SQL | Join (Inner, Left, Right and Full Joins)

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN

Consider the two tables below as follows:

Student

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

StudentCourse

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

The simplest Join is INNER JOIN.

A. INNER JOIN

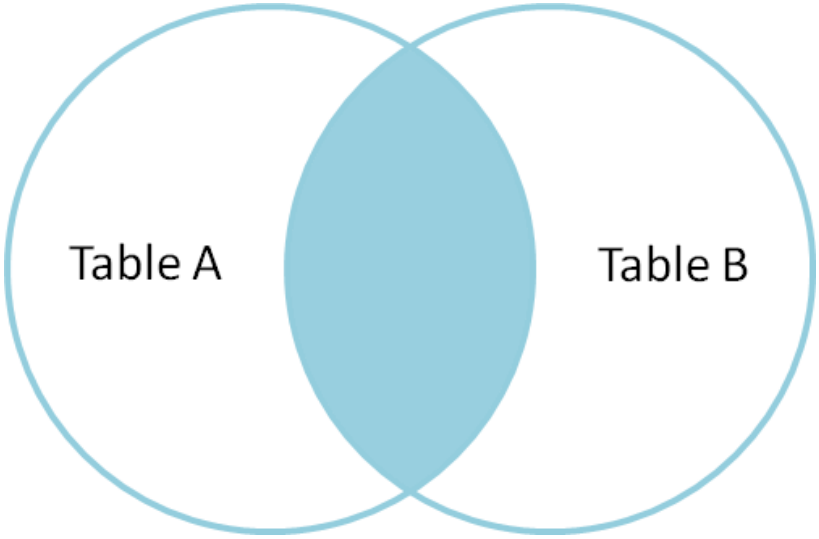
The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

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

table1: First table.
table2: Second table
matching_column: Column common to both the tables.

***Note**: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.*



Example Queries(INNER JOIN)

This query will show the names and age of students enrolled in different courses.

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

Output:

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

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

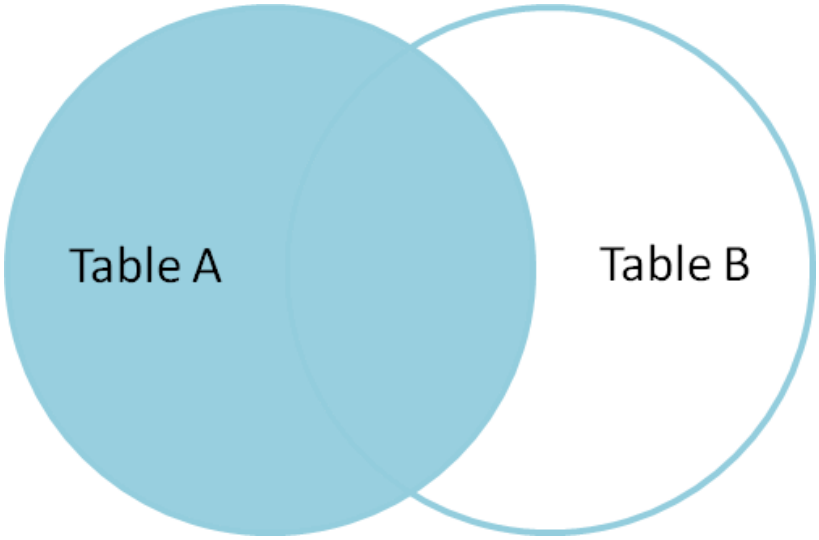
Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
```

