MySQL JOIN Types

Understanding JOIN statements in MySQL

A join enables 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 datatype 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 involves more than two tables. It is a good practice to think of the query as a series of two table joins when the involvement of three or more tables in joins.

Types of MySQL JOINs

- Inner Join
- Left Join
- Right Join
- Straight Join
- Cross Join
- Natural Join

Here is the sample tables table_A and table_B, which are used below to explain the technologies behind the joins.

A	M
1	m
2	n
4	o

table_A

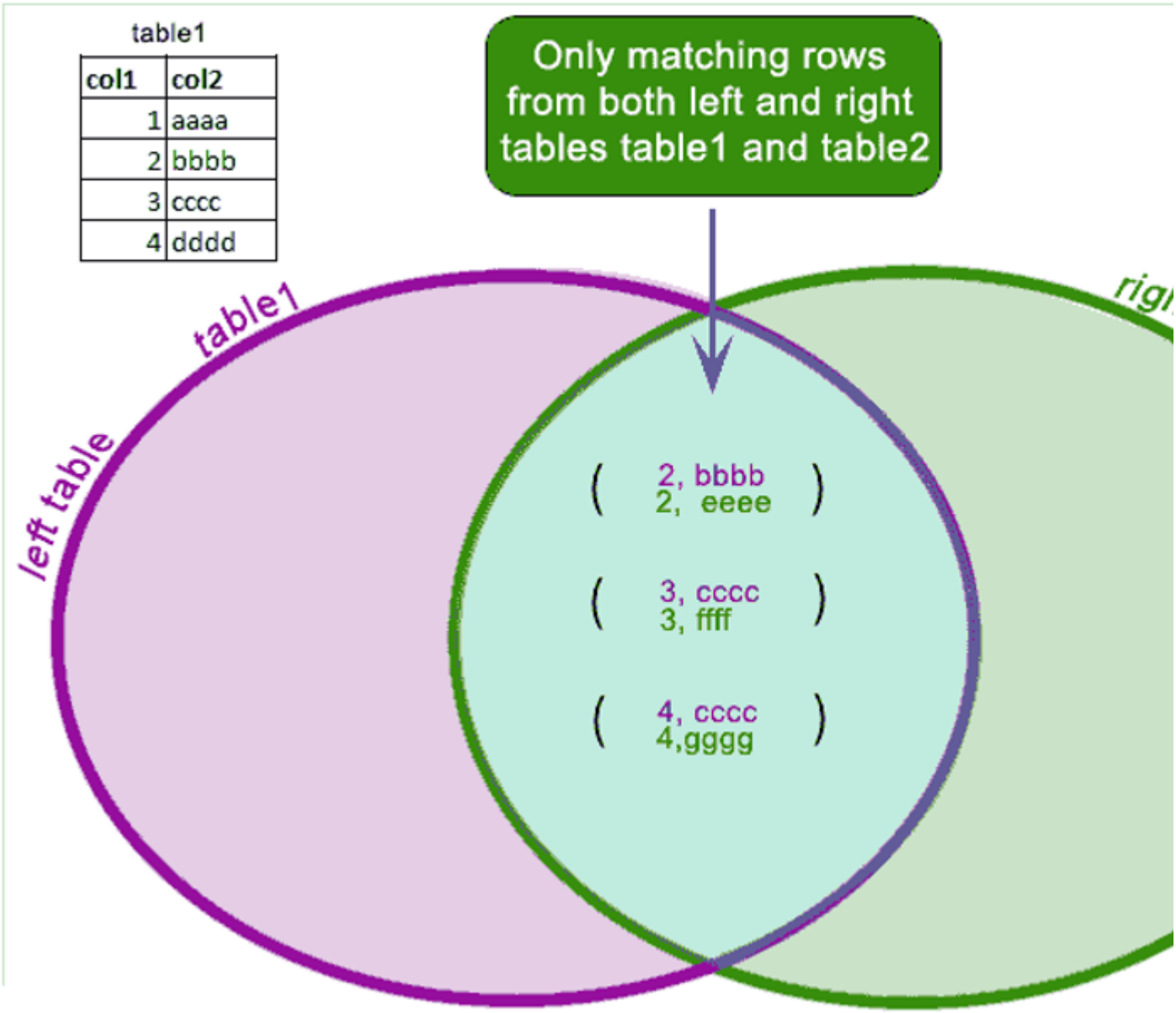
A	N
2	p
3	q
5	r

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.

