

I am a new comer to mssql server and you are experieced teacher and who teaches from scratch gives comprehensive notes. so explain me from scratch all the joins concepts from basics 1. self 2. right 3. left 4.Inner 5. Cartesion join . what are they ?why used ? use emp and dept example . Also include driven and driver table while explaining left and right joins and whenever required.

Alright! Let's go step by step and learn **SQL Joins** in **MS SQL Server** from scratch with clear explanations and examples using `emp` (Employee) and `dept` (Department) tables.



1. What is a JOIN in SQL?

A `JOIN` is used to combine rows from two or more tables based on a related column between them. It helps retrieve meaningful data from multiple tables in a database.

Why use Joins?

- To avoid data redundancy by splitting related data into multiple tables.
- To fetch meaningful data by combining information from multiple tables.
- To perform data analysis efficiently.



2. Sample Tables for Examples

We will use Employee (`emp`) and Department (`dept`) tables.

Employee Table (`emp`)

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

Department Table (`dept`)

dept_id	dept_name
10	HR
20	Marketing
30	Finance
40	Operations
50	IT

Department Table (`dept`)

dept_id	dept_name
10	HR
20	Marketing
30	Finance
40	Operations
50	IT

Department Table (dept)

dept_id		dept_name
10		HR
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

Department Table (dept)

dept_id		dept_name
10		HR
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

Department Table (dept)

dept_id		dept_name
10		HR
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

Department Table (dept)

dept_id		dept_name
10		HR

Result

emp_id	emp_name	dept_name
1	Alice	HR
2	Bob	IT
3	Charlie	Sales
4	David	NULL
5	Eve	HR

Driver and Driven Table

- Left Table (emp) → Driver table
- Right Table (dept) → Driven table

Explanation:

- Employee David is included even though he has no department (NULL in dept_name).
- Department Finance is not included since it has no employees.

3. RIGHT JOIN (or RIGHT OUTER JOIN)

- Retrieves all records from the right table (dept) and matching records from the left table (emp).
- If no match is found, NULL is returned for columns from the left table.

Query

sql

↓

Copy Edit

```
SELECT emp.emp_id, emp.emp_name, dept.dept_name
FROM emp
RIGHT JOIN dept ON emp.dept_id = dept.dept_id;
```

Result

emp_id	emp_name	dept_name
1	Alice	HR
2	Bob	IT
3	Charlie	Sales
5	Eve	HR
NULL	NULL	Finance

Driver and Driven Table

- Right Table (dept) → Driver table
- Left Table (emp) → Driven table

Explanation:

- Department Finance is included even though it has no employees (NULL in emp_name).
- Employee David is not included since his dept_id is NULL.

4. SELF JOIN

- A SELF JOIN is when a table joins itself.
- Used to compare rows within the same table.

Use Case

Suppose we have an emp table where employees have a manager_id referring to another employee in the same table.

Modified Employee Table

emp_id	emp_name	manager_id
1	Alice	NULL
2	Bob	1
3	Charlie	1
4	David	2
5	Eve	3

Query

sql

↓

Copy Edit

```
SELECT e1.emp_name AS Employee, e2.emp_name AS Manager
FROM emp e1
LEFT JOIN emp e2 ON e1.manager_id = e2.emp_id;
```

Result

Employee	Manager
Alice	NULL
Bob	Alice
Charlie	Alice

David	Bob
Eve	↓ Charlie

Explanation:

- Alice has **no manager** (`NULL`).
- Bob and Charlie report to Alice.
- David reports to Bob.
- Eve reports to Charlie.



5. CROSS JOIN (Cartesian Join)

- Produces all possible combinations of rows from both tables.
- No need for a `JOIN` condition.

Query

sqlCopyEdit

```
SELECT emp.emp_name, dept.dept_name
FROM emp
CROSS JOIN dept;
```

↓

Result

emp_name	dept_name
Alice	HR
Alice	IT
Alice	Sales
Alice	Finance
Bob	↓ HR
Bob	IT
Bob	Sales
Bob	Finance
...	...

Explanation:

- If `emp` has 5 employees and `dept` has 4 departments, the result will have $5 \times 4 = 20$ rows.
- This is mostly used in testing or when generating all possible combinations.



Summary Table

Join Type	What it Does
INNER JOIN	Returns only matching records from both tables
LEFT JOIN	Returns all records from the left table and matching ones from the right
RIGHT JOIN	Returns all records from the right table and matching ones from the left
SELF JOIN	Joins a table with itself
CROSS JOIN	Returns all possible combinations of rows from both tables

Final Thoughts

- Use **INNER JOIN** when you want only matching data.
- Use **LEFT JOIN** when you need **all records from the left table** and matching ones from the right.
- Use **RIGHT JOIN** when you need **all records from the right table** and matching ones from the left.
- Use **SELF JOIN** for hierarchical or recursive relationships.
- Use **CROSS JOIN** when you need all possible row combinations.