```
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

```
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

***Note**: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.*



Example Queries(LEFT JOIN):

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

Output:

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

C. RIGHT JOIN

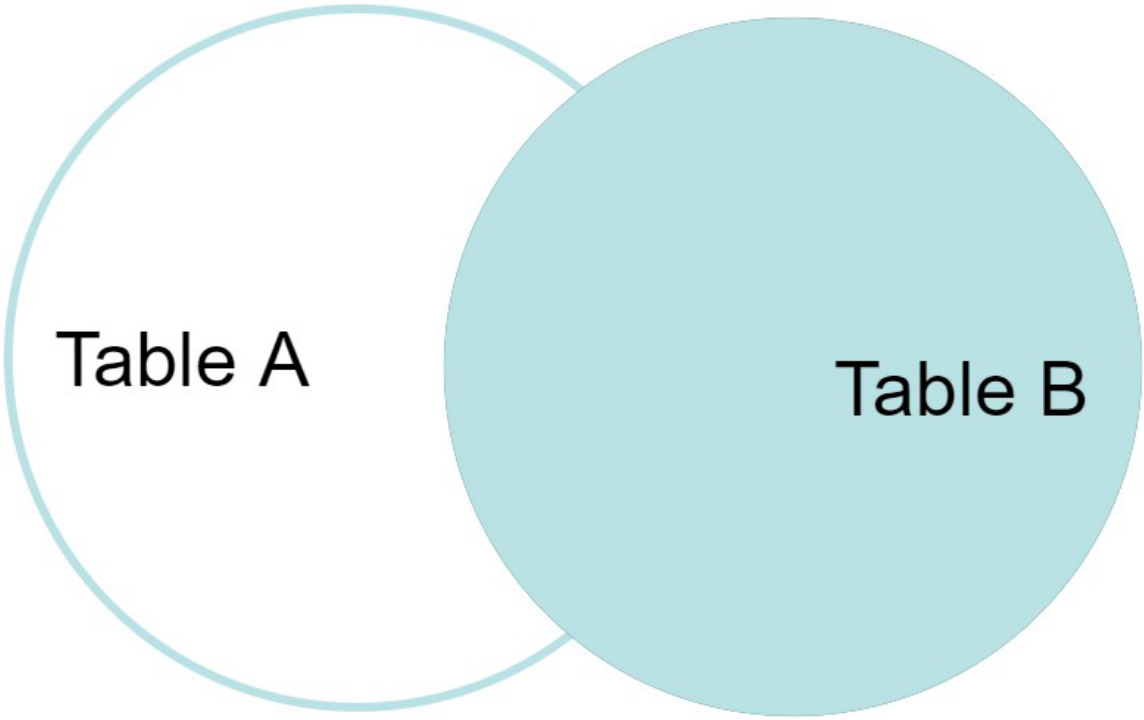
RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

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


table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

***Note**: We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are the same.*



Example Queries(RIGHT JOIN):

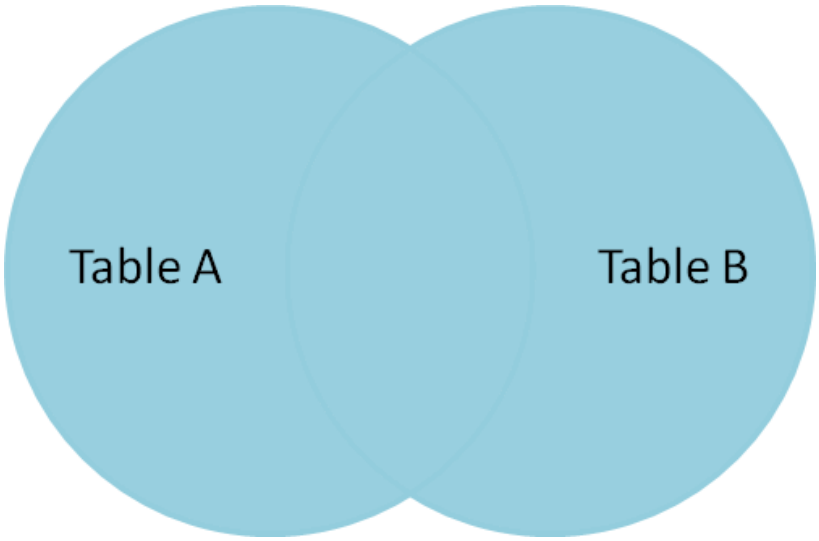
```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

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

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.



Syntax:

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


table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

Example Queries(FULL JOIN):

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

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2

NAME	COURSE_ID
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4

Left JOIN (Video)
Right JOIN (Video)
Full JOIN (Video)
SQL | JOIN (Cartesian Join, Self Join)

E. Natural join (⋈)

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both table must have at list one common column with same column name and same data type.

The two table are joined using Cross join.

DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Example:

Employee		
Emp_id	Emp_name	Dept_id
1	Ram	10
2	Jon	30
3	Bob	50

Department	
Dept_id	Dept_name
10	IT
30	HR
40	TIS

Query: Find all Employees and their respective departments.

Solution: (Employee) ⋈ (Department)

Emp_id	Emp_name	Dept_id	Dept_id	Dept_name
1	Ram	10	10	IT
2	Jon	30	30	HR
Employee data			Department data	

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 24 Apr, 2023

324

Similar Reads

SQL Inner Join

akshatsachan

Overview :

Structured Query Language or SQL is a standard Database language that is used to create, maintain and retrieve the data from relational databases like MySQL, Oracle, etc. A join is a combination of a Cartesian product followed by a selection process. A join operation pairs two tuples from different relations if and only if a given join condition is satisfied. An inner join is the one in which only those tuples are included that satisfy some conditions. In this article, we will be using MySQL to demonstrate the working of SQL Inner Join.

Steps to implement the SQL Inner Join :

Here, we will discuss the implementation of SQL Inner Join as follows.

