SQL: Interrogazioni base

Prof. Viglietti Francesco

Www.in4matika.altervista.org

Introduzione

Abbiamo fin qui visto come progettare, creare e popolare una base di dati usando i linguaggi DDL e DML dello standard SQL, nel nostro caso particolare MySQL. In questa parte e nella successiva conoscerete l'altro aspetto di questo processo: come ritrovare i dati in un DB.

Si tratteranno quindi alcuni aspetti del comando SELECT di SQL, che è il comando usato per selezionare le righe in una o più tabelle del DB.

In questa parte vedremo come selezionare le righe da una sola tabella, mentre nella parte successiva, si affronteranno le interrogazioni più complesse, in particolare quelle che usano più tabelle, i diversi tipi di join e le sottointerrogazioni.

Introduzione

Partiamo dai seguenti argomenti:

- una panoramica del comando SELECT
- le interrogazioni semplici
- come selezionare particolari colonne
- come usare la condizione WHERE per selezionare delle righe specifiche
- come usare la condizione GROUP BY
- come selezionare gruppi di righe con HAVING
- come ordinare i risultati di una ricerca con ORDER BY
- come restringere i risultati di una ricerca con LIMIT.

Il comando ***SELECT***

Il comando SELECT ha la seguente sintassi generale:

SELECT colonne
FROM tabelle
[WHERE condizioni]
[GROUP BY gruppo]
[HAVING condizioni sul gruppo]
[ORDER BY criterio d'ordine sulle colonne]
[LIMIT limiti];

Questa non è la sintassi completa, che vedremo nel prossimo modulo, ma vi dà un'idea generale della struttura del comando.

Il comando SELECT ha molte proprietà opzionali. Potete usarle oppure no, a vostra scelta, ma se lo fate esse devono seguire l'ordine in cui appaiono nell'elenco.

Interrogazioni semplici

Un esempio della forma più semplice del comando SELECT è:

SELECT * FROM dipartimento;

Facciamo riferimento alla solita base di dati di esempio impiegati, popolata con i dati che abbiamo inserito nel modulo precedente.

Se facessimo eseguire questa interrogazione sui dati presenti nella base di dati degli impiegati, otterremmo qualcosa del tipo:

```
+----------------+-------------------------+
| IDdipartimento | nome                    |
+----------------+-------------------------+
|             42 | Finanza                 |
|            128 | Ricerca e Sviluppo      |
|            129 | Risorse Umane           |
|            130 | Vendite                 |
+----------------+-------------------------+

4 rows in set (0.00 sec)
```

Interrogazioni semplici

L’interrogazione ha selezionato tutti i dati nella tabella voluta, cioè tutte le righe e tutte le colonne della tabella dipartimento.

Possiamo provare il comando su un’altra tabella, ad esempio selezioniamo tutte le righe e tutte le colonne della tabella **impiegatoCompetenze.**

Naturalmente, la potenza di DB relazionale non consiste nel eseguire solo query per la restituzione di tutti i dati, ma nel permettere di fare delle ricerche e trovare determinate e particolari informazioni.

Selezione di particolari colonne

Il prossimo passo consiste nel restringere il numero di colonne restituito. L'asterisco (*) nell'interrogazione precedente indica "tutte le colonne nella tabella".

Al posto dell'asterisco si può elencare un insieme di colonne che specifico. L'elenco può essere formato da una colonna, da un sottoinsieme di colonne della tabella o persino dall'intero insieme di colonne nell'ordine preferito. Si Deve specificare l'elenco separando i nomi delle colonne con una virgola.

Ad esempio, l'interrogazione seguente seleziona soltanto le colonne nome ed IDimpiegato dalla tabella impiegato:

SELECT nome, IDimpiegato

FROM impiegato;

Selezione di particolari colonne

Se facessimo seguire l'interrogazione sulla base di dati degli impiegati otterremmo un risultato simile a:

```
+---------------+-------------+
| nome          | IDimpiegato |
+---------------+-------------+
| Ajay Patel    |        6651 |
| Nora Edwards  |        7513 |
| Candy Burnett |        9006 |
| Ben Smith     |        9842 |
+-------------+---------------+

4 rows in set (0.00 sec)
```

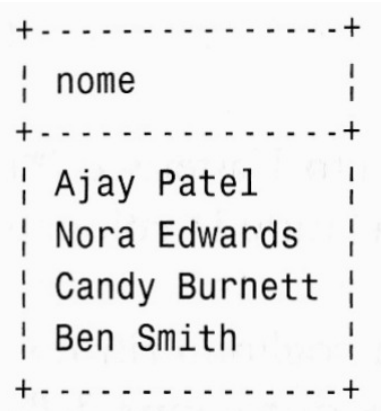
Vediamo che appaiono soltanto le due colonne specificate nella query, e sono visualizzate nell'ordine in cui sono state richieste, piuttosto che nell'ordine in cui compaiono nella base di dati.

