

[Home Page](#) [Novità](#) [Aiuto](#) 

Il linguaggio SQL

L'operatore JOIN

http://www.vbsimple.net/database/db_08_09.htm

Difficoltà:  3 / 5

La possibilità di suddividere i dati in più tabelle sarebbe inutile se non si potesse unire il contenuto di più tabelle in un unico set di risultati; abbiamo infatti già anticipato il concetto di JOIN accennando alla possibilità di specificare più origini nella clausola **FROM** trattata [qualche pagina addietro](#). Tuttavia l'insieme dei JOIN è molto più complesso di quanto presentato nelle pagine precedenti.

Un'operazione di JOIN unisce il contenuto di una o più tabelle con l'origine specificata nella clausola FROM; il suo funzionamento è infatti basato sull'uso della clausola FROM e si presenta immediatamente accanto, in un'operazione di selezione:

```
SELECT <campi>  
FROM <origine>  
<tipo di join> <origine>  
[ ON <campo> <relazione> <campo> ]
```

Anzichè specificare più origini nella clausola FROM la soluzione ideale è quella di specificare un'unica origine in FROM e tutte le altre origini in tante clausole JOIN, ognuna con la propria relazione con l'origine principale. Prima di affrontare in dettaglio il comportamento dell'operatore JOIN è necessaria una presentazione dei diversi tipi di join:

- **INNER JOIN**

Il JOIN interno, sicuramente il principale, stabilisce che i dati che verranno restituiti sono **soltanto quelli** che corrispondono al criterio formato dalla relazione dei due campi; tutte le righe che non rientrano nella relazione sono escluse, sia dall'origine principale che da quelle coinvolte nelle relazioni. Corrisponde alla stessa modalità di JOIN trattata nell'articolo dedicato all'istruzione SELECT.

- **LEFT OUTER JOIN**

Il primo dei JOIN esterni, aggiunge ai risultati del JOIN interno anche quelle righe presenti nell'origine presente a sinistra (LEFT) della clausola JOIN; le nuove righe formate da questi dati aggiuntivi sono riempiti con valori [NULL](#).

- **RIGHT OUTER JOIN**

Il JOIN esterno opposto al precedente, aggiunge ai risultati del JOIN interno anche le righe dell'origine che si trova alla destra (RIGHT) della clausola JOIN; le righe non corrispondenti sono riempite con valori NULL.

- **FULL OUTER JOIN**

La combinazione di JOIN interno, esterno sinistro ed esterno destro, include tutte le righe presenti nelle origini a sinistra e a destra della clausola; i dati presenti in una

delle origini ma non corrispondenti nell'altra conterranno valori NULL.

- **CROSS JOIN**

Un JOIN non supportato direttamente da tutti i database, basato praticamente sull'assenza di una relazione che limiti i dati restituiti. Corrisponde alla modalità di JOIN trattata negli articoli precedenti senza tuttavia la presenza della clausola di relazione. Comporta la permutazione di tutte le righe delle origini coinvolte nella relazione.

Vediamo in esame le singole modalità di JOIN, il loro funzionamento ed una serie di esempi per cercare di chiarire questo concetto alquanto complesso. A scopo di esempio immaginiamo una situazione di un'azienda che mantiene in due tabelle separate il personale generale ed i soli dipendenti.

PERSONALE		
NOME	COGNOME	RUOLO
Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere
Maurizio	Maugeri	Usciere
Pasquale	Catozzo	Manager
Milena	Piconi	Responsabile Acq.

DIPENDENTI		
NOME	COGNOME	RUOLO
Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere
Silvana	Perugia	Consulente
Maurizio	Maugeri	Usciere

In questa presunta realtà aziendale abbiamo tre persone (*Capelli*, *Tardi* e *Maugeri*) che lavorano come dipendenti e prestano la loro attività nell'azienda; altri due soggetti (*Catozzo* e *Piconi*) fanno parte del personale perché prestano la loro attività nell'azienda ma come professionisti esterni e quindi non rientrano tra i dipendenti; la signora *Perugia* invece che appare soltanto come dipendente è stipendiata dall'azienda solo per questioni di bilancio ma presta la sua attività in un'altra azienda, collegata alla prima.