Step-1: Creating Database :

Here, we will create the database by using the following SQL query as follows.

```
CREATE DATABASE geeks;
```

Step-2: Using the Database :

Here, we will use the geeks database.


```
USE geeks;
```

Step-3: Adding Tables :

We will add 2 tables to the database as follows.

1. The first table will be the professor which will contain ID, the name of the professor, and salary.

2. The second table will be taught which will contain the ID of the course, professor's ID, and name of the course.

Adding table professor –

```
CREATE TABLE professor(
    ID int,
    Name varchar(20),
    Salary int
);
```

Adding table teaches –

```
CREATE TABLE teaches(
    course_id int,
    prof_id int,
    course_name varchar(20)
);
```

Step-4: Description of the Tables :

We can get the description of the 2 tables using the following SQL command as follows.

```
DESCRIBE professor
```

Output :

Field	Type	Null	Key	Default	Extra
ID	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Salary	int	YES		NULL	

```
DESCRIBE teaches
```

Output :

Field	Type	Null	Key	Default	Extra
course_id	int	YES		NULL	

Field	Type	Null	Key	Default	Extra
prof_id	int	YES		NULL	
course_name	varchar(20)	YES		NULL	

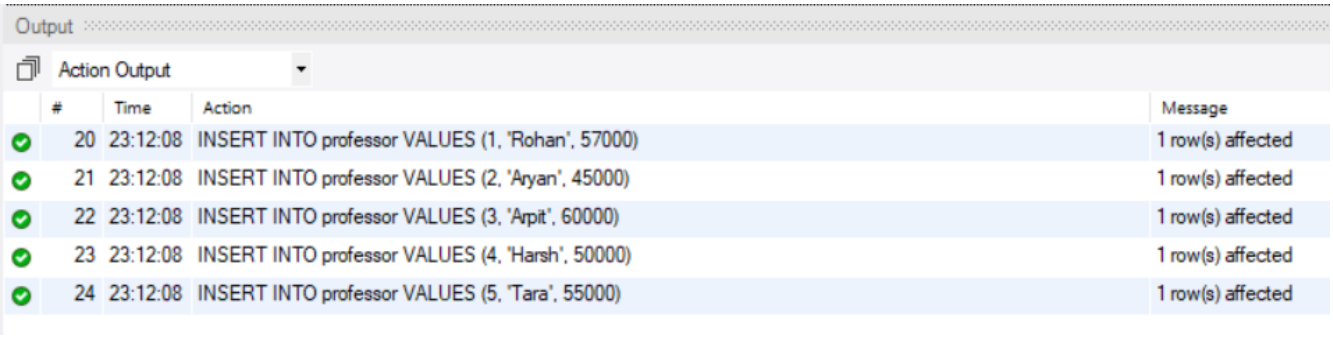
Step-5: Inserting the rows :

Here, we will insert the rows in both tables one by one as follows.

Inserting rows inside professor table –

```
INSERT INTO professor VALUES (1, 'Rohan', 57000);
INSERT INTO professor VALUES (2, 'Aryan', 45000);
INSERT INTO professor VALUES (3, 'Arpit', 60000);
INSERT INTO professor VALUES (4, 'Harsh', 50000);
INSERT INTO professor VALUES (5, 'Tara', 55000);
```

Output :



Inserting rows inside teaches table –

```
INSERT INTO teaches VALUES (1, 1, 'English');
INSERT INTO teaches VALUES (1, 3, 'Physics');
INSERT INTO teaches VALUES (2, 4, 'Chemistry');
INSERT INTO teaches VALUES (2, 5, 'Mathematics');
```

Output :

Step-6: Current State of the Tables :

Verifying the data in both tables as follows.

professor Table –

```
SELECT * FROM professor;
```

Output :

ID	Name	Salary
1	Rohan	57000
2	Aryan	45000
3	Arpit	60000
4	Harsh	50000
5	Tara	55000

teaches Table –

```
SELECT * FROM teaches;
```

Output :

course_id	prof_id	course_name
1	1	English

course_id	prof_id	course_name
1	3	Physics
2	4	Chemistry
2	5	Mathematics

Step-7: INNER JOIN Query :

Syntax :

```
SELECT comma_separated_column_names
FROM table1 INNER JOIN table2 ON condition
```

Example –

```
SELECT teaches.course_id, teaches.prof_id, professor.Name, professor.Salary
FROM professor INNER JOIN teaches ON professor.ID = teaches.prof_id;
```

Output :

Using the Inner Join we are able to combine the information in the two tables based on a condition and the tuples in the Cartesian product of the two tables that do not satisfy the required condition are not included in the resulting table.

course_id	prof_id	Name	Salary
1	1	Rohan	57000
1	3	Arpit	60000
2	4	Harsh	50000
2	5	Tara	55000

Last Updated : 10 May, 2021

3

Similar Reads

1. Difference between Inner Join and Outer Join in SQL
2. Difference between Natural join and Inner Join in SQL

SQL Outer Join

neeraj kumar 13

In a relational DBMS, we follow the principles of normalization that allows us to minimize the large tables into small tables. By using a select statement in Joins, we can retrieve the big table back. Outer joins are of following three types.

