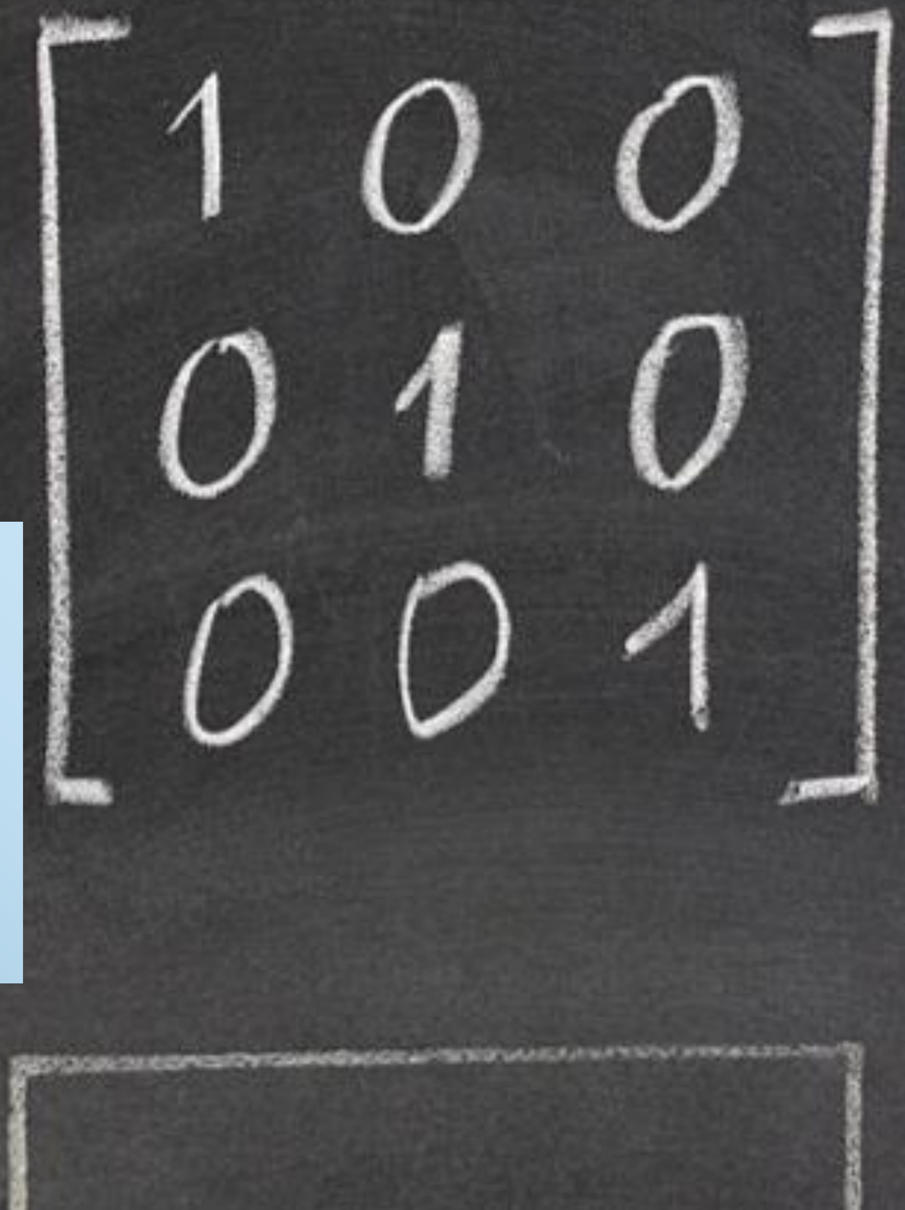
SQL Joins under the hood

by Nicola Orecchini, 31/07/2025



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It all started with a question

What’s the difference between these 2 queries?

```
SELECT *
FROM customer
LEFT JOIN staff ON o.order_id=p.order_id
```

```
SELECT *
FROM customer
LEFT JOIN staff ON 1=1
WHERE o.order_id=p.order_id
```

Apparently equal, only to find out that they don’t produce the same output. To find out why, read this deck, which gives the answer towards the end. Before doing that, it goes through some foundation concepts that will help you derive an answer by yourself

