SQL - Structured Query Language

Alessandro Lori

Università di Pisa

25 marzo 2011

Query su più tabelle

Sintassi **SELECT**

```
SELECT Attributo1, Attributo2, ...
FROM Tabella1, Tabella2, ...
```

- Cosa significa inserire più di una tabella nel **FROM**?
- La query viene eseguita sul *prodotto cartesiano* delle tabelle
- Nel caso di due tabelle, ogni riga della prima tabella viene collegata ad ogni riga della seconda

Prodotto cartesiano

- Sia data la tabella a destra (Persone)
- *Esempio 1:* Costruire una query che produca come risultato tutte le coppie ordinate di persone. Per ogni coppia di persone visualizzare il nome di ciascuna delle persone della coppia

Nome	Eta	Reddito
Aldo	25	15
Andrea	27	21
Anna	50	35
Filippo	26	30
Franco	60	20
Gigi	26	21
Luigi	50	40
Luisa	75	87
Maria	55	42
Olga	30	41
Sergio	85	35

Esempio 1

Soluzione

```
SELECT P1.Nome, P2.Nome
FROM Persone AS P1,
     Persone AS P2
```

Nome	Nome
Aldo	Aldo
Andrea	Aldo
Filippo	Aldo
Gigi	Aldo
...	...

E se volessi ottenere solo le coppie non ordinate?

Esempio 1

Soluzione

```
SELECT P1.Nome, P2.Nome
FROM Persone AS P1,
     Persone AS P2
```

Nome	Nome
Aldo	Aldo
Andrea	Aldo
Filippo	Aldo
Gigi	Aldo
...	...

E se volessi ottenere solo le coppie non ordinate?

Soluzione

```
SELECT P1.Nome, P2.Nome
FROM Persone AS P1,
     Persone AS P2
WHERE P1.Nome < P2.Nome
```

Nome	Nome
Aldo	Andrea
Aldo	Anna
Andrea	Anna
Aldo	Filippo
...	...

Esercizio 1

Trovare tutte le persone che hanno la stessa età. Ogni riga del risultato dovrà mostrare una coppia di persone (distinte) che hanno la stessa età e la loro età.

Esercizio 1

Trovare tutte le persone che hanno la stessa età. Ogni riga del risultato dovrà mostrare una coppia di persone (distinte) che hanno la stessa età e la loro età.

Soluzione

```
SELECT P1.Nome, P2.Nome, P1.Eta
FROM Persone P1, Persone P2
WHERE P1.Nome < P2.Nome
AND   P1.Eta = P2.Eta
```

Nome	Nome	Eta
Filippo	Gigi	26
Anna	Luigi	50

Join

- Progettando le basi di dati in maniera oculata le informazioni vengono disperse su più tabelle.
- Nelle query si può aver necessità di ricostruire l'informazione dispersa utilizzando il **`JOIN`**
- Col **`JOIN`** si collegano solo le righe di tabelle diverse che hanno lo stesso valore su uno o più campi
- Si considerino le seguenti tabelle
  - Persone(Nome, Età, Reddito)
  - Maternità(Madre, Figlio)
  - Paternità(Padre, Figlio)

Esercizio 2

Costruire una query che produca come risultato i padri di persone che hanno un reddito superiore a 20

Esercizio 2

Costruire una query che produca come risultato i padri di persone che hanno un reddito superiore a 20

Soluzione

```
SELECT DISTINCT Padre
FROM Persone, Paternita
WHERE Figlio = Nome
AND Reddito > 20
```

Padre
Franco
Luigi

Esercizio 2 - Parola chiave `JOIN`

Costruire una query che produca come risultato i padri di persone che hanno un reddito superiore a 20

Esercizio 2 - Parola chiave `JOIN`

Costruire una query che produca come risultato i padri di persone che hanno un reddito superiore a 20

Soluzione

```
SELECT DISTINCT Padre
FROM Persone JOIN Paternita
ON Figlio=Nome
WHERE Reddito > 20
```

Padre
Franco
Luigi

Esercizio 3

Costruire una query che produca come risultato il padre e la madre di ogni persona

Esercizio 3

Costruire una query che produca come risultato il padre e la madre di ogni persona

Soluzione

```
SELECT Padre, Madre, P.Figlio
FROM Paternita P, Maternita M
WHERE P.Figlio = M.Figlio
```

Esercizio 3

Costruire una query che produca come risultato il padre e la madre di ogni persona

Soluzione

```
SELECT Padre, Madre, P.Figlio
FROM Paternita P, Maternita M
WHERE P.Figlio = M.Figlio
```

Soluzione alternativa

```
SELECT Padre, Madre, P.Figlio
FROM Paternita P JOIN Maternita M
ON P.Figlio = M.Figlio
```