1. Left outer join
2. Right outer join
3. Full outer join

Creating a database : Run the following command to create a database.

```
Create database testdb;
```

Using the database : Run the following command to use a database.

```
use testdb;
```

Adding table to the database : Run the following command to add tables to a database.


```
CREATE TABLE Students (
   StudentID int,
   LastName varchar(255),
   FirstName varchar(255),
   Address varchar(255),
   City varchar(255)
);
```

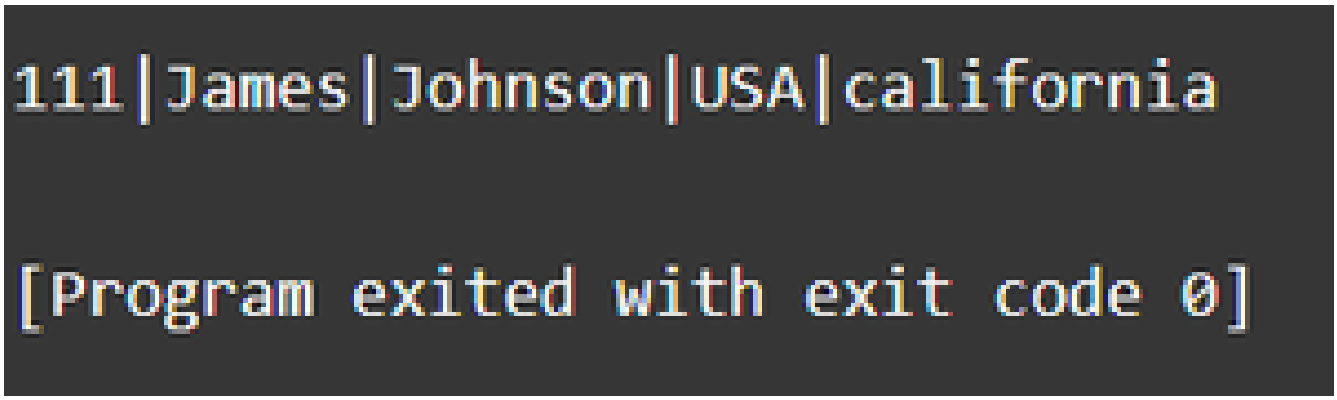
Inserting rows into database :

```
INSERT INTO students (
StudentID,
LastName,
FirstName,
Address,
City
)
VALUES
(
111,
'James',
 'Johnson',
 'USA',
 california
);
```

Output of database :

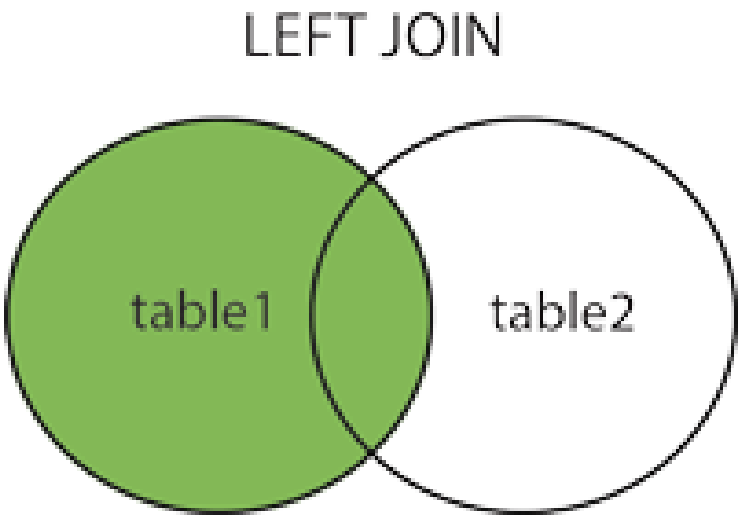
Type the following command to get output.

```
SELECT  * FROM students;
```



Types of outer join :

