MySQL Join

A join enable you to retrieve records from two (or more) logically related tables in a single result set.

JOIN clauses are used to return the rows of two or more queries using two or more tables that shares a meaningful relationship based on a common set of values. These values are usually the same column name and data type that appear in both the participating tables being joined.

These columns, or possibly a single column from each table, are called the join key or common key. Mostly *but not all of the time*, the join key is the primary key of one table and a foreign key in another table. The join can be performed as long as the data in the columns are *matching*.

It can be difficult when the join involving more than two tables. It is a good practice to think of the query as a series of two table joins, when involvement of three or more tables in joins.

Types of Joins:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- STRAIGHT JOIN
- CROSS JOIN
- NATURAL JOIN

Here is the sample tables table_A and table_B, which we will use to understand the technologies behind the joins.

A	M
1	m
2	n
4	o

table_A

A	N
2	p
3	q
5	r

table_B

MySQL INNER JOIN

The INNER JOIN is such a JOIN in which all rows can be selected from both participating tables as long as there is a match between the columns. Usage of INNER JOIN combines the tables. An INNER JOIN allows rows from either table to appear in the result if and only if both tables meet the conditions specified in the ON clause.

Example

```
SELECT * FROM table_A  INNER JOIN table_B  ON table_A.A=table_B.A;
```

Output

A	M	N
2	n	p

MySQL LEFT JOIN

The LEFT JOIN is such a join which specifies that all records be fetched from the table on the left side of the join statement. If a record returned from the left table has no matching record in the table on the right side of the join, it is still returned, and the corresponding column from the right table returns a NULL value.

Example

SELECT * FROM table_A LEFT JOIN table_B ON table_A.A=table_B.A;

Output

A	M	A	N
2	n	2	p
1	m	NULL	NULL
4	o	NULL	NULL

MySQL RIGHT JOIN

The RIGHT JOIN is such a join which specifies that all records be fetched from the table on the right side of the join statement, even if the table on the left has no matching record. In this case, the columns from the left table return NULL values.

Example

SELECT * FROM table_A   RIGHT JOIN table_B ON table_A.A=table_B.A;

Output

A	M	A	N
2	n	2	p
NULL	NULL	3	q
NULL	NULL	5	r

MySQL STRAIGHT JOIN

A STRAIGHT_JOIN is such a join which scans and combines matching rows (if specified any condition) which are stored in associated tables otherwise it behaves like an INNER JOIN or JOIN without any condition.

Example

SELECT * FROM table_A STRAIGHT JOIN table_B;

Output

A	M	A	N
1	m	2	p
2	n	2	p
4	o	2	p
1	m	3	q
2	n	3	q
4	o	3	q
1	m	5	r
2	n	5	r
4	o	5	r

MySQL CROSS JOIN

A CROSS JOIN is such a join which specifies the complete cross product of two tables. For each record in the first table, all the records in the second table are joined, creating a potentially huge result set. This command has the same effect as leaving off the join condition, and its result set is also known as a **Cartesian product**.

Example

SELECT * FROM table_A CROSS JOIN table_B;

Output

A	M	A	N
1	m	2	p
2	n	2	p
4	o	2	p
1	m	3	q
2	n	3	q
4	o	3	q
1	m	5	r
2	n	5	r
4	o	5	r

MySQL NATURAL JOIN

A NATURAL JOIN is such a join that performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.
Example