Agenda

1. Introduction to relational data model
2. SQL basic operations

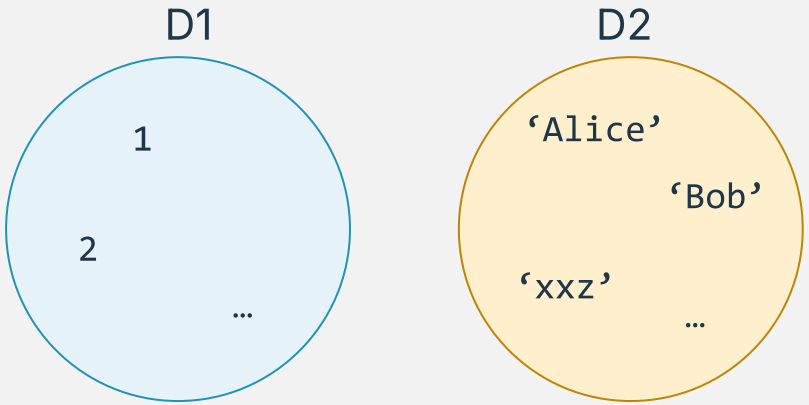
Deep dive on JOIN

Agenda

1. Introduction to relational data model
2. SQL basic operations

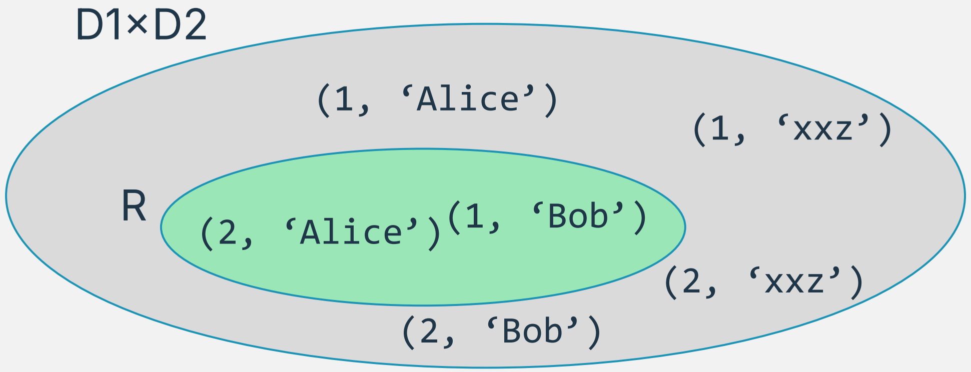
   Deep dive on JOIN

A relational database is a database that stores data in tables (called *relations*)



$D_1 \times D_2 \times \cdots \times Dn$ are domains: sets of possible values for a variable. Each domain corresponds to a database column type, e.g.:

- $D_1 = \mathbb{Z}$ (integers for id)
- D_2=strings (for name)



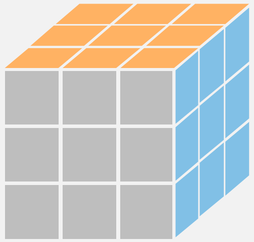
A relation R is a subset of the cartesian product $D_1 \times D_2 \times \cdots \times D_n$:
$R \subseteq D_1 \times D_2 \times \cdots \times D_n$

Each element of R is a n-tuple $(d_1, d_2, \ldots, d_n)$ where $d_i \in D_i$.

So, a relation corresponds to the concept of table, in a database. A table is thus a set of tuples

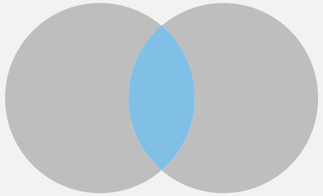
The word «relational» in relational databases refers to the fact that data is stored in structured tables (relations), and not to the fact that tables have relationships between them (e.g., keys)

Users can manipulate tables through Structured Query Language, a language that allows multiple operations



Filters

- σ Selection
- π Projection



Set operations

- ∪ Union
- ∩ Intersection
- \ Exclusion

Relational operations

- × Cartesian Product
- ⋈ Join

...and many more, but for this presentation we focus on these

Source: https://blog.jooq.org/say-no-to-venn-diagrams-when-explaining-joins/; https://www.geeksforgeeks.org/dbms/difference-between-selection-and-projection-in-dbms/

Agenda

1. Introduction to relational data model
2. SQL basic operations

   Deep dive on JOIN

Filters allow to take only specific rows or columns of a table

	σ Selection	π Projection
What it does	Considers only rows of a table that meet a specified condition	Considers only columns of a table as specified by a list of names

