Azzolini Riccardo 2019-10-10

SQL: Join e funzioni di gruppo

1 Operazione di join

Il **join** di due relazioni A e B genera tutte le coppie formate da una tupla di A e una tupla di B *semanticamente legate*.

Esso è il metodo che permette di attraversare le associazioni rappresentate mediante le chiavi esterne.[1]

Semanticamente, il join tra A e B corrisponde a una sequenza di due operazioni:

1. si effettua il *prodotto cartesiano* $A \times B$;
2. si esegue una selezione in base a un **predicato di join**, il quale specifica l'associazione che le tuple del risultato devono verificare, scartando quindi le coppie di tuple di A e B che non sono semanticamente legate.

Lo schema della relazione risultato di un join è dato dall'unione (concatenazione) degli schemi delle relazioni operandi. Quindi, il grado della relazione risultato è uguale alla somma dei gradi delle relazioni operandi.

2 Prodotto cartesiano in SQL

Nel linguaggio SQL, il prodotto cartesiano si esegue specificando più relazioni nella clausola `FROM`. Ad esempio:

```
SELECT * FROM Corso, Docente;
```

Esiste anche una sintassi alternativa, `CROSS JOIN`:[2]

```
SELECT * FROM Corso CROSS JOIN Docente;
```

[1] I join funzionano anche senza vincoli di chiave esterna: questi garantiscono che i collegamenti tra relazioni siano validi quando si inseriscono/modificano/cancellano dati, ma non servono per le interrogazioni.

[2] Nonostante il nome, di per sé `CROSS JOIN` indica solo un prodotto cartesiano, non un join.

3 Join in SQL

Per effettuare un join, si esegue un prodotto cartesiano (specificando più relazioni nella clausola `FROM`) e si aggiunge nella clausola di qualificazione il predicato di join (messo in `AND` con eventuali altre condizioni di ricerca).

3.1 Forma del predicato di join

- Siano S e S' le relazioni di cui si vuole effettuare il join.
- Siano A un attributo di S e A' un attributo di S'.

Un *predicato di join* per S ed S' è definito come:

- $A\,\theta\,A'$, dove θ è un operatore di SQL,
- oppure una combinazione booleana di tali predicati.

3.2 Esempi

Determinare i titoli dei video noleggiati dai clienti della videoteca:

```sql
SELECT titolo
FROM Video, Noleggio
WHERE Video.colloc = Noleggio.colloc; -- predicato di join
```

Determinare i titoli dei video noleggiati il 15 marzo 2006 dal cliente con codice 6635:

```sql
SELECT titolo
FROM Video, Noleggio
WHERE Video.colloc = Noleggio.colloc -- predicato di join
  AND codCli = 6635                  -- condizione di ricerca
  AND dataNol = DATE '15-Mar-2006';  -- condizione di ricerca
```

4 Sintassi alternative

Il join può essere specificato anche con alcune sintassi alternative, che facilitano la scrittura delle query in quanto separano il predicato di join da eventuali condizioni di ricerca:

- `JOIN ON` permette di specificare il predicato di join direttamente all'interno della clausola `FROM`:

  ```
  <nome relazione> JOIN <nome relazione> ON <predicato>
  ```

- `JOIN USING` permette di specificare una lista di nomi di colonne che devono avere valori uguali nelle due relazioni:

  `<nome relazione> JOIN <nome relazione> USING (<lista nomi colonne>)`

  Di conseguenza, si può applicare solo se le due relazioni hanno colonne con lo stesso nome.

- `NATURAL JOIN` richiede l'uguaglianza dei valori di tutte le colonne che hanno lo stesso nome nelle due relazioni.

  `<nome relazione> NATURAL JOIN <nome relazione>`

  In pratica, questa sintassi si usa raramente, sia perché introduce il rischio di commettere errori, sia perché ci potrebbero essere più collegamenti tra le stesse due relazioni, e in tal caso non sarebbe possibile specificare quale di questi considerare.

Nel caso di `JOIN USING` e `NATURAL JOIN`, le colonne con lo stesso nome nelle due relazioni (in particolare, quelle elencate nel `JOIN USING`, oppure tutte se si usa il `NATURAL JOIN`) vengono incluse un'unica volta nel risultato.

