Der Verbund von mehreren Tabellen wird als JOIN bezeichnet. Dazu werden ein oder mehrere Felder beider Tabellen miteinander verglichen. Die Datensätze, die der Vergleichsbedingung genügen, werden in die Ergebnistabelle (Dynaset) aufgenommen. In SQL (Entry Level) gibt es für einige JOIN-Arten äquivalente Befehle. Sonst können diese JOIN's durch Kombination anderer JOIN's sowie Projektionen und Selektionen nachgebildet werden.

Als Beispiel für verschiedene JOIN's dienen die beiden Tabellen :

Tab A : Person

Nr	Nachname	Vor
1	Naumann	Karl
2	Baumann	Beate
3	Hartmann	Heike
4	Naumann	Jens

Tab B : Telefon

Nr	Telefon
1	12345
7	98765
10	24680

Der **CROSS-JOIN** bildet das kartesische Produkt. Jeder Datensatz der Tabelle A wird mit jedem Datensatz der Tabelle B kombiniert.

```
SELECT * FROM TabA CROSS JOIN TabB ;
```

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	1	12345
1	Naumann	Karl	7	98765
1	Naumann	Karl	10	24680
2	Baumann	Beate	1	12345
3	Hartmann	Heike	10	24680
4	Naumann	Jens	1	12345
4	Naumann	Jens	7	98765
4	Naumann	Jens	10	24680

Mit einem **THETA-JOIN** werden bestimmte Datensätze aus dem kartesischen Produkt durch eine Bedingung ausgewählt (z.B.: TabA.Nr = TabB.Nr).

```
SELECT * FROM TabA CROSS JOIN TabB
         WHERE TabA.Nr < TabB.Nr ;
```

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	7	98765
2	Baumann	Beate	7	98765
3	Hartmann	Heike	7	98765
4	Naumann	Jens	7	98765
1	Naumann	Karl	10	24680
2	Baumann	Beate	10	24680
3	Hartmann	Heike	10	24680
4	Naumann	Jens	10	24680

Der **INNER-JOIN (EQUI-JOIN)** ist einSpezialfall des THETA-JOIN . Dabei werden zwei Felder auf Gleichheit (deshalb EQUI-JOIN) getestet (also hier: TabA.Nr = TabB.Nr) .

```
SELECT * FROM TabA INNER JOIN TabB ON TabA.Nr = TabB.Nr;
```

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	1	12345

Der **NATURAL-JOIN** ist mit dem INNER-JOIN vergleichbar, außer dass in der Ergebnistabelle keine identischen Spalten enthalten sind.

```
SELECT * FROM TabA NATURAL JOIN TabB ;
```

A.Nr	Nachname	Vor	Telefon
1	Naumann	Karl	12345

Beim **LEFT-OUTER-JOIN** werden von der linken Tabelle alle Datensätze in die Ergebnismenge aufgenommen. Von der rechten Tabelle werden nur dazu korrespondierenden Datensätze übernommen (andernfalls bleibt das Feld leer → NULL).

```
SELECT * FROM TabA LEFT OUTER JOIN TabB ON TabA.Nr = TabB.Nr ;
```

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	1	12345
2	Baumann	Beate	NULL	NULL
3	Hartmann	Heike	NULL	NULL
4	Naumann	Jens	NULL	NULL

Analog werden beim **RIGHT-OUTER-JOIN** von der rechten Tabelle alle Datensätze in die Ergebnismenge aufgenommen. Von der Linken Tabelle werden nur dazu korrespondierendenDatensätze übernommen.

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	1	12345
NULL	NULL	NULL	7	98765
NULL	NULL	NULL	10	24680

```
SELECT * FROM TabA RIGHT OUTER JOIN TabB ON TabA.Nr = TabB.Nr ;
```

Die Ergebnismenge des **FULL-OUTER-JOIN** bildet eine Kombination aus LEFT-JOIN und RIGHT JOIN.

A.Nr	Nachname	Vor	B.Nr	Telefon
1	Naumann	Karl	1	12345
2	Baumann	Beate	NULL	NULL
3	Hartmann	Heike	NULL	NULL
4	Naumann	Jens	NULL	NULL
NULL	NULL	NULL	7	98765
NULL	NULL	NULL	10	24680

```
SELECT * FROM TabA LEFT OUTER JOIN TabB
         ON TabA.Nr = TabB.Nr
UNION
SELECT * FROM TabA RIGTH OUTER JOIN TabB
         ON TabA.Nr = TabB.Nr ;
```

Beim **SEMI-JOIN** werden die Tabellen über einen NATURAL-JOIN verbunden. Anschließend erfolgt eine Projektion auf die Spalten der ersten Tabelle.

Nr	Nachname	Vor
1	Naumann	Karl

```
SELECT TabA.Nr , Nachname , Vor FROM TabA NATURAL JOIN TabB ;
```

Beim **SELF-JOIN** wird eine der zuvor genannten JOIN-Arten angewendet, wobei aber nicht zwei verschiedene, sondern eine Tabelle mit sich selbst verbunden wird.

Nr	Nachname	Vor
1	Naumann	Karl
4	Naumann	Jens

```
SELECT C.* FROM TabA AS C INNER JOIN TabA AS D
WHERE C.Nachname = D.Nachname AND C.Nr != D.Nr ;
```

In MySQL gibt es aus Kompatibilitätsgründen eine Reihe von (beinahe) gleichwertigen JOIN-Varianten, die sich teilweise nur in der internen Optimierung bei der Verarbeitung unterscheiden. Nachfolgend ist ein (unvollständiger) Auszug der MySQL-Join Syntax dem Handbuch entnommen:

```
table_reference [INNER | CROSS] JOIN table_factor [join_condition]
table_reference STRAIGHT_JOIN table_factor
table_reference STRAIGHT_JOIN table_factor ON condition
table_reference LEFT [OUTER] JOIN table_reference join_condition
table_reference NATURAL [LEFT [OUTER]] JOIN table_factor
table_reference RIGHT [OUTER] JOIN table_reference join_condition
table_reference NATURAL [RIGHT [OUTER]] JOIN table_factor
```

Von besonderer praktischer Relevanz sind dabei der INNER-JOIN und der LEFT-JOIN, mit denen die meisten anderen Verbundarten abgebildet werden können.

Quellen : RRZN „SQL-Grundlagen und Datenbankdesign“8.Auflage April 2010
MySQL Onlinerefernz 5.1 (http://dev.mysql.com/doc/refman/5.1/de/join.html)