Let me know if you need more clarifications! 🚀

Good . now explain me 1. Joining 3 or more tables 2.use of subqueries for joins 3. set operators

Glad you found the explanation useful! Now, let's move forward with:

1. **Joining Three or More Tables**
2. **Using Subqueries for Joins**
3. **Set Operators**

I'll explain each in detail with examples.

1. Joining Three or More Tables

When working with databases, you often need to fetch data from more than two tables. The concept of joins remains the same—you simply extend the join conditions to more tables.

Example: Employees, Departments, and Projects

Let's assume we have the following three tables:

Employee Table (emp)

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

Department Table (dept)

dept_id	dept_name
10	HR
20	IT
30	Sales
40	Finance

Project Table (project)

project_id	project_name	emp_id
101	Alpha	1
102	Beta	2
103	Gamma	3
104	Delta	5



Joining Three Tables: emp, dept, project

Let's say we want to retrieve employee names, their department names, and the projects they are working on.

Query:



sql

Copy

Edit

```
SELECT emp.emp_name, dept.dept_name, project.project_name
FROM emp
LEFT JOIN dept ON emp.dept_id = dept.dept_id
LEFT JOIN project ON emp.emp_id = project.emp_id;
```

Result:

emp_name	dept_name	project_name
Alice	HR	Alpha
Bob	IT	Beta
Charlie	Sales	Gamma
David	NULL	NULL
Eve	HR	Delta

Explanation:

- LEFT JOIN ensures that even employees without projects (David) or departments (Finance) are included.
- Employees with multiple projects will have multiple rows.

2. Using Subqueries for Joins

A subquery is a query inside another query. Instead of using JOINS, you can sometimes use subqueries to fetch data.

Example 1: Subquery Instead of INNER JOIN

If we want employees and their department names, we can use either a join or a subquery.

Using INNER JOIN :

sql

Copy

Edit

```
SELECT emp.emp_name, dept.dept_name
FROM emp
INNER JOIN dept ON emp.dept_id = dept.dept_id;
```

Using SUBQUERY :

sql

Copy

Edit

```
SELECT emp_name,
       (SELECT dept_name FROM dept WHERE dept.dept_id = emp.dept_id) AS dept_name
FROM emp;
```

Result:

emp_name	dept_name
Alice	HR

emp_name	dept
Bob	IT
Charlie	Sales
David	NULL
Eve	HR

When to Use a Subquery Instead of a Join?

- If you need **only one column** from the joined table.
- If the **join is not complex** (subqueries can be slower for large datasets).
- If you don't need to fetch multiple columns from the related table.

3. SQL Set Operators

Set operators combine the results of two or more queries. Unlike joins (which merge columns), set operators combine **rows**.

Types of Set Operators

Operator	Description
UNION	Combines results from two queries and removes duplicates.
UNION ALL	Combines results from two queries and keeps duplicates .
INTERSECT	Returns only the common rows between two queries.
EXCEPT	Returns rows from the first query that are not in the second query.

Example Dataset

Assume we have another table storing employees who have attended training programs.

Training Table (training)

emp_name	training_completed
Alice	Yes
Bob	No
Charlie	Yes
David	Yes

1. UNION (Removes Duplicates)

Fetch a list of all employees and training attendees (without duplicates).

```
sql
SELECT emp_name FROM emp
UNION
SELECT emp_name FROM training;
```

Result:

emp_name
Alice
Bob
Charlie

emp_name
David
Eve

2. UNION ALL (Keeps Duplicates)

If we want duplicates to be included, we use `UNION ALL`.

```
sql
SELECT emp_name FROM emp
UNION ALL
SELECT emp_name FROM training;
```

Result (With Duplicates):

emp_name
Alice
Bob
Charlie
David
Eve
Alice
Bob
Charlie
David

3. INTERSECT (Common Records)

Find employees who are both in the employee table (`emp`) and have attended training (`training`).

```
sql
SELECT emp_name FROM emp
INTERSECT
SELECT emp_name FROM training;
```

Result:

emp_name
Alice
Bob
Charlie
David

4. EXCEPT (Records Present in First Query but Not in Second)

Find employees who have NOT attended training.

```
sql
SELECT emp_name FROM emp
EXCEPT
SELECT emp_name FROM training;
```


Result:

emp_name
Eve

Final Summary

Concept	Explanation
Joining 3 Tables	Use multiple <code>JOIN</code> conditions to link more than two tables.
Subqueries for Joins	Use subqueries when retrieving single values from related tables.
Set Operators	<code>UNION</code> , <code>INTERSECT</code> , <code>EXCEPT</code> are used to combine row sets from multiple queries.

Which One to Use?

- Use **Joins** when merging **columns** from multiple tables.
- Use **Subqueries** when extracting **single values** from related tables.
- Use **Set Operators** when merging **rows** from multiple queries.

Hope this clears everything! Let me know if you have any doubts! 🚀

📄 👍 🗨️ 🔊 ✎️ ↺ Compare with Claude ↕

Ask anything

+

🌐 Search

💡 Reason

🔊

ChatGPT can make mistakes. Check important info.

SQL Queries and Topics

Remove City Distance Duplicat

Delete All Records SQL

Upgrade plan
More access to the best models