SQL essentials

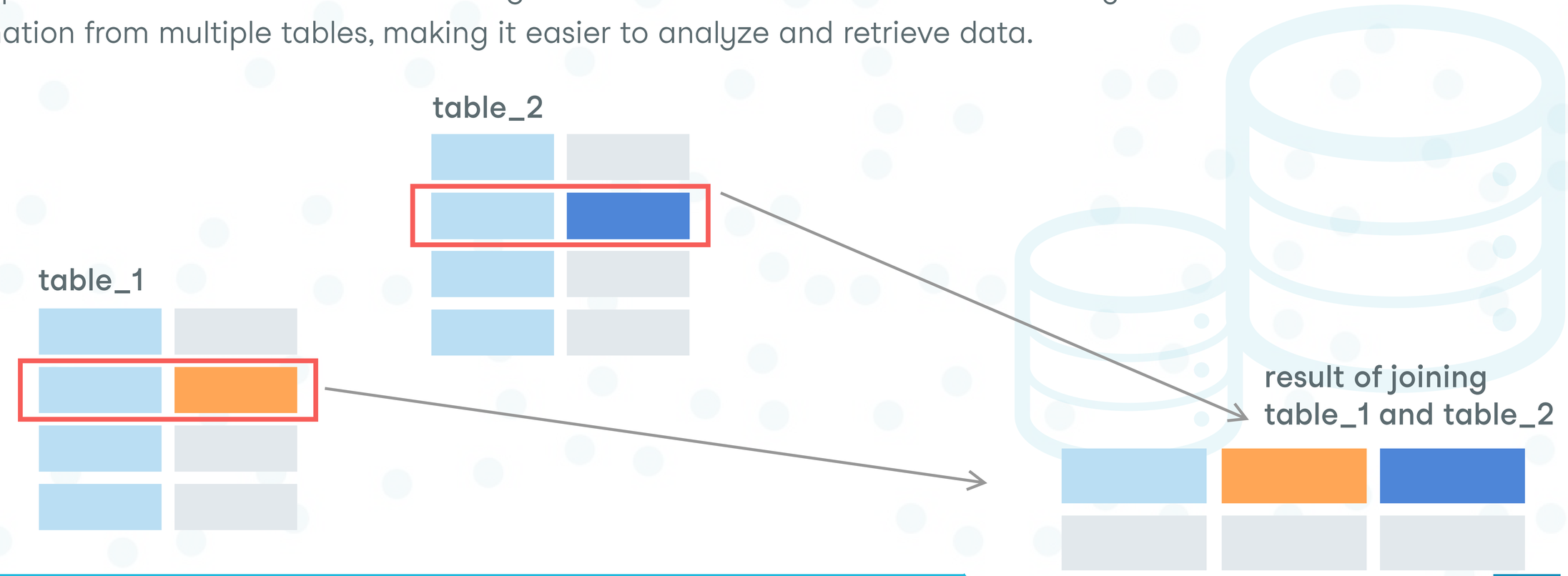
Mastering database queries: a comprehensive guide to SQL

Learning session 03
Combining data from multiple tables

Instructor
Péter Fülöp
peteristvan.fulop@hcl.com

Extracting information from multiple tables

Joining tables refers to the process of **combining data from two or more database tables** based on a related column, typically a common column or key, to produce a single result set.
This operation is fundamental to working with relational databases and allows you to extract and consolidate information from multiple tables, making it easier to analyze and retrieve data.



Extract information from multiple tables

employees

id	first_name	last_name	business_unit_id
3	Emily	Johnson	1
7	Michael	Davis	2

business_units

id	name
1	Consulting 1
2	Managed services
3	Consulting 2

? Retrieve the first names, last names and department names of all employees.