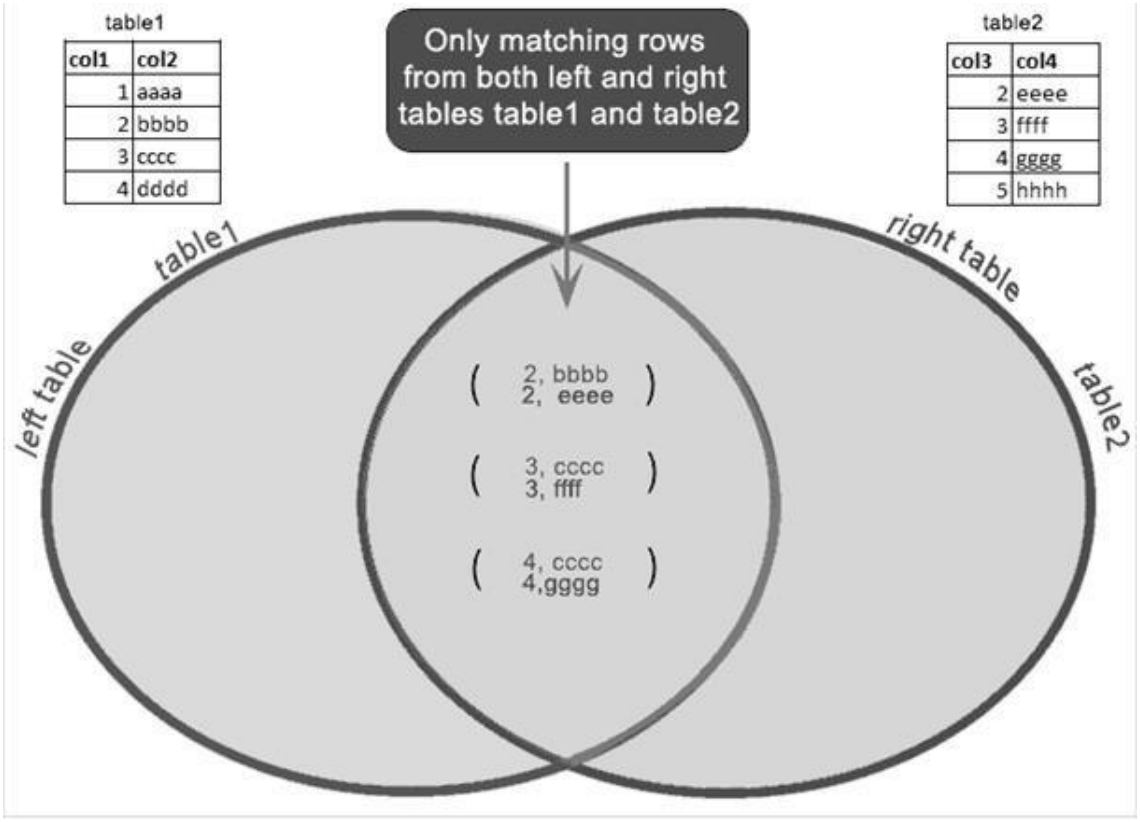
SELECT * FROM table_A NATURAL JOIN table_B;

Output

A	M	N
2	n	p

Pictorial Description of different JOINS

INNER JOIN



Inner join with three tables: Consider

the following three tables:

table - doctors

+-------+---------+

| docid | dname |
+-------+---------+
| 1 | A.VARMA |
| 2 | D.GOMES |
+-------+---------+

table - specialize +------+----------+-------+
| spid | desc | docid |
+------+----------+-------+
| 1 | special1 | 1 |
| 2 | special2 | 2 |
+------+----------+-------+

table - timeschedule
+-----+------+----------+-------+
| tid | tday | sit_time | docid |
+-----+------+----------+-------+
1	MON	17:00:00	1
2	WED	08:00:00	1
3	TUE	16:00:00	2
4	FRI	09:00:00	2
+-----+------+----------+-------+

Here, in the above tables, they are related to each other.
In *doctors*, *specialize* and *timeschedule* tables the *docid*, *spid* and *tid* are primary key consecutively. The *docid* in *specialize* table and *timeschedule* tables are foreign key, which are reference to primary key *docid* of *doctors* table.

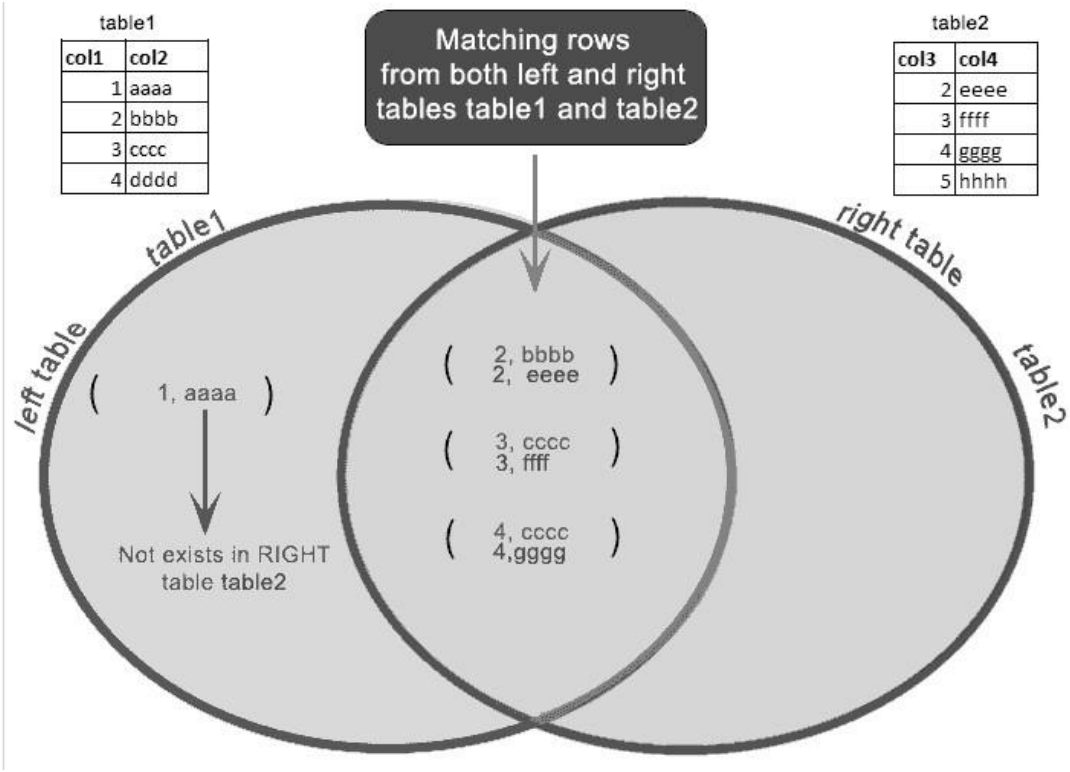
If we want all records for a doctor who are specialized in *special1* and seat in his chamber on wednesday (WED) in his schedule time, the following sql can be used.

```
SELECT a.docid,a.dname,
       b.desc,c.tday,c.sit_time
FROM doctors a
    INNER JOIN specialize b
     ON a.docid=b.docid
    INNER JOIN timeschedule c
     ON a.docid=c.docid
        WHERE a.docid=1 AND c.tday='WED';
```