Example	`SELECT *`
	`FROM customer`
	`WHERE name=‘bob’`
		`SELECT name, surname`
		`FROM customer`

Set operations combine or compare the results of multiple queries that return the same column structure

	∪ Union	∩ Intersection	\\ Exclusion
What it does	Starting from 2 (or more) tables, creates a new table containing all records of the 1st and all those of the 2nd. The 2 tables must be made of the same type of tuples	Starting from 2 (or more) tables, creates a new table containing only records that are present in both tables. The 2 tables must be made of the same type of tuples	Starting from 2 (or more) tables, creates a new table containing only records that are in the first but not in the second. The 2 tables must be made of the same type of tuples

Example	`SELECT name, surname`
	`FROM customer`
	`UNION`
	`SELECT name, surname`
	`FROM staff`
		`SELECT name, surname`
		`FROM customer`
		`INTERSECT`
		`SELECT name, surname`
		`FROM staff`
			`SELECT name, surname`
			`FROM customer`
			`EXCEPT`
			`SELECT name, surname`
			`FROM staff`

Relational operations combine information from two or more tables by matching rows based on a condition

	× Cartesian product	⋈ Join
What it does	Given 2 (or more) tables, creates a new table whose records are tuples obtained by combining a record of the first table with one of the second, until all possible pairs have been generated	Given 2 (or more) tables, calculates the Cartesian product between them, and then filters only tuples where a specific condition (specified by the user) is met. Then, depending on the join type, special extra records can be returned

Example	`SELECT *`
	`FROM customer, staff`
		`SELECT *`
		`FROM customer`
		`JOIN staff ON o.order_id=p.order_id`

Agenda

1. Introduction to relational data model
2. SQL basic operations

❯ Deep dive on JOIN

Joins are of 2 types: inner & outer

Inner join

What it does

A Cartesian product (also called “Cross join”) between 2 (or more) tables in which only combinations that fulfil a given predicate are retained

Examples (equivalent variants)

```
-- "Classic" ANSI JOIN syntax
SELECT *
FROM customer c
JOIN staff s ON c.cust_id=s.staff_id
```

```
-- "Old" syntax using a "CROSS JOIN"
SELECT *
FROM customer c, staff s
WHERE c.cust_id=s.staff_id
```

Outer join

An inner join with extra records. These extra records are rows from either the LEFT, the RIGHT, or both (FULL) tables, for which no rows satisfying the predicate were found in the inner join results

```
SELECT *
FROM customer
LEFT JOIN staff ON
o.order_id=p.order_id
```