4.1 Esempio

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Selezionare padre e madre di ogni persona:

```sql
SELECT Paternita.Figlio, Padre, Madre
FROM Maternita, Paternita
WHERE Paternita.Figlio = Maternita.Figlio;
-- oppure
SELECT Paternita.Figlio, Padre, Madre
FROM Maternita
  JOIN Paternita ON Paternita.Figlio = Maternita.Figlio;
-- oppure
SELECT *
FROM Maternita
  JOIN Paternita USING (Figlio);
```

```sql
-- oppure
SELECT *
FROM Maternita NATURAL JOIN Paternita;
```

Risultato

Figlio	Padre	Madre
Olga	Luigi	Anna
Filippo	Luigi	Anna
Andrea	Franco	Maria
Aldo	Franco	Maria

5 Alias di relazione

Quando una stessa relazione è coinvolta più volte in un join, è necessario assegnare a essa nomi diversi. A tale scopo, nella clausola `FROM` si possono definire degli **alias di relazione**, con la sintassi:

```
<nome relazione> <alias>
```

5.1 Esempi

Selezionare le persone che guadagnano più dei rispettivi padri, mostrando nome, reddito, e reddito del padre:

```sql
SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p, Paternita, Persone f
WHERE p.Nome = Padre
  AND f.Nome = Figlio
  AND f.Reddito > p.Reddito;
-- oppure, con la sintassi JOIN ON:
SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p
  JOIN Paternita ON p.Nome = Padre
  JOIN Persone f ON f.Nome = Figlio
WHERE f.Reddito > p.Reddito;
```

Selezionare i codici dei video che sono stati noleggiati almeno due volte dallo stesso cliente:

```sql
SELECT DISTINCT X.colloc
FROM Noleggio X, Noleggio Y
WHERE X.colloc = Y.colloc
```

```
AND X.codCli = Y.codCli
AND X.dataNol <> Y.dataNol;
```

6 Outer join

In S `JOIN` T non si ha traccia delle tuple di S che non corrispondono ad alcuna tupla di T (e viceversa). Se questo non è ciò che si desidera, bisogna invece usare l'operatore `OUTER JOIN`, che aggiunge al risultato anche le tuple di S, di T, o di entrambe che non hanno partecipato al join, mettendo a `NULL` i valori delle colonne appartenenti all'altra relazione (che sarebbero presi dalla tupla associata, la quale però non esiste).

Per contrasto, l'operatore `JOIN` originario è anche detto `INNER JOIN`.

6.1 Varianti

L'operatore S `OUTER JOIN` T ha tre varianti:

- `LEFT OUTER JOIN`: le tuple di S che non partecipano al join vengono completate con `NULL` e aggiunte al risultato;
- `RIGHT OUTER JOIN`: le tuple di T che non partecipano al join vengono completate con `NULL` e aggiunte al risultato;
- `FULL OUTER JOIN`: sia le tuple di S che quelle di T che non partecipano al join vengono completate con `NULL` e aggiunte al risultato.

Inoltre, anche per gli `OUTER JOIN` si possono usare le sintassi `JOIN ON`, `JOIN USING` e `NATURAL JOIN`.

6.2 Esempi

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Selezionare il padre e, se nota, la madre di ogni persona:

```sql
SELECT Paternita.Figlio, Padre, Madre
FROM Paternita
  LEFT OUTER JOIN Maternita
    ON Paternita.Figlio = Maternita.Figlio;
-- oppure
SELECT Figlio, Padre, Madre
FROM Paternita
  LEFT OUTER JOIN Maternita USING (Figlio);
```

Risultato

Figlio	Padre	Madre
Franco	Sergio	NULL
Olga	Luigi	Anna
Filippo	Luigi	Anna
Andrea	Franco	Maria
Aldo	Franco	Maria

Per ogni video contenente un film di Tim Burton, visualizzare la sua collocazione, il titolo, e i codici dei clienti che l'hanno *eventualmente* noleggiato:

```sql
SELECT colloc, titolo, codCli
FROM Film
  NATURAL JOIN Video
  NATURAL LEFT OUTER JOIN Noleggio
WHERE regista = 'tim burton';
```

