WHAT ARE SQL JOINS

- In SQL (Structured Query Language), a join is a way to combine data from two or more database tables based on a related column between them.

- Joins are used when we want to query information that is distributed across multiple tables in a database, and the information we need is not contained in a single table. By joining tables together, we can create a virtual table that contains all of the information we need for our query.

BUT WHY HAVE DATA IN MULTIPLES TABLE

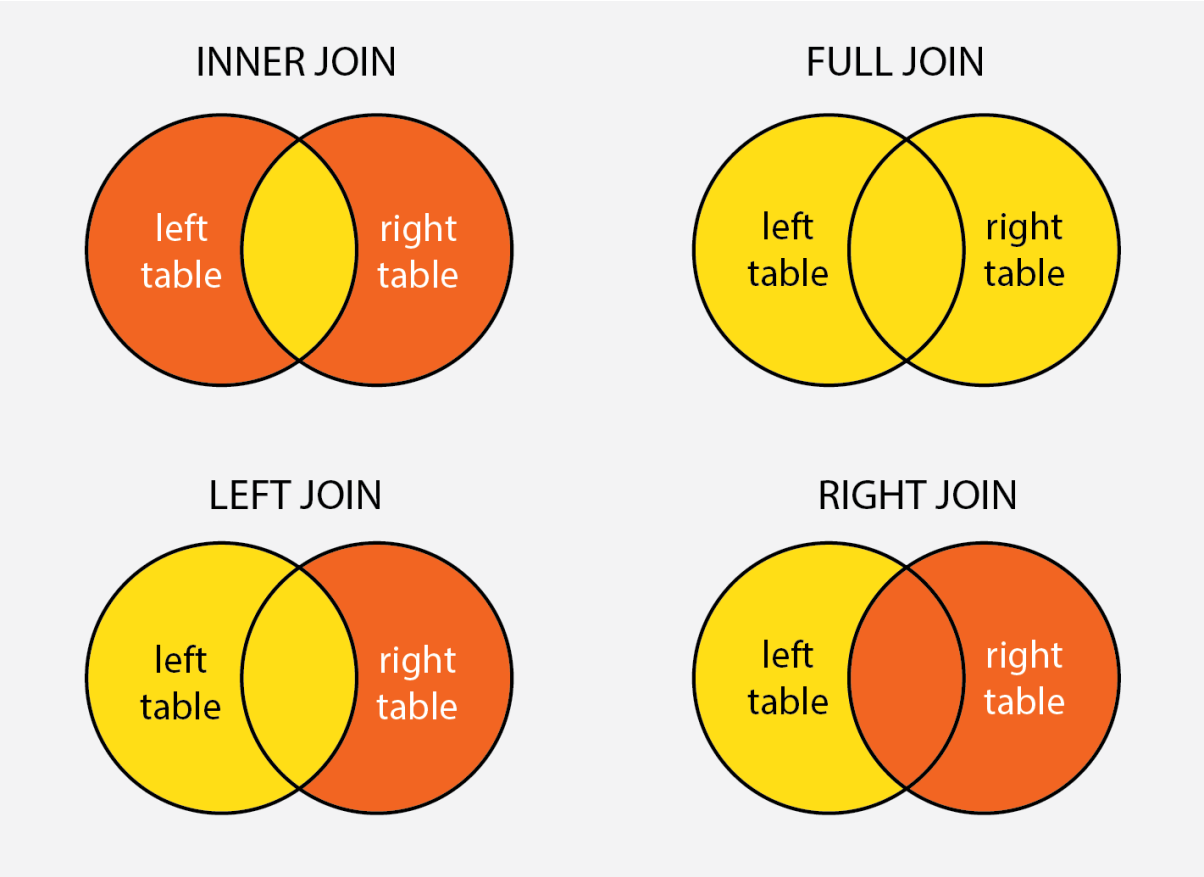
- Reduced Redundancy: By breaking down data into smaller, more focused tables, you avoid repeating the same information across multiple records. This reduces redundancy and helps maintain data consistency.

- Enhanced Performance: Smaller tables are generally faster to search and query, especially when dealing with large datasets. This improves the overall performance of database operations such as reading, updating, and deleting records.

- Easier Maintenance: With data organized into separate tables based on their logical relationships, it becomes easier to manage and maintain the database. You can update, modify, or delete data in one table without affecting unrelated data in other tables.

- Easier Data Analysis: Separating data into distinct tables based on their attributes or relationships makes it easier to perform complex data analysis and reporting. You can join tables together to retrieve specific information or generate meaningful insights.

TYPES OF SQL JOINS



INNER JOINS

In SQL, an inner join is a type of join operation that combines data from two or more tables based on a specified condition. The inner join returns only the rows from both tables that satisfy the specified condition, i.e., the matching rows.

When you perform an inner join on two tables, the result set will only contain rows where there is a match between the joining columns in both tables. If there is no match, then the row will not be included in the result set.

