
Basi di Dati

Modulo Laboratorio

Lezione 4: Introduzione ai raggruppamenti in SQL

DR. SARA MIGLIORINI

Sommario

- Data Query Language (DQL/QL)
 - Comando SELECT (lezione precedente)
 - Interrogazioni con raggruppamento: GROUP BY e HAVING
 - Interrogazioni con JOIN
 - INNER JOIN
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN

Interrogazioni con raggruppamento

- Un raggruppamento è un insieme di tuple che hanno medesimi valori su uno o più attributi caratteristici del raggruppamento.
- La clausola `GROUP BY attr [, ...]` permette di determinare tutti i raggruppamenti delle tuple della relazione risultato (tuple selezionate con la clausola `WHERE`) in funzione degli attributi dati.
- In una interrogazione che usa `GROUP BY`, la clausola `SELECT` contiene solamente gli attributi (da 0 a tutti) utilizzati per il raggruppamento ed eventuali funzioni di aggregazione sugli altri attributi.

Interrogazioni con raggruppamento

- Si assuma una tabella così popolata

| matricola | cognome | nome | indirizzo | città | media |
|-----------|---------|--------|------------|--------|---------|
| IN0001 | Rossi | Marco | Via nobile | Verona | 27.50 |
| IN0002 | Paoli | Paolo | Via dritta | Padova | 28.6666 |
| IN0003 | Bianchi | Dante | Via storta | VERONA | 18.345 |
| IN0004 | Rossi | Matteo | Via nobile | Verona | 24.567 |
| IN0005 | Verdi | Luca | Via nuova | Va | 28.6666 |
| IN0006 | Nocasa | Caio | | | |

Interrogazioni con raggruppamento

- Visualizzare tutte le città raggruppate della tabella Studente

```
SELECT città  
FROM studente  
GROUP BY città
```

| città |
|--------|
| Padova |
| Verona |
| VERONA |
| Va |

Interrogazioni con raggruppamento

- Visualizzare tutte le città e i cognomi raggruppati.

```
SELECT cognome , LOWER(città)
FROM Studente
GROUP BY cognome , LOWER(città);
```

| cognome | città |
|---------|--------|
| Verdi | Va |
| Rossi | verona |
| Paoli | padova |
| Nocasa | |
| Bianchi | verona |

- **NOTA:** nel GROUP BY non si possono usare espressioni con operatori di aggregazione.

Interrogazioni con raggruppamento

- Nota: nella SELECT con GROUP BY non si possono specificare attributi che non sono raggruppati.

```
SELECT cognome, città  
FROM Studente  
GROUP BY città;
```

```
ERROR: COLUMN  
"studente.cognome" must  
appear IN the GROUP BY clause  
OR be used IN an aggregate  
function
```

Interrogazioni con raggruppamento

- Si possono specificare espressioni con operatori di aggregazione su attributi non raggruppati.

```
SELECT LOWER(città) AS "Città",  
       AVG(media)::DECIMAL(5,2)  
FROM Studente  
GROUP BY "Città";
```

| Città | avg |
|--------|-------|
| | |
| padova | 28.67 |
| va | 28.67 |
| verona | 23.47 |

- NOTA: Da PostgreSQL 10, GROUP BY può accettare l'alias di colonna!

Interrogazioni con raggruppamento

- La clausola `WHERE` permette di selezionare le righe che devono far parte del risultato.
- La clausola `HAVING` permette di selezionare i raggruppamenti che devono far parte del risultato.
- La sintassi è `HAVING bool_expr`, dove `bool_expr` è un'espressione booleana che può usare gli attributi usati nel `GROUP BY` e/o gli altri attributi mediante operatori di aggregazione.

Interrogazioni con raggruppamento

- Visualizzare tutte le città raggruppate che iniziano con 'V' della tabella Studente.

```
SELECT città  
FROM Studente  
GROUP BY città  
HAVING città LIKE 'V%';
```

| Città |
|--------|
| Verona |
| VERONA |
| Va |

Interrogazioni con raggruppamento

- Visualizzare tutte le città e i cognomi raggruppati con almeno due studenti con lo stesso cognome.

```
SELECT cognome , LOWER(città)
FROM Studente
GROUP BY cognome , LOWER (città)
HAVING COUNT ( cognome ) >1;
```

| cognome | città |
|---------|--------|
| Rossi | verona |

Interrogazioni con raggruppamento

- Si possono specificare espressioni con operatori di aggregazione su attributi non raggruppati anche nella clausola **HAVING**.

```
SELECT LOWER(città) AS "Città"  
      AVG (media)::DECIMAL(5 ,2) AS "mediaArr."  
FROM Studente  
GROUP BY LOWER(città)  
HAVING AVG(media) >23.47 ;
```

| Città | mediaArr |
|--------|----------|
| padova | 28.67 |
| va | 28.67 |
| Verona | 23.47 |

Perchè il risultato contiene anche la tupla (verona, 23.47) quando la condizione HAVING non ammette tale valore?