Specifica DB e tabelle

Un’altra istruzione permette di specificare in modo assoluto a quale base di dati e a quale tabella si sta riferendo. Ad esempio, si può far riferimento alla colonna nome della tabella impiegato con la notazione impiegato.nome:

SELECT impiegato.nome FROM impiegato;

Si otterrebbe un risultato simile a quello in figura.



```
+--------------+
| nome         |
+--------------+
| Ajay Patel   |
| Nora Edwards |
| Candy Burnett|
| Ben Smith    |
+--------------+
```

Allo stesso modo, è possibile indicare di quale tabella di quale base di dati si sta parlando, ad esempio:

SELECT nome FROM impiegati.impiegato;

Il risultato dovrebbe essere lo stesso della query precedente.

Specifica DB e tabelle

In questo esempio ci si riferisce esplicitamente alla tabella impiegato che si trova nella base di dati impiegati. La notazione è ***nomeDB.tabella.***

Volendo, può essere specificato anche il database e la tabella a cui appartiene la colonna. Lo stesso esempio potrebbe essere riscritto usando la notazione nomeDB.tabella.colonna:

SELECT impiegati.impiegato.nome

FROM impiegato;

Questa sintassi non è molto utile per queste semplici interrogazioni, ma lo è in interrogazioni più complesse permettendo di non essere ridondanti nello specificare quale informazioni si cercano.

Alias

A questo punto occorre trattare il concetto di alias di tabella e di colonna. È possibile rinominare le colonne o le espressioni in un comando SELECT e il nuovo nome può essere usato nella visualizzazione del risultato. Ad esempio, si può scrivere la seguente interrogazione:
SELECT nome AS nomeImpiegato FROM impiegato;
In questo esempio, è stata rinominata la colonna nome in nomeImpiegato soltanto nel contesto dell'interrogazione. Il risultato dell'interrogazione eseguita sulla base di dati degli impiegati sarà simile a:

Si può vedere che nel risultato l'etichetta della colonna ***nome*** è visualizzata come ***nomeImpiegato***.
L'identificatore nomeImpiegato è detto alias

```
+---------------+
| nomeImpiegato |
+---------------+
| Ajay Patel    |
| Nora Edwards  |
| Candy Burnett |
| Ben Smith     |
+---------------+

4 rows in set (0.01 sec)
```

Alias

Esistono alcune regole riguardo cosa si possa o non possa fare con gli alias.
Si possono anche usare gli alias per le tabelle, ad esempio:

SELECT i.nome FROM impiegato AS i;

Si otterrà lo stesso risultato scrivendo l'interrogazione senza gli alias. Questa notazione sarà utile quando, si eseguiranno le interrogazioni su più tabelle.
Negli ultimi due esempi, la parola chiave AS è opzionale. Si potrebbe scrivere semplicemente:

SELECT nome nomeImpiegato FROM impiegato;

oppure

SELECT i.nome FROM impiegato i;

Come si può vedere, esistono parecchi modi per scrivere la stessa interrogazione SQL. Lo stile di programmazione individuale può cambiare in SQL come in altri linguaggi.