TABLE - EMPLOYEE

Employee id	name	Department id	salary
1	John	1	20000
2	Jane	2	30000
3	Bob	3	355000
4	Lisa	3	34000
5	Mike	4	56000
6	tim	5	67000

TABLE – DEPARTMENT

Department id	Department Name
1	Engineering
2	Sales
3	Finance
6	Operations

SELECT * FROM database.employee t1

INNER JOIN database.department t2

ON t1.department id = t2.department id

If we perform inner join on Department id then resultant table :

Employee id	name	Department id	salary	Department id	Department Name
1	John	1	20000	1	Engineering
2	Jane	2	30000	2	Sales
3	Bob	3	355000	3	Finance
4	lisa	3	34000	3	Finance

Left join

A left join, also known as a left outer join, is a type of SQL join operation that returns all the rows from the left table (also known as the "first" table) and matching rows from the right table (also known as the "second" table). If there are no matching rows in the right table, the result will contain NULL values in the columns that come from the right table.

In other words, a left join combines the rows from both tables based on a common column, but it also includes all the rows from the left table, even if there are no matches in the right table. This is useful when you want to include all the records from the first table, but only some records from the second table.

Right table->

Employee id	name	Department id	salary
1	John	1	20000
2	Jane	2	30000
3	Bob	3	355000
4	Lisa	3	34000
5	Mike	4	56000
6	tim	5	67000

Left table->

Department id	Department Name
1	Engineering
2	Sales
3	Finance
6	Operations

SELECT * FROM database.employee t1

LEFT JOIN database.department t2

ON t1.department id = t2.department id

Employee id	name	Department id	salary	Department id	Department Name
1	John	1	20000	1	Engineering
2	Jane	2	30000	2	Sales
3	Bob	3	355000	3	Finance
4	Lisa	3	34000	3	Finance
5	Mike	4	56000	Null	Null
6	tim	5	67000	Null	Null

Right join

A right join, also known as a right outer join, is a type of join operation in SQL that returns all the rows from the right table and matching rows from the left table. If there are no matches in the left table, the result will still contain all the rows from the right table, with NULL values for the columns from the left table.

Right Table ->

Employee id	name	Department id	salary
1	John	1	20000
2	Jane	2	30000
3	Bob	3	355000
4	Lisa	3	34000
5	Mike	4	56000
6	tim	5	67000

Left table ->

Department id	Department Name
1	Engineering
2	Sales
3	Finance
6	Operations

SELECT * FROM database.employee t1

RIGHT JOIN database.department t2

ON t1.department id = t2.department id

Resultant table ->

Employee id	name	Department id	salary	Department id	Department Name
1	John	1	20000	1	Engineering
2	Jane	2	30000	2	Sales
3	Bob	3	355000	3	Finance
4	lisa	3	34000	3	Finance
null	null	null	Null	6	Operations

Full outer join

A full outer join, sometimes called a full join, is a type of join operation in SQL that returns all matching rows from both the left and right tables, as well as any nonmatching rows from either table. In other words, a full outer join returns all the rows from both tables and matches rows with common values in the specified columns, and fills in NULL values for columns where there is no match.

Right table ->

Employee id	name	Department id	salary
1	John	1	20000
2	Jane	2	30000
3	Bob	3	355000
4	Lisa	3	34000
5	Mike	4	56000
6	tim	5	67000

Left table ->

Department id	Department Name
1	Engineering
2	Sales
3	Finance
6	Operations

Full outer directly can’t perform on mysql

But we can perform full outer join by this way :

```
SELECT * From database.employee t1
LEFT JOIN database.department t2
ON t1.department id = t2.department id
UNION
SELECT * FROM database.employee t1
RIGHT JOIN database.department t2
ON t1.department id = t2.department id
```

Resultant table ->

Employee id	name	Department id	salary	Department id	Department Name
1	John	1	20000	1	Engineering
2	Jane	2	30000	2	Sales
3	Bob	3	355000	3	Finance
4	Lisa	3	34000	3	Finance
5	Mike	4	56000	Null	Null
6	tim	5	67000	null	Null
null	null	null	null	6	Operations

Cross join (cartesian product)

In SQL, a cross join (also known as a Cartesian product) is a type of join that returns the Cartesian product of the two tables being joined. In other words, it returns all possible combinations of rows from the two tables.

Cross joins are not commonly used in practice, but they can be useful in certain scenarios, such as generating test data or exploring all possible combinations of items in a product catalogue. However, it's important to be cautious when using cross joins with large tables, as they can generate a very large result set, which can be resource-intensive and slow to process.

Table 1 ->

column 1	Column 2
A	1
B	2

Table 2 ->

column 3	column 4
X	Y
Z	W

Resultant table->

Column 1	Column 2	Column 3	Column 4
A	1	X	Y
A	1	Z	W
B	2	X	Y
B	2	Z	W

SQL SET OPERATIONS

UNION: The UNION operator is used to combine the results of two or more SELECT statements into a single result set. The UNION operator removes duplicate rows between the various SELECT statements.

