Query SQL Annidate, EXISTS e Operatori di Raggruppamento

Prof. Fedeli Massimo - IIS Fermi Sacconi Cpia

1 Query SQL annidate

Una **query annidata** (o subquery) è una query SQL scritta all'interno di un'altra query. La subquery è racchiusa tra parentesi tonde ed è valutata prima della query esterna.

2 Schema di esempio

Tabelle utilizzate negli esempi:

STUDENTI

id_studente, nome, classe

VOTI

id_studente, materia, voto

3 Query annidata con IN

Trovare gli studenti che hanno ottenuto almeno un voto pari a 10.

```
SELECT nome
FROM STUDENTI
WHERE id_studente IN (
SELECT id_studente
FROM VOTI
WHERE voto = 10
);
```

4 Subquery con valore singolo

Trovare gli studenti con voto superiore alla media.

```
SELECT nome
FROM STUDENTI S
JOIN VOTI V ON S.id_studente = V.id_studente
WHERE V.voto > (
SELECT AVG(voto)
FROM VOTI
);
```

5 Operatori ANY e ALL

`ANY` richiede che la condizione sia vera per almeno un valore, `ALL` per tutti i valori restituiti dalla subquery.

5.1 Esempio con ANY

```
SELECT DISTINCT nome
FROM STUDENTI S
JOIN VOTI V ON S.id_studente = V.id_studente
WHERE V.voto > ANY (
SELECT voto
FROM VOTI
WHERE materia = 'Matematica'
);
```

5.2 Esempio con ALL

```
SELECT DISTINCT nome
FROM STUDENTI S
JOIN VOTI V ON S.id_studente = V.id_studente
WHERE V.voto > ALL (
SELECT voto
FROM VOTI
WHERE materia = 'Matematica'
);
```

6 La clausola EXISTS

La clausola `EXISTS` è utilizzata per verificare se una subquery restituisce **almeno una riga**.

A differenza di altre subquery:

- non restituisce valori da confrontare;
- non produce insiemi da analizzare;
- risponde esclusivamente alla domanda: *esiste almeno una riga?*

Se la subquery restituisce almeno una riga, la condizione `EXISTS` è vera; se non restituisce alcuna riga, la condizione è falsa.

6.1 EXISTS come quantificatore logico

Dal punto di vista logico, `EXISTS` corrisponde al quantificatore:

esiste almeno un elemento tale che...

Questo lo rende particolarmente adatto a esprimere condizioni del tipo:

- uno studente *ha almeno un voto*;
- uno studente *ha almeno un voto insufficiente*;
- uno studente *ha almeno un voto in una certa materia*.

6.2 Subquery correlata

Nella maggior parte dei casi, `EXISTS` è usato con subquery correlate, cioè subquery che fanno riferimento alla query esterna.

Esempio 1: studenti con almeno un voto

```
SELECT nome
FROM STUDENTI S
WHERE EXISTS (
SELECT *
FROM VOTI V
WHERE V.id_studente = S.id_studente
);
```

La subquery viene valutata per ogni studente. Se esiste almeno un voto associato, lo studente viene selezionato.

Esempio 2: studenti con almeno un voto insufficiente

```
SELECT nome
FROM STUDENTI S
WHERE EXISTS (
SELECT *
FROM VOTI V
WHERE V.id_studente = S.id_studente
AND V.voto < 6
);
```

La condizione è vera solo se esiste almeno un voto minore di 6.

Esempio 3: studenti con almeno un voto di matematica

```
SELECT nome
FROM STUDENTI S
WHERE EXISTS (
SELECT *
FROM VOTI V
WHERE V.id_studente = S.id_studente
AND V.materia = 'Matematica'
);
```

6.3 EXISTS vs IN

Dal punto di vista concettuale:

- `IN` confronta un valore con un insieme;
- `EXISTS` verifica l'esistenza di righe;

- `EXISTS` non è influenzato da valori `NULL`;
- `EXISTS` è spesso più efficiente su tabelle di grandi dimensioni.