Natural Join

Soluzione alternativa

```
SELECT Padre, Madre, P.Figlio
FROM Paternita P NATURAL JOIN Maternita
```

Il **NATURAL JOIN** effettua il join sugli attributi che hanno lo stesso nome

Esercizio 4

Considerare le persone che guadagnano più dei rispettivi padri. Costruire una query che produca come risultato nome del figlio, reddito del figlio e reddito del padre

Esercizio 4

Considerare le persone che guadagnano più dei rispettivi padri. Costruire una query che produca come risultato nome del figlio, reddito del figlio e reddito del padre

Soluzione

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

Esercizio 4

Considerare le persone che guadagnano più dei rispettivi padri. Costruire una query che produca come risultato nome del figlio, reddito del figlio e reddito del padre

Soluzione

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

Soluzione alternativa con JOIN

```
SELECT F.Nome, F.Reddito, P.Reddito AS RedditoPadre
FROM (Persone P JOIN Paternita Pa ON
P.Nome = Pa.Padre) JOIN Persone F ON
Pa.Figlio = F.Nome WHERE F.Reddito > P.Reddito
```

INNER JOIN - OUTER JOIN

- Di default viene usato l'**INNER JOIN**. Fanno parte del risultato solo le righe della prima e della seconda tabella che verificano la condizione
- Con l'**OUTER JOIN** si possono includere nel risultato anche le righe che non hanno una corrispondenza nell'altra tabella
  - **LEFT** [**OUTER**] **JOIN** Partecipano anche le righe della prima tabella che non hanno una corrispondenza. Gli attributi della seconda tabella sono fissati a **NULL**
  - **RIGHT** [**OUTER**] **JOIN** Partecipano anche le righe della seconda tabella che non hanno una corrispondenza. Gli attributi della prima tabella sono fissati a **NULL**
  - **FULL** [**OUTER**] **JOIN** Partecipano tutte le righe che non hanno una corrispondenza. Gli altri attributi sono fissati a **NULL**. Non supportato da `MySQL`

INNER JOIN - OUTER JOIN - Esempi

LEFT JOIN

```
SELECT *
FROM Maternita Ma LEFT JOIN Persone Pe
ON Ma.Figlio = Pe.Nome
```

INNER JOIN - OUTER JOIN - Esempi

LEFT JOIN

```
SELECT *
FROM Maternita Ma LEFT JOIN Persone Pe
ON Ma.Figlio = Pe.Nome
```

RIGHT JOIN

```
SELECT *
FROM Maternita Ma RIGHT JOIN Persone Pe
ON Ma.Figlio = Pe.Nome
```

Operatori aggregati

- Si possono usare degli operatori che effettuano calcoli sui valori di una colonna appartenenti a più righe
  - **COUNT** conta il numero di valori in un insieme
  - **MIN** calcola il minimo un insieme di valori
  - **MAX** calcola il massimo un insieme di valori
  - **AVG** calcola la media di un insieme di valori
  - **SUM** calcola la somma di un insieme di valori
- L'operatore aggregato **COUNT** supporta una lista di attributi (e quindi anche un attributo singolo)
- Gli operatori aggregati **MIN**, **MAX**, **SUM** e **AVG** supportano solo un singolo attributo

Operatori aggregati - Esempi

Calcolare la somma dei redditi delle persone con meno di 40 anni

Operatori aggregati - Esempi

Calcolare la somma dei redditi delle persone con meno di 40 anni

Soluzione

```
SELECT SUM(Reddito)
FROM Persone
WHERE Eta<40
```

Operatori aggregati - Esempi

Calcolare la somma dei redditi delle persone con meno di 40 anni

Soluzione

```
SELECT SUM(Reddito)
FROM Persone
WHERE Eta<40
```

Calcolare quante persone guadagnano fra 30 e 40

Operatori aggregati - Esempi

Calcolare la somma dei redditi delle persone con meno di 40 anni

Soluzione

```
SELECT SUM(Reddito)
FROM Persone
WHERE Eta<40
```

Calcolare quante persone guadagnano fra 30 e 40

Soluzione

```
SELECT COUNT(*)
FROM Persone
WHERE Reddito BETWEEN 30 AND 40
```

L'operatore COUNT

La query

```
SELECT COUNT(*)
FROM Persone
WHERE Reddito BETWEEN 30 AND 40
```

si ottiene applicando **COUNT** al risultato della query

```
SELECT *
FROM Persone
WHERE Reddito BETWEEN 30 AND 40
```

L'operatore **COUNT** e i valori **NULL**

- **SELECT COUNT**(∗) **FROM** Persone
  - Conta il numero di righe nella tabella
- **SELECT COUNT**(Reddito) **FROM** Persone
  - Conta il numero di righe con Reddito diverso da **NULL**
- **SELECT COUNT**(**DISTINCT** Reddito) **FROM** Persone
  - Conta il numero di valori diversi nella colonna Reddito
- **SELECT COUNT**(**ALL** Reddito) **FROM** Persone
  - Conta il numero di righe hanno un Reddito diverso da **NULL**. La parola chiave **ALL** può essere omessa

