JOIN, creare query relazionali

Capita spesso la necessità di selezionare dati prelevandoli fra più tabelle, evidentemente correlate tra loro. A tale scopo si utilizzano le **join**.

Esistono diverse tipologie di join. Fondamentalmente sono tre: la **inner join**, la **outer join** e la **cross join**.

La *cross join* è concettualmente la più semplice, ma raramente trova applicazione in quanto è difficile che abbia un senso logico: si tratta del "prodotto cartesiano" di due tabelle. In pratica, ogni riga della prima tabella viene combinata con tutte le righe della seconda. Ipotizzando di avere una tabella di 5 righe e una di 6, il risultato sarà una tabella di 30 righe.

La *inner join* si effettua andando a cercare righe corrispondenti nelle due tabelle, basandosi sul valore di determinate colonne.

Immaginiamo, in un esempio classico, di avere una *tabella ordini* e una *tabella clienti*. Diciamo che la prima contiene le colonne *idOrdine, idCliente, articolo, quantità* mentre la seconda contiene *idCliente, nome, cognome*.

Evidentemente il campo '**idCliente**' della tabella ordini è una chiave esterna sulla tabella clienti, che ci permette di recuperare i dati anagrafici del cliente che ha effettuato l'ordine (abbiamo limitato al minimo, per semplicità, il numero dei campi contenuti nelle due tabelle). In questo caso quindi potremo fare le join basandoci sulla corrispondenza dei valori dei campi '**idCliente**' nelle due tabelle (naturalmente non è necessario che le colonne abbiano lo stesso nome).

Le righe estratte con una inner join saranno **solo** quelle che hanno il valore di una tabella corrispondente a quello nell'altra tabella.

Le *outer join*, come le inner join, vengono effettuate in base alla corrispondenza di alcuni valori sulle tabelle. La differenza è che, nel caso delle outer join, è possibile estrarre anche le righe di una tabella che **non** hanno corrispondenti nell'altra.

Vediamo alcuni esempi:

SELECT * FROM ordini AS o, clienti AS c WHERE o.idCliente = c.idCliente AND idOrdine > 1000; SELECT * FROM ordini AS o JOIN clienti AS c on o.idCliente = c.idCliente WHERE idOrdine > 1000;

Queste due query sono equivalenti e rappresentano una inner join: estraggono i dati relativi ad ordine e cliente per quegli ordini che hanno un identificativo maggiore di

1000. La prima è una join implicita: infatti non l'abbiamo dichiarata esplicitamente e abbiamo messo la condizione di join nella clausola WHERE. Quando elenchiamo più tabelle nella FROM senza dichiarare esplicitamente la JOIN stiamo facendo una inner join (oppure una cross join se non indichiamo condizioni di join nella WHERE).

Nella seconda, al posto di 'JOIN' avremmo potuto scrivere per esteso 'INNER JOIN'; in alcune vecchie versioni di MySQL ciò è obbligatorio.

SELECT * FROM ordini as o LEFT JOIN clienti as c ON o.idCliente = c.idCliente WHERE idOrdine > 1000;

In questo caso abbiamo effettuato una **left outer join**: la query ottiene gli stessi risultati delle due precedenti, ma in più restituirà le eventuali righe della tabella ordini il cui valore di idCliente non ha corrispondenti sulla tabella clienti. In queste righe della tabella risultato, **i campi dell'altra tabella saranno valorizzati a NULL**. Quindi potremmo eseguire una query che estrae solo le righe della prima tabella senza corrispondente, così:

SELECT * FROM ordini as o LEFT JOIN clienti as c ON o.idCliente = c.idCliente WHERE idOrdine > 1000 AND c.idCliente IS NULL

Le outer join si dividono in **left outer join**, **right outer join** e **full outer join**.

Con le prime otterremo le righe senza corrispondente che si trovano nella tabella di sinistra (cioè quella dichiarata per prima nella query). Le right outer join restituiscono invece le righe della seconda tabella che non hanno corrispondente nella prima. Con le full outer join infine si ottengono le righe senza corrispondente da entrambe le tabelle.

Nella sintassi di MySQL, la parola OUTER è facoltativa: scrivere LEFT JOIN o LEFT OUTER JOIN è equivalente. Allo stesso modo avremmo potuto scrivere RIGHT JOIN o RIGHT OUTER JOIN per una right join.
In MySQL non è invece possibile effettuare le full outer join.

Quando, come nel nostro esempio, le colonne su cui si basa la join hanno lo stesso nome nelle due tabelle, è possibile utilizzare una sintassi abbreviata per effettuare la join: la clausola **USING**. Vediamo la join precedente con questa clausola:

SELECT * FROM ordini LEFT JOIN clienti USING (idCliente) WHERE idOrdine > 1000;

Naturalmente la join può essere basata anche su più colonne. In questo caso elencheremo più condizioni, separate da virgole, nella ON, oppure elencheremo i nomi delle colonne, sempre separati da virgole, nella clausola USING.

C'è una ulteriore possibilità di abbreviare la sintassi, quando la join è basata su **tutte** le

colonne che nelle due tabelle hanno lo stesso nome: si tratta della clausola **NATURAL**.

Anche questa è applicabile al nostro esempio:

SELECT * FROM ordini NATURAL LEFT JOIN clienti WHERE idOrdine > 1000;

Le clausole USING e NATURAL possono essere utilizzate sia con le inner join che con le outer join.

Join fra più tabelle

Finora abbiamo visto esempi di join fra due tabelle, ma è possibile effettuarne anche fra più di due. In questo caso l'operazione sarà logicamente suddivisa in più join, ciascuna delle quali viene effettuata fra due tabelle; il risultato di ogni join diventa una delle due tabelle coinvolte nella join successiva. L'ordine con cui vengono effettuate le diverse join dipende dall'ordine in cui sono elencate le tabelle e (a partire da MySQL 5.0.1) dall'uso di eventuali parentesi:

FROM *t1* JOIN *t2* ON t1.col1 = t2.col2 LEFT JOIN *t3* ON t2.col3 = t3.col3

In questo caso viene effettuata prima la join fra t1 e t2; di seguito, il risultato di questa join viene utilizzato per la left join con t3.

FROM *t1* JOIN (*t2* LEFT JOIN *t3* ON t2.col3 = t3.col3) ON t1.col1 = t2.col2

La presenza delle parentesi fa sì che venga effettuata prima la left join fra t2 e t3, e di seguito il risultato venga utilizzato per la inner join con t1.

Una nota importante: se si mischiano join implicite con join esplicite, a partire da MySQL 5.0.12 queste ultime prendono la precedenza anche in assenza di parentesi. Questo può far sì che alcune query che in precedenza funzionavano possano causare errori, soprattutto se non scritte in maniera ortodossa.