Condizione WHERE per selezionare determinate righe

Ora vediamo come selezionare alcune righe particolari, che soddisfino alcuni determinati criteri di ricerca. Diventa importante anche se si devono ritrovare alcune righe utili da una tabella molto più grande.
Si può fare grazie alla condizione WHERE del comando SELECT.
Un semplice esempio è il seguente:
SELECT IDimpiegato, nome FROM impiegato
WHERE mansione = Programmatore;
NB: è possibile scrivere le query su più righe. Ciascuna query termina con il punto e virgola ";".
Il risultato di questa interrogazione sulla base di dati degli impiegati è:

```
+-------------+--------------+
| IDimpiegato | nome         |
+-------------+--------------+
|        6651 | Ajay Patel   |
|        7513 | Nora Edwards |
+-------------+--------------+
2 rows in set (0.42 sec)
```

Condizione WHERE per selezionare determinate righe

È stata usata la condizione WHERE per trovare nella tabella soltanto le righe che corrispondevano ai criteri selezionati; in questo esempio, dovevano essere selezionati solo gli impiegati programmatori.
Notate che è stata utilizzata una lista di colonne desiderate (IDimpiegato e nome) per estrarre solo le informazioni che si volevano.
In questo caso è stato eseguito, nella condizione WHERE, un confronto con uguaglianza. SQL indica con '=' il confronto con uguaglianza, ricordando invece ciò che avviene in altri linguaggi che adottano '=='.
Esiste una grande varietà di funzioni che possono essere usate nella condizione WHERE.

Condizione WHERE per selezionare determinate righe

Alcuni tra gli operatori più adoperati sono:

- uguaglianza ‘**=**’ ;
- diversità espressa come ‘**!=**’ oppure ‘**<>**’;
- tutte le possibili combinazioni di: **>** (maggiore di), **<** (minore di), **>=** (maggiore o uguale a), **<=** (minore o uguale a);
- IS NULL e IS NOT NULL usati per verificare se un certo valore è o non è NULL. Questa verifica non può farsi eseguendo il confronto = NULL
- i soliti operatori matematici, utilizzati tipicamente insieme agli operatori di confronto. Es., si potrebbe voler verificare se x > y*10;
- gli operatori booleani standard AND, OR e NOT, che si possono usare per raggruppare i confronti. Questi hanno un livello di precedenza inferiore agli operatori di confronto.

Condizione WHERE

Oltre agli operatori, possono essere usate le funzioni. La funzione count( ) permette di contare le righe selezionate da una interrogazione.
Es: **SELECT count(*) FROM impiegato;**
L'interrogazione dice quante righe sono contenute nella tabella impiegato.
Si possono regolare le priorità tra operatori mediante l'uso di parentesi.
Altro esempio che usa la condizione WHERE:
SELECT * FROM assegnamento WHERE IDimpiegato=7513 AND oreEffettuate> 8;
L'interrogazione seleziona tutti i lavori assegnati a IDimpiegato 7513 (Nora Edwards) che abbiano svolto più di 8 ore lavorative.
E' importante notare che non potete utilizzare nessun alias di colonna nella condizione WHERE. Dovete usare il nome originale. Questo è un limite ANSI SQL. La ragione è che il valore della colonna con l'alias potrebbe non essere noto al momento della valutazione della condizione WHERE.

Rimozione duplicati: DISTINCT

Nelle interrogazione si usa la parola chiave DISTINCT per indicare che non si desiderano i duplicati nel risultato. Ad esempio:

SELECT mansione FROM impiegato;

L'interrogazione restituisce il seguente risultato:

```
+---------------------------+
¦ mansione                  ¦
+---------------------------+
¦ Programmatore             ¦
¦ Programmatore             ¦
¦ Amministratore di Sistema ¦
¦ DBA                       ¦
+---------------------------+

4 rows in set (0.01 sec)
```

la mansione programmatore appare due volte perché è presente in due righe. L'interrogazione restituisce semplicemente l'elenco completo dei valori presenti nella colonna mansione della tabella..

Rimozione duplicati: DISTINCT

Adesso ripetiamo l'interrogazione aggiungendo il comando DISTINCT:

SELECT DISTINCT mansione FROM impiegato;

L'interrogazione restituisce le righe seguenti:

```
+---------------------------+
¦ mansione                  ¦
+---------------------------+
¦ Programmatore             ¦
¦ Amministratore di Sistema ¦
¦ DBA                       ¦
+---------------------------+
3 rows in set (0.04 sec)
```

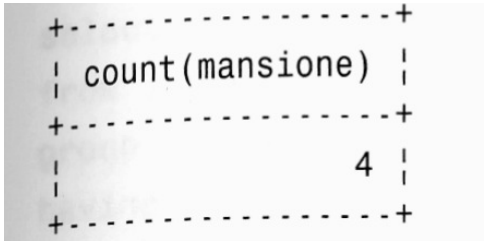
Sono stati eliminati i duplicati. In questo caso, non sembra che ci sia questa gran differenza; di sicuro il secondo insieme di risultati è un po' più piccolo, ma questo non migliora di molto le cose. La riduzione sarebbe stata un po' più importante con una tabella grande con molte ripetizioni, pur mantenendo una informazione accurata.