Interrogazioni con JOIN

- La clausola **FROM** ammette come argomento:

```
table_name [[AS] name]] [, ...]
```

- Si è visto che se sono presenti due o più nomi di tabelle, si esegue il prodotto cartesiano tra tutte le tabelle e lo schema del risultato può contenere tutti gli attributi del prodotto cartesiano.
- Il prodotto cartesiano di due o più tabelle è un **CROSS JOIN**.
- A partire da SQL-2, esistono altri tipi di JOIN (join_type):
INNER JOIN, **LEFT [OUTER] JOIN**, **RIGHT [OUTER] JOIN** e **FULL [OUTER] JOIN**.

Interrogazioni con JOIN

Sintassi generale

```
table_name [ NATURAL ] join_type table_name [ON join_condition [, ...]].
```

- dove `join_condition` è un'espressione booleana che seleziona le tuple del join da aggiungere al risultato. Le tuple selezionate possono essere poi filtrate con la condizione della clausola `WHERE`.

Interrogazioni con JOIN

Prodotto Cartesiano

- Formulazione con JOIN:

```
SELECT I. cognome , R. nomeRep  
FROM Impiegato AS I  
      CROSS JOIN Reparto AS R;
```

- Formulazione con ',':

```
SELECT I. cognome , R. nomeRep  
FROM Impiegato AS I, Reparto AS R;
```

| cognome | nome |
|---------|----------|
| Rossi | Acquisti |
| Verdi | Acquisti |
| Rossi | Vendite |
| Verdi | Vendite |

Interrogazioni con INNER JOIN

Sintassi

```
table1 [ INNER ] JOIN table2 ON join_condition [, ...].
```

- Rappresenta il tradizionale θ join dell'algebra relazionale.
- Combina ciascuna riga $r1$ di `table1` con ciascuna riga di `table2` che soddisfa la condizione della clausola `ON`.

Interrogazioni con JOIN

```
SELECT I. cognome , R. nomeRep, R.sede  
FROM Impiegato AS I  
    INNER JOIN Reparto AS R  
    ON I.nomerep = R.nomerep;
```

| cognome | nomerep | sede |
|---------|----------|--------|
| Rossi | Acquisti | Verona |
| Verdi | Vendite | Verona |

Interrogazioni con LEFT OUTER JOIN

Sintassi generale

```
table1 LEFT [ OUTER ] JOIN table2 ON join_condition [, ...].
```

- Si esegue un INNER JOIN. Poi, per ciascuna riga r1 di table1 che non soddisfa la condizione con qualsiasi riga di table2, si aggiunge una riga al risultato con i valori di r1 e assegnando NULL agli altri attributi..

Interrogazioni con JOIN

```
INSERT INTO Reparto (nomerep, sede, telefono) VALUES  
( 'Finanza', 'Padova', '02 8028888');
```

```
SELECT R. nomeRep, I.cognome  
FROM Reparto AS R  
LEFT OUTER JOIN Impiegato AS I  
ON I.nomerep = R.nomerep;
```

| nomerep | cognome |
|----------|---------|
| Acquisti | Rossi |
| Vendite | Verdi |
| Finanza | |

Interrogazioni con JOIN

Nota

Il `LEFT [OUTER] JOIN` non è simmetrico!

Con le medesime tabelle si possono ottenere risultati diversi invertendo l'ordine delle tabelle nel join!

```
SELECT I. cognome , R. nomeRep  
FROM Impiegato I LEFT OUTER JOIN RepartoR  
ON I. nomerep = R. nomerep ;
```

| cognome | nomerep |
|---------|----------|
| Rossi | Acquisti |
| Verdi | Vendite |

Interrogazioni con RIGHT OUTER JOIN

Sintassi

```
table1 RIGHT [ OUTER ] JOIN table2 ON join_condition [, ...].
```

- Si esegue un INNER JOIN. Poi, per ciascuna riga r2 di table2 che non soddisfa la condizione con qualsiasi riga di table1, si aggiunge una riga al risultato con i valori di r2 e assegnando NULL agli altri attributi..

Interrogazioni con JOIN

```
SELECT I. cognome , R. nomeRep  
FROM Impiegato I RIGHT OUTER JOIN RepartoR  
ON I. nomerep = R. nomerep ;
```

| nomerep | cognome |
|----------|---------|
| Acquisti | Rossi |
| Vendite | Verdi |
| Finanza | |

Interrogazioni con FULL OUTER JOIN

Sintassi

```
table1 FULL [ OUTER ] JOIN table2 ON join_condition [, ...].
```

- È equivalente a:
INNER JOIN + LEFT OUTER JOIN + RIGHT OUTER JOIN.
- Non è equivalente a CROSS JOIN!