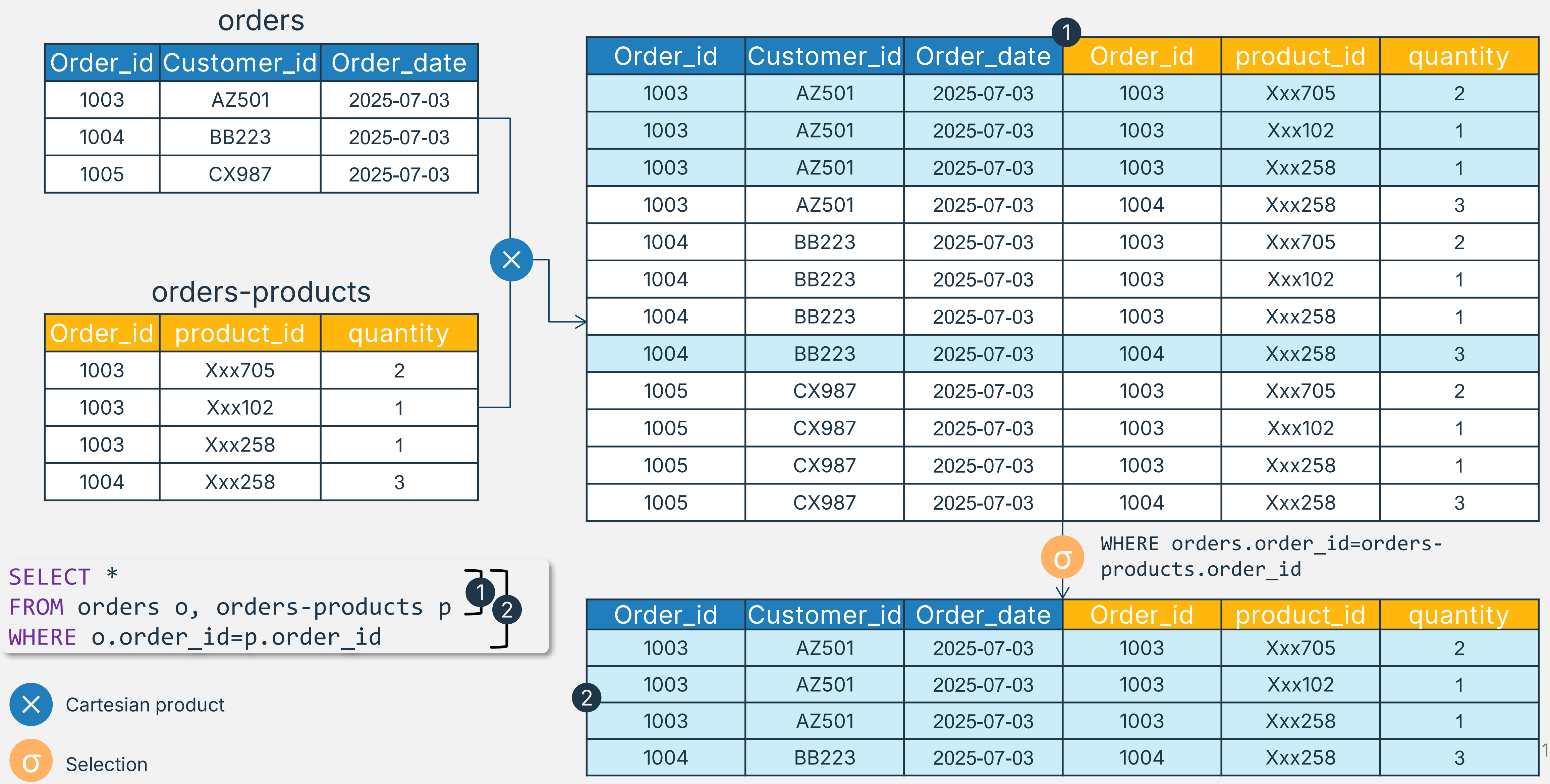
LEFT/RIGHT/FULL

```
SELECT *
FROM customer c
JOIN staff s ON c.cust_id=s.st_id

UNION

SELECT rows_not_matched.*, NULL, ..., NULL
FROM (
  SELECT rows_not_matched.*
  FROM customer c

  EXCEPT

  SELECT c.*
  FROM customer c
  JOIN staff s ON c.cust_id=s.st_id
) rows_not_matched
```

Source: https://blog.jooq.org/say-no-to-venn-diagrams-when-explaining-joins/

Click here for interactive example

Inner Join is a filtered Cartesian product



orders

Order_id	Customer_id	Order_date
1003	AZ501	2025-07-03
1004	BB223	2025-07-03
1005	CX987	2025-07-03

orders-products

Order_id	product_id	quantity
1003	Xxx705	2
1003	Xxx102	1
1003	Xxx258	1
1004	Xxx258	3

1

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1003	AZ501	2025-07-03	1004	Xxx258	3
1004	BB223	2025-07-03	1003	Xxx705	2
1004	BB223	2025-07-03	1003	Xxx102	1
1004	BB223	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3
1005	CX987	2025-07-03	1003	Xxx705	2
1005	CX987	2025-07-03	1003	Xxx102	1
1005	CX987	2025-07-03	1003	Xxx258	1
1005	CX987	2025-07-03	1004	Xxx258	3

σ

```
WHERE orders.order_id=orders-
products.order_id
```

2

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3

```
SELECT *
FROM orders o, orders-products p      ] 1 ] 2
WHERE o.order_id=p.order_id
```