Rimozione duplicati: DISTINCT

Per apprezzare meglio DISTINCT, consideriamo la seguente istruzione:

SELECT count(mansione) FROM impiegato;

Questa interrogazione dice che esistono 4 mansioni. Questo è fuorviante, infatti non afferma che esistono 4 diverse mansioni perché se osservate i dati vedete che ce ne sono solo 3.

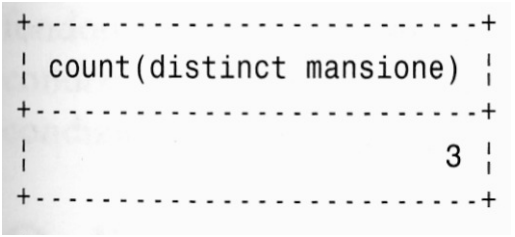


```
+-----------------+
| count(mansione) |
+-----------------+
|               4 |
+-----------------+
```

È abbastanza semplice scrivere per errore l'interrogazione precedente, mentre ciò che si intendeva era:

SELECT count(DISTINCT mansione) FROM impiegato;

L'interrogazione restituisce quanti valori diversi si trovano nella colonna mansione:



```
+--------------------------+
| count(distinct mansione) |
+--------------------------+
|                        3 |
+--------------------------+
```

Uso della condizione GROUP BY

Questa condizione permette di raggruppare le righe restituite da un'interrogazione. E' un comando davvero utile soltanto se viene usato in combinazione con operatori che lavorano su gruppi di righe. L'unico comando di questo tipo visto finora è count( ), ma ne vedremo altri. Consideriamo l'interrogazione seguente:

SELECT count(*), mansione FROM impiegato GROUP BY mansione;

L'interrogazione conta il numero di impiegati che svolgono una mansione. Eseguendo l'interrogazione sulla base di dati degli impiegati, si dovrebbe avere il seguente risultato:

count(*)	mansione
1	Amministratore di Sistema
1	DBA
2	Programmatore

condizione GROUP BY ordinamento

MySQL inoltre permette di scegliere l'ordine dei gruppi in cui deve apparire il risultato. Lo standard è quello ascendente. Se volessimo ripetere l'ultima interrogazione ma vedere il risultato in ordine inverso, potremmo usare:

SELECT count(*), mansione FROM impiegato GROUP BY mansione DESC;

L'interrogazione restituisce:

count(*)	mansione
2	Programmatore
1	DBA
1	Amministratore di Sistema

Come possiamo vedere, i nomi delle mansioni si trovano adesso in ordine alfabetico inverso.

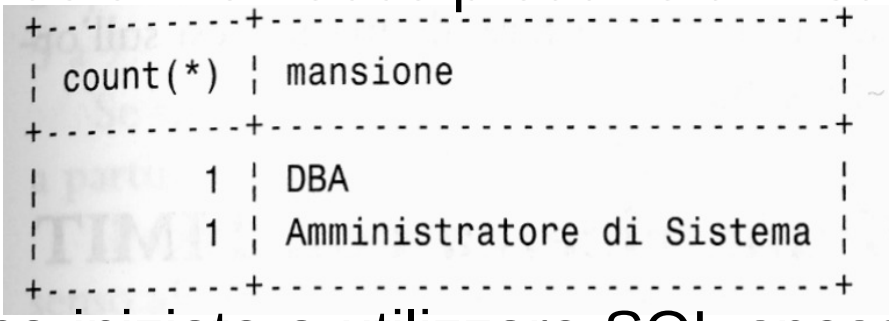
Potete specificare ASC (ascend in ascendente), ma, poiché questo è il valore standard, farlo sarebbe ridondante.