INNER JOIN

La forma di JOIN più semplice, combina dei criteri per estrarre soltanto le righe che presentano i campi indicati in entrambe le origini. Data la situazione di analisi precedente è possibile stabilire un JOIN interno che relazioni i due gruppi di persone e riporti soltanto quelle che costituiscono sia figura di personale sia figura di dipendente. La rappresentazione SQL di questa situazione può essere la presente:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p INNER JOIN dipendenti d
ON p.nome=d.nome AND p.cognome=d.cognome AND p.ruolo=d.ruolo
```

Questa estrazione riporta i 3 campi della prima tabella (PERSONALE) ed i 3 della seconda (DIPENDENTI), combinando le righe della seconda la relazione indicata a seguito di ON. La stessa poteva essere rappresentata utilizzando la forma già vista in precedenza:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p, dipendenti d
WHERE p.nome=d.nome AND p.cognome=d.cognome AND p.ruolo=d.ruolo
```

I risultati riportati da entrambe le forme sono i seguenti:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere	Marco	Tardi	Ragioniere
Maurizio	Maugeri	Usciere	Maurizio	Maugeri	Usciere

Infatti le tre figure fanno parte di entrambe le tabelle di origine; tutte le altre righe sono scartate. Se non è specificata il tipo di JOIN (cioè è indicato solamente JOIN seguito dall'origine) questo è il tipo predefinito.

LEFT OUTER JOIN

Questo JOIN esterno tiene conto di tutte le righe che soddisfano la relazione ed include anche quelle righe presenti nella tabella che si trova a sinistra del JOIN, riempiendo i campi della seconda tabella con NULL. Data l'analisi della situazione aziendale potremmo voler estrarre tutti i nominativi del personale, mostrando la loro relazione di dipendenza con l'azienda e per far ciò useremo una query SQL del genere:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p LEFT JOIN dipendenti d
ON p.nome=d.nome AND p.cognome=d.cognome AND p.ruolo=d.ruolo
```

Il risultato ottenuto sarà il seguente:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere	Marco	Tardi	Ragioniere
Maurizio	Maugeri	Usciere	Maurizio	Maugeri	Usciere
Pasquale	Catozzo	Manager			
Milena	Piconi	Responsabile Acq.			

Naturalmente i valori qui presentati in bianco conterranno in realtà il valore NULL. Possiamo ben capire il comportamento osservando la forma dell'interrogazione SQL: la tabella PERSONALE si trova alla sinistra del JOIN, mentre la DIPENDENTI si trova alla destra; utilizzando il LEFT JOIN estrarremo tutte le righe presenti nella tabella PERSONALE, assegnando i campi corrispondenti della tabella DIPENDENTI e riempiendo con NULL quelli il cui valore non esiste nella seconda tabella. Le righe presenti unicamente nella tabella DIPENDENTI non saranno affatto estratte.



Alcuni database (Microsoft SQL Server ad esempio) supportano l'uso di una sintassi dedicata alla creazione di un LEFT JOIN utilizzando le clausole FROM e WHERE anziché JOIN e ON, sfruttando un operatore di relazione particolare: *=

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p, dipendenti d
WHERE p.nome*=d.nome AND p.cognome*=d.cognome AND p.ruolo*=d.ruolo
```

Tuttavia Microsoft stessa sconsiglia l'uso di questi operatori e indica che nelle prossime versioni del suo database potrebbe non essere presente e già adesso comportarsi in una maniera inaspettata. Alcuni database invece supportano l'uso dell'operatore **LEFT OUTER JOIN** al posto del semplice LEFT JOIN.