× Cartesian product

σ Selection

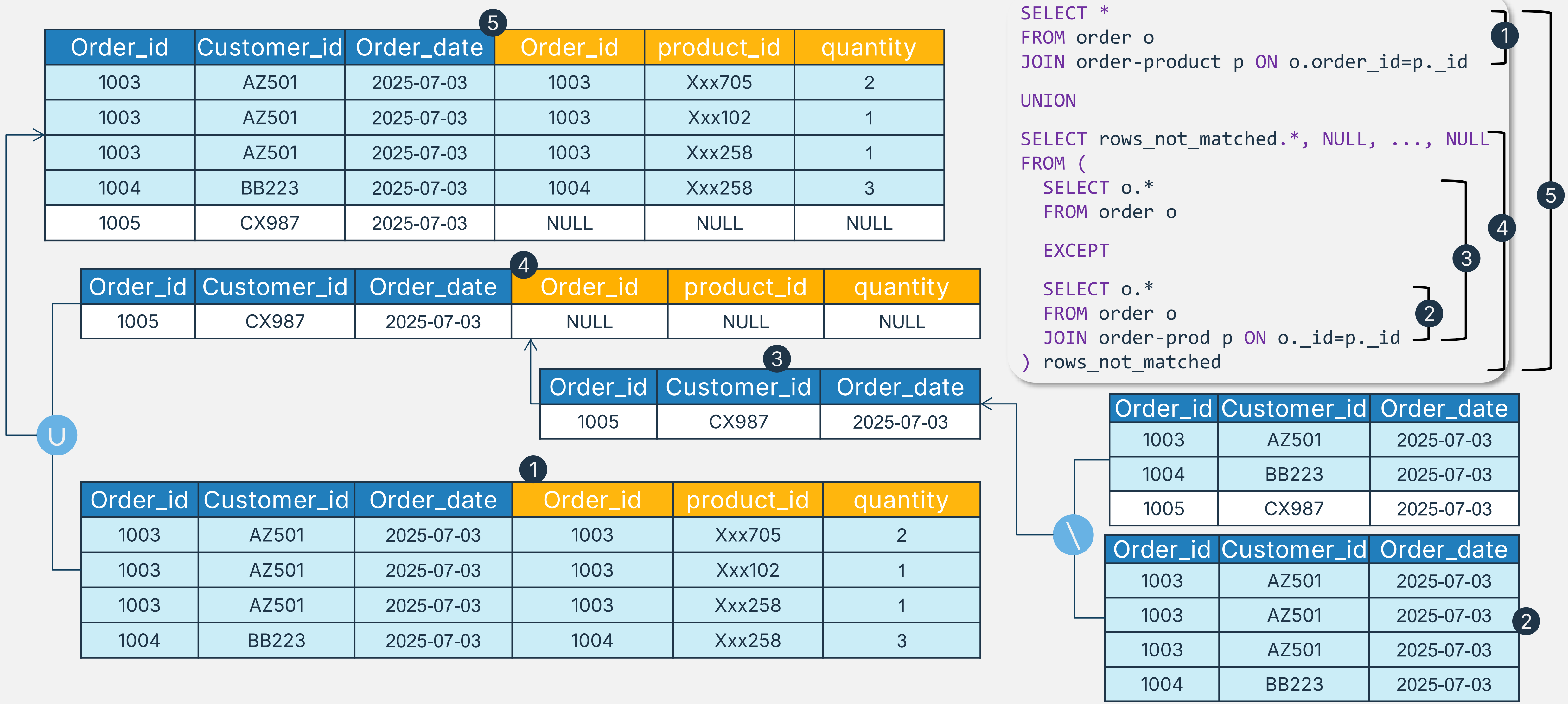
Click here for interactive example

U Union

\ Exclusion

Left Join is an Inner Join unioned with unmatched records from the left table padded with NULLs

```
SELECT *
FROM order o
JOIN order-product p ON o.order_id=p._id

UNION

SELECT rows_not_matched.*, NULL, ..., NULL
FROM (
  SELECT o.*
  FROM order o

  EXCEPT

  SELECT o.*
  FROM order o
  JOIN order-prod p ON o._id=p._id
) rows_not_matched
```



5

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3
1005	CX987	2025-07-03	NULL	NULL	NULL

4

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1005	CX987	2025-07-03	NULL	NULL	NULL

3

Order_id	Customer_id	Order_date
1005	CX987	2025-07-03

1

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3

Order_id	Customer_id	Order_date
1003	AZ501	2025-07-03
1004	BB223	2025-07-03
1005	CX987	2025-07-03

2

Order_id	Customer_id	Order_date
1003	AZ501	2025-07-03
1003	AZ501	2025-07-03
1003	AZ501	2025-07-03
1004	BB223	2025-07-03

So, at a high-level, a left join has an extra step than inner

Inner Join

1. Cartesian product
2. Selection

Left Join

1. Cartesian product
2. Selection
3. Union

The key difference between INNER and OUTER JOINS is that the first ends with a selection (i.e., removing some rows), while the second ends with a UNION (i.e., adding some rows)