Selezione di gruppi particolari con HAVING

La prossima istruzione nel comando di SELECT è HAVING. Un GROUP BY con una condizione HAVING funziona come un SELECT con un WHERE. Ad esempio:

SELECT count(*), mansione FROM impiegato GROUP BY mansione HAVING count(*) = 1;

L'interrogazione seleziona le mansioni per le quali esiste un solo impiegato in quel ruolo. Dovrebbe produrre un risultato del tipo:



count(*)	mansione
1	DBA
1	Amministratore di Sistema

N.B. chi ha appena iniziato a utilizzare SQL spesso confonde WHERE e HAVING. Si può usare WHERE in qualsiasi interrogazione per verificare condizioni relative a singole righe. Invece si usa HAVING quando si vuole applicare una condizione a interi gruppi.

Ordinamento risultati di una ricerca con ORDER BY

ORDER BY permette di ordinare le righe del risultato sulla base di una o più colonne. L'ordine può essere ascendente, indicato con ASC, o discendente, indicato con DESC. Ad esempio:
SELECT * FROM impiegato ORDER BY mansione ASC, nome DESC;
L'istruzione seleziona tutte le righe e le colonne della tabella impiegato che verranno ordinate secondo la mansione in ordine alfabetico, e, se due o più persone dovessero avere la stessa mansione, verranno visualizzate in ordine alfabetico inverso secondo il nome. Il risultato è il seguente:

IDimpiegato	nome	mansione	IDdipartimento
9006	Candy Burnett	Amministratore di Sistema	128
9842	Ben Smith	DBA	42
7513	Nora Edwards	Programmatore	128
6651	Ajay Patel	Programmatore	128

Indicando ORDER BY senza ASC o DESC, lo standard sarà ASC.
NB: se non viene indicato il comando ORDER BY, non si può fare alcuna ipotesi sull'ordine in cui le righe verranno visualizzate nel risultato.

Limitazione dei risultati di una ricerca

L'ultimo comando dell'istruzione SELECT che conoscerete in questo capitolo è LIMIT. L'istruzione LIMIT viene usata per limitare il numero e l'intervallo di righe che vengono restituite da un'interrogazione. Ad esempio, considerate:

```
SELECT *
FROM impiegatoCompetenze
LIMIT 5;
```

L'interrogazione restituirà solo le prime 5 righe che corrispondono ai criteri di ricerca. In questo particolare caso, si avranno semplicemente le prime 5 righe trovate nella tabella:

IDimpiegato	competenza
6651	Java
6651	VB
7513	C
7513	Java
7513	Perl

Limitazione dei risultati di una ricerca con LIMIT

Si può inoltre specificare che si vuole un sottoinsieme di righe piuttosto che le prime n righe.

SELECT * FROM impiegatoCompetenze LIMIT 5;

Ad esempio, se volessimo ritrovare le righe che si trovano dalla sesta all'ottava dovremmo eseguire:

SELECT * FROM impiegatoCompetenze LIMIT 5, 3;

Quando si passano al comando limit due parametri, il primo indica il punto di partenza (detto offset) e il secondo il numero totale di righe che volete nel risultato.

Quando si passa un solo parametro, esso indica il massimo numero di righe che si vogliono visualizzare.

Quando si vuole indicare l'offset, la numerazione delle righe inizia da zero (come potete vedere dall'esempio precedente, per indicare la sesta riga è stato dato offset pari a 5). Il primo esempio di LIMIT ha selezionato le righe da 0 a 4, il secondo quelle da 5 a 7.

Limitazione dei risultati di una ricerca con LIMIT

Se viene specificato -1 come secondo parametro l'interrogazione restituirà tutte le righe a partire dall'offset fino alla fine della tabella.
Il comando LIMIT viene generalmente usato insieme a ORDER BY così da dare più senso all'ordine in cui le righe vengono restituite.
Il comando è utile specialmente quando si costruiscono delle applicazioni Web o GUI (Graphical User Interface, interfacce utente grafiche) usando MySQL, in quanto esso fornisce un comodo meccanismo per impaginare i risultati.

Esercizi

- Scrivere una interrogazione che elenchi tutte le informazioni degli impiegati che lavorano per il dipartimento 128.
- Scrivere un'interrogazione che elenchi tutti gli IDimpiegato degli impiegati che hanno lavorato per il cliente num. 1.
- Scrivere un'interrogazione che restituisca il numero di impiegati associati a ciascuna competenza tra quelle elencate nella tabella impiegatoCompetenze.