In MySQL, the INNER JOIN selects all rows from both participating tables to appear in the result if and only if both tables meet the conditions specified in the ON clause. JOIN, CROSS JOIN, and INNER JOIN are syntactic equivalents. In standard SQL, they are not equivalent. INNER JOIN is used with an ON clause, CROSS JOIN is used otherwise.



When combining records from more than one tables, a user needs to indicate how records in a table can be matched to records in the other table.

Example Query:

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

MySQL INNER JOIN Sample using 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 |
+-----+------+----------+-------+
```

The above tables are related to each other. In *doctors*, *specialize* and *timeschedule* tables the *docid*, *spid* and *tid* are primary key respectively. The *docid* in *specialize* table and *timeschedule* tables are a foreign key, which is the 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';
```

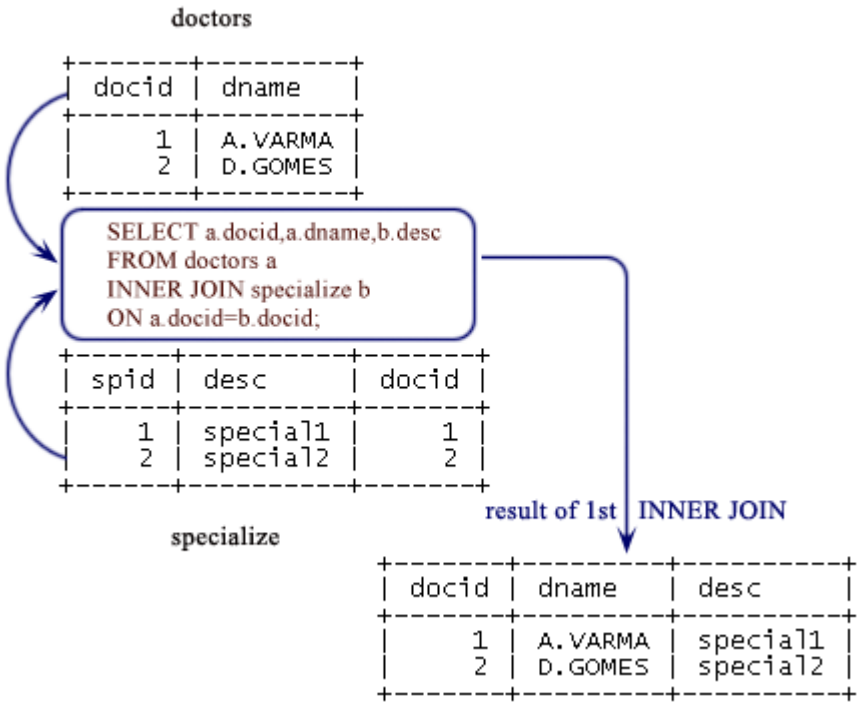
Sample Output:

```
+-------+---------+----------+------+----------+
| docid | dname   | desc     | tday | sit_time |
+-------+---------+----------+------+----------+
|     1 | A.VARMA | special1 | WED  | 08:00:00 |
+-------+---------+----------+------+----------+
```

Detailed Explanation (how this works internally):