L'operatore `COUNT` e i valori `NULL` - Esempi

Data la tabella

Nome	Eta	Reddito
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

- **SELECT COUNT**(∗) **FROM** Persone
- **SELECT COUNT**(Reddito) **FROM** Persone
- **SELECT COUNT**(**DISTINCT** Reddito) **FROM** Persone

L'operatore `COUNT` e i valori `NULL` - Esempi

Data la tabella

Nome	**Eta**	**Reddito**
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

- **`SELECT COUNT`**$(*)$ **`FROM`** Persone **4**
- **`SELECT COUNT`**(Reddito) **`FROM`** Persone
- **`SELECT COUNT`**(**`DISTINCT`** Reddito) **`FROM`** Persone

L'operatore `COUNT` e i valori `NULL` - Esempi

Data la tabella

Nome	**Eta**	**Reddito**
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

- `SELECT COUNT`$(*)$ `FROM` Persone **4**
- `SELECT COUNT`(Reddito) `FROM` Persone **3**
- `SELECT COUNT`(`DISTINCT` Reddito) `FROM` Persone

L'operatore `COUNT` e i valori `NULL` - Esempi

Data la tabella

Nome	**Eta**	**Reddito**
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

- **SELECT COUNT**(∗) **FROM** Persone **4**
- **SELECT COUNT**(Reddito) **FROM** Persone **3**
- **SELECT COUNT**(**DISTINCT** Reddito) **FROM** Persone **2**

Esercizio

Costruire una query che produca come risultato la media dei redditi dei figli di Franco

Esercizio

Costruire una query che produca come risultato la media dei redditi dei figli di Franco

Soluzione

```
SELECT AVG(Reddito)
FROM Persone JOIN Paternita
ON Nome=Figlio
WHERE Padre='Franco'
```

Eventuali valori nulli non vengono considerati nel calcolo della media

Creare gruppi - la clausola `GROUP BY`

- Finora utilizzando gli operatori aggregati il risultato è un unico valore
- Si possono raggruppare insiemi di tuple su cui applicare singolarmente gli operatori aggregati
- Indicando **`GROUP BY`** listaAttributi la tabella viene partizionata in gruppi
- Tutte le tuple che appartengono allo stesso gruppo hanno lo stesso valore sulla lista degli attributi specificata nel **`GROUP BY`**

Esempio

Costruire una query che produca come risultato il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)
FROM Paternita
GROUP BY Padre
```

Esempio

Costruire una query che produca come risultato il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)
FROM Paternita
GROUP BY Padre
```

1. Si sceglie le tabelle da utilizzare per produrre il risultato
   FROM

Esempio

Costruire una query che produca come risultato il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)
FROM Paternita
GROUP BY Padre
```

1. Si sceglie le tabelle da utilizzare per produrre il risultato **FROM**
2. Si decide quali tuple eliminare **WHERE**

Esempio

Costruire una query che produca come risultato il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)
FROM Paternita
GROUP BY Padre
```

1. Si sceglie le tabelle da utilizzare per produrre il risultato **FROM**
2. Si decide quali tuple eliminare **WHERE**
3. Si raggruppano le tuple rimaste **GROUP BY**

Esempio

Costruire una query che produca come risultato il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)
FROM Paternita
GROUP BY Padre
```

1. Si sceglie le tabelle da utilizzare per produrre il risultato **FROM**
2. Si decide quali tuple eliminare **WHERE**
3. Si raggruppano le tuple rimaste **GROUP BY**
4. Si applica l'operatore aggregato a ciascun gruppo

Attributi nel `SELECT` e `GROUP BY`

Query scorretta

```
SELECT Padre, AVG(F.Reddito), P.Reddito
FROM (Persone AS F JOIN Paternita ON Figlio = Nome)
     JOIN Persone AS P ON Padre=P.Nome
GROUP BY Padre
```

Attributi nel SELECT e GROUP BY

Query scorretta

```
SELECT Padre, AVG(F.Reddito), P.Reddito
FROM (Persone AS F JOIN Paternita ON Figlio = Nome)
     JOIN Persone AS P ON Padre=P.Nome
GROUP BY Padre
```

Query corretta

```
SELECT Padre, AVG(F.Reddito), P.Reddito
FROM (Persone AS F JOIN Paternita ON Figlio = Nome)
     JOIN Persone AS P ON Padre=P.Nome
GROUP BY Padre, P.Reddito
```

Filtrare i gruppi

Con la clausola **`HAVING`** possiamo inserire solo alcuni dei gruppi creati

Filtrare i gruppi

Con la clausola **HAVING** possiamo inserire solo alcuni dei gruppi creati

Esempio