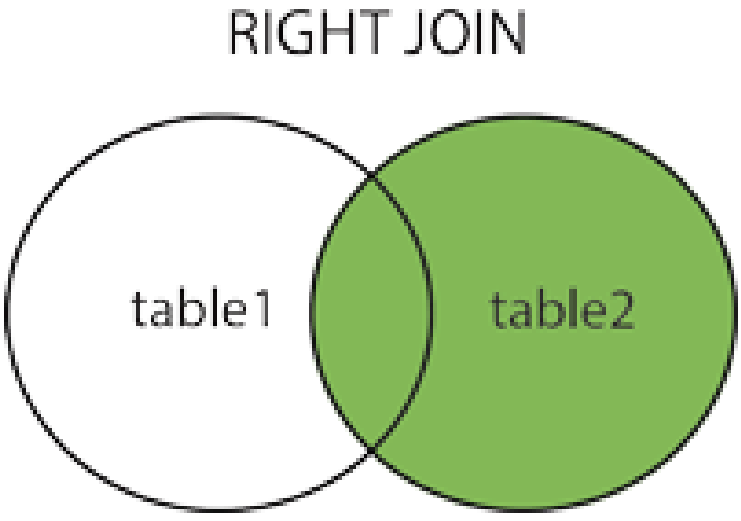
1.**Left Outer Join** : The left join operation returns all record from left table and matching records from the right table. On a matching element not found in right table, NULL is represented in that case.



Syntax :

```
SELECT column_name(s)
FROM table1
LEFT JOIN Table2
ON Table1.Column_Name=table2.column_name;
```

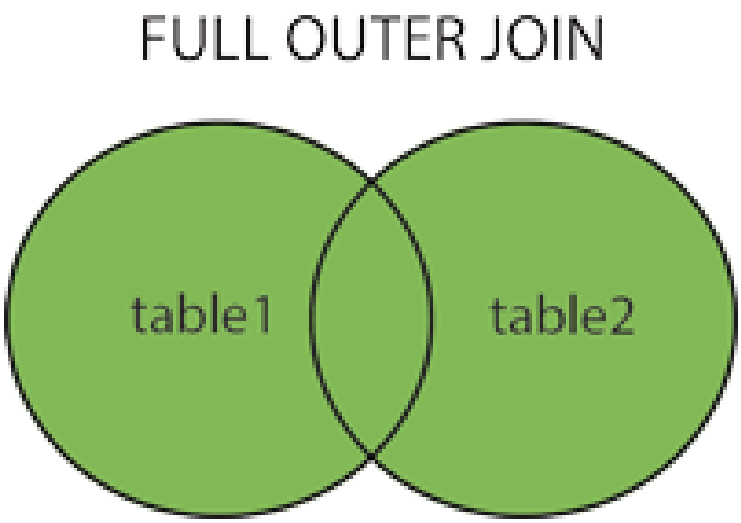
2. Right Outer Join : The right join operation returns all record from right table and matching records from the left table. On a matching element not found in left table, NULL is represented in that case.



Syntax :

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

3. Full Outer Join : The full outer Join keyword returns all records when there is a match in left or right table records.



Syntax:

```
SELECT column_name
FROM table1
FULL OUTER JOIN table2
ON table1.columnName = table2.columnName
WHERE condition;
```

Example :

Creating 1st Sample table students.

```
CREATE TABLE students (
 id INTEGER,
 name TEXT NOT NULL,
 gender TEXT NOT NULL
);
-- insert some values
INSERT INTO students VALUES (1, 'Ryan', 'M');
INSERT INTO students VALUES (2, 'Joanna', 'F');
INSERT INTO students Values (3, 'Moana', 'F');
```

Creating 2nd sample table college.

```
CREATE TABLE college (
 id INTEGER,
 classTeacher TEXT NOT NULL,
 Strength TEXT NOT NULL
);
-- insert some values
```

```
INSERT INTO college VALUES (1, 'Alpha', '50');
INSERT INTO college VALUES (2, 'Romeo', '60');
INSERT INTO college Values (3, 'Charlie', '55');
```

Performing outer join on above two tables.

SELECT College.classTeacher, students.id

FROM College

FULL OUTER JOIN College ON College.id=students.id

ORDER BY College.classTeacher;

The above code will perform a full outer join on tables students and college and will return the output that matches the id of college with id of students. The output will be class Teacher from college table and id from students table. The table will be ordered by class Teacher from college table.

Class Teacher	Id
Alpha	1
Romeo	2
Charlie	3

Last Updated : 13 Apr, 2021

9

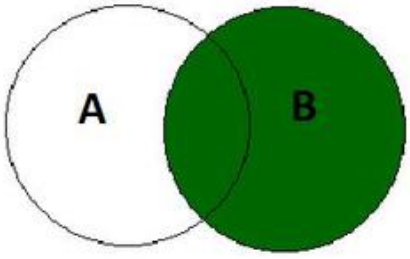
Similar Reads

1. SQL Full Outer Join Using Left and Right Outer Join and Union Clause
2. Difference between Inner Join and Outer Join in SQL
3. Difference between “INNER JOIN” and “OUTER JOIN”
4. Inner Join vs Outer Join
5. SQL | Join (Cartesian Join & Self Join)
6. SQL Full Outer Join Using Union Clause
7. SQL Full Outer Join Using Where Clause

SQL Right Join

disha55handa

The RIGHT JOIN keyword in SQL returns the all **matching records(or rows)** and the **records(or rows) which are present in the right table but not in the left table** .That means that, if a certain row is present in the right table but not in the left, the result will include this row but with a NULL value in each column from the left . If a record from the left table is not in the right, it will not be included in the result.



