MySQL
Practical 8 MySQL Join & Advance Query

MySQL Join

A relational database consists of multiple related tables linking together using common columns which are known as foreign key columns.

For example, Table client_master and Sales_order have are linked via clientno column.

A MySQL join is a method of linking data between one (self-join) or more tables based on values of the common column between tables.

MySQL supports the following types of joins:

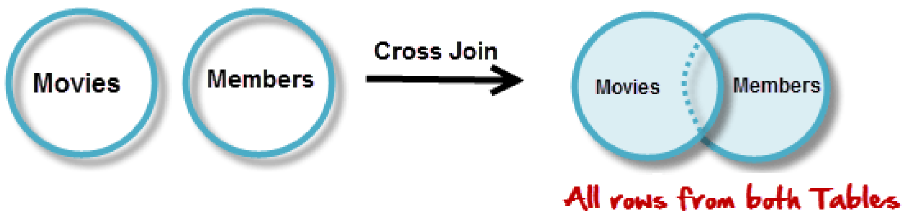
1. Cross join
2. Inner join
3. Left join
4. Right join

The join clause is used in the SELECT statement appeared after the FROM clause. **Notice that MySQL does not support full outer join**.

Cross Join

cross JOIN is a simplest form of JOINs which matches each row from one database table to all rows of another.

It is a cartasian product between two table. (r1Xr2) r1 and r2 are two tables.



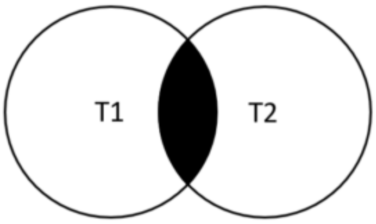
e.g **mysql> select *from student cross join stud_sub;**
it will display the all columns of student and stud_sub table.
It will display all record combination with _sub record students records and stud.

mysql> select *from student cross join subject;

```
| sid | name    | sname   | subno |
+-----+---------+---------+-------+
|   1 | Simon   | CONSM   |     1 |
|   1 | Simon   | DBMS    |     2 |
|   1 | Simon   | physics |     3 |
|   1 | Simon   | Maths   |     4 |
|   1 | Simon   | Biology |     5 |
|   2 | Alvin   | CONSM   |     1 |
|   2 | Alvin   | DBMS    |     2 |
|   2 | Alvin   | physics |     3 |
|   2 | Alvin   | Maths   |     4 |
```

INNER JOIN

The inner JOIN is used to return rows from both tables that satisfy the given condition.



Syntax : SELECT *column_name(s)*
FROM *table1*
INNER JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

e.g select *from student inner join stud_sub on student.sid=stud_sub.sid ;

```
----+---------+-----+-------+-------------+-------+
| sid | name    | sid | subid | teachername | marks |
+-----+---------+-----+-------+-------------+-------+
|   1 | Simon   |   1 |     1 | Reshma      |    62 |
|   1 | Simon   |   1 |     2 | Vihar       |    50 |
|   1 | Simon   |   1 |     3 | Bhavik      |    55 |
|   2 | Alvin   |   2 |     1 | Jigar       |    64 |
|   2 | Alvin   |   2 |     2 | kamlesh     |    68 |
|   2 | Alvin   |   2 |     3 | suhana      |    72 |
|   2 | Alvin   |   2 |     4 | Reshma      |    59 |
|   2 | Alvin   |   2 |     5 | Vihar       |    71 |
|   3 | vidya   |   3 |     1 | Jigar       |    65 |
|   3 | vidya   |   3 |     2 | Bhavik      |    66 |
|   3 | vidya   |   3 |     3 | suhana      |    54
```

OUTER JOIN

LEFT JOIN

The LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right.

Select *from category;

```
| cat_id | name     |
|      2 | spritual |
|      3 | business |
|      4 | food     |
+--------+----------+
```

|mysql> select *from post;

```
+--------------+----------------------------+------------+------+
| title        | content                    | createdon  | id   |
+--------------+----------------------------+------------+------+
| punjabi      | sabji recipe is here       | 2020-10-14 |    4 |
| south indian | south indian recipe is here | 2020-10-15 |    4 |
| newshare     | newshare rises highh       | 2020-12-12 |    3 |
```

mysql> select id,name,title from category left join post on post.id=category.cat_id;

```
+------+----------+--------------+
| id   | name     | title        |
+------+----------+--------------+
| NULL | spritual | NULL         |
|    3 | business | newshare     |
|    4 | food     | punjabi      |
|    4 | food     | south indian |
+------+----------+--------------+
```

- **RIGHT JOIN**

RIGHT JOIN is obviously the opposite of LEFT JOIN. The RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL

mysql> select id,name,title from category right join post on post.id=category.cat_id;

```
+------+----------+--------------+
| id   | name     | title        |
+------+----------+--------------+
|    4 | food     | punjabi      |
|    4 | food     | south indian |
|    3 | business | newshare     |
```

- **USING CLAUSE**
  USING clause can also be used for the same purpose. The difference with USING is it needs to **have identical names for matched columns in both tables.**

ADVANCE QUERY

- **CASE STATEMENT WITH SELECT CLAUSE**

MySQL CASE expression is a control flow structure that allows you to add if-else logic to a query. Generally speaking, you can use the CASE expression anywhere that allows a valid expression e.g., SELECT, WHERE and ORDER BY clauses.