RIGHT OUTER JOIN

Il caso opposto del precedente è quello che riporta quelle righe che si trovano nella tabella indicata a destra del JOIN e riempie i campi non corrispondenti della prima tabella con il valore NULL. Analizzando ancora una volta il nostro problema potremmo cercare di ottenere l'elenco di tutti i dipendenti, segnalando anche quelli che non sono considerati personale dell'azienda. Basterà sostituire alla query precedente il LEFT con il RIGHT:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p RIGHT JOIN dipendenti d
ON p.nome=d.nome AND p.cognome=d.cognome AND p.ruolo=d.ruolo
```

Ed il risultato ottenuto sarà il seguente:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere	Marco	Tardi	Ragioniere
			Silvana	Perugia	Consulente
Maurizio	Maugeri	Usciere	Maurizio	Maugeri	Usciere

Possiamo notare la presenza di tutte le righe della tabella DIPENDENTI, le tre righe corrispondenti presenti nella tabella PERSONALE ed i campi della prima tabella riguardo la signora *Perugia* riempiti con valore NULL. I nominativi presenti esclusivamente nella prima tabella non sono stati estratti affatto.



In maniera analoga al caso precedente, alcuni database (Microsoft SQL Server ad esempio) supportano l'uso di una sintassi dedicata alla creazione di un RIGHT JOIN utilizzando le clausole FROM e WHERE anziché JOIN e ON, sfruttando un altro operatore di relazione particolare: =*

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p, dipendenti d
WHERE p.nome=*d.nome AND p.cognome=*d.cognome AND p.ruolo=*d.ruolo
```

Anche per questo Microsoft avverte che non sarà più supportato nelle versioni successive.

È anche possibile per certi database utilizzare la clausola **RIGHT OUTER JOIN** al posto del semplice **RIGHT JOIN**.

FULL OUTER JOIN

L'ultimo tipo di JOIN esterno è il **FULL JOIN**, che mettendo assieme sia **LEFT** che **RIGHT** comprende quindi tutti i dati presenti in entrambe le origini indicate, cioè includendo le righe presenti nella prima tabella ma non nella seconda, quelle presenti nella seconda tabella ma non nella prima, oltre che naturalmente tutte quelle che soddisfano la relazione. L'analisi della nostra problematica potrebbe ad esempio richiedere l'estrazione di tutto il personale e di tutti i dipendenti e ciò può essere esposto con una selezione del genere:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p FULL JOIN dipendenti d
ON p.nome=d.nome AND p.cognome=d.cognome AND p.ruolo=d.ruolo
```

Ed il risultato ottenuto sarà il seguente:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere	Marco	Tardi	Ragioniere
Maurizio	Maugeri	Usciere	Maurizio	Maugeri	Usciere
Pasquale	Catozzo	Manager			
Milena	Piconi	Responsabile Acq.			
			Silvana	Perugia	Consulente

Abbiamo cioè estratto le 5 righe presenti nella tabella **PERSONALE** e le 4 righe contenute all'interno della tabella **DIPENDENTI**. La relazione dov'era possibile è avvenuta, mentre dove non lo era ha riportato i dati esistenti accompagnati da un valore **NULL**. Al posto di **FULL JOIN** alcuni database consentono l'uso di **FULL OUTER JOIN**.

CROSS JOIN

Un JOIN particolare non supportato da tutti i database è quello incrociato, in realtà il più semplice perché non richiede la specifica dei campi con cui effettuare la relazione e pertanto permuta tutte le righe in modo che ciascuna riga presente nella prima tabella abbia una corrispondenza con tutte le righe nella seconda tabella; il numero di righe risultati è quindi dato dalla moltiplicazione del numero di righe nella prima tabella col numero di righe presenti nella seconda tabella.

Utilizza una sintassi del genere:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p CROSS JOIN dipendenti d
```

