Joins

Joins are used to retrieve data from two or more tables.

Types of Join

- Cross Join
- Inner Join
- Equi Join
- Non Equi Join
- Outer Join
 - o Left Outer Join
 - o Right Outer Join
 - o Full Outer Join
- Natural Join
- Self Join

CROSS JOIN

- It is unconditional join between any two tables.
- Each row of first table is joined with each row of the second table. So the no. of rows after joining will be **no. of rows of first table * no. of rows in 2**nd table.
- Syntax:

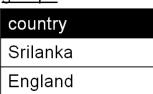
SELECT col1, col2 FROM table1 CROSS JOIN table2;

Example#1

group1

country
India
Australia
South Africa

group2



INSERT INTO group1 VALUES('India'), ('Australia'), ('South Africa');

INSERT INTO group2 VALUES('Srilanka'), ('England');

SELECT group1.country AS Team1,group2.country as Team2 FROM group1 CROSS JOIN group2;

Example#2

product

pid	pname	p_cost
1	Simple Pizza	100
2	Chicken Pizza	200

subproduct

sid	sname	subp_cost
1	Cold Drink	40
2	Bread	50

Pname	Sname	Total cost
Simple Pizza	Cold Drink	140
Simple Pizza	Bread	150
Chicken Pizza	Cold Drink	240
Chicken Pizza	Bread	250

select pname, sname, product.s_cost+subp_cost total_price from product cross join subproduct;

INNER JOIN

- equi join retrieves data from multiple table based on equality condition.
- Here joining conditional columns must belongs to same data types.
- Whenever tables having common columns then only we are using equi join.
- Syntax:

select col1,col2 from table1 INNER JOIN table2 ON table1.col1=table2.col2;

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
INSERT INTO departments (department_id, department_name) VALUES
(1, 'IT'),
```

```
(2, 'HR'),
(3, 'Finance');

INSERT INTO employees (employee_id, employee_name, department_id) VALUES
(101, 'Ram', 1),
(102, 'Raj', 2),
(103, 'Tushar', 1),
(104, 'Alok', 3);

SELECT * FROM departments;
SELECT * FROM employees;

SELECT employee_id, employee_name, department_name
FROM employees
INNER JOIN departments ON employees.department_id = departments.department_id;

SELECT employee_id, employee_name, department_name, department_id;

SELECT employee_id, employee_name, department_name, department_id;

NNER JOIN departments ON employees.department_id = department_id;
```

Conclusion

- When we are trying to display common column in join then database servers returns ambiguity error.
- To overcome this problem we have to specify table name along with common column name using dot operator.
- Syntax:

tablename.commoncolumnname

SELECT employee_id, employee_name, department_name, employees.department_id FROM employees

INNER JOIN departments ON employees.department id = departments.department id;

Example#4

 ${\tt SELECT\ employees.employee_id,\ employees.employee_name,\ departments. department_name,\ employees. department_id}$

FROM employees

INNER JOIN departments ON employees.department id = departments.department id;

Conclusion

• We should specify column name along with table name by using dot operator with in select list to avoid future ambiguity.

Using Table Alias Name in Joins

Example

SELECT emp.employee_id, emp.employee_name, emp.department_id, dept.department_name

FROM employees emp

INNER JOIN departments dept ON emp.department id = dept.department id;

Conclusion

- It will improve readability of the query.
- It will make the query concise.

Example

SELECT employees.employee_id, employees.employee_name, employees.department_id, dept.department_name
FROM employees emp
INNER JOIN departments dept ON emp.department_id = dept.department_id;

Conclusion

• We will not be able to use original name after assigning original name.

Inner Join with more than 2 tables

• If we are joining n tables, we are using n-1 join condition.

<u>student</u>

stdid	Name
101	Alok
102	Sunil
103	Pritam

course details

course_ name	fee	duration
Java	3500	2 month
PHP	4000	2 month
Oracle	4000	2 month

<u>course</u>

course_ name	Stdid
Java	101
PHP	101
Java	102
Oracle	102
Oracle	103

Course_name stdid Name fee duration 3500 102 Sunil 2 month ja∨a 101 Alok 3500 2 month ja∨a 101 Alok PHP 2 month 4000 103 Pritam 4000 2 month oracle 102 Sunil oracle 4000 2 month

```
INSERT INTO student values (101,'Alok'), (102,'Sunil'), (103,'Pritam');
INSERT INTO course_details values
('Java',3500,'2 months'),
('PHP',4000,'2 months'),
('Oracle',4000,'2 months');
INSERT INTO course values
('Java',101),
('PHP',101),
('Java',102),
('Oracle',103),
('Oracle',103);
```

Join Query