RIGHT JOIN

The syntax for a RIGHT JOIN is :-

```
SELECT column_name(s)
FROM tableA
RIGHT JOIN tableB ON tableA.column_name = tableB.column_name;
```

SQL RIGHT JOIN EXAMPLE :

In this example we will consider two tables **employee** table containing details of the employees working in the particular department the and **department** table containing the details of the department

employee table :

emp_no	emp_name	dept_no
E1	Varun Singhal	D1
E2	Amrita Aggarwal	D2
E3	Ravi Anand	D3

department table :

dept_no	d_name	location
D1	IT	Delhi
D2	HR	Hyderabad
D3	Finance	Pune
D4	Testing	Noida
D5	Marketing	Mathura

To perform right- join on these two tables we will use the following SQL query:

```
select emp_no , emp_name ,d_name, location
from employee
right join dept on employee.dept_no = department.dept_no;
```

The output that we will get is as follows :

emp_no	emp_name	d_name	location
E1	Varun Singhal	IT	Delhi
E2	Amrita Aggarwal	HR	Hyderabad
E3	Ravi Anand	Finance	Pune
[NULL]	[NULL]	Testing	Noida
[NULL]	[NULL]	Marketing	Mathura

As right join gives the matching rows and the rows that are present in the left table but not in the right table .Here in this example ,we see that the department that contains no employee contains [NULL] values of emp_no and emp_name after performing the right join.

Last Updated : 26 Apr, 2021

2

Similar Reads

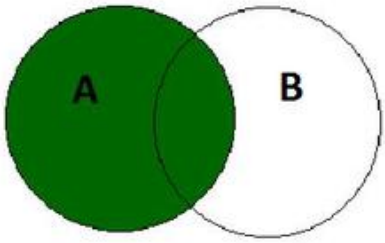
1. SQL | Join (Cartesian Join & Self Join)
2. SQL Full Outer Join Using Left and Right Outer Join and Union Clause
3. Left join and Right join in MS SQL Server
4. Difference between Inner Join and Outer Join in SQL
5. Difference between Natural join and Inner Join in SQL
6. Difference between Natural join and Cross join in SQL
7. Full join and Inner join in MS SQL Server
8. Self Join and Cross Join in MS SQL Server
9. SQL | EQUI Join and NON EQUI JOIN
10. Implicit Join vs Explicit Join in SQL

SQL Left Join

disha55handa

The LEFT JOIN keyword in SQL returns all matching records(or rows) and the records(or rows) that are present in the left table but not in the right table.

That means that, if a certain row is present in the left table but not in the right, the result will include this row but with a NULL value in each column from the right. If a record from the right table is not on the left, it will not be included in the result.



LEFT JOIN

Syntax

SELECT column_name(s)

FROM tableA

LEFT JOIN tableB ON tableA.column_name = tableB.column_name;

Example

In this example, we will consider two tables Emp containing details of the Employee working in the particular department, and department table containing the details of the department

Employee Table

Query:

```
CREATE TABLE Emp (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Country VARCHAR(50),
    Age INT,
    Salary INT,
    department_id INT
);

INSERT INTO Emp (EmpID, Name, Country, Age, Salary, department_id)
VALUES (1, 'Shubham', 'India', 23, 30000, 101),
       (2, 'Aman', 'Australia', 21, 45000, 102),
       (3, 'Naveen', 'Sri Lanka', 24, 40000, 103),
       (4, 'Aditya', 'Austria', 21, 35000, 104),
       (5, 'Nishant', 'Spain', 22, 25000, 101);
```

Output:

EmpID	Name	Country	Age	Salary	department_id
1	Shubham	India	23	30000	101
2	Aman	Australia	21	45000	102
3	Naveen	Sri Lanka	24	40000	103
4	Aditya	Austria	21	35000	104
5	Nishant	Spain	22	25000	101