E riporta il seguente incredibile risultato:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Lucio	Capelli	Ragioniere
Lucio	Capelli	Ragioniere	Marco	Tardi	Ragioniere
Lucio	Capelli	Ragioniere	Maurizio	Maugeri	Usciere
Lucio	Capelli	Ragioniere	Silvana	Perugia	Consulente
Marco	Tardi	Ragioniere	Lucio	Capelli	Ragioniere
Marco	Tardi	Ragioniere	Marco	Tardi	Ragioniere
Marco	Tardi	Ragioniere	Maurizio	Maugeri	Usciere
Marco	Tardi	Ragioniere	Silvana	Perugia	Consulente
Maurizio	Maugeri	Usciere	Lucio	Capelli	Ragioniere
Maurizio	Maugeri	Usciere	Marco	Tardi	Ragioniere
Maurizio	Maugeri	Usciere	Maurizio	Maugeri	Usciere
Maurizio	Maugeri	Usciere	Silvana	Perugia	Consulente
Pasquale	Catozzo	Manager	Lucio	Capelli	Ragioniere
Pasquale	Catozzo	Manager	Marco	Tardi	Ragioniere
Pasquale	Catozzo	Manager	Maurizio	Maugeri	Usciere
Pasquale	Catozzo	Manager	Silvana	Perugia	Consulente
Milena	Piconi	Responsabile Acq.	Lucio	Capelli	Ragioniere
Milena	Piconi	Responsabile Acq.	Marco	Tardi	Ragioniere
Milena	Piconi	Responsabile Acq.	Maurizio	Maugeri	Usciere
Milena	Piconi	Responsabile Acq.	Silvana	Perugia	Consulente

Da questa confusa estrazione possiamo notare la combinazione di ciascuna persona presente nella prima tabella con qualsiasi altra persona presente nella seconda tabella; dalle 5 righe presenti nella prima tabella, e 4 righe presenti nella seconda sono estratte 20 righe (4*5) permutando qualsiasi possibile combinazione tra i nominativi di ciascuna tabella.

Equivale anche all'uso della specifica delle due origini nella clausola FROM senza l'uso di una clausola WHERE:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p, dipendenti d
```

Nel qual caso che il database non dovesse apprezzare la sintassi CROSS JOIN sarà sempre possibile ricorrere a questa forma alternativa.

SELF JOIN

Non si tratta di JOIN veri e propri ma vale la pena citarli; in questi esempi abbiamo sempre utilizzato due differenti tabelle come origini ma nulla vieta l'uso di un'unica tabella ripetuta due volte e di una relazione con la stessa; si tratta di una pratica singolare ma utile in certi casi quando non è semplice combinare in una sola relazione più caratteristiche.

Il tipo di JOIN può essere uno qualsiasi tra quelli esposti in precedenza e cerchiamo di inventarci una relazione sensata per questo tipo di caso: l'unica che mi possa venire in mente con questi pochi dati è una sorta di estrazione che riporti tutti i dipendenti che svolgono lo stesso ruolo di un'altra persona e chi sia questa persona; la query può essere esposta come segue:

```
SELECT p.nome, p.cognome, p.ruolo, d.nome, d.cognome, d.ruolo
FROM personale p INNER JOIN dipendenti d
ON p.ruolo=d.ruolo
WHERE p.nome<>d.nome AND p.cognome<>d.cognome
```

Ed il corrispondente risultato:

P_NOME	P_COGNOME	P_RUOLO	D_NOME	D_COGNOME	D_RUOLO
Lucio	Capelli	Ragioniere	Marco	Tardi	Ragioniere
Marco	Tardi	Ragioniere	Lucio	Capelli	Ragioniere

Come già detto si tratta di casi rari ma in alcuni casi di complesse relazioni tra molte origini può essere una strada applicabile, se non altro per cercare di semplificare un problema risolvibile in una maniera molto più complessa.

Aggiungiamo per completezza la sintassi da utilizzare quando la relazione integra tre o più origini:

```
SELECT <campi>
FROM <origine>
<tipo di join> <origine>
ON <campo> <relazione> <campo>
<tipo di join> <origine>
ON <campo> <relazione> <campo>
....
```

[Fibia.FBI](#)
30 Marzo 2004



[Torna all'indice della sezione Database](#)