7 Funzioni di gruppo

Nelle espressioni della clausola di proiezione (`SELECT`) si possono avere anche funzioni che calcolano valori a partire da insiemi di tuple. Esse sono dette **funzioni di gruppo**:

- operano su insiemi di valori;
- producono come risultato un unico valore (detto *aggregato* perché dipende da tutti i valori in input), quindi la relazione risultato di un'interrogazione con funzioni di gruppo contiene una singola tupla;
- nella clausola `SELECT` si possono specificare più funzioni di gruppo contemporaneamente, anche su attributi diversi, ma *non* si può avere un misto di funzioni di gruppo e attributi/espressioni "normali" (perché il risultato è una singola tupla, quindi non può contenere tutti i valori di una colonna).

Le principali funzioni di gruppo previste da SQL sono:

- `MAX`: determina il massimo di un insieme di valori;
- `MIN`: determina il minimo di un insieme di valori;
- `SUM`: calcola la somma dei valori di un insieme;
- `AVG`: calcola la media dei valori di un insieme;
- `COUNT`: determina la cardinalità di un insieme.

Ad eccezione di `COUNT`, queste funzioni si applicano solo su insiemi di valori semplici, e non su insiemi di tuple. In particolare, `SUM` e `AVG` sono definite solo per insiemi di valori numerici.

L'insieme di valori è denotato nel caso più semplice dal nome di una colonna, ma in generale può essere una qualsiasi espressione contenente nomi di colonne. Inoltre, queste funzioni possono a loro volta essere usate all'interno di espressioni aritmetiche (composte da valori costanti o altre funzioni di gruppo, ma non nomi di colonne).

Eventuali valori nulli vengono eliminati dall'insieme prima del calcolo della funzione di gruppo. Se l'insieme dei valori è vuoto (dopo l'eventuale rimozione dei valori nulli), `COUNT` restituisce 0, mentre le altre funzioni restituiscono `NULL`.

7.1 Funzione `COUNT`

La funzione di gruppo `COUNT` può avere come argomento:

- il carattere `*`, per contare il numero di tuple;
- un nome di colonna o un'espressione, per contare il numero di tuple per le quali tale colonna/espressione *non ha valore nullo*.

7.2 `DISTINCT`

Tutte le funzioni di gruppo possono essere usate con il qualificatore `DISTINCT`, per eliminare eventuali valori duplicati prima di applicare la funzione.

L'eliminazione dei duplicati è significativa (cioè ha effetto sul risultato) solo per le funzioni `SUM`, `AVG` e `COUNT` (e, comunque, si usa soprattutto per `COUNT`).

7.3 Esempi

Selezionare il minimo, la media, e il massimo delle valutazioni di tutti i film:

```
SELECT MIN(valutaz), AVG(valutaz), MAX(valutaz)
FROM Film;
```

Selezionare il minimo, la media, e il massimo delle valutazioni dei film di genere drammatico:

```
SELECT MIN(valutaz), AVG(valutaz), MAX(valutaz)
FROM Film
WHERE genere = 'drammatico';
```

Contare il numero di persone:

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

Contare il numero di valori distinti del reddito delle persone:

```
SELECT COUNT(DISTINCT Reddito)
FROM Persone;
```

Contare il numero di figli di Franco:

```
SELECT COUNT(*) AS NumFigliDiFranco
FROM Paternita
WHERE Padre = 'Franco';
```

7.4 Valutazione

Le funzioni di gruppo sono valutate dopo l'applicazione di tutti i predicati della clausola di selezione (`WHERE`). Di conseguenza, esse non possono essere utilizzate in tale clausola (se non all'interno di sotto-interrogazioni).

Se, ad esempio, si vogliono determinare il titolo e il regista del film drammatico di valutazione minima, l'interrogazione

```
SELECT titolo, regista
FROM Film
WHERE genere = 'drammatico' AND valutaz = MIN(valutaz);
```

non è corretta: serve invece una sotto-interrogazione.