROSS|UNION|NATURAL]
              [LEFT|RIGHT|FULL]JOIN

opins ::= (UNION|INTERSECT|EXCEPT)
  [CORRESPONDING [BY”(Attributo {,Attributo} “)”]]
```