Joins

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

Types of Join

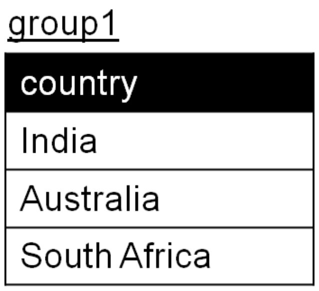
- Cross Join
- Inner Join
- Equi Join
- Non Equi Join
- Outer Join
    - Left Outer Join
    - Right Outer Join
    - Full Outer Join
- Natural Join
- Self Join

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 2nd** table.
- Syntax:

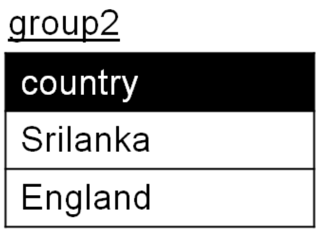
  SELECT col1, col2 FROM table1 CROSS JOIN table2;

Example#1



group1

country
India
Australia
South Africa



group2

country
Srilanka
England

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

- Inner 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
FROM employees
INNER JOIN departments ON employees.department_id = departments.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

```
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

<u>Example</u>

```
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;
```

<u>Conclusion</u>

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

<u>Example</u>

```
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;
```

<u>Conclusion</u>

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

Inner Join with more than 2 tables

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

student

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

course

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

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

```
CREATE TABLE student(
        stdid int,
        name varchar(20)
);
CREATE TABLE course_details(
  course_name varchar(20),
  fee int,
  duration varchar(30)
);
CREATE TABLE course(
        course_name varchar(20),
        stdid int
);
```

```
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

Syntax

```
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;
```

Example

```
SELECT s.stdid, s.name, c.course_name, cd.fee, cd.duration
FROM student s, course c, course_details cd
```

```
WHERE s.stdid = c.stdid AND c.course_name = cd.course_name;
```