Join - Combinazione di tabelle

Questo capitolo tratta un importante tipo di operazione tra le tabelle: il Join.
Il vocabolo join significa unione e nel caso di SQL sta ad indicare unione tra tabelle. Esistono vari tipi di join, ma tutti derivano o possono essere ricondotti a vari operatori dell'algebra insiemistica. ***L'importanza principale delle join risiede nella possibilità che ci offre per correlare e visualizzare dati appartenenti a tabelle diverse o alla medesima tabella, logicamente correlati tra di loro***. I semplici dati, da noi uniti, possono assumere la forma di complesse informazioni così come noi li vogliamo.

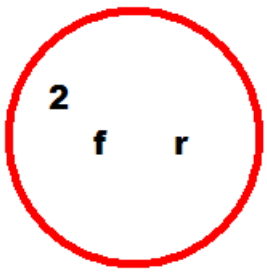
CROSS JOIN (prodotto cartesiano fra insiemi)

Per comprendere a pieno l'operazione CROSS JOIN (unione incrociata) bisogna aver ben chiaro il concetto di prodotto cartesiano:

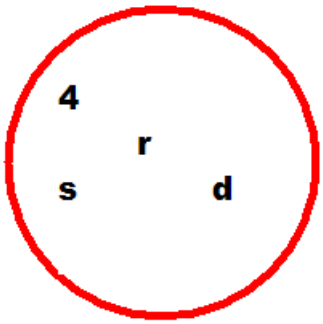
.....

Dati due insiemi D1 e D2 si chiama prodotto cartesiano di D1 e D2, l'insieme delle coppie ordinate (v1, v2), tali che v1 è un elemento di D1 e v2 un elemento di D2.

Vediamo cosa significa quanto affermato con un esempio:



Insieme A



Insieme B

A x B = {(2, r), (2, s), (2, d), (2, 4), (f, r), (f, s), (f, d), (f, 4), (r, r), (r, s), (r, d), (r, 4)}

Come possiamo vedere il prodotto cartesiano fra i due insiemi è dato da tutti gli elementi di A combinati con ogni elemento di B. Nella rappresentazione delle varie coppie dobbiamo rispettare l'ordine di apparizione degli elementi, in quanto l'appartenenza dell'elemento all'insieme è individuabile proprio dalla suo ordine di apparizione. Nell'esempio abbiamo usato solo due insiemi ma il prodotto cartesiano è applicabile anche a più di due insiemi.

Ora considerando che le tabelle non sono altro che insiemi i cui elementi sono le righe ecco che possiamo individuare l'operazione di **CROSS JOIN** in quella di prodotto cartesiano appartenente alle teorie degli insiemi. Dunque il prodotto cartesiano tra due o più tabelle si traduce in una istruzione chiamata CROSS JOIN. Il CROSS JOIN si ottiene in maniera molto semplice elencando dopo la FROM le tabelle che devono essere coinvolte. Vediamo un esempio di CROSS JOIN:

Per lo scopo usiamo due tabelle: TAB1 e TAB2

TAB1	TAB2
COLONTAB1	COLONTAB2
----------	----------
RIG1 TAB1	RIG1 TAB2
RIG2 TAB1	RIG2 TAB2
RIG3 TAB1	RIG3 TAB2
RIG4 TAB1	
RIG5 TAB1	

SELECT *
FROM TAB1, TAB2;

COLONTAB1	COLONTAB2
----------	----------
RIG1 TAB1	RIG1 TAB2
RIG2 TAB1	RIG1 TAB2
RIG3 TAB1	RIG1 TAB2
RIG4 TAB1	RIG1 TAB2
RIG5 TAB1	RIG1 TAB2
RIG1 TAB1	RIG2 TAB2
RIG2 TAB1	RIG2 TAB2
RIG3 TAB1	RIG2 TAB2
RIG4 TAB1	RIG2 TAB2
RIG5 TAB1	RIG2 TAB2
RIG1 TAB1	RIG3 TAB2
RIG2 TAB1	RIG3 TAB2
RIG3 TAB1	RIG3 TAB2
RIG4 TAB1	RIG3 TAB2
RIG5 TAB1	RIG3 TAB2

Questo è il risultato che si ottiene dal CROSS JOIN delle tabelle TAB1 e TAB2, come si può vedere non è altro che un prodotto cartesiano. Chiaramente avremmo **potuto usare anche più di due tabelle**.

