Appunti Basi di Dati

Luca Seggiani

29 Febbraio 2024

Gestione delle date in SQL
Come per tutti i sistemi UNIX, le date in SQL vengono rappresentate come una distanza in secondi dalla mezzanotte del primo gennaio 1970 (il cosiddetto tempo di riferimento EPOCH, *EPOCH reference*).

Formato di data Esistono 2 tipi di dato data in SQL: DATE (formato YYYY-MM-DD) e TIMESTAMP (formato YYYY-MM-DD HH:MM:SS). La manipolazione di date è possibile attraverso la funzione DATE_FORMAT, che consente di cambiare il formato o effettuare manipolazioni d'utilità. Ad esempio potrò avere:

```
SELECT Matricola, DATE_FORMAT(DataLaurea, '%d|%m|%Y, %w')
FROM Studente
WHERE DataLaurea > 12-7-2004
```

che prende tutti gli studenti laureati dopo il 12 luglio 2004 e ne riporta il nome di matricola e la data formattata come DD-MM-YYYY + settimana. ancora, per includere ad esempio gli studenti laureati di mercoledì:

```
SELECT Matricola
FROM Studente
WHERE DATE_FORMAT(DataLaurea, '%w') = 3
```

dove il '%w' restituisce il giorno della settimana di DataLaurea. Qualsiasi ulteriore informazione sui formati delle date può essere trovata al link:
`https://www.w3schools.com/sql/func_mysql_date_format.asp`

Poniamoci adesso il problema di restituire i cognomi di tutti gli studenti laureati 5 anni fa, ovvero con un offset di 5 anni rispetto alla data attuale. Servirà chiaramente un qualche riferimento alla data di oggi, che l'SQL fornisce con la parola chiave CURRENT_DATE. Potrò quindi avere:

```
SELECT DISTINCT Cognome
FROM Studente
WHERE DataLaurea IS NOT NULL
  AND YEAR(DataLaurea) = YEAR(CURRENT_DATE) - 5
```

si noti che abbiamo usato l'operatore "-": l'SQL fornisce anche la funzione DATEDIFF(dataRecente, dataRemota), che restituisce il numero di giorni che separano due date. Nota bene: somme e sottrazioni di date non hanno senso: tutto quello che possiamo fare è sommare i singoli valori (anno, giorno, ecc...) di date diverse. Per qualsiasi altro caso occorre usare DATEDIFF(). Possiamo avere ad esempio:

```
SELECT Matricola, DATEDIFF('2005-07-15', DataIscrizione)
FROM Studente
WHERE DataIscrizione < '2005-07-15'
  AND DataLaurea > '2005-07-15'
```

che restituirà la matricola e giorni da quando si erano iscritti degli studenti ad oggi laureati e che non si erano ancora laureati il 15 Luglio 2005.
Inoltre, per sommare e sottrarre lassi di tempo a date possiamo usare le funzioni DATE_ADD e DATE_SUB. I suddetti lassi andranno espressi con la parola chiave INTERVAL:

```
INTERVAL NumeroIntero [YEAR|MONTH|DAY]
```

riportiamo ad esempio matricola e mese di iscrizione degli studenti che si sono laureati dopo cinque anni esatti dal giorno dell'iscrizione:

```
SELECT Matricola, MONTH(DataIscrizione)
FROM Studente
WHERE DataLaurea = DATE_ADD(DataIscrizione, INTERVAL 5 YEAR)
```

quando la somma è definita fra date e intervalli, possiamo usare anche solo l'operatore "+":

```
  AND DataLaurea = DataIscrizione + INTERVAL 5 YEAR
```

esistono poi diverse altre funzioni di utilità sulle date, che possono essere trovate nel sito precedentemente citato.

Operatori di aggregazione

Gli operatori di aggregazione permettono di fare determinati calcoli i cui operandi sono i valori assunti da un attributo in un insieme di record (ovvero conteggio, somma, minimo / massimo, media), e di collassarli in un unico attributo numerico. Gli operatori di aggregazione sono disponibili solamente nel SELECT, in quanto il WHERE non ha alcuna visibilità a livello globale della tabella su cui lavora, ma solo a livello di record.