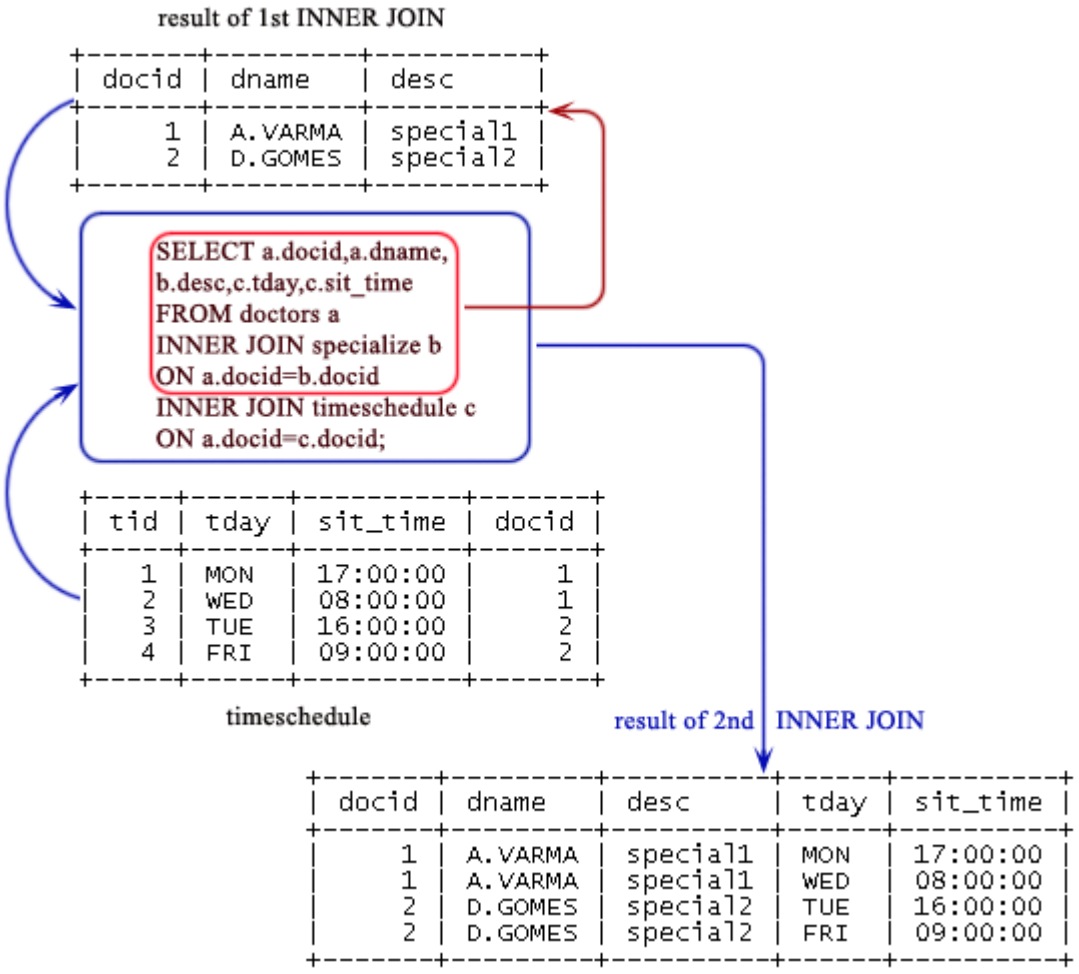
Step-1:

```
SELECT a.docid, a.dname, b.desc
FROM doctors a
INNER JOIN specialize b
   ON a.docid=b.docid;
```