```
SELECT s.stdid, s.name, c.course_name, cd.fee, cd.duration
FROM student s INNER JOIN course c
ON s.stdid = c.stdid
INNER JOIN course_details cd
ON c.course_name = cd.course_name;
```

Equi Join

<u>Syntax</u>

select col1,col2 from table1,table2 where table1.col1=table2.col2;

Example

SELECT emp.employee_id, emp.employee_name, dept.department_id, dept.department_name FROM employees emp, departments dept where emp.department_id = dept.department_id;

Non Equi Join

• Non equi join is used to retrieve data from multiple tables based on other than equality operator (<, <=, >, >=, between and).

Example#1

employee_id	employee_name	department_id	d	lepartment_id	department_name
101	Ram	1	1		Π
102	Raj	2	2		HR
103	Tushar	1	3		Finance
104	Alok	3			

Approach-1

SELECT emp.employee_id, emp.employee_name, dept.department_id, dept.department_name FROM employees emp INNER JOIN departments dept ON emp.department_id > dept.department_id;

Approach2

SELECT emp.employee_id, emp.employee_name, dept.department_id, dept.department_name FROM employees emp, departments dept where emp.department id > dept.department id;

Example#2

<u>student</u>

stdid	name	score
101	Badal	555
102	Ajit	459
103	Archana	245
104	Barsha	385
106	Jayanta	95

<u>grade</u>

lowmark	highmark	grade	gradevalue		
540	600	A1	Outstanding		
480	539	A2	Excellent		
420	479	B1	Very Good		
360	419	B2	Good		
300	359	С	Above Average		
240	299	D	Average		
198	239	E	Fair		
0	197	F	Un Satisfactory		

stdid	name	score	Lowmark	Highmark	Grade	gradevalue
101	Badal	555	540	600	A1	Outstanding
102	Ajit	459	420	479	B!	Very Good
103	Archana	245	140	299	D	Average
104	Barsha	385	360	419	B2	Good
106	Jayanta	95	0	197	F	Un Satisfactory

```
create table student(
       stdid int,
       name varchar(10),
       score int
);
create table grade(
       lowmark int,
       highmark int,
       grade varchar(5),
       gradevalue varchar(20)
);
insert into student values(101, 'Badal', 555),
(102, 'Ajit', 459),
(103,'Archana',245),
(104, 'Barsha', 385),
(101, 'Jayanta', 95);
insert into grade values(540,600,'A1','Outstanding'),
(480,539,'A2','Excellent'),
(420,479,'B1','Very Good'),
(360,419,'B2','Good'),
(300,359,'C','Above Average'),
(240,299,'D','Average'),
(198,239, 'E', 'fair'),
(000,197,'F','Un Satisfactory');
```

SELECT * FROM student, grade WHERE score BETWEEN lowmark AND highmark;

SELECT * FROM student IINER JOIN grade ON score BETWEEN lowmark AND highmark;

<u>Example</u>

<u>employee</u>

empid	ename
101	Rahul
102	Ram
103	Tusar
104	Raj
105	Alok

project

Pname	Pduration	Empid
HDFC	1 year	103
HRFC	1 year	104
SBI	6 month	
ICICI	1 year	

ename	pname	pduration
Tusar	HDFC	1 year
Raj	HDFC	1 year