(**vedi esercizio prodotto-cartesiano-esempio.txt**)

Il CROSS JOIN non è particolarmente utile e viene usato raramente, ma se in una CROSS JOIN si utilizza la clausola WHERE potremmo ottenere join molto più interessanti.

Analizzeremo ora i vari tipi di Join su due tabelle esempio: Impiegati e Dipartimenti in cui lavorano.

File SQL per effettuare le prove:

```sql
/* ESEMPIO - join, INNER, NATURAL, LEFT, RIGHT */

DROP TABLE impiegati;
DROP TABLE dipartimenti;


CREATE TABLE dipartimenti (
id_dipartimento       INTEGER PRIMARY KEY,
nome_dipartimento     VARCHAR(25)
);

CREATE TABLE impiegati (
cognome varchar(25) PRIMARY KEY,     -- Non ha molto senso cognome==PK,ma e' una tabella i
prova
id_dipartimento INTEGER REFERENCES dipartimenti(id_dipartimento)
);

INSERT INTO dipartimenti VALUES(31,'VENDITE');
INSERT INTO dipartimenti VALUES(33,'TECNICO');
INSERT INTO dipartimenti VALUES(34,'RISORSE UMANE');
INSERT INTO dipartimenti VALUES(35,'PRODUZIONE');


INSERT INTO impiegati VALUES('ROSSI',31);
INSERT INTO impiegati VALUES('BIANCHI',33);
INSERT INTO impiegati VALUES('MANCINI',33);
INSERT INTO impiegati VALUES('SANTORO',34);
INSERT INTO impiegati VALUES('MONTI',34);
INSERT INTO impiegati VALUES('GRASSI',NULL);
```

Tabella Impiegati

Cognome	ID_dipartimento
Rossi	31
Bianchi	33
Mancini	33
Santoro	34
Monti	34
Grassi	Null

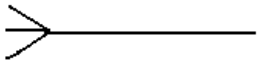


Tabella Dipartimenti

ID_dipartimento	Nome_dipartimento
31	Vendite
33	Tecnico
34	Risorse umane
35	Promozione

Nota: Il dipartimento "Promozione" della tabella "Dipartimenti" non ha alcuna corrispondenza nella tabella "Impiegati". Mentre l'impiegato "Grassi" non è stato assegnato ad alcun dipartimento (`Null`).

Inoltre nella tabella Impiegati, non e' stato inserita nessuna Primary Key, per semplificare gli esempi.
L'associazione fra l'entita' Dipartimenti (ramo 1) e l'entita' Impiegati (ramo N) e' di tipo 1 a N.

Inner join

E' la forma di join piu' usata. Di solito la otteniamo uguagliando i Campi PK della tab di ramo 1 con la FK della tabella di ramo N:

```
SELECT *
FROM   Impiegati, Dipartimenti
WHERE  Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

Esiste la possibilita' di scrivere la Join anche nella forma che evidenzia il TIPO di Join (Inner Join):

```
SELECT * FROM   Impiegati

INNER JOIN Dipartimenti

ON Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

Oppure nella forma piu' compatta (senza INNER)

```
SELECT * FROM   Impiegati

JOIN Dipartimenti ON Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

Impiegati.Cognome	Impiegati.ID_dipartimento	Dipartimenti.Nome_dipartimento	Dipartimenti.ID_dipartimento
Santoro	34	Risorse umane	34
Bianchi	33	Tecnico	33
Monti	34	Risorse umane	34
Mancini	33	Tecnico	33
Rossi	31	Vendite	31

NB: FORMA GENERALE INNER JOIN CON 3 (o piu' TABELLE A,B,C...)