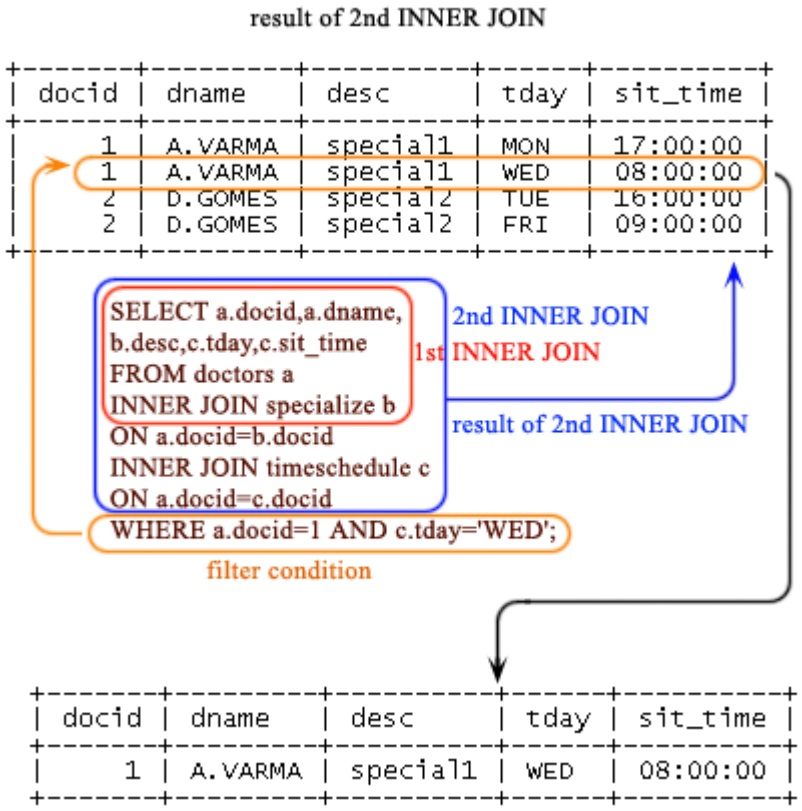
Step-2:

```
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;
```



Step-3:

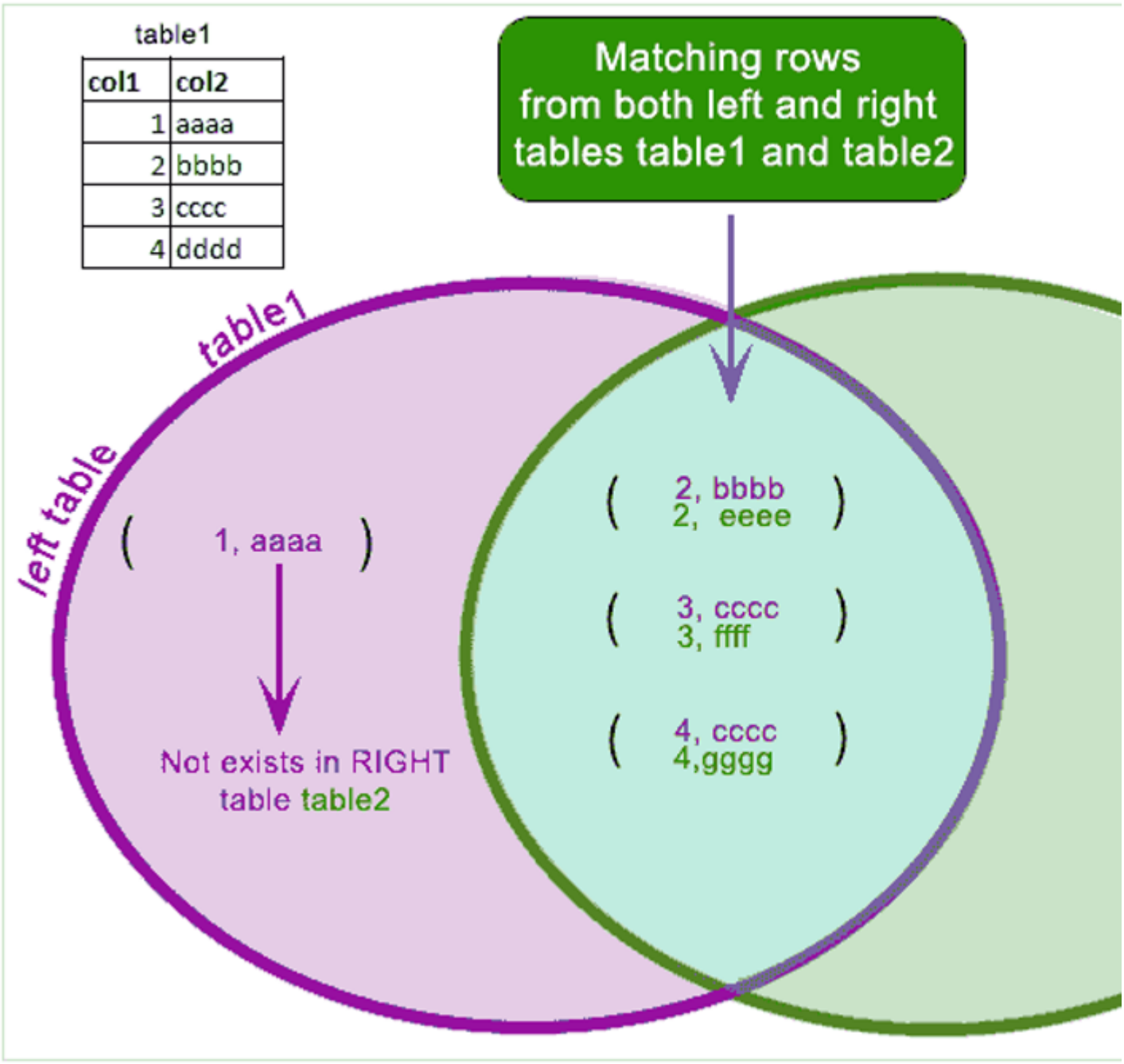
```
SELECT a.docid,a.dname, b.desc,c.tday,c.sit_time
FROM doctors a
     INNER JOIN specialize b
        ON a.docid=b.docid
     INNER JOIN timeschedule c
        ON a.docid=c.docid
WHERE a.docid=1 AND c.tday='WED';
```