Emily	Johnson	Consulting 1

The INNER JOIN

Extracting information from multiple tables using INNER JOIN

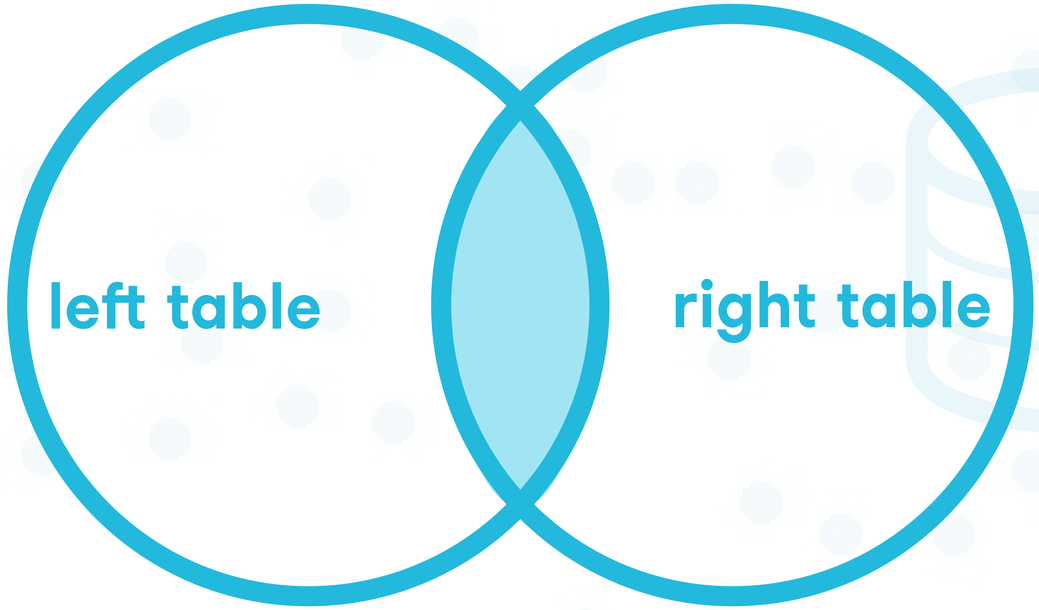
INNER JOIN is a method for combining rows from two or more tables based on a related column between them. It only returns rows for which there is a match in both tables, excluding unmatched rows.

SQL

```sql
SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;
```

Extracting information from multiple tables using INNER JOIN

When you use INNER JOIN in PostgreSQL, you retrieve a result set that contains rows from multiple tables, emphasizing the records that **meet the specified conditions**, while excluding those that do not have matching values in the common field(s).



INNER JOIN

Example of using INNER JOIN

Display the id, first name, last name, and business unit ids for every individual in the employees table.

SQL

```
SELECT id, first_name, last_name, business_unit_id
FROM practice_03.employees;
```

OUTPUT

id	first_name	last_name	business_unit_id
7	Anita	Molnar	3
8	Ferenc	Kovacs	4
9	Lilla	Toth	<null>
10	Laszlo	Nagy	

29 rows

Show the id and name of all business units.

SQL

```
SELECT id, name
FROM practice3.business_units;
```

OUTPUT

id	name
1	Consulting 1
2	Consulting 2
3	Managed se

6 rows

Examples of using INNER JOIN

SQL

```sql
SELECT first_name, last_name, name as business_unit
FROM employees
INNER JOIN business_units
ON employees.business_unit_id = business_units.id;
```

OUTPUT

id	first_name	last_name	business_unit
24	Katalin	Varga	Consulting 1
11	Anna	Kovacs	Consulting 1
1	Gabor	Nagy	Consulting 1

23 rows

Examples of using INNER JOIN

Retrieve the first names, last names, business_unit, band, and position of all employees where the business unit, band and position are not NULL.

