Join operations :-

Page No.:
Date: / /

In real life, we store our data in multiple logical tables that are linked together by a common key value in relational databases like SQL server, Oracle, MySQL and others. As a result, we constantly need to get data from two or more tables into the desired output based on some conditions. We can quickly achieve this type of data in SQL server using the SQL JOIN clause.

Join :- The join clause allows us to retriere data from two or more related tables into a meaningful result set.

- A join clause is used to combine rows from two or more tables, based on a related column between them.

- We can join the table using a SELECT statement and a join condition.

- It indicates how SQL server can use data from one table to select rows from another table. In general, tables are related to each other using foreign key constraints.

- In a Join query; a condition indicates how two tables are related:

- Choose columns from each table that should be used in the join. A join condition indicates a foreign key from one table and its corresponding key in the other table.

- Specify the logical operator to compare values from the columns like =, < or >.

Syntax of Joining two tables :-

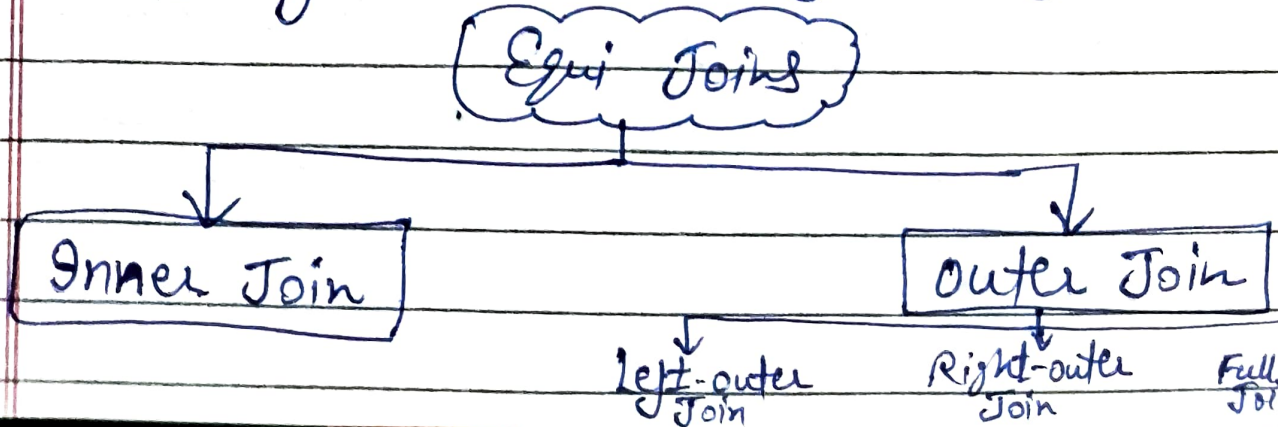
```
SELECT col1, col2, col3, col4 ....
FROM table_name1 , table_name2
WHERE table_name1. col1 = table_name2. col1;
```

Date: / /

- If a SQL join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined.
  - e.g.; if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10 = 200 rows. This query takes a long time to execute.

Types of Joins :- (1) Equi- Joins
(2) Non-Equi Joins

(1) Equi-Joins :- It is a simple SQL join condition which uses the equal sign (=) as the comparison operator. Two types of Equi-joins are SQL Outer join & SQL Inner Join.



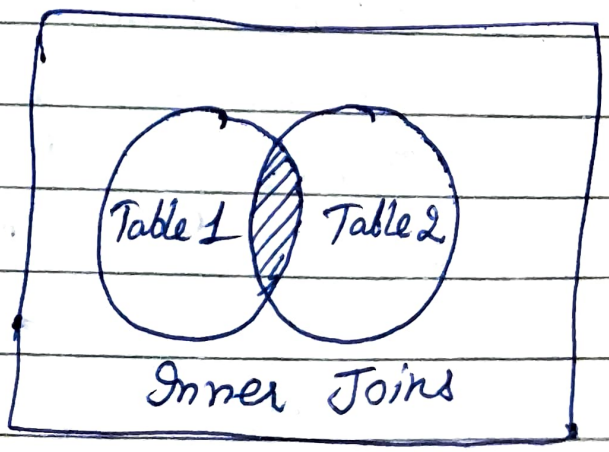
Page No.:
Date: / /

(2) SQL Non-equi Joins :- It is a Sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=.

(1) SQL Equi Joins :- An equi join is further classified into two categories:

(i) SQL Inner Join :- All the rows returned by the Sql query satisfy the Sql join condition specified. A JOIN that displays only rows that have a match in both the joined tables is known as inner Join.