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.

In MySQL, LEFT JOIN joins two tables and fetches rows based on a condition, which is matching in both the tables and the unmatched rows will also be available from the table written before the JOIN clause.



In MySQL LEFT JOIN, the condition is used to decide how to retrieve rows from the second table. If there is a row in 1st_table that matches the WHERE clause, but there is no row in 2nd_table that matches the ON condition, an extra 2nd_table row is generated with all columns set to NULL.

So, in case of LEFT JOIN or LEFT OUTER JOIN, MySQL:

1. takes all selected values from the left table
2. combines them with the column names (specified in the condition) from the right table
3. retrieve the matching rows from both the associated tables.
4. sets the value of every column from the right table to NULL which is unmatched with the left table.

Example Query:

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

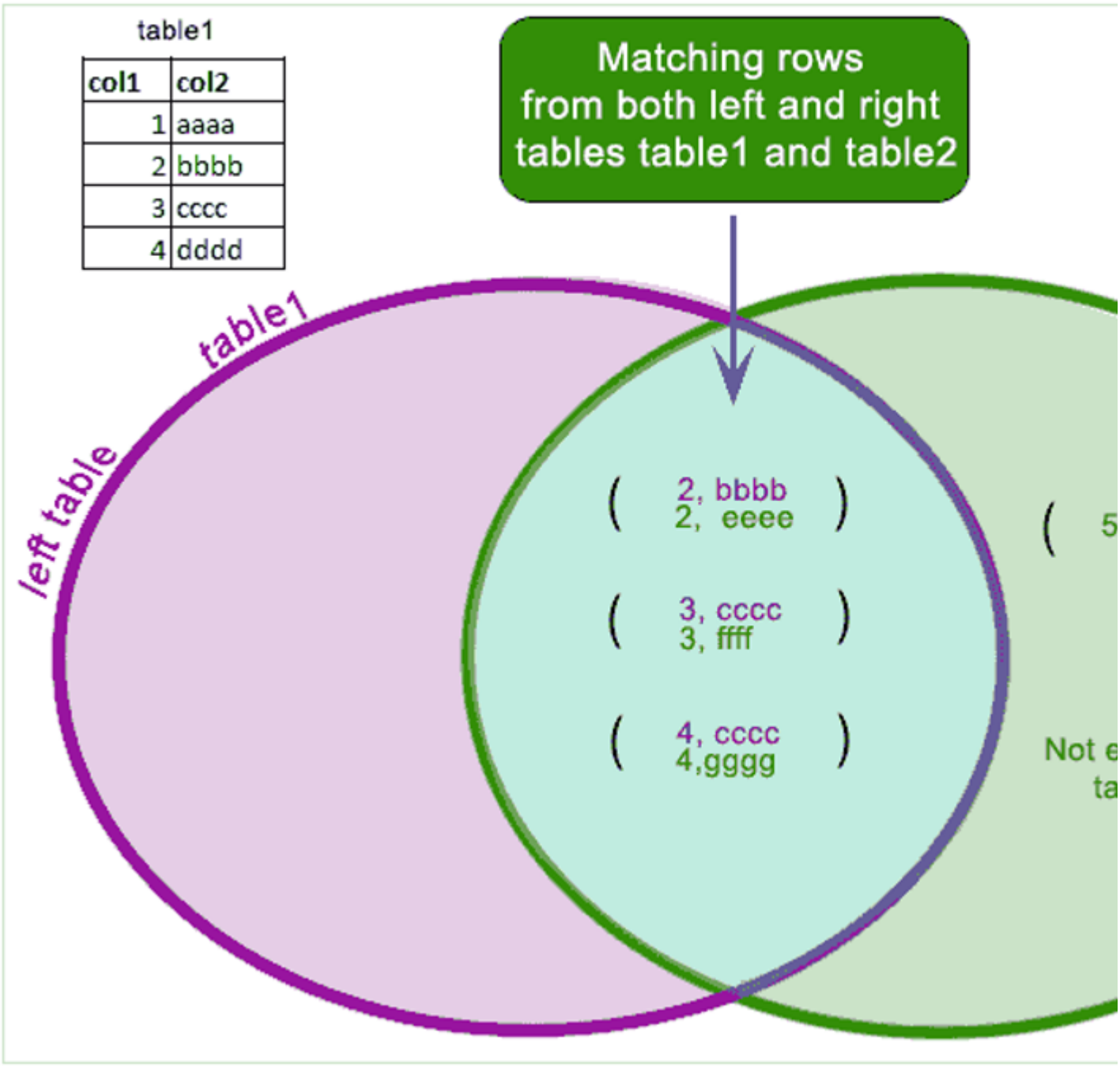
Output

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.

In MySQL, Right Join joins two tables and fetches rows based on a condition, which is matching in both the tables and the unmatched rows will also be available from the table written after the JOIN clause.



Example Query:

```
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 other wise it behaves like an INNER JOIN or JOIN of without any condition.

"STRAIGHT_JOIN is similar to JOIN, except that the left table is always read before the right table. This can be used for those (few) cases for which the join optimizer puts the tables in the wrong order." - **MySQL Manual** **(https://dev.mysql.com/doc/refman/8.0/en/join.html)** .

Example Query:

```
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

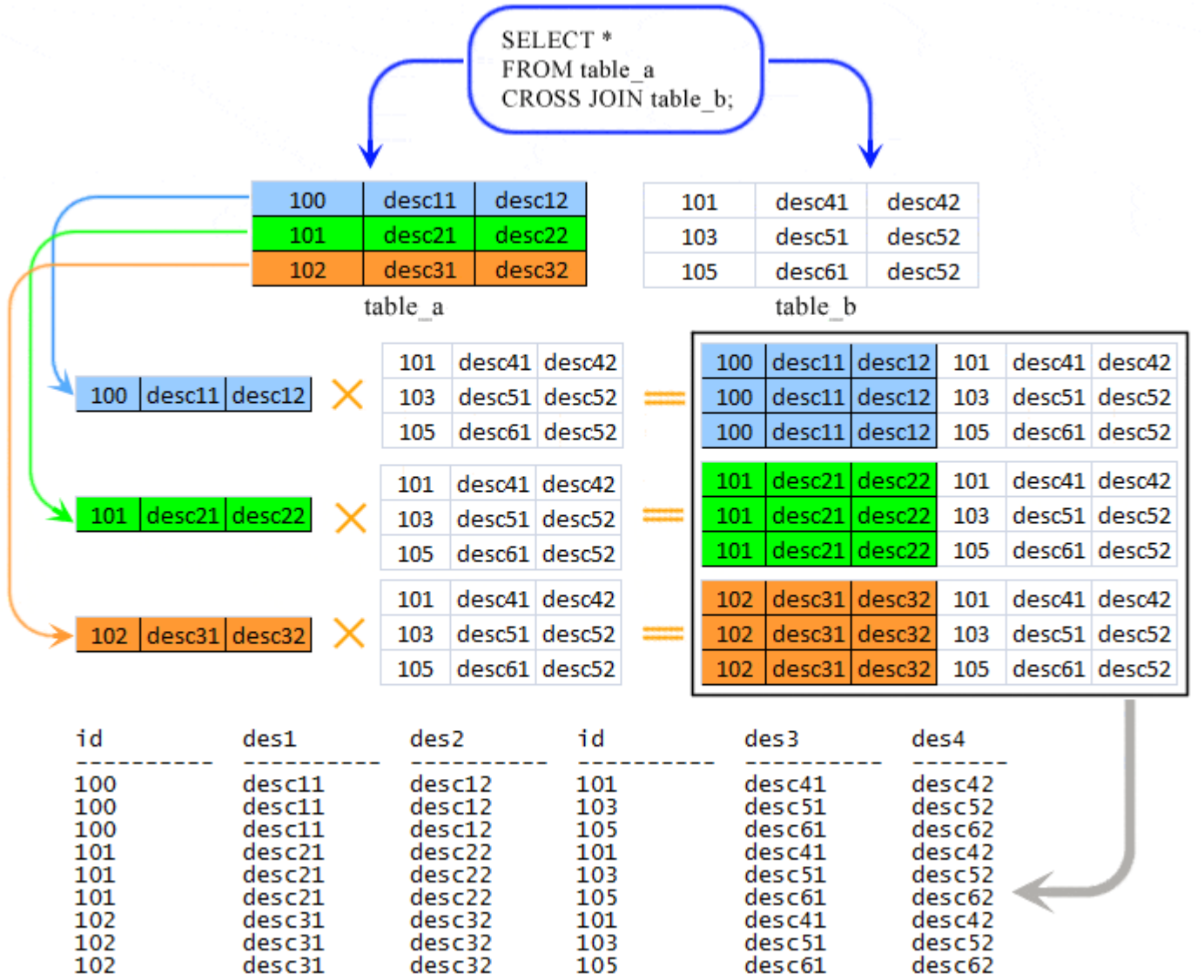
A Cross Join is such a join which specifies the complete cross product of two tables. For each record in the first table, all the records in the second table are joined, creating a potentially huge result set. This command has the same effect as leaving off the join condition, and its result set is also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

In MySQL, the CROSS JOIN produces a result set which is the product of rows of two associated tables when no WHERE clause is used with CROSS JOIN.

In this join, the result set appeared by multiplying each row of the first table with all rows in the second table if no condition introduced with CROSS JOIN. This kind of result is called as Cartesian Product.

In MySQL, the CROSS JOIN behaves like JOIN and INNER JOIN of without using any condition.

In standard SQL, the difference between INNER JOIN and CROSS JOIN is ON clause that can be used with INNER JOIN, on the other hand ON clause can't be used with CROSS JOIN.