```
SELECT *
FROM A JOIN B ON PKA=FKB JOIN C ON PKB=FKC;
```

La sintassi ormai e' accettata dalla maggior parte dei data base relazionali.

NATURAL JOIN

o Join Naturale corrisponde all'inner Join dove, se i campi identici delle due tabella

non vengono ripetuti

*(Nota: e'comunque possibile ottenere lo stesso risultato specificando i campi nella select anziche' mettere *)*

```
SELECT *
FROM   Impiegati NATURAL JOIN Dipartimenti
```

ID_dipartimento	Impiegati.Cognome	Dipartimenti.Nome_dipartimento
34	Monti	Risorse umane
33	Bianchi	Tecnico
34	Santoro	Risorse umane
33	Mancini	Tecnico
31	Rossi	Vendite

(NB: se i campi PK-FK hanno nomi diversi (es id,id_dipartimento) la natural join produce un prodotto cartesiano!!!) provare ad esempio a rinominare il campo id_dipartimento in id nella tabella dipartimenti.)

CROSS JOIN (prodotto cartesiano):

Esempio (implicito) di cross join:

```
SELECT *
FROM   Impiegati, Dipartimenti;
```

La sintassi puo' anche essere esplicita:

Esempio di cross join esplicito:

```
SELECT *
FROM   Impiegati CROSS JOIN Dipartimenti
```

(vedi sotto nella prossima pagina il risultato)

Il risultato della cross Join:

Impiegati.Cognome	Impiegati.ID_dipartimento	Dipartimenti.Nome_dipartimento	Dipartimenti.ID_dipartimento
Rossi	31	Vendite	31
Bianchi	33	Vendite	31
Mancini	33	Vendite	31
Monti	34	Vendite	31
Santoro	34	Vendite	31
Grassi	Null	Vendite	31
Rossi	31	Tecnico	33
Bianchi	33	Tecnico	33
Mancini	33	Tecnico	33
Monti	34	Tecnico	33
Santoro	34	Tecnico	33
Grassi	Null	Tecnico	33
Rossi	31	Risorse umane	34
Bianchi	33	Risorse umane	34
Mancini	33	Risorse umane	34
Monti	34	Risorse umane	34
Santoro	34	Risorse umane	34
Grassi	Null	Risorse umane	34
Rossi	31	Promozione	35
Bianchi	33	Promozione	35
Mancini	33	Promozione	35
Monti	34	Promozione	35
Santoro	34	Promozione	35
Grassi	Null	Promozione	35

OUTER JOIN – (Join meno usate) sono 3: left, right, full

Con l'OUTER JOIN è possibile estrapolare anche quei dati, appartenenti ad una delle tabelle, che non verrebbero estrapolati nei tipi di join visti fino a questo momento.

Infatti OUTER significa esterno; dati esterni al normale tipo di join.
Dobbiamo specificare quale è la tabella di cui vogliamo estrapolare i dati anche se non soddisfano la condizione di join, questo lo facciamo indicando con LEFT o RIGHT se la tabella in questione è quella che appare a sinistra o a destra del comando JOIN:

Left outer join (o left join)

Il risultato di una query *left outer join* (o semplicemente **left join**) per le tabelle A e B contiene sempre tutti i record della tabella di sinistra ("left") A, mentre vengono estratti dalla tabella di destra ("right") B solamente le righe che trovano corrispondenza nella regola di confronto della join.

```
SELECT *
FROM   Impiegati  LEFT JOIN Dipartimenti
          ON Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

Risultato della left join:

Impiegati.Cognome	Impiegati.ID_dipartimento	Dipartimenti.Nome_dipartimento	Dipartimenti.ID_dipartimento
Bianchi	33	Tecnico	33
Rossi	31	Vendite	31
Santoro	34	Risorse umane	34
Monti	34	Risorse umane	34
Grassi	Null	Null	Null
Mancini	33	Tecnico	33

Right outer join (o right join)

Una **right outer join** (o **right join**) semplicemente ricalca il funzionamento della left outer join, ma invertendo l'ordine delle tabelle interessate.

