Appunti Basi di Dati

Luca Seggiani

1 Marzo 2024

Riprendiamo la trattazione delle operazioni che permettono l'unione (join) di più tabelle.

Natural join

Il join naturale combina i record della prima tabella con i record della seconda tabella aventi valori uguali su tutti gli attributi omonimi. Ad esempio, in pseudocodice, ammettendo che la tabella visita abbia un attributo ominimo ad un'altro attributo sulla tabella medico:

```
SELECT M.Nome, M.Cognome
FROM Visita V
  NATURAL JOIN
  Medico M
```

Nota bene: **tutti** gli attributi omonimi dovranno avere valori identici, ed a quel punto il record della tabella di provenienza verrà unito alla tabella su cui lavoriamo una sola volta.

External join

Il join esterno combina tutti i record della prima tabella con tutti i record della seconda che soddisfano una condizione, mantenendo tutti i record in una delle tabelle, a differenza del join esterno che invece scarterebbe i record non appaiati con nessuno dei record della tabella madre. Il join esterno può essere pensato come una sorta di prodotto cartesiano fra tabelle. I record spaiati che verrano uniti avranno tutti valori NULL sugli attributi della tabella di provenienza (inesistenti). Poniamo ad esempio di voler indicare le visite effettuate da medici che non lavorano più presso la clinica. Supponiamo quindi che loro anagrafica sia stata quindi eliminata dal database.

```
SELECT V.*
FROM Visita V
  LEFT OUTER JOIN
  Medico M ON V.Medico = M.Matricola
WHERE M.Matricola IS NULL;
```

Così facendo potremo unire tutti i record di tutti i medici, anche quelli che non lavorano più nella mia clinica. Controllando a questo punto chi di questi medici ha elemento chiave (si controlla, per convenzione, proprio l'elemento chiave) NULL, otterremo le visite svolte dai suddetti. Notiamo infine la differenza fra join esterno destro e join esterno sinistro: il join esterno sinistro mantiene tutti i record della tabella di sinistra mettendo NULL a destra per i record non appaiati. Il join esterno destro fa la cosa opposta, quindi mantenendo i record a destra e inserendo NULL a sinistra.

Query con join e condizioni sui record

Possiamo a questo punto combinare il FROM con instruzione di join allo WHERE con condizioni di selezione dei record d'interesse applicate alla tabella *dopo* il join. Ad esempio, se voglio ottenere matricola, cognome e specializzazione dei medici che hanno visitato almeno un paziente il giorno 1 Marzo 2013

```
SELECT DISTINCT M.Matricola, M.Cognome, M.Specializzazione
FROM Medico M
  INNER JOIN
  Visita V ON M.Matricola = V.Medico
WHERE V.Data = '2013-01-03'
```

Unioni multiple

Il join multiplo può essere effettuato su più di 2 tabelle. Diciamo che oltre al medico, la nostra tabella visita contiene un'attributo corrispondente al paziente visitato. Vogliamo allora ottenere nomi e cognomi di tutti i pazienti visitati da un certo medico. Notiamo che il nome del medico sta nella sottotabella corrispondente al medico, e il nome del paziente sta nella sottotabella corrispondente al paziente, ovvero dovremmo unire non una ma due tabelle alla nostra tabella madre. In codice:

```
SELECT DISTINCT P.Nome, P.Cognome
FROM Paziente P
  INNER JOIN
  Visita V ON P.CodFiscale = V.Paziente
  INNER JOIN
  Medico M ON V.Medico = M.Matricola
WHERE M.Nome = 'Rino'
  AND M.Cognome = 'Neri';
```

notiamo che in questo caso, per assicurare la sicurezza dell'operazione, nome e cognome del medico dovrebbero essere una cosiddetta *chiave candidata*, ovvero un insieme di attributi sempre diversi in tutti i record (come la chiave primaria ma senza lo status di chiave primaria).

Si noti inoltre di come gli alias (P V e M) nell'esempio precedente hanno il compito di evitare ambiguità su attributi con lo stesso nome ma appartenenti

a tabelle diverse. Ulteriore esempio, supponiamo di voler indicare nome e cognome dei pazienti visitati nel mese di dicembre 2013, e il nome e cognome dei medici che li hanno visitati:

```
SELECT DISTINCT P.Nome AS NomePaziente, P.Cognome AS
    CognomePaziente
                M.Nome AS NomeMedico, M.Cognome AS
    CognomeMedico
FROM Paziente P
  INNER JOIN
  Visita V ON P.CodFiscale = V.Paziente
  INNER JOIN
  Medico M ON V.Medico = M.Matricola
WHERE YEAR(V.Data) = 2013
  AND MONTH(V.Data) = 12
```

Self join

Il self join combina le righe di una tabella con righe della stessa tabella. Più propriamente, non esiste una keyword SELF JOIN, ma possiamo sfruttare le possibilità dell'INNER JOIN. Diciamo ad esempio di volere il codice fiscale dei pazienti che sono stati visitati più di una volta da uno stesso medico della clinica, nel mese corrente:

```
SELECT DISTINCT V1.Paziente
FROM Visita V1
  INNER JOIN
  Visita v2 ON (
                V2.Medico = V1.Medico
                AND V2.Paziente = V1.Paziente
                AND V2.Data<> V1.Data
                )
WHERE MONTH(V1.Data) = MONTH(CURRENT_DATE)
  AND YEAR(V1.Data) = MONTH(CURRENT_DATE)
  AND MONTH(V2.Data) = MONTH(CURRENT_DATE)
  AND YEAR(V2.Date) = YEAR(CURRENT_DATE);
```

qui unisco alla mia tabella visita un record proveniente dalla stessa tabella quando medico, paziente corrispondono e la data differisce. A questo punto seleziono soltanto i record della nuova tabella creata che hanno date diverse fra le due visite. Si noti che per n visite, il paziente comparirà nel result set n - 1 volte (da cui il DISTINCT), ovvero il join accade in una sola direzione.

Common Table Expression

Le common table expression (CTE) sono result set dotati di identificatori che possono essere usati prima di una query per costruire risultati intermedi. Si scrivono, separate da virgole, prima della query che li usa, tramite la parola chiave WITH e separati da virgole. Sono parti di codice i cui risultati vengono stoccati e poi restituiti alle query che nli usano. Ad esempio:

```
WITH
  name1 AS (query1)
  name2 AS (query2)
  ...
  nameN AS (queryN)
query finale;
```

Diciamo per esepmio, di volere indicare il numero di pazienti di Siena mai visitati da ortopedici. Avrò allora la CTE che trova tutti gli ortopedici:

```
WITH ortopedici AS
  (
  SELECT M.Matricola AS Medico
  FROM Medico M
  WHERE M.Specializzazione = 'Ortopedia'
  )
```

poi la CTE che trova tutti i pazienti visitati da tali ortpedici:

```
paz_visitati_ortopedici AS
  (
  SELECT V.Paziente AS CodFiscale
  FROM Visita V NATURAL JOIN Ortopedici O
  )
```

infine ottengo, usando le CTE precedenti, tutti i pazienti senesi:

```
SELECT COUNT(*)
FROM Paziente P
  NATURAL LEFT OUTER JOIN
  paz_visitati_ortopedici PVO
WHERE Citta = 'Siena'
  AND PVO.Matricola IS NULL
```