Example Query:

```
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

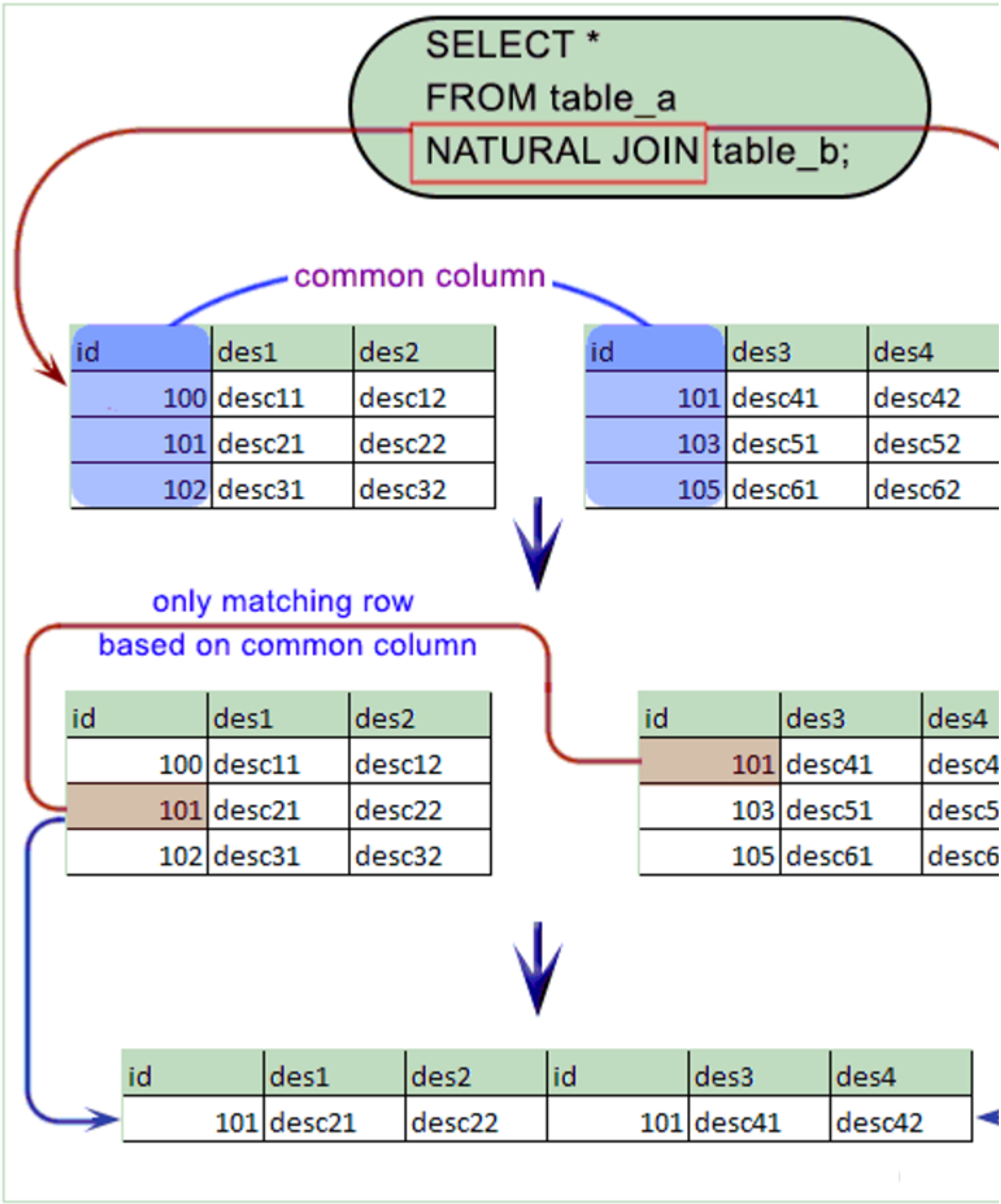
A Natural Join is such a join that performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.

In MySQL, the 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.

The MySQL NATURAL JOIN is structured in such a way that, columns with the same name of associate tables will appear once only.

Natural Join Guidelines:

- The associated tables have one or more pairs of identically named columns.
- The columns must be the same data type.
- Don’t use ON clause in a NATURAL JOIN.



Example Query:

```
SELECT * FROM table_A
NATURAL JOIN table_B;
```