fingerTips

Module-5

Joins in SQL

1) Joins in SQL

Join statement in SQL is used to combine data or rows from two or more tables based on a common field between them.

There are many types of joins in SQL:

i. Inner Join
ii. Left Join
iii. Right Join
iv. Full Outer Join
v. Cross Join
vi. Self Join
vii. Equi Join
viii. Natural Join

Let's understand most important joins (inner, left, right, outer joins) with an example:

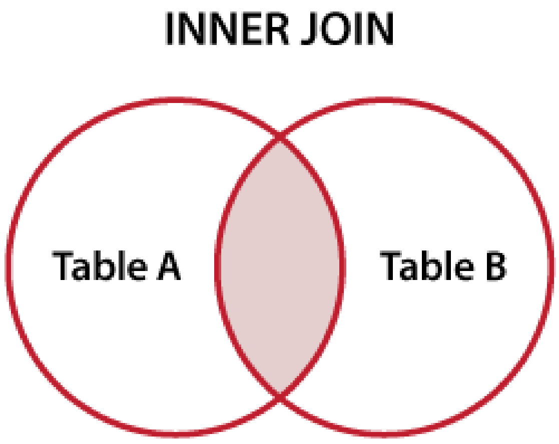
Student Table:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

Student Course:

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

Inner Join:

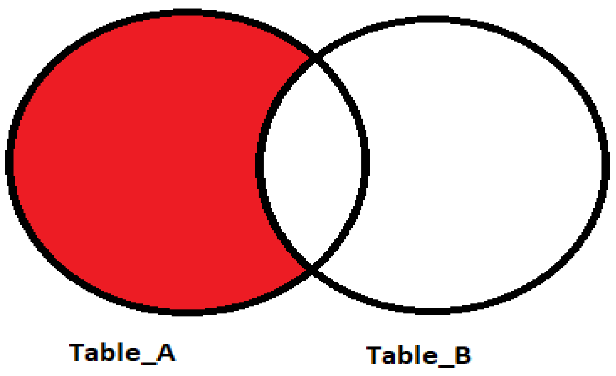


Syntax:

SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student INNER JOIN StudentCourse ON Student.ROLL_NO = StudentCourse.ROLL_NO;

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

Left Join:

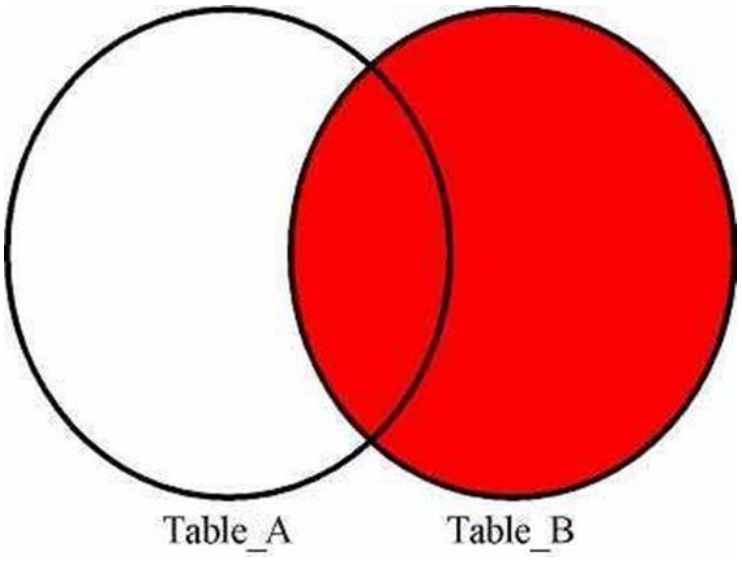


Syntax:

SELECT Student.NAME,StudentCourse.COURSE_ID FROM Student LEFT JOIN StudentCourse ON StudentCourse.ROLL_NO = Student.ROLL_NO;

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

Right Join:

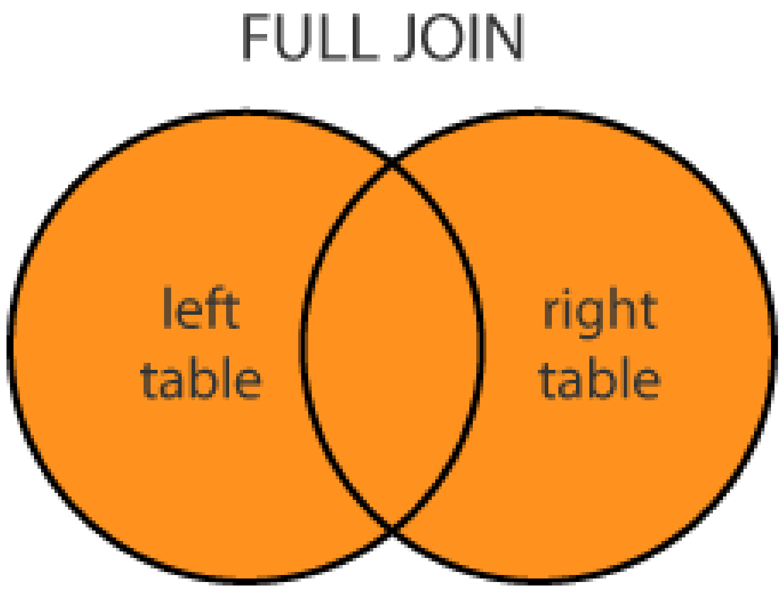


Syntax:

SELECT Student.NAME,StudentCourse.COURSE_ID FROM Student RIGHT JOIN StudentCourse ON StudentCourse.ROLL_NO = Student.ROLL_NO;

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

Full Join:



Syntax:

SELECT Student.NAME,StudentCourse.COURSE_ID FROM Student FULL JOIN StudentCourse ON StudentCourse.ROLL_NO = Student.ROLL_NO;

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*
NULL	9
NULL	10
NULL	11

2) Inner Join

An inner join returns only the matching rows between two tables. It compares the columns of both tables and returns the matching rows based on the join condition.

Syntax:

```
SELECT column1, column2, ...
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

Q. How many quantities in total were ordered by customers?

```
select c.name, sum(o.quantity)
from customers c inner join orders o
on c.id=o.customer_id
group by c.id;
```

3) Left Join

A left join returns all the rows from the left table and the matching rows from the right table. If there is no matching row in the right table, then NULL values are returned for the columns of the right table.

Syntax:

```
SELECT column1, column2, ...
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

Q. How many quantities of each sized product are ordered?

```
select p.size, sum(o.quantity) from
products p left join orders o
on p.id=o.product_id
group by p.size;
```

4) Right Join

A right join returns all the rows from the right table and the matching rows from the left table. If there is no matching row in the left table, then NULL values are returned for the columns of the left table.

Syntax:

```
SELECT column1, column2, ...
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

Q. Name the products which were ordered and number of quantities ordered.

```
select distinct(p.name)
from products p right join orders o
on o.product_id=p.id;
```

5) Full Outer Join

A full join returns all the rows from both tables, including the non-matching rows. If there is no matching row in one of the tables, then NULL values are returned for the columns of that table. We can emulate FULL OUTER JOIN using UNION of left join & right join.

Example:

Q. What are the total number of products and average amount spent on each product?

```
select p.id, count(p.id), avg(p.price)
from products p left join orders o
on p.id=o.product_id
group by p.id
union
select p.id, count(p.id), avg(p.price)
from products p right join orders o
on p.id=o.product_id
group by p.id;
```

6) Cross or Cartesian Join

A cross join returns the combination of all rows from both tables. It does not use any join condition. Cross join joins every row of a table to every row of some other table.

Syntax:

```
SELECT column1, column2, ...
FROM table1
CROSS JOIN table2;
```

Example:

```
select *
from products p cross join orders o;
```

OR, we can also specify a condition on columns.

```
select *
from products p cross join orders o
```

where p.id=o.product_id;

7) Self Join

A self join is used to join a table to itself. It is used when we need to compare the rows of the same table.

Syntax:

```
SELECT column1, column2, ...
FROM table1 t1
INNER JOIN table1 t2
ON t1.column_name = t2.column_name;
```

Example:

Q. Name the products that are having same price.

```
SELECT A.name as product1, B.name AS product2, A.price
FROM products A, products B
WHERE A.id<> B.id
AND A.price = B.price
ORDER BY A.color;
```

8) Equi Join

An equi join is a type of join that uses the equality operator (=) to compare the columns of both tables. It is used to return the matching rows between two tables based on the join condition. The word equi is used in reference to the = operator.

Syntax:

```
SELECT column1, column2, ...
FROM table1
JOIN table2
ON table1.column_name = table2.column_name;
```

Points to be noted:

- Inner join can have equality (=) and other operators (like <,>,<>) in the join condition.
- Equi join only have an equality (=) operator in the join

condition.

- Equi join can be an Inner join, Left Outer join, Right Outer join.

Example:

Q. Details of all those customers who have ordered something.

```
select *
from customers c join orders o
on c.id=o.customer_id;
```

9) Natural Join

A natural join is a type of join that returns the matching rows between two tables based on the same column names and data types. It does not require a join condition.

Syntax:

```
SELECT column1, column2, ...
FROM table1
NATURAL JOIN table2;
```

Example:

Q. Join products & orderes table without applying ON condition, check results & make conclusions out of that.

```
select *
from customers c natural join orders o;
```

Conclusion--> the drawback of using natural join is that it joins tables based on same column names.

10) Multiple Join (Combining 3 tables in SQL)

Name the customers who have ordered at-least 6 quantities and for price>140.

```
select c.name, sum(quantity), sum(price)
from (customers c join orders o on c.id=o.customer_id)
join products p on p.id=o.product_id
group by c.id
```

```
having sum(quantity)>=6 and sum(price)>140;
```