Department Table

Query:

```
CREATE TABLE department (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50),
    department_head VARCHAR(50),
    location VARCHAR(50)
```

```
);

INSERT INTO department (department_id, department_name, department_head,
location)
VALUES (101, 'Sales', 'Sarah', 'New York'),
       (102, 'Marketing', 'Jay', 'London'),
       (103, 'Finance', 'Lavish', 'San Francisco'),
       (104, 'Engineering', 'Kabir', 'Bangalore');
Select * from department;
```

Output:

department_id	department_name	department_head	location
101	Sales	Sarah	New York
102	Marketing	Jay	London
103	Finance	Lavish	San Francisco
104	Engineering	Kabir	Bangalore

To perform left-join on these two tables we will use the following SQL query :

```
SELECT Emp.EmpID, Emp.Name, department.
department_name, department.department_head,
department.location
FROM Emp
LEFT JOIN department ON Emp.department_id
 = department.department_id;
```

Output:

EmpID	Name	department_name	department_head	location
1	Shubham	Sales	Sarah	New York
2	Aman	Marketing	Jay	London
3	Naveen	Finance	Lavish	San Francisco
4	Aditya	Engineering	Kabir	Bangalore
5	Nishant	Sales	Sarah	New York

As left join gives the matching rows and the rows that are present in the left table but not in the right table. Here in this example, we see that the employees that do not work in a particular department, i.e, having dept no values as [NULL], contain [NULL] values of dept name and location after the left join.

SQL Join as Aliases

Now in this query, we will use the aliases “e” for the Emp table and “d” for the department table. Then the SELECT statement then references these aliases for each of the columns being returned, making our query easier to read and type out. Aliases are especially useful when working with tables that have long or complicated names, as they can help simplify the code and make it easier to understand.

Query:

```
SELECT e.EmpID, e.Name, d.department_name,
d.department_head, d.location
FROM Emp e
LEFT JOIN department d ON
e.department_id = d.department_id;
```

Output:

EmpID	Name	department_name	department_head	location
1	Shubham	Sales	Sarah	New York
2	Aman	Marketing	Jay	London
3	Naveen	Finance	Lavish	San Francisco
4	Aditya	Engineering	Kabir	Bangalore
5	Nishant	Sales	Sarah	New York

SQL Join with WHERE Clause

Now in this query, we will add a WHERE clause that specifies to only return results where the “location” column in the department table equals ‘Bangalore’. This will filter the results to only show employees who belong to a department located in Bangalore, and departments that have no employees will not be returned in the results.

Query:

```
SELECT e.EmpID, e.Name, d.department_name,
d.department_head, d.location
FROM Emp e
LEFT JOIN department d ON e.department_id
 = d.department_id
WHERE d.location = 'Bangalore';
```

Output:

EmpID	Name	department_name	department_head	location
4	Aditya	Engineering	Kabir	Bangalore

Last Updated : 14 Apr, 2023

3

Similar Reads

1. SQL | Join (Cartesian Join & Self Join)
2. SQL Full Outer Join Using Left and Right Outer Join and Union Clause
3. Left join and Right join in MS SQL Server
4. Difference between Inner Join and Outer Join in SQL
5. Difference between Natural join and Inner Join in SQL
6. Difference between Natural join and Cross join in SQL
7. Full join and Inner join in MS SQL Server
8. Self Join and Cross Join in MS SQL Server
9. SQL | EQUI Join and NON EQUI JOIN
10. Implicit Join vs Explicit Join in SQL

Previous Next

Article Contributed By :

disha55handa
disha55handa

Vote for difficulty

Current difficulty : Hard

Easy | Normal | Medium | Hard | Expert

SQL | Join (Cartesian Join & Self Join)

SQL| JOIN(Inner, Left, Right and Full Joins)

In this article, we will discuss about the remaining two JOINS:

- **CARTESIAN JOIN**
- **SELF JOIN**

Consider the two tables below:

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18

StudentCourse

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4

1. **CARTESIAN JOIN**: The CARTESIAN JOIN is also known as CROSS JOIN. In a CARTESIAN JOIN there is a join for each row of one table to every row of another table. This usually happens when the matching column or WHERE condition is not specified.
   - In the absence of a WHERE condition the CARTESIAN JOIN will behave like a CARTESIAN PRODUCT . i.e., the number of rows in the result-set is the product of the number of rows of the two tables.
   - In the presence of WHERE condition this JOIN will function like a INNER JOIN.
   - Generally speaking, Cross join is similar to an inner join where the join-condition will always evaluate to True

   Syntax:

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

   table1: First table.
   table2: Second table

Example Queries(CARTESIAN JOIN):

- In the below query we will select NAME and Age from Student table and COURSE_ID from StudentCourse table. In the output you can see that each row of the table Student is joined with every row of the table StudentCourse. The total rows in the result-set = 4 * 4 = 16.

```
SELECT Student.NAME, Student.AGE, StudentCourse.COURSE_ID
FROM Student
CROSS JOIN StudentCourse;
```

