MySQL
Practical 8 MySQL Join & Stored Procedure

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:

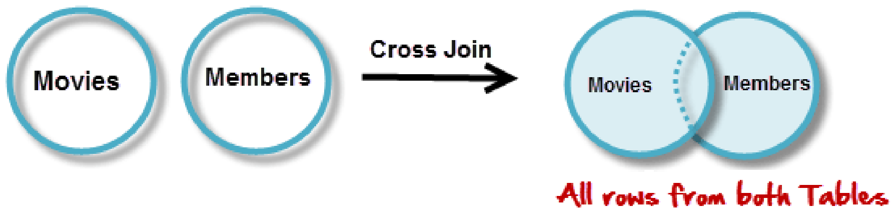
1. Cross join
2. Inner join
3. Left join
4. Right join

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.



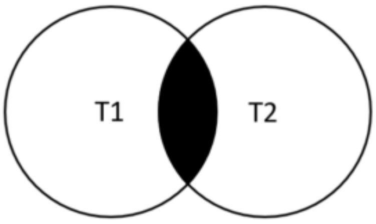
e.g **mysql> select *from student cross join stud_sub;**
it will display the all columns of student and stud_sub table.
It will display all record combination with students records and stud_sub record.

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.**

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 .

Compound Statement for stored procedure

1. BEGIN ... END Compound Statement

which can appear within stored programs (stored procedures and functions, triggers.

```
BEGIN
  Statement list
END
```

2. DECLARE Statement

The DECLARE statement is used to define various items local to a program: Local variable,conditions .

3. DECLARE LOCAL variable.
This statement declares local variables within stored programs.

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
e.g declare no int
```

4. Assign values to variable
Set is used to assign the value to variable.

```
SET variable_name = value;
```

```
e.g  DECLARE total INT DEFAULT 0;
     SET total = 10;
```

Another way to assign value to the variable **which is fetch from select statement**.

e.g

```
DECLARE productCount INT DEFAULT 0;

SELECT COUNT(*)
INTO productCount
FROM products;
```

5. IF Statement

```
IF search_condition THEN
       statement_list;
   [ELSEIF search_condition THEN
        statement_list] ...;
   [ELSE
         statement_list ;]
END IF;
```

6. WHILE Statement

```
WHILE search_condition DO
   statement_list
END WHILE;
```

STORED PROCEDURE

Syntax :

```
CREATE PROCEDURE procedure_name(parameter_list)
BEGIN
   statements;
END
```

- To execute a stored procedure, you use the CALL statement:
  Mysql>CALL stored_procedure_name(argument_list);

- To drop the procedure

Mysql> drop procedure procedure_name;

- **To Update procedure**
  1. First drop the procedure
  2. create again the procedure

E.G : simple procedure to display hello message.

Step 1 : change delimiter to $$

```
mysql> delimiter $$
```

step 2 : create procedure

```
mysql> create procedure disp()
 begin
  select "hello";
end $$
```

step 3 : change delimiter to ;

```
mysql>delimiter;
```

call the procedure

```
mysql> call disp();

+-------+
| hello |
+-------+
| hello |
+-------+
```

e.g **declare a variable and assign value and display value**

mysql>delimiter $$ // change delimiter to $$

```
mysql> create procedure vartest()
         begin
             declare n int(2);
             set n =10;
            select n;
             end $$
Query OK, 0 rows affected, 1 warning (0.13 sec)

mysql> delimiter ;     / / change delimiter back to ;
mysql> call vartest();
+------+
| n    |
+------+
|   10 |
+------+
```

e.g **procedure containing if loop**

```
mysql > delimiter //

mysql > create procedure iftest()
             begin
            declare n int(2);
             set n = 10;
                if n>10 then
                          select concat(n,' is greater');
              else
               select concat(n,' is smaller') ;
            end if;
           end //

mysql> delimiter ;                              -> set delimiter semicolon
```

call the procedure

```
mysql>   call iftest();
+---------+
| smaller |
+---------+
| smaller |
```

e.g procedure of while loop

step 1 change delimiter to $$

```
mysql> delimiter $$
```

step 2 : create procedure

```
create procedure whiletest()
begin
  declare n int(2) ;
  set n = 10;
  while n > 0 do
      select n;
     set n = n -1;
  end while;
end $$
```

step 3 : change delimiter to ;

```
mysql> delimiter ;
```

step 4 : call the procedure

```
mysql> call whiletest();
```

e.g procedure using the query

```
mysql > delimiter $$

mysql>  create procedure selecttest()
begin
  declare n int(2) ;
  set n = 20;
  select *from employee where age > n;
end $$

mysql> delimiter ;

mysql> call selecttest();

+--------------+------+--------+-------------+------------+----------+------------+
| name         | age  | city   | designation | department | salary   | joindate   |
+--------------+------+--------+-------------+------------+----------+------------+
| rohan patel  |   26 | NULL   | salesman    | sales      |  9000.00 | NULL       |
| virat        |   32 | mumbai | admin       | admin      | 10000.00 | NULL       |
| sameer       |   32 | mumbai | accountant  | admin      | 12000.00 | 2011-09-27 |
| hares      |  24 | NULL  | salesman   | sales     | 11000.00 | NULL      |
```

- **Parameters in stored procedure.**

Parameter syntax :

```
[IN | OUT | INOUT] parameter_name datatype[(length)]
```

IN Parameter

IN is the default mode. When you define an IN parameter in a stored procedure, the calling program has to pass an argument to the stored procedure.

OUT Parameter

The value of an OUT parameter can be changed inside the stored procedure and its new value is passed back to the calling program.

IN-OUT Parameter:

n INOUT parameter is a combination of IN and OUT parameters. It means that the calling program may pass the argument, and the stored procedure can modify the INOUT parameter.

IN parameter example

e.g   Create a procedure which takes two parameters for department and age. And display records from employee table for department and age is greater than given age in parameter.

 IN parameter will pass the value to the procedure

Step 1 : delimiter $$

Step 2 :

```
  mysql> create procedure display(IN dept varchar(30), IN a int(2) )
    begin
           select *from employee where department=dept and age > a ;
     end $$
```

Step 3 : delimiter ;

Step 4 : call procedure.

Mysql > call display('production',25);

-OUT Prarameter

e.g procedure for demonstrate OUT parameter

create procedure which pass the department name as arguments calculate total of the salary of given procedure. Total of salary is stored in OUT parameter.

Step 1 : delimiter $$

Step 2 : -

```
        create procedure outtest(IN dept varchar(10) , OUT total int)
    begin
     select sum(salary) into total from employee where department=dept;
    end $$
```

step 3 : delimiter ;

step 4 : **call outtest('production',@total);**

step 5 : **select @total;**

Display the List of procedure in the database

Syntax : show procedure status where db = 'databasename';

e.g mysql>show procedure status where db='testdb';

Exercise for procedure

1 Create procedure called proc1 which declare one integer variable and one varchar variable and display both the variables.

2. Create procedure called proc2 in which declare the variable counter = and execute while loop until counter > 0 .

3. create procedure called proc3, which pass the argurment N. and procedure make total of first N number. E.g N =5 then sum = (1+2+3+4+5) = 15 use while loop.

4. create a procedure called proc4 which pass the student id in parameter and find average of marks of given student id from stud_sub table. E.g call proc2(1)

5. Create procedure called proc5 in which pass the number and display whether number is odd or even. [hint if mod(n,2) = 0 then]

6. create procedure called proc6 which pass the orderid as parameter and find the total quantity order form sales_order_detail, total of quantity order should be stored in OUT parameter.