Table -> person1

id	Name
1	Yogi
2	Jogi
3	lofi

Table -> person2

Id	Name
4	Ram
5	Mohan
6	yogi

SELECT * FROM database.person1

UNION

SELECT * FROM database.person2

Id	Name
1	Yogi
2	Jogi
3	Lofi
4	Ram
5	Mohan

UNION ALL: The UNION ALL operator is similar to the UNION operator, but it does not remove duplicate rows from the result set.

Table -> person1

id	Name
1	Yogi
2	Jogi
3	lofi

Table -> person2

Id	Name
4	Ram
5	Mohan
3	yogi

```
SELECT * FROM database.person1

UNION ALL

SELECT * FROM database.person2
```

Id	Name
1	Yogi
2	Jogi
3	Lofi
4	Ram
5	Mohan
3	Yogi

INTERSECT: The INTERSECT operator returns only the rows that appear in both result sets of two SELECT statements.

Table -> person1

id	Name
1	Yogi
2	Jogi
3	lofi

Table -> person2

Id	Name
4	Ram
5	Mohan
1	yogi

SELECT * FROM database.person1

INTERSECT

SELECT * FROM database.person2

id	Name
1	yogi

EXCEPT: The EXCEPT or MINUS operator returns only the distinct rows that appear in the first result set but not in the second result set of two SELECT statements.

Table -> person1

id	Name
1	Yogi
2	Jogi
3	lofi

Table -> person2

Id	Name
4	Ram
5	Mohan
3	yogi

SELECT * FROM database.person1

EXCEPT

SELECT * FROM database.person2

id	Name
2	Jogi
3	lofi

SELF JOIN

A self join is a type of join in which a table is joined with itself. This means that the table is treated as two separate tables, with each row in the table being compared to every other row in the same table. Self joins are used when you want to compare the values of two different rows within the same table.

For example, you might use a self join to compare the salaries of two employees who work in the same department, or to find all pairs of customers who have the same billing address.

Table -> users

User_id	name	age	Friend_id
1	Rohit	39	7
2	Kohli	37	4
4	Gill	27	2
5	Rahul	30	11
7	Jadeja	32	1
10	Bumrah	29	null
11	kuldeep	31	5

```
SELECT * FROM database.users t1
Join database.users t2
ON t1.Friend_id = t2.User_id
```

Resultant table :

User_id	name	age	Friend_id	User_id	name	age	Friend_id
1	Rohit	39	7	7	Jadeja	32	1
2	Kohli	37	4	4	Gill	27	2
4	Gill	27	2	2	Kohli	37	4
5	Rahul	30	11	11	Kuldeep	31	5
7	Jadeja	32	1	1	Rohit	39	7
11	Kuldeep	31	5	5	Rahul	30	11

Joining on more then one columns

Table – students :

Student_id	First_name	Last_name	Class_id	Enrollment_year
1	John	Smith	1	2021
2	Jane	Doe	2	2020
3	Bob	Johnson	1	2021
4	Sally	Brown	3	2022
5	Tom	Williams	2	2022
6	Alice	Davis	4	2020

Table – branch :

Class_id	Class_name	Teacher	Class_year
1	Math 101	Mr. smith	2021
2	English 1	Ms.johnson	2021
3	Science 1	Dr. lee	2022
4	History 1	Ms. Williams	2022

```
SELECT * FROM database.Student t1
JOIN database.brach t2
ON t1.class_id = t2.class_id
AND t1.enrollment_year = t2.class_year
```

Student_id	First_name	Last_name	Class_id	Enrollment_number	Class_id	Class_name	teacher	Class_y
1	John	Smith	1	2021	1	Math 101	Mr.smith	2021
3	Bob	Johnson	1	2021	1	Math 101	Mr.smith	2021
4	Sally	Brown	3	2022	3	Science 1	Dr.lee	2022

Joining more then 2 tables

Table – users:

User_id	Name	State
1	Rohan	Bhopal
2	Ram	Indore
3	Krishna	Pune

Table – order :

Order_id	User_id	Order_date
b-102	1	12-12-2024
b-103	2	1-1-2025
b-104	1	2-2-2025

Table – order_details

Order_id	Amount	Quantity
b-102	10000	100
b-103	20000	200
b-104	30000	300

```
SELECT * FROM database.users t1
JOIN database.order t2
On t1.User_id = t2.User_id
Join database.order_details t3
On t2.Order_id = t3.Order_id
```

User_id	Name	State	Order_id	Order_date	amount	Quantity
1	Rohan	Bhopal	b-102	12-12-2024	10000	100
2	Ram	Indore	b-103	1-1-2025	20000	200
1	Rohan	Bhopal	b-104	2-2-2025	30000	300

Yogeshchouhan263@gmail.com