Appunti Basi di Dati

Luca Seggiani

27 Marzo 2024

1 Subquery

Le subquery (query annidate) permettono di incapsulare query in altre query, in modo correlato o non correlato. Presentano un modo alternativo di risolvere i problemi su cui normalmente si userebbe il join: infatti per ogni problema che si può risolvere usando il join esiste sempre un corrispettivo che usa le subquery. Chiamiamo outer query la query più esterna, cioè quella che incapsula, e inner query la subquery incapsulata.

Subquery non correlate

Le subquery non correlate (*noncorrelated*) sono incapsulate nel WHERE, e permettono di ottenere un certo result set intermedio, che verrà poi usato dall'outer query per calcolare il result set finale. Il result set della subquery non correlata non dipende dall'outer query. Poniamo ad esempio di voler indicare nome, cognome e parcella degli ortopedici che hanno effettuato almeno una visita nell'anno 2013. Definiremo allora la subquery:

```
SELECT V.Medico
FROM Visita V
WHERE YEAR(V.Data) = 2013
```

che innesteremo poi in:

```
SELECT M.Nome, M.Cognome, M.Parcella
FROM Medico M
WHERE M.Specializzazione = 'Ortopedia'
  AND M.Matricola IN  (
                      SELECT V.Medico
                      FROM VIsita V
                      WHERE YEAR(V.Data) = 2013
                      );
```

La parola chiave IN permette di controllare la presenza dell'attributo M.Matricola nel result set della subquery. La subquery calcolerà quindi un result set composto da tutte le visite fatte nel 2013. L'outer query userà la subquery, confrontandola con tutti i medici specializzati in ortopedia la cui matricola trova

un riscontro nel result set calcolato dalla subquery. A questo punto si restituiscono gli attributi richiesti. Il meccanismo è del tutto analogo a quello di un join naturale fra le tabelle "Medico" e "Visita". Notiamo che non è necessario alcuna istruzione DISTINCT, in quanto ogni record della tabella Medico viene considerato comunque una volta sola, e cercato nel result set delle visite.
Avremmo potuto porre la stessa query usando un comune join, ad esempio come:

```
SELECT M.Nome, M.Cognome, M.Parcella
FROM Medico M
  INNER JOIN Visita V ON V.Medico = M.Matricola
WHERE YEAR(V.Data) = 2013
  AND Medico.Specializzazione = 'Ortopedico';
```

Si può anche usare la negazione di quanto appena fatto, usando la parola chiave NOT IN. Ad esempio, se vogliamo indicare i cognomi dei pazienti che non appartengono anche a un medico, abbiamo la subquery:

```
SELECT M.Cognome
FROM Medico M
```

innestata in:

```
SELECT DISTINCT P.Cognome
FROM Paziente P
WHERE P.Cognome NOT IN  (
                        SELECT M.Cognome
                        FROM Medico M
                        );
```

Analogamente a prima, esisterà una versione che usa il join:

```
SELECT DISTINCT P.Cognome
FROM Paziente P
  LEFT OUTER JOIN Medico M ON
    M.Cognome = P.Cognome
WHERE M.Cognome IS NULL;
```

Annidamento multiplo

Non c'è teoricamente limite al numero di subquery che si possono innestare l'una dentro l'altra. A volte può infatti tornare utile avere innesti multipli (e frontali!). Vediamo come ottenere il numero di tutti i pazienti di Siena mai visitati da pazienti. Innanzitutto si trovano i medici specializzati in ortopedia:

```
SELECT M.Matricola
FROM Medico M
WHERE M.Specializzazione = 'Ortopedia'
```

poi i pazienti visitati da suddetti medici:

```
SELECT V.Paziente
FROM Visita V
WHERE V.Medico IN (...)
```

e si rimuovono da una tabella dei soli pazienti senesi:

```
SELECT COUNT(*)
FROM Paziente P
WHERE P.Citta = 'Siena'
  AND P.CodFiscale NOT IN (
    SELECT V.Paziente
    FROM Visita V
    WHERE V.Medico IN (
      SELECT M.Matricola
      FROM Medico M
      WHERE M.Specializzazione = 'Ortopedia'
    )
  );
```

Subquery scalari

Una subquery scalare produce, invece che un insieme di record che poi verrà controllato con IN e NOT IN, un singolo record di un singolo attributo, come ad esempio farebbe la funzione COUNT(). A questo punto, il valore scalare prodotto può essere controllato con i vari operatori di confronto. Vogliamo ad esempio indicare il numero degli otorini aventi parcella più alta della media delle parcelle della loro specializzazione. Troviamo innanzitutto il la parcella media degli otorini (un valore scalare):

```
SELECT AVG(Medico.Parcella)
FROM Medico M
WHERE M.Specializzazione = 'Otorinolaringoiatria'
```

e la useremo in un'outer query:

```
SELECT COUNT(*)
FROM Medico M1
WHERE M1.Parcella >  (
                      SELECT AVG(Medico.Parcella)
                      FROM Medico M
          WHERE M.Specializzazione = 'Otorinolaringoiatria'
                      );
```

Un caso interessante: mettiamo di voler trovare le entrate complessive generate dai cardiologi della clinica negli ultimi 2 anni. Ciò risulta semplice con un join:

```
SELECT SUM(M.Parcella) AS IncassoTotale
FROM Visita V
```

```
INNER JOIN Medico M ON V.Medico = M.Matricola
WHERE V.Data > CURRENT_DATE - INTERVAL 12 YEAR
  AND M.Specializzazione = 'Cardiologia'
```

La versione con subquery è invece più complessa, richiedendo una subquery nel SELECT:

```
SELECT SUM((
  SELECT M.Parcella
  FROM Medico M
  WHERE M.Matricola = V.Medico
     AND M.Specializzazione = 'Cardiologia'
)) AS IncassoTotale
FROM Visita V
WHERE V.Data > CURRENT_DATE - INTERVAL 12 YEAR
```

Notiamo che la subquery è correlata, come vedremo nella prossima lezione.