Syntax :

```
CASE
    WHEN  condition1 THEN result1
    WHEN  Condition2 THEN result2
    WHEN  conditionN THEN resultN
    ELSE  result
END [ AS LABEL]
```

Parameters :

- *condition1, condition2, ...conditionN* : Required. The conditions. These are evaluated in the same order as they are listed
- *result1, result2, ...resultN* : Required. The value to return once a condition is true

- If no conditions are true, it will return the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.

E.g 1. Sort the data accordint to city , if city is null then sort according to country.

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
   WHEN City IS NULL THEN Country
   ELSE City
END);
```

E.G 2

E.g 2. Display the location according to city

```
select name,city,
case city
      when 'AHMD' then 'near location'
      when 'baroda' then 'far distance'
      when  'anand' then 'mid distance'
      else 'too far to reach'
end as location
 from persons;
```

output :

```
+---------+--------+------------------+
| name    | city   | location         |
+---------+--------+------------------+
| abcd    | AHMD   | near location    |
| aaaa    | AHMD   | near location    |
| bbbb    | AHMD   | near location    |
| aaaa    | baroda | far distance     |
| sdkskdl | anand  | mid distance     |
| dimpal  | AHMD   | near location    |
| harshil | baroda | far distance     |
```

e.g 3 Display the grade of students for every marks from stud_sub table.
if marks is 50-65 then grade is ‘average’
if marks is 66-80 then grade is ‘good’
if mark is 81-99 then grade is ‘very good’
else grade is ‘not acceptable marks’

sql>

```
  select *,
 case
     when marks >=50 and marks <= 65 Then 'average'
     when marks >= 66 and marks <= 80 Then 'good'
     when marks >= 81 and marks <= 99 Then 'very good'
     else 'not acceptable marks'
 end as class
  from stud_sub;
```

output

```
sid | subid | teachername | marks | class     |
+-----+-------+-------------+-------+-----------+
|   1 |     1 | Reshma      |    62 | average   |
|   1 |     2 | Vihar       |    50 | average   |
|   1 |     3 | Bhavik      |    55 | average   |
|   2 |     1 | Jigar       |    64 | average   |
|   2 |     2 | kamlesh     |    68 | good      |
```

- Case statement with group by clause

  e.g Find the max marks for each subect. If max mark is < 70 then print ok o.w print good as a result

```
sql> select subid,max(marks) as maxmarks,
case
   when max(marks) <= 70 then 'ok'
   else 'good'
 end as result
 from stud_sub group by subid;
```

output :

```
-------+----------+--------+
| subid | maxmarks | result |
+-------+----------+--------+
|     1 |       81 | good   |
|     2 |       70 | ok     |
|     3 |       72 | good   |
|     4 |       69 | ok     |
|     5 |       79 | good   |
```

- Case statement with Update clause

Case statement can be used with the update

e.g add the grade into stud_sub table according to marks
if marks is 50-65 then grade is ‘average’
if marks is 66-80 then grade is ‘good’
if mark is 81-99 then grade is ‘very good’
else grade is ‘not acceptable marks’

```
sql> update stud_sub set grade = case
  when marks >=50 and marks <= 65 Then 'average'
  when marks >= 66 and marks <= 80 Then 'good'
  when marks >= 81 and marks <= 99 Then 'very good'
  else 'not acceptable marks'
end ;
```

output:
```
| sid | subid | teachername | marks | grade     |
+-----+-------+-------------+-------+-----------+
|   1 |     1 | Reshma      |    62 | average   |
|   1 |     2 | Vihar       |    50 | average   |
|   1 |     3 | Bhavik      |    55 | average   |
|   2 |     1 | Jigar       |    64 | average   |
|   2 |     2 | kamlesh     |    68 | good      |
```

Exercise on JOIN

1. display all details of every client clients as well as order details of clients. Using leftjoin
2. Display only those salesman name who has supplies the order (hint innerjoin)
3. display only those product name who has been ordered.(hint inner join)
4. Display salesman name,city,saleamout,clientno,orderno,orderdate,orderstatus of only those sales man who have order. Using right join.
5. displaydescription, description,qtyonhand,reorderlvl,sellprice,qtyorder,orderno for all product Using left join .

Exercise on Case Statement

based on practical -2 table

1. Based on sale price of product display the message as result.
   If saleprice is less than 500 , result message is “its not costaly”
   o.w display the result message “it is costaly”
   Output should be like
   Prductno , sellprice, result
2. Calculate the bonus for the salesman based on sales amount
   If sales_amount is 1000-2000 give 2% bonus
   If sales_amount is 2001-3000 give 3% bonus
   If sales_amount greater than 3000 give 5% bonus
   o.w don’t give any bonus.

   Output should be like
   Salesmanno, sales_amount, bounus

3. calculate total quantity ordered of each product .

if is 5-10 message is “good sailing item”
if is 10-15 message is “very good sailing item”
else message “no sale”

Based on employee table

4. Add field bonus float(10,2) in office table;
   Update the bonus of every employee based on salary
   If salary is 5000-10000 then bonus is 2%
   If salary is 10,001 – 15 then bonus is 3 %
   If salary is greater than 15,000 then bonus is 5%
   Else bonus is 0%