Conteggio

Iniziamo col contare il numero di righe di una tabella o di un suo sottoinsieme. Definita la tabella di una semplice realtà medica, abbiamo:

```
SELECT COUNT(*) AS VisitePrimoMarzo
FROM Visita
WHERE Data = '2013-03-01'
```

per trovare il numero di visite effettuate in data 1 marzo 2013. La parola chiave AS serve solamente per rinominare la tabella generata dalla funzione COUNT(). Possiamo effettuare anche, anzichè il conteggio delle righe, il conteggio dei valori diversi assunti da un attributo. Ad esempio, se cercassimo il numero di pazienti visitati nel mese di marzo 2013, visto che lo stesso paziente potrà essere stato visitato più volte in un solo mese, dovremmo fare:

```
SELECT COUNT(DISTINCT Paziente) as PazientiMarzo
FROM Visita
WHERE MONTH(Data) = '03'
  AND YEAR(Data) = '2013'
```

Somma

Posso sommare i valori numerici di degli attributi di più record usando SUM(). Supponiamo di voler calcolare, data una tabella del reddito di più persone, il reddito totale di una sola famiglia:

```
SELECT SUM(Reddito) AS RedditoTotale
FROM Paziente
WHERE Cognome = 'Lepre'
```

Media

Allo stesso modo, potrò calcora il reddito medio:

```
SELECT AVG(Reddito) AS RedditoMedio
FROM Paziente
WHERE Cognome = 'Lepre'
```

Minimo / Massimo

Potrò inoltre calcolare valori massimi e minimi:

```
SELECT MIN(Reddito)
FROM Paziente

SELECT MAX(Reddito)
FROM Paziente
```

A questo punto, se volessimo cercare il reddito massimo, e il nome e cognome di chi lo detiene, incontreremmo un'ostacolo: non si può infatti ottenere un qualsiasi altro attributo da un insieme risultato ormai collassato ad un solo valore numerico. Non otterremmo nulla dal codice:

```
SELECT MAX(Reddito), Nome, Cognome
FROM Paziente
```

Riassumendo: non si possono affiancare agli operatori di aggregazione i nomi di attributi ormai collassati.

Query su più tabelle

In un database le informazioni sono spesso frammentate su più tabelle. Questo aiuta a evitare ridondanze, anomalie, ed a vere la possibilità di distribuire i dati. Ad esempio, immaginiamo il database di una clinica medica, che dovrà quindi immagazinare i dati di pazienti, dottori, visite mediche ecc... Potremo allora definire più tabelle separate per ciascuna di queste categorie di dati, ognuna con i propri specifici attributi. A questo punto, ogni tabella potrà contenere come attributi altre tabelle, o meglio record provienienti da altre tabelle in qualche modo "collassati" nel singolo attributo della tabella. Questo meccanismo si concrettizza immagazzinando nella tabella l'informazione minimale per ritrovare il record desiderato nella tabella d'appartenenza, ovvero riportandone soltanto la chiave (che sappiamo essere diversa per ogni record archviato). La successiva esplosione della chiave fino al record completo nella tabella d'appartenenza viene effettutato in SQL attraverso le operazioni di unione (join).

Inner join

L'inner join trova il record nella tabella d'appartenenza che corrisponde alla chiave nella tabella su cui lavoriamo, e semplicemente lo affianca. Ad esempio, magari vogliamo indicare nome e cognome dei medici che hanno effettuato, nella nostra clinica, almeno una visita. Dovremmmo allora prendere la nostra tabella visita, che conterrà la chiave di un medico nella tabella medico, che andremo quindi a sostituire con il record completo del medico corrispondente. In codice:

```
SELECT DISTINCT M.Nome, M.Cognome
FROM Visita V
  INNER JOIN
  Medico M ON V.MEDICO = M.Matricola
```