Output:
+-------+---------+----------+------+----------+
| docid | dname | desc | tday | sit_time |
+-------+---------+----------+------+----------+

| 1 | A.VARMA | special1 | WED | 08:00:00 |
+-------+---------+----------+------+----------+

LEFT JOIN



RIGHT JOIN

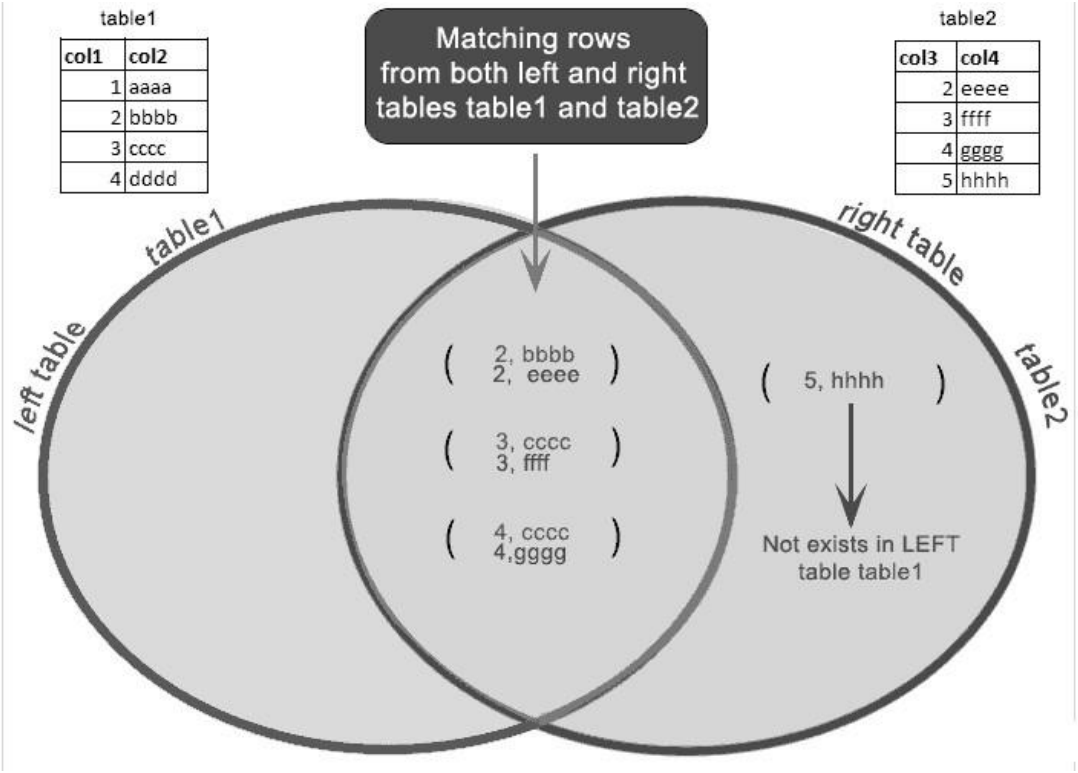


table1
col1 col2
1 aaaa
2 bbbb
3 cccc
4 dddd
Matching rows from both left and right tables table1 and table2
table2
col3 col4
2 eeee
3 ffff
4 gggg
5 hhhh
left table
table1
right table
table2
(2, bbbb 2, eeee)
(3, cccc 3, ffff)
(4, cccc 4,gggg)
(5, hhhh)
Not exists in LEFT table table1