Source: https://stackoverflow.com/questions/354070/sql-join-what-is-the-difference-between-where-clause-and-on-clause

Click here for interactive example

What's the difference between putting the pairing condition in the ON clause vs in the WHERE?

```sql
SELECT *
FROM orders o
JOIN products p ON o.order_id=p.order_id
```

- We saw how a Join behaves as a Cartesian product followed by a filter
- In modern Join SQL statement, the filter condition is put inside the ON statement (e.g., ON o.order_id=p.order_id)
- But the WHERE clause is also used to filter rows of a table

```sql
SELECT *
FROM orders o
JOIN products p ON 1=1
WHERE o.order_id=p.order_id
```

- So, technically, we could obtain the same behaviour of a JOIN by using an always true ON condition (e.g., 1=1), and then filter records with a WHERE, right?
- Is this correct? What is the difference between ON and WHERE?
- Does something change for INNER and OUTER Joins?

Source: https://stackoverflow.com/questions/354070/sql-join-what-is-the-difference-between-where-clause-and-on-clause

Click here for interactive example

To answer this question, let’s look at the steps the INNER JOIN query goes through

```
SELECT *
FROM orders o
JOIN products p ON o.order_id=p.order_id
```

1 Cartesian product orders X products

2 Selection o.order_id=p.order_id

```
SELECT *
FROM orders o
JOIN products p ON 1=1
WHERE o.order_id=p.order_id
```

1 Cartesian product orders X products

2 Selection 1=1

3 Selection o.order_id=p.order_id

At the end, even if the second query has an extra step, the result is the same, because:

- 1 = 1
- 2 does nothing
- 2 = 3

For an Inner Join, putting the join predicate in the ON or in the WHERE produces the same output

Source: https://stackoverflow.com/questions/354070/sql-join-what-is-the-difference-between-where-clause-and-on-clause

Click here for interactive example

For Outer Joins, things are a bit different

```
SELECT *
FROM orders o
LEFT JOIN products p ON o.order_id=p.order_id
```

1	Cartesian product	orders X products
2	Selection	o.order_id=p.order_id
3	Union	rows of orders that aren’t in products

```
SELECT *
FROM orders o
LEFT JOIN products p ON 1=1
WHERE o.order_id=p.order_id
```

1	Cartesian product	orders X products
2	Selection	1=1
3	Union	rows of orders that aren’t in products
4	Selection	o.order_id=p.order_id

This time, results are different, because:

- 1 = 1
- 2 does nothing
- 3 = 4 ⎫ Step 3 and 4 do the exact same things
- 4 = 3 ⎭ in both queries, but in an inverted order

Thus, the query on the left will filter records as per predicate and then add back unmatched records, whereas query on the right will first add unmatched records and then filter those out

For an Outer Join, putting the join predicate in the WHERE clause produces the same output of an Inner Join

Source: https://stackoverflow.com/questions/354070/sql-join-what-is-the-difference-between-where-clause-and-on-clause

This is what outputs would look like

```sql
SELECT *
FROM orders o
LEFT JOIN products p ON o.order_id=p.order_id
```

1. Cartesian product — orders X products
2. Selection — o.order_id=p.order_id
3. Union — rows of orders that aren’t in products

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3
1005	CX987	2025-07-03	NULL	NULL	NULL

```sql
SELECT *
FROM orders o
LEFT JOIN products p ON 1=1
WHERE o.order_id=p.order_id
```

1. Cartesian product — orders X products
2. Selection — 1=1
3. Union — rows of orders that aren’t in products
4. Selection — o.order_id=p.order_id

Order_id	Customer_id	Order_date	Order_id	product_id	quantity
1003	AZ501	2025-07-03	1003	Xxx705	2
1003	AZ501	2025-07-03	1003	Xxx102	1
1003	AZ501	2025-07-03	1003	Xxx258	1
1004	BB223	2025-07-03	1004	Xxx258	3

So, we can answer the initial question

These 2 queries are different.

```
SELECT *
FROM orders o
LEFT JOIN products p ON o.order_id=p.order_id
```

```
SELECT *
FROM orders o
LEFT JOIN products p ON 1=1
WHERE o.order_id=p.order_id
```

If the Join was an Inner Join, the result would have been the same. But, being a Left Join, results are different: the second query subtely produces the same output as an Inner Join

Nicola Orecchini

nicolaorecchini@gmail.com