6.4 Forma tipica di EXISTS

La struttura tipica è:

```
SELECT ...
FROM tabella_esterna T
WHERE EXISTS (
SELECT *
FROM tabella_interna S
WHERE condizione_di_collegamento
);
```

7 Operatori di raggruppamento

Gli operatori di raggruppamento, detti anche **funzioni di aggregazione**, permettono di eseguire calcoli su insiemi di righe e di restituire un singolo valore per ciascun gruppo.

Essi sono utilizzati frequentemente insieme alla clausola `GROUP BY`, ma possono essere impiegati anche senza raggruppamento, quando l'aggregazione riguarda l'intera tabella.

Le principali funzioni di aggregazione sono:

- `COUNT`
- `SUM`
- `AVG`
- `MIN`
- `MAX`
- `STDDEV`

7.1 COUNT

La funzione `COUNT` restituisce il numero di righe considerate.

Conteggio totale delle righe

```
SELECT COUNT(*)
FROM VOTI;
```

Conta il numero totale di record presenti nella tabella `VOTI`.

Conteggio per gruppo

```
SELECT id_studente, COUNT(*) AS numero_voti
FROM VOTI
GROUP BY id_studente;
```

In questo caso il numero di voti viene calcolato per ciascuno studente.

7.2 SUM

La funzione `SUM` calcola la somma dei valori di una colonna numerica.

Esempio

Calcolare la somma dei voti per ciascuno studente.

```
SELECT id_studente, SUM(voto) AS somma_voti
FROM VOTI
GROUP BY id_studente;
```

`SUM` è utile quando si vogliono ottenere totali o punteggi complessivi.

7.3 AVG

La funzione `AVG` calcola la media aritmetica dei valori di una colonna numerica.

Media globale

```
SELECT AVG(voto) AS media_generale
FROM VOTI;
```

Media per gruppo

```
SELECT materia, AVG(voto) AS media_materia
FROM VOTI
GROUP BY materia;
```

Questa funzione è molto utilizzata per valutazioni, statistiche e confronti.

7.4 MIN

La funzione `MIN` restituisce il valore minimo presente in una colonna.

Esempio

Trovare il voto minimo per ogni materia.

```
SELECT materia, MIN(voto) AS voto_minimo
FROM VOTI
GROUP BY materia;
```

7.5 MAX

La funzione `MAX` restituisce il valore massimo presente in una colonna.

Esempio

Trovare il voto massimo per ogni studente.

```
SELECT id_studente, MAX(voto) AS voto_massimo
FROM VOTI
GROUP BY id_studente;
```

7.6 STDDEV

La funzione `STDDEV` calcola la **deviazione standard**, che misura la dispersione dei valori rispetto alla media.

Significato

- valore basso: voti concentrati vicino alla media;
- valore alto: voti molto variabili.

Esempio

Calcolare la deviazione standard dei voti per ciascuna materia.

```
SELECT materia, STDDEV(voto) AS deviazione_standard
FROM VOTI
GROUP BY materia;
```

La funzione `STDDEV` è particolarmente utile per analisi statistiche e confronti sulla variabilità dei dati.

7.7 Uso combinato di più funzioni

È possibile utilizzare più funzioni di aggregazione nella stessa query.

Esempio

```
SELECT materia,
COUNT(*) AS numero_voti,
AVG(voto) AS media,
MIN(voto) AS minimo,
MAX(voto) AS massimo
FROM VOTI
GROUP BY materia;
```

7.8 GROUP BY: regola fondamentale

Quando si utilizza `GROUP BY`:

- tutte le colonne presenti nel `SELECT` che non sono funzioni di aggregazione devono comparire nel `GROUP BY`;
- ogni riga del risultato rappresenta un gruppo.

7.9 HAVING e funzioni di aggregazione

La clausola `HAVING` consente di filtrare i gruppi in base ai risultati delle funzioni di aggregazione.

Esempio

Trovare le materie con media dei voti superiore a 7.

```
SELECT materia, AVG(voto) AS media
FROM VOTI
GROUP BY materia
HAVING AVG(voto) > 7;
```

7.10 Errori comuni

- usare funzioni di aggregazione nella clausola `WHERE`;
- dimenticare colonne nel `GROUP BY`;
- confondere `WHERE` e `HAVING`;
- interpretare male il significato dei risultati aggregati.