Il risultato di una query *right outer join* per le tabelle A e B contiene sempre tutti i record della tabella di destra ("right") B, mentre vengono estratti dalla tabella di sinistra ("left") A solamente le righe che trovano corrispondenza nella regola di confronto della join.

```
SELECT *
FROM   Impiegati  RIGHT JOIN Dipartimenti
          ON Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

Impiegati.Cognome	Impiegati.ID_dipartimento	Dipartimenti.Nome_dipartimento	Dipartimenti.ID_dipartimento
Monti	34	Risorse umane	34
Bianchi	33	Tecnico	33
Santoro	34	Risorse umane	34
Mancini	33	Tecnico	33
Rossi	31	Vendite	31
`Null`	`Null`	*Promozione*	*35*

Full outer join (o full join)

Una **full outer join** combina i risultati delle due tabelle A e B tenendo conto di tutte le righe delle tabelle, anche di quelle che non hanno corrispondenza tra di loro.

```
SELECT *

FROM   Impiegati
       FULL OUTER JOIN Dipartimenti
          ON Impiegati.ID_dipartimento = Dipartimenti.ID_dipartimento
```

NOTA: Alcuni database (come ad esempio MySQL) non supportano direttamente questa funzionalità,

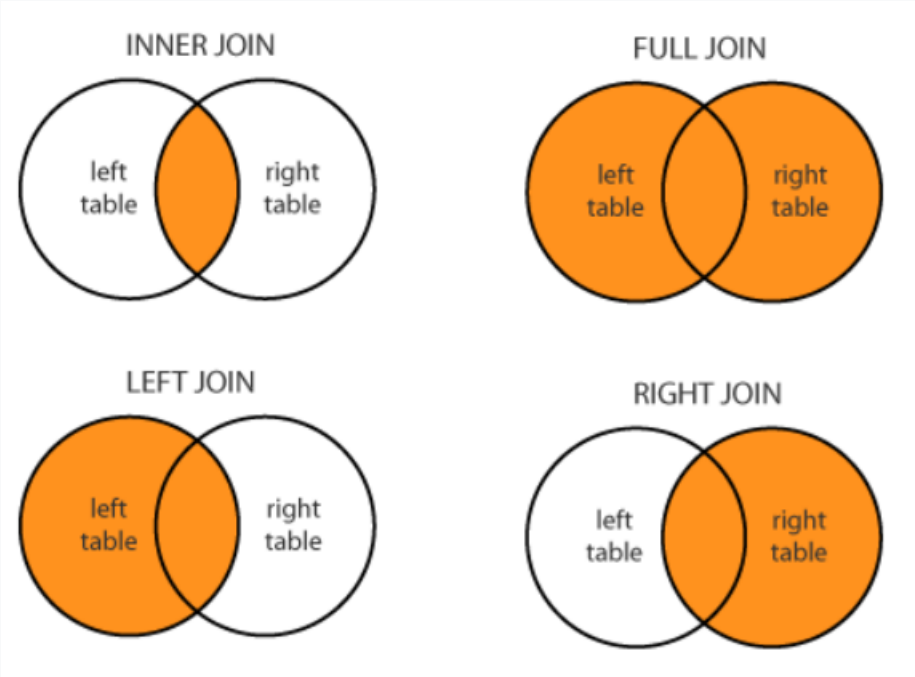
Risultato della full join:

Impiegati.Cognome	Impiegati.ID_dipartimento	Dipartimenti.Nome_dipartimento	Dipartimenti.ID_dipartimento
Monti	34	Risorse umane	34
Bianchi	33	Tecnico	33
Santoro	34	Risorse umane	34
Grassi	`Null`	`Null`	`Null`
Mancini	33	Tecnico	33
Rossi	31	Vendite	31
`Null`	`Null`	*Promozione*	*35*

SELF JOIN
Il SELF JOIN ci consente di unire una tabella con se stessa. Molto interessanti, hanno una sintassi particolare.

Tratteremo in seguito in una lezione a parte in modo esaustivo il SELF JOIN.

Schema riassuntivo vari tipi di Join:



Infine un buon programma di test PHP per provare JOIN con basi di dati un po' piu' grandi:

Bigfile.php → produce un file sql (bigfile.sql) per creare tre tabelle:

```
CREATE TABLE A (
PKA INTEGER PRIMARY KEY,
DESCA VARCHAR(30) );