```
CREATE TABLE employee(
       empid int,
  ename varchar(20)
);
CREATE TABLE project(
       pname varchar(20),
  pduration varchar(20),
       empid int
);
INSERT INTO employee VALUES (101, 'Rahul'),
(102, 'Ram'),
(103, 'Tushar'),
(104, 'Raj'),
(105,'Alok');
INSERT INTO project VALUES ('HDFC', '1 year', 103),
('HDFC', '1 year', 104),
('SBI', '6 month', null),
('ICICI', '1 year', null);
SELECT emp.ename, proj.pname, proj.pduration
FROM employee emp INNER JOIN project proj
ON emp.empid = proj.empid;
```

Requirements

empid	ename	pname	pduration
101	Rahul	NULL	NULL
102	Ram	NULL	NULL
103	Tushar	HDFC	1 year
104	Raj	HDFC	1 year
105	Alok	NULL	NULL

Outer Join

- Outer join is used to fetch both matching and non matching row from multiple table.
- Outer join can be divided into 3 types:
 - Left outer join
 - o Right outer
 - o Full outer join

Left Outer Join

• Left Outer join is used to fetch matching row from both table and non matching row from left table or first table.

Right Outer Join

• Right Outer join is used to fetch matching row from both tables and non matching row from right table or second table.

Full Outer Join

• Full Outer join is used to fetch matching row and non matching row from both tables.

Example

SELECT emp.empid,emp.ename, proj.pname, proj.pduration

FROM employee emp LEFT JOIN project proj

ON emp.empid = proj.empid;

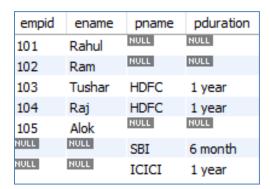
Requirement

empid	ename	pname	pduration
103	Tushar	HDFC	1 year
104	Raj	HDFC	1 year
NULL	NULL	SBI	6 month
NULL	NULL	ICICI	1 year

SELECT emp.empid,emp.ename, proj.pname, proj.pduration

FROM employee emp RIGHT JOIN project proj

ON emp.empid = proj.empid;



```
SELECT emp.empid,emp.ename, proj.pname, proj.pduration
FROM employee emp LEFT JOIN project proj
ON emp.empid = proj.empid
UNION
SELECT emp.empid,emp.ename, proj.pname, proj.pduration
FROM employee emp RIGHT JOIN project proj
ON emp.empid = proj.empid;
```

Question#1

Write a query to display employee_id, name, department_name, location who are working in the location 'bbsr' from employee and department tables using **inner join**.

department_id	department_name	employee_id	employee_name	location	department_id
1	HR	101	Α	bbsr	1
1		102	В	pune	2
2	Engineering	103	С	pune	1
Marketing		104	D	bbsr	3

employee_id	employee_name	location	department_name
101	Α	bbsr	HR
104	D	bbsr	Marketing

```
CREATE TABLE departments (
    department_id INT ,
    department_name VARCHAR(50)
);
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    location VARCHAR(50),
    department_id INT
);
INSERT INTO departments VALUES
(1, 'HR'),
(2, 'Engineering'),
(3, 'Marketing');
```

```
INSERT INTO employees VALUES (101, 'A', 'bbsr', 1), (102, 'B', 'pune',2), (103, 'C', 'pune',1), (104, 'D', 'bbsr',3);
```

SELECT emp.employee_id, emp.employee_name, emp.location, dept.department_name FROM employees emp INNER JOIN departments dept

ON emp.department_id = dept.department_id

WHERE emp.location = 'bbsr';

SELECT emp.employee_id, emp.employee_name, emp.location, dept.department_name FROM employees emp INNER JOIN departments dept

ON emp.department_id = dept.department_id

AND emp.location = 'bbsr';

Note

 If we want to filter data after joining condition then we can use either AND operator or WHERE clause.

Question#2

Write a query to display employee_id, name, department_name, location who are working in the location 'bbsr' from employee and department tables using **equi join**.

SELECT emp.employee_id, emp.employee_name, emp.location, dept.department_name FROM employees emp, departments dept WHERE emp.department_id = dept.department_id AND emp.location = 'bbsr';

<u>Note</u>

If we want to filter data after joining condition then we have to use AND operator.

Question#3

Write a query to display department name, sum of salary of each department from employee department table by using **inner join**.

department_id	department_name	employee_id	employee_name	location	salary	department_id
1	HR	101	Α	bbsr	30000.00	1
2	Engineering	102	В	pune	40000.00	2
3	Marketing	103	С	pune	50000.00	1
		104	D	bbsr	50000.00	3

department_name	sum(salary)
HR	80000.00
Engineering	40000.00
Marketing	50000.00

SELECT department_name, sum(salary)

FROM departments dept INNER JOIN employees emp

ON dept.department_id = emp.department_id

GROUP BY department_name;

Question#4

Write a query to display department name, sum of salary of each department from employee department table by using **equi join**.

SELECT department_name, sum(salary)
FROM departments dept, employees emp
WHERE dept.department_id = emp.department_id
GROUP BY department_name;

Question#5

Write a query to display department name, sum of salary of those department having more than 40,000 from employee department table by using **equi join**.

department_name	salary
HR	80000.00
Marketing	50000.00

SELECT department_name, sum(salary)

FROM departments dept, employees emp
WHERE dept.department_id = emp.department_id
GROUP BY department_name
HAVING sum(salary)> 40000;

Natural Join

- Join condition is not required.
- Natural join display the common column of both the tables only once. While inner join display the common column two times in the output.

Example#1

2	<u>student</u>	
	stdid	Name
	101	Alok
	102	Sunil
	103	Pritam

course			
course_ name	Stdid		
Java	101		
PHP	101		
Java	102		
Oracle	102		
Oracle	103		

stdid	name	course_name
101	Alok	Java
101	Alok	PHP
102	Sunil	Java
103	Pritam	Oracle
103	Pritam	Oracle