  Output:

NAME	AGE	COURSE_ID
Ram	18	1
Ram	18	2
Ram	18	2
Ram	18	3
RAMESH	18	1
RAMESH	18	2
RAMESH	18	2
RAMESH	18	3
SUJIT	20	1
SUJIT	20	2
SUJIT	20	2
SUJIT	20	3
SURESH	18	1
SURESH	18	2
SURESH	18	2
SURESH	18	3

2. **SELF JOIN**: As the name signifies, in SELF JOIN a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.**Syntax:**

```
SELECT a.coulmn1 , b.column2
FROM table_name a, table_name b
WHERE some_condition;
```

table_name: Name of the table.
some_condition: Condition for selecting the rows.

Example Queries(SELF JOIN):

```
SELECT a.ROLL_NO , b.NAME
FROM Student a, Student b
WHERE a.ROLL_NO < b.ROLL_NO;
```

Output:

ROLL_NO	NAME
1	RAMESH
1	SUJIT
2	SUJIT
1	SURESH
2	SURESH
3	SURESH

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 09 Nov, 2020

69

Similar Reads

1. Self Join and Cross Join in MS SQL Server
2. SQL Query to Avoid Cartesian Product
3. SQL Full Outer Join Using Left and Right Outer Join and Union Clause
4. Difference between Inner Join and Outer Join in SQL
5. Difference between Natural join and Inner Join in SQL
6. Difference between Natural join and Cross join in SQL
7. Full join and Inner join in MS SQL Server
8. Left join and Right join in MS SQL Server
9. SQL | EQUI Join and NON EQUI JOIN
10. Implicit Join vs Explicit Join in SQL

Previous Next

Article Contributed By :

Combining aggregate and non-aggregate values in SQL using Joins and Over clause

ishaan bhatnagar

Prerequisite – Aggregate functions in SQL, Joins in SQL

Aggregate functions perform a calculation on a set of values and return a single value. Now, consider an employee table EMP and a department table DEPT with following structure:

Table – EMPLOYEE TABLE

Name	Null	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7, 2)
COMM		NUMBER(7, 2)
DEPTNO		NUMBER(2)

Table – DEPARTMENT TABLE

Name	Null	Type
DEPTNO		NUMBER(2)

Name	Null	Type
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

And the following results are needed:

1. DISPLAY NAME, SAL, JOB OF EMP ALONG WITH MAX, MIN, AVG, TOTAL SAL OF THE EMPS DOING THE SAME JOB.
2. DISPLAY DEPTNAME WITH NUMBER OF EMP WORKING IN IT.

The aggregated values can't be directly used with non-aggregated values to obtain a result. Thus one can use the following concepts:

1. Using Joins –

1. Create a sub-table containing the result of aggregated values.
2. Using Join, use the results from the sub-table to display them with non-aggregated values.

Solutions for problem 1 using JOIN:

```
SELECT ENAME, SAL, EMP.JOB,
            SUBTABLE.MAXSAL, SUBTABLE.MINSAL,
            SUBTABLE.AVGSAL, SUBTABLE.SUMSAL
FROM EMP
INNER JOIN
        (SELECT JOB, MAX(SAL) MAXSAL, MIN(SAL)
                MINSAL, AVG(SAL) AVGSAL, SUM(SAL) SUMSAL
         FROM EMP
          GROUP BY JOB) SUBTABLE
                  ON EMP.JOB = SUBTABLE.JOB;
```

Output for sample data:

Ename	Sal	Job	MaxSal	MinSal	AvgSal	SumSal
SCOTT	3300	ANALYST	3300	1925	2841.67	8525
HENRY	1925	ANALYST	3300	1925	2841.67	8525
FORD	3300	ANALYST	3300	1925	2841.67	8525
SMITH	3300	CLERK	3300	1045	1746.25	6985
MILLER	1430	CLERK	3300	1045	1746.25	6985

2. Using 'Over' clause –

1. OVER CLAUSE ALONG WITH PARTITION BY IS USED TO BRAKE UP DATA INTO PARTITIONS.
2. THE SPECIFIED FUNCTION OPERATES FOR EACH PARTITION.

Solutions for problem 2 using OVER Clause:

```
SELECT DISTINCT(DNAME),
COUNT(ENAME) OVER (PARTITION BY EMP.DEPTNO) EMP
FROM EMP
RIGHT OUTER JOIN DEPT
ON EMP.DEPTNO=DEPT.DEPTNO
ORDER BY EMP DESC;
```

Dname	Emp
SALES	6
RESEARCH	5
ACCOUNTING	3
OPERATIONS	0
OTHERS	0