CREATE TABLE B (
PKB INTEGER PRIMARY KEY,
DESCB VARCHAR(30),
FKB integer REFERENCES A(PKA) );

CREATE TABLE C (
PKC INTEGER PRIMARY KEY,
DESCC VARCHAR(30),
FKC integer REFERENCES B(PKB) );
```

..con dimensioni fino a TABELLA A=50 Righe, TABELLA B=2500 Righe, TABELLA C=125000 Righe:

```php
<?php

if (!isset($_REQUEST['nrighe']))

{
    echo "<form action=bigfile.php>\n";
    echo "Numero di insert da creare (N X N X N) ->";
    echo "<input type=number name=nrighe value=10>\n";
    echo "<input type=submit value=\"Crea bigfile.sql\" >";
    echo "</form>";
}
else  /// creo 3 tabelle
{
    $drop="DROP TABLE C;\nDROP TABLE B;\nDROP TABLE A;\n";
    $sqla="CREATE TABLE A ( PKA INTEGER PRIMARY KEY, DESCA VARCHAR(30) );";
    $sqlb="CREATE TABLE B ( PKB INTEGER PRIMARY KEY, DESCB VARCHAR(30), FKB integer
REFERENCES A(PKA) );";
    $sqlc="CREATE TABLE C ( PKC INTEGER PRIMARY KEY, DESCC VARCHAR(30), FKC integer
REFERENCES B(PKB) );";

    $fd=fopen("bigfile.sql","w");
    fprintf($fd,"%s\n%s\n%s\n%s\n",$drop,$sqla,$sqlb,$sqlc);

    $nrighe=$_REQUEST['nrighe'];
    if ($nrighe>50)
    {
        echo "<p style=color:red;>Attenzione MAX 50 Righe!</p>";
        die("");
    }
    $RA=$nrighe;
    $RB=$RA*$RA;
    $RC=$RB*$RA;   /// numero righe per tabella
    $indice=0;   /// contatore
    echo "<hr>Creo file .\bigfile.sql con TABELLE in associazione 1 a N: A-&lt;B-
&lt;C\n<br><br>";
    echo "TABELLA A=$RA Righe, TABELLA B=$RB Righe, TABELLA C=$RC Righe<br><hr>\n";
    $indicea=0;
    $indiceb=0;
    $indicec=0;  // per TABELLE A,B e C
    fprintf($fd,"-- TABELLA A=$RA Righe, TABELLA B=$RB Righe, TABELLA C=$RC
Righe\n\n\n");
    for($i=0; $i <$nrighe; $i++)
    {
        $indicea++;
        fprintf($fd,"INSERT INTO A VALUES ($indicea,'TAB A->Descrizione$indicea');\n");
        for($j=0; $j <$nrighe; $j++)
        {
            $indiceb++;
            fprintf($fd,"INSERT INTO B VALUES ($indiceb,'TAB B->Descrizione$indiceb',
$indicea);\n");
            for($k=0; $k < $nrighe; $k++)
            {
                $indicec++;
                fprintf($fd,"INSERT INTO C VALUES ($indicec,'TAB C->Descrizione$indicec',
$indiceb);\n");
            }
        }
    }
    fprintf($fd,"\n\n\n\n--QUERY test 1:\nSELECT DESCA,DESCB,DESCC\nFROM A,B,C\nWHERE
PKA=FKB AND PKB=FKC;\n");
    fprintf($fd,"\n\n\n\n--QUERY test 2:\nSELECT DESCA,DESCB,DESCC\nFROM A JOIN B ON
PKA=FKB JOIN C ON PKB=FKC;\n");
    fclose($fd);
```

```
    echo "<br>Fine creazione file <a href=bigfile.sql>bigfile.sql!</a><hr>\n";
    echo "Query per test:<textarea>";
    echo "--QUERY test 1:\nSELECT DESCA,DESCB,DESCC\nFROM A,B,C\nWHERE PKA=FKB AND
PKB=FKC;\n";
    echo "--QUERY test 2:\nSELECT DESCA,DESCB,DESCC\nFROM A JOIN B ON PKA=FKB JOIN C ON
PKB=FKC;\n";
    echo "</textarea>";

}
?>
```