- This is the default type of Join in query and view designer.



Date: / /

- The SQL **INNER JOIN** Statement joins two tables based on a common column and selects rows that have matching values in these columns.

Syntax :-

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

⇒ Here; table1 & table2 – two tables that are to be joined.
- column1 and column2 – columns common to in table1 and table2.

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

Example :- Let's look at the example of INNER JOIN clause and understand its working.

Page No.:
Date: / /

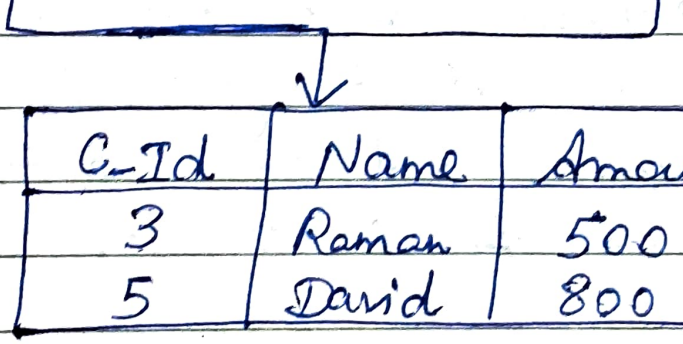
Let's take two tables [customers] and [orders] and perform Inner Join

Customers

C_Id	Name
1	John
2	Ram
3	Raman
4	John
5	David

Orders

O_Id	Amount	C_Id
1	200	10
2	500	3
3	300	6
4	800	5
5	150	8



C_Id	Name	Amount
3	Raman	500
5	David	800

INNER JOIN Query on above tables :-

⇒
```
SELECT customers.cId, customers.name,
       orders.amount FROM customers
INNER JOIN Orders
ON customers.C_Id = orders.C_Id;
```

Here, the SQL command selects the specified rows from both tables if the values of ~~customer~~ [C_Id] (of the [customers] table) and [C_Id] (of [orders] table) are a match.

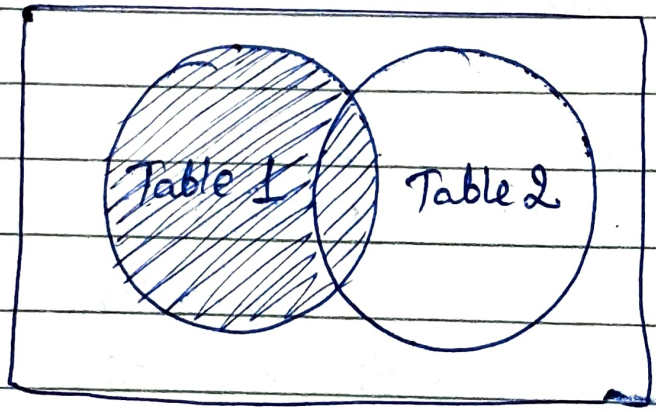
⇒ As you can see, [INNER JOIN] excludes all the rows that are not common b/w two tables.

(ii) SQL Outer Joins :-

- This SQL Join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables.
- The SQL outer Join operator in oracle is (+) and is used on one side of the Join condition only.
- A Join that includes rows even if they do not have related rows in the joined table is an Outer Join. You can create three different outer JOINS to specify the unmatched rows to be included:

Left Outer Join :- • The SQL LEFT JOIN combines two tables based on a common column.

• In Left Outer Join, all rows in the first-named table, i.e. "left" table, which appears leftmost in the JOIN clause, are included. Unmatched rows in the right table do not appear.



Left outer JOIN

Syntax:

```
SELECT columns_from_both_tables
FROM table1
LEFT JOIN table2
ON table1.column1 = table2.column2
```

⇒ Here, • table1 is the left table to be joined.

• table2 is the right table to be joined.

- Left Join Keyword returns all rows from left table, even if there are no matches in the Right table.