```
SELECT Padre, AVG(F.Reddito), P.Reddito
FROM (Persone AS F JOIN Paternita ON Figlio = Nome)
     JOIN Persone AS P ON Padre=P.Nome
GROUP BY Padre, P.Reddito
HAVING AVG(F.Reddito) < 20
```

Esercizio

Costruire una query che produca come risultato i padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 25 e il reddito medio di tali figli

Esercizio

Costruire una query che produca come risultato i padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 25 e il reddito medio di tali figli

Soluzione

```
SELECT Padre, AVG(Reddito)
FROM Persone JOIN Paternita ON Figlio = Nome
WHERE Eta < 30
GROUP BY Padre
HAVING AVG (Reddito) > 25
```

Operatori insiemistici

- Il risultato di una **SELECT** è una tabella (insieme)
- Fra i risultati di due query si possono operare le comune operazioni sugli insiemi:
  - $\cup$ Unione - **UNION**
  - $\cap$ Intersezione - **INTERSECT**
  - $\setminus$ Complemento - **EXCEPT**

Sintassi

```
SELECT .... FROM ....
UNION / INTERSECT / EXCEPT [ALL]
SELECT .... FROM ....
```

Di norma i duplicati sono rimossi, a meno che non si aggiunga la parola chiave **ALL**

Operatori insiemistici - Requisiti

- Non è necessario che gli schemi a cui si applicano gli operatori insiemistici siano identici
- Tuttavia, gli schemi devono avere lo stesso numero di attributi e lo stesso tipo, così come specificato nelle clausole **`SELECT`**
- Nel risultato, si utilizzano come nomi di attributo quelli del primo schema

Notazione posizionale

Nel risultato di una query, gli attributi compaiono nel risultato secondo l'ordine specificato nella **`SELECT`**

Esempio - Elencare tutte le relazioni Genitore - Figlio

Notazione posizionale

Nel risultato di una query, gli attributi compaiono nel risultato secondo l'ordine specificato nella **SELECT**

Esempio - Elencare tutte le relazioni Genitore - Figlio

Soluzione errata

```
SELECT Padre, Figlio FROM Paternita
UNION
SELECT Figlio, Madre FROM Maternita
```

Notazione posizionale

Nel risultato di una query, gli attributi compaiono nel risultato secondo l'ordine specificato nella **SELECT**
Esempio - Elencare tutte le relazioni Genitore - Figlio

Soluzione errata

```
SELECT Padre, Figlio FROM Paternita
UNION
SELECT Figlio, Madre FROM Maternita
```

Soluzione corretta

```
SELECT Padre, Figlio FROM Paternita
UNION
SELECT Madre, Figlio FROM Maternita
```

Ridenominazioni

Ridenominare non cambia le cose **Esempio** - Elencare tutte le relazioni Genitore - Figlio

Ridenominazioni

Ridenominare non cambia le cose **Esempio** - Elencare tutte le relazioni Genitore - Figlio

Soluzione errata

```
SELECT Padre AS Genitore, Figlio FROM Paternita
UNION
SELECT Figlio, Madre AS Genitore FROM Maternita
```

Ridenominazioni

Ridenominare non cambia le cose **Esempio** - Elencare tutte le relazioni Genitore - Figlio

Soluzione errata

```
SELECT Padre AS Genitore, Figlio FROM Paternita
UNION
SELECT Figlio, Madre AS Genitore FROM Maternita
```

Soluzione corretta

```
SELECT Padre AS Genitore, Figlio FROM Paternita
UNION
SELECT Madre, Figlio FROM Maternita
```

Esercizio

Trovare tutte le persone che sono genitori

Esercizio

Trovare tutte le persone che sono genitori

Soluzione

```
SELECT Padre AS Genitore FROM Paternita
UNION
SELECT Madre FROM Maternita
```

Altri operatori insiemistici - **EXCEPT**

Complemento - Esempio

```
SELECT Figlio FROM Paternita
EXCEPT
SELECT Figlio FROM Maternita
```

- Restituisce le persone di cui si sa chi è il padre ma non chi è la madre
- **EXCEPT** restituisce tutti i risultati della prima query che non appartengono alla seconda query
- Non supportato da `MySQL`
- Si potrà emulare utilizzando le query annidate

Altri operatori insiemistici - INTERSECT

Intersezione - Esempio

```
SELECT Figlio FROM Paternita
INTERSECT
SELECT Figlio FROM Maternita
```

- Restituisce le persone di cui si sa sia chi è il padre sia chi è la madre
- **INTERSECT** restituisce tutti i risultati della prima query che appartengono anche alla seconda query
- Non supportato da MySQL
- Si potrà emulare utilizzando le query annidate