```
CREATE TABLE student(
    stdid int,
    name varchar(20)
);
CREATE TABLE course(
    course_name varchar(20),
    stdid int
```

```
);
INSERT INTO student values (101, 'Alok'), (102, 'Sunil'), (103, 'Pritam');
INSERT INTO course values
('Java',101),
('PHP',101),
('Java',102),
('Oracle',103),
('Oracle',103);
Approach1:
SELECT * FROM student INNER JOIN course ON student.stdid = course.stdid;
Approach2:
SELECT * FROM student NATURAL JOIN course;
Example#2
CREATE TABLE student(
       stdid int,
       name varchar(20)
);
CREATE TABLE course(
       course_name varchar(20),
       std_id int
);
SELECT * FROM student NATURAL JOIN course;
SELECT * FROM student CROSS JOIN course;
```

<u>Note</u>

• If we don't have common column in both the table then natural join will behave like cross join.

SELECT * FROM student INNER JOIN course ON student.stdid = course.std_id;

Self Join

• Joining a table to itself is called self join.

empid	ename	managerid	salary	location
1	Raj	2	40000.00	bbsr
2	Rahul	5	30000.00	bbsr
3	Tushar	2	60000.00	pune
4	Alok	3	40000.00	bbsr
5	Ram	NULL	80000.00	bbsr

```
create table employee(
empid int,
ename varchar(20),
managerid int,
salary decimal(8,2),
location varchar(20)
);
insert into employee values
(1,'Raj',2, 40000, 'bbsr'),
(2,'Rahul',5, 30000, 'bbsr'),
(3,'Tushar',2, 60000, 'pune'),
(4,'Alok',3, 40000, 'bbsr'),
(5,'Ram',null, 80000, 'bbsr');
```

SELECT * FROM employee;

Example#1

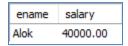
Write a query to display employee names and their manager names.



SELECT e1.ename Employee, e2.ename Manager FROM employee e1, employee e2 WHERE e1.managerid = e2.empid;

Example#2

Write a query to display the employees who are getting same salary as 'Raj' salary from employee table by using self join.



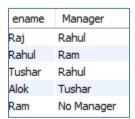
SELECT e2.ename, e2.salary

FROM employee e1, employee e2

WHERE e1.ename='Raj' AND e1.salary = e2.salary AND e2.ename!='Raj';

Example#3

Write a query to display employee names and their manager names, if manager is not available for an employee then display 'No Manager'.



SELECT e1.ename, coalesce(e2.ename, 'No Manager') Manager FROM employee e1 LEFT JOIN employee e2 ON e1.managerid = e2.empid;

Example#4

Write a query to display the employee salary and their manager salary.

ename	salary	ename	Manager
Tushar	60000.00	Rahul	30000.00
Raj	40000.00	Rahul	30000.00
Alok	40000.00	Tushar	60000.00
Rahul	30000.00	Ram	80000.00

SELECT e1.ename, e1.salary, e2.ename, e2.salary FROM employee e1, employee e2 WHERE e1.managerid = e2.empid;

Example#5

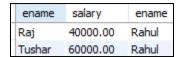
Write a query to display the employees who are getting more salary than their manager salary.

ename	salary	ename	Manager
Tushar	60000.00	Rahul	30000.00
Raj	40000.00	Rahul	30000.00

SELECT e1.ename, e1.salary, e2.ename, e2.salary Manager FROM employee e1, employee e2
WHERE e1.managerid = e2.empid
AND e1.salary > e2.salary;

Example#6

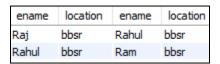
Display employee name who is working under manager of 'Rahul'.



SELECT e1.ename, e1.salary, e2.ename FROM employee e1, employee e2 WHERE e2.ename='Rahul' AND e1.managerid = e2.empid;

Example#7

Write a query to display those employee and manager who belongs same location.



SELECT e1.ename, e1.location, e2.ename, e2.location FROM employee e1, employee e2 WHERE e1.managerid = e2.empid AND e1.location = e2.location;