Page No.
Date: / /

Example:- Left Join the customers and orders tables.

```
SELECT customers.C_Id, customers.name,
       orders.amount FROM customers
LEFT JOIN orders
ON customers.C_Id = orders.C_Id;
```

Customers

C_Id	Name
1	John
2	Ram
3	Raman
4	John
5	David

orders

O_Id	Amount	C_Id
1	200	10
2	500	3
3	300	6
4	800	5
5	150	8

Left table is customers & Right table is orders

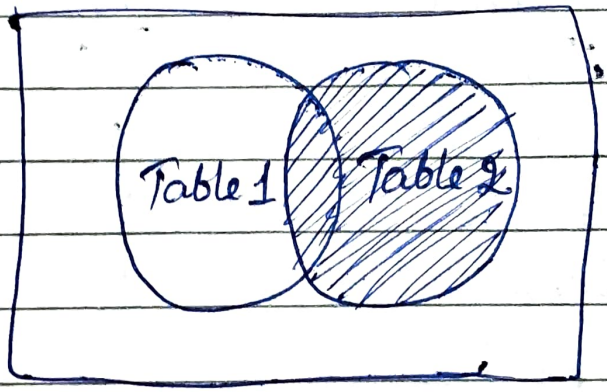
C_Id	Name	Amount
1	John	—
2	Ram	—
3	Raman	500
4	John	—
5	David	800

Here, the results includes rows where C_Id from customers matches C_Id from orders.

Page No.:
Date: / /

Right Outer Join :- • In Right outer Join all rows in the second-named table, i.e "right" table, which appears rightmost in the JOIN clause, are included. Unmatched rows in the left table are not included.

• The Right Join keyword return all rows from the right table, even if there are no matches in left table.



Right outer Join

Syntax :

```
SELECT columns from both tables
FROM table1
RIGHT JOIN table2
ON table1.column1=table2.column2;
```

Here, table1 is left table.
table2 is right table.

Page No.:
Date: / /

Example:- Right Join the customers & orders table. based on C_Id of customers & orders.
- customers is left table, orders is right table.

```
SELECT customers.C_Id, customers.Name,
    orders.amount
FROM customers RIGHT JOIN Orders
ON customers.C_Id = orders.C_Id;
```

Tables

Customers

C_Id	Name
1	John
2	Ram
3	Raman
4	John
5	David

Orders

O_Id	Amount	C_Id
1	200	10
2	500	3
3	300	6
4	800	5
5	150	8

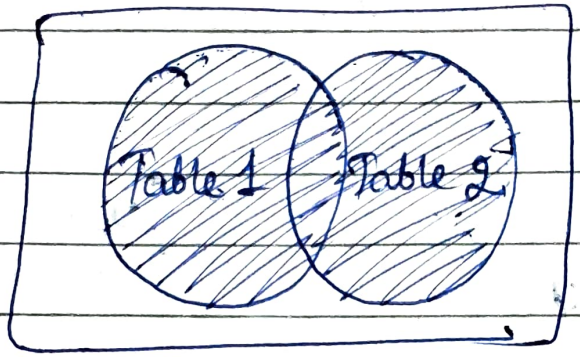
C_Id	Name	Amount
3	Raman	500
5	David	800
–	–	200
–	–	300
–	–	150

Here, result set will contain these rows where there is match b/w both tables, along with all the remaining rows from the orders table.

Page No.:
Date: / /

Full Outer Join :- The Full Outer Join statement joins two tables based on a common column.

- In full outer Join, all rows in all the joined tables are included, whether they are matched or not.



Full outer JOIN

Syntax :

```
SELECT columns
FROM table1
FULL OUTER JOIN table2
ON table1.column1 = table2.column2;
```

Example :- Full outer Join: Customers and orders table.

⇒

```
SELECT customers.cId, customers.name,
    orders.amount
FROM customers
FULL OUTER JOIN orders
ON customers.C-Id = orders.C-Id;
```

Page No.:
Date: / /

The result set will contain all rows of both the tables, regardless of whether there is a match b/w C_Id of customers table and C_Id of orders table.

Table:

Customers

C_Id	Name
1	John
2	Ram
3	Raman
4	John
5	David

Orders

O_Id	Amount	C_Id
1	200	10
2	500	3
3	300	6
4	800	5
5	150	8

C_Id	Name	Amount
1	John	—
2	Ram	—
3	Raman	500
4	John	—
5	David	800
—	—	200
—	—	300
—	—	150

(2) SQL Non Equi Join :- A non-equi join is a SQL Join whose condition is establish using all comparison operators except the equal(=) operator. like <, >, <=, >=.

For eg. if you want to find the name of students who are not studing either economics, the SQL query would be like,

```
SELECT first_name, last_name, subject
FROM student_details
WHERE subject != 'Economics';
```

Student_details

firstname	lastname	subject
Anjali	Bhagwat	Maths
Shekhar	Gowda	Maths
Rahul	Sharma	Science
Amrit	Kaur	Economics

Output

first_name	last_name	subject
Anjali	Bhagwat	Maths
Shekhar	Gowda	Maths
Rahul	Sharma	Science