Lectrue 7.1. JOIN

What is a JOIN?

> A **JOIN query** is any query that accesses **multiple tables at once** (or multiple copies of the same table). It **combines rows** from one table with rows from another **based on a condition**.

Typiacal use: data is normalized into multiple tables; JOIN brings it back together

Basic generic syntax:

```
SELECT column_name(s)
FROM table1
JOIN table2;
```

Explicit vs implicit JOIN syntax

Explicit

```
SELECT *
FROM weather
JOIN cities ON city = name;
```

- Compares `weather.city` with `cities.name`
- Returns only rows where they match.

Better style is to **list columns explixity amd qualify** them:

```
SELECT weather.city, weather.temp_lo, weather.temp_hi,
weather.prcp, weather.date, cities.location
FROM weather JOIN cities ON weather.city = cities.name;
```

Reason:

- Avoid query breaking if a new column with same name as added later
- Avoid duplicate column confusion

Implicit join

Same JOIN can be written as:

```
SELECT *
FROM weather, cities
WHERE city = name;
```

- Old syntax
- Equivalent result but worse readability: join condition is mixed with other WHERE filters
- Prefer expliciy `JOIN ... ON` in modern SQL

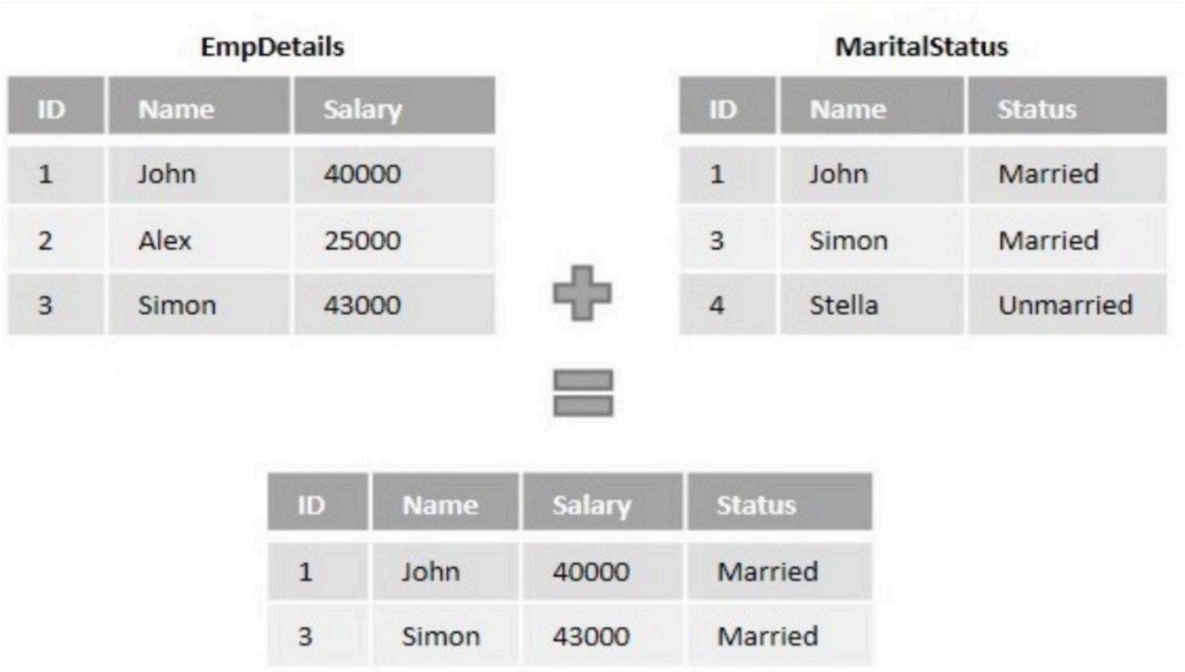
INNER JOIN

Defenition:

> **Inner join** = return only rows that have **matching values in both tables** (in the join condition)
> Rows without matches on either side are **ignored** (dropped fron result)

Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.colum_name;
```



OUTER JOIN (Left / Right / Full)

General idea:

- **Outer join** returns rows **even if one side has no matching row.**
- Types on slides: **LEFT**, **RIGHT**, **FULL**.
- In PostgreSQL/SQL, the keyword **OUTER** is optional:
  - `LEFT JOIN` == `LEFT OUTER JOIN`
  - `RIGHT JOIN` == `RIGHT OUTER JOIN`
  - `FULL JOIN` == `FULL OUTER JOIN`

LEFT JOIN (Left Outer Join)

Definition:

> Returns **all rows from the left table**, and the matching rows from the right table.

> If there is **no match** on the right side, right columns are **NULL**

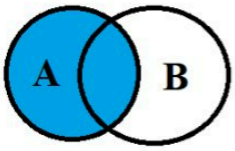
Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN cities
   ON table1.column_name = table2.column_name;
```

Student ID	Name
1001	A
1002	B
1003	C
1004	D

+

Student ID	Department
1004	Mathematics
1005	Mathematics
1006	History
1007	Physics
1008	Computer Science



Student ID	Name	Department
1001	A	NULL
1002	B	NULL
1003	C	NULL
1004	D	Mathematics

RIGHT JOIN (Right Outer Join)

Defenition:

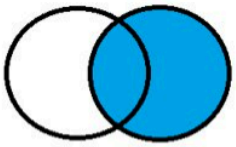
> Returns **all rows from the right table**, and matching rows from the left.
> Left column become **NULL** for rows that don't match.

Syntax:

```
SELECT table1.column1, table2.column2, ...
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Student ID	Name
1001	A
1002	B
1003	C
1004	D

Student ID	Department
1004	Mathematics
1005	Mathematics
1006	History
1007	Physics
1008	Computer Science



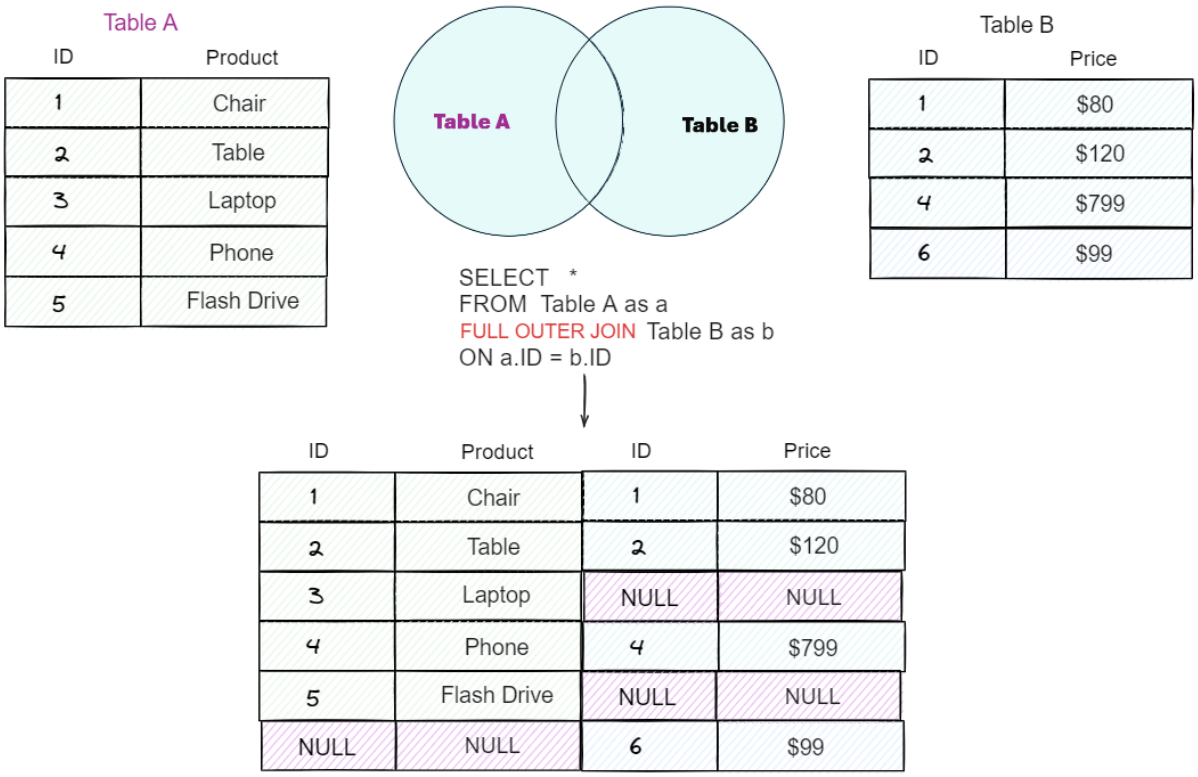
Student ID	Name	Department
1004	D	Mathematics
1005	NULL	Mathematics
1006	NULL	History
1007	NULL	Physics
1008	NULL	Computer Science

FULL JOIN (Full Outer Join)

> Returns **all rows from both tables.**
> Where there is a match → combined row.
> Where there is no match on one side → NULL on that side.

Syntax:

```
SELECT column_name(s)
FROM table1
FULL JOIN table2
ON table1.column_name = table2.column_name;
```



CROSS JOIN (Cartesian product)

Definition:

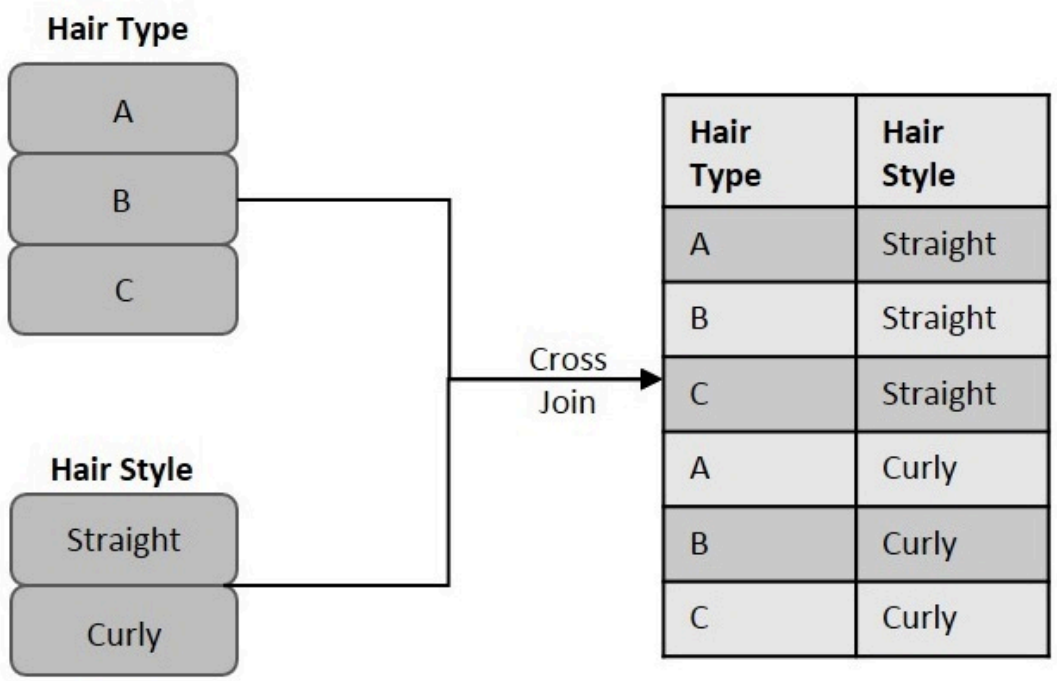
> `CROSS JOIN` produces a **Cartesian product** of two tables:
>
> - Every row of `T1` with **every** row of `T2`
>
> If `T1` has `m` rows and `T2` has `n` rows → result has `m * n` rows.

Syntax:

```
SELECT select_list
FROM T1
CROSS JOIN T2
```

Use cases:

- Generate combinatins (e.g. all dates × all shifts)
- Test data



NATURAL JOIN

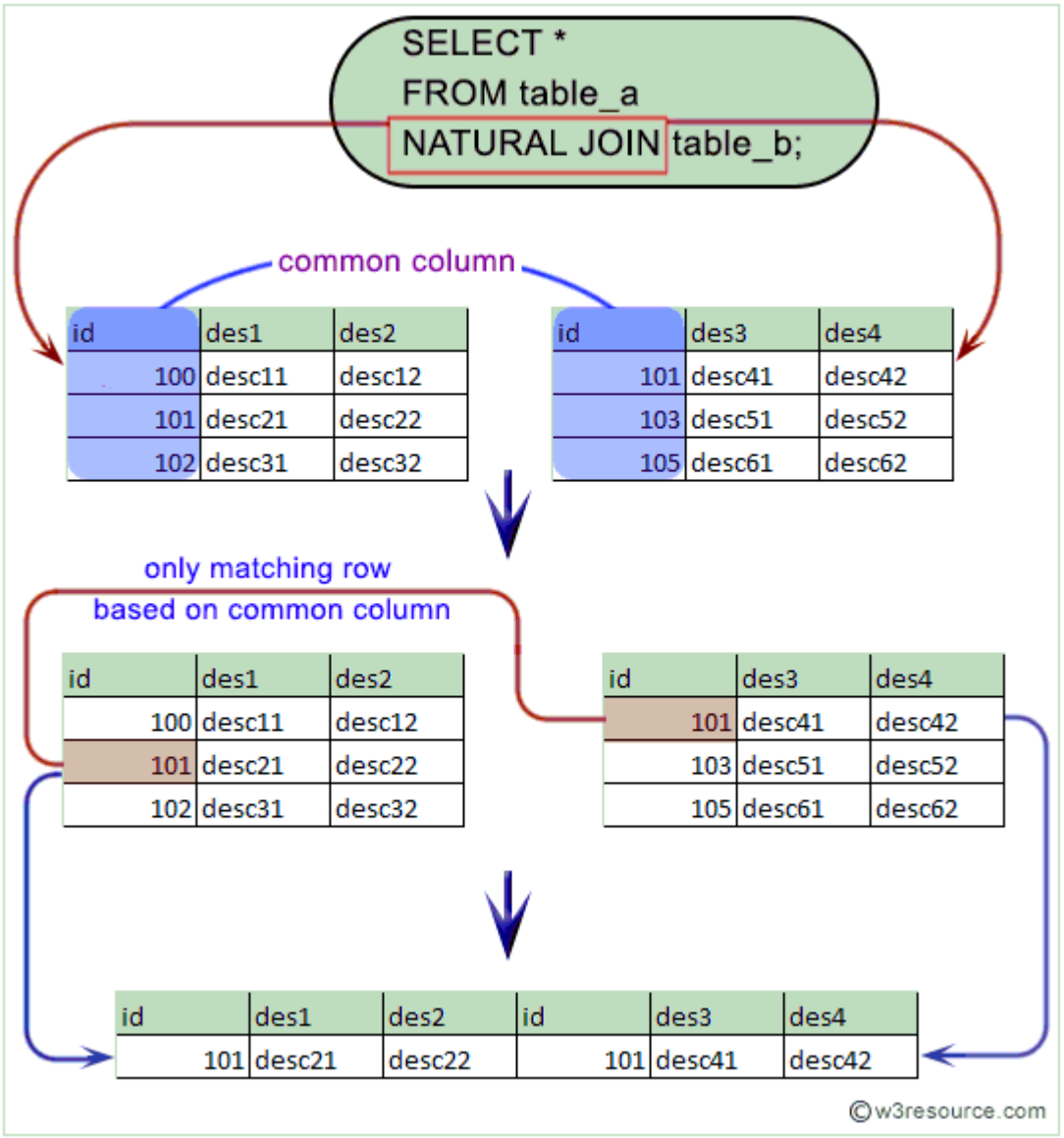
Definition:

> **A natural join** automatically joins tables on **all columns with the same name** in both tables.
> You don't specify the join condition explicity: it's implicit.

Syntax:

```
SELECT select_list
FROM T1 NATURAL [INNER | LEFT | RIGHT] JOIN T2;
```

If you don't write `INNER/LEFT/RIGHT`, default is usually `INNER`



Differences vs other joins:

NATURAL JOIN:

- Convenient if tables have **multiple columns in common.**
- But can give **unexpected results** if:
  - schema changes (new common column appears),
  - or if you forget which columns are common.

INNER JOIN:

- Explicit condition, more **precise** and **predictable**.

LEFT / RIGHT / FULL OUTER JOIN

- More **flexibility**: you control which side is fully included even when matches do not exist.

SELF JOIN

Defenition:

> Self-join = a table is joined with itself.

Often used for:

- Hierarchical data (employees → managers)
- Comparing rows in the same table

Synytax:

```
SELECT select_list
FROM table_name T1
INNER JOIN table_name T2
    ON join_predicate;
```

Color	Name	Color
Blue	John	Red
Green	Alex	Blue
Red	Simon	Green

Color	Name	Color
Blue	John	Red
Green	Alex	Blue
Red	Simon	Green

Name	Secret_Santa
John	Simon
Alex	John
Simon	Alex