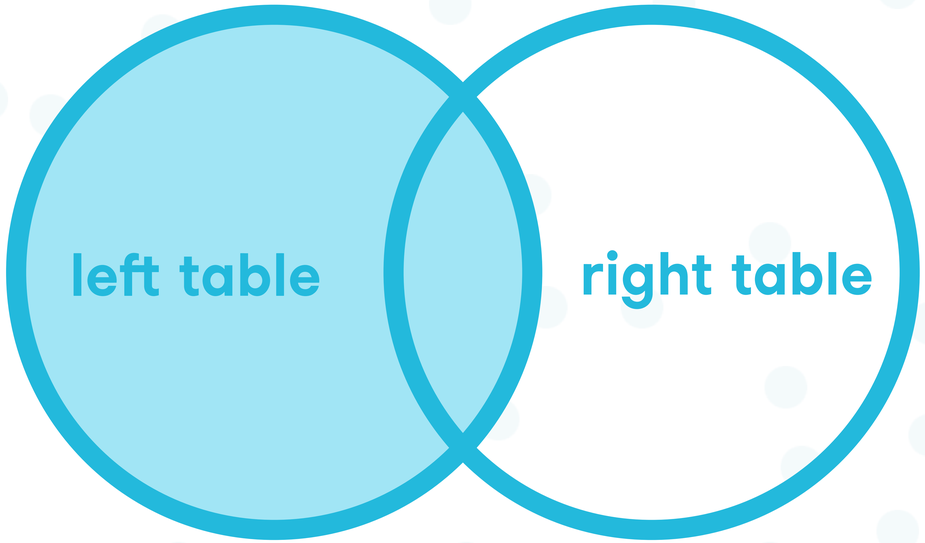
SQL

```
SET search_path TO practice_03;
SELECT emp.first_name, emp.last_name,
    bu.name as business_unit,
    jt.name job_title,
    b.name as band
FROM employees AS emp
    INNER JOIN business_units AS bu ON emp.business_unit_id = bu.id
    INNER JOIN job_titles AS jt ON emp.job_title_id = jt.id
    INNER JOIN bands AS b ON emp.band_id = b.id;
```

OUTER JOINS

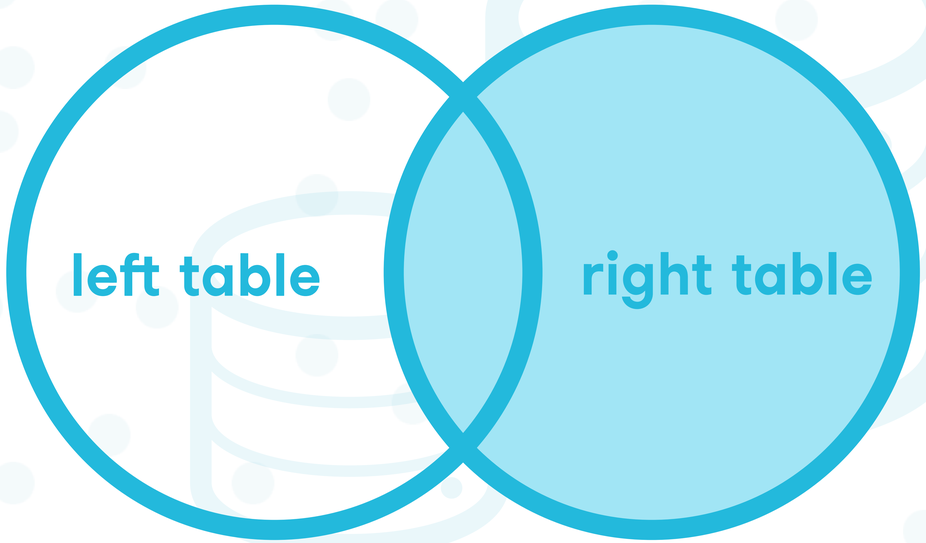
Extracting information from multiple tables using OUTER JOIN

LEFT JOIN



Returns **all records from the left table** along with any matching record from the right table.

RIGHT JOIN



Returns **all records from the right table** along with any matching record from the left table.

Extracting information from multiple tables using LEFT OUTER JOIN

Retrieve the first names, last names, and business unit names of all employees. Ensure that employees without an assigned business unit are also included in the list.

SQL

```sql
SELECT emp.first_name, emp.last_name, bu.name as business_unit
FROM practice_03.employees AS emp
LEFT OUTER JOIN practice_03.business_units AS bu ON emp.business_unit_id = bu.id;
```

OUTPUT

first_name	last_name	business_unit
Laszlo	Nagy	Team Augmentation 2
Katalin	Szabo	Team Augmentation 2
Noemi	Biro	<null>

29 rows

Extracting information from multiple tables using RIGHT OUTER JOIN

Retrieve the first names, last names, and business unit names of all employees. Include business units that have no employees assigned, but exclude employees who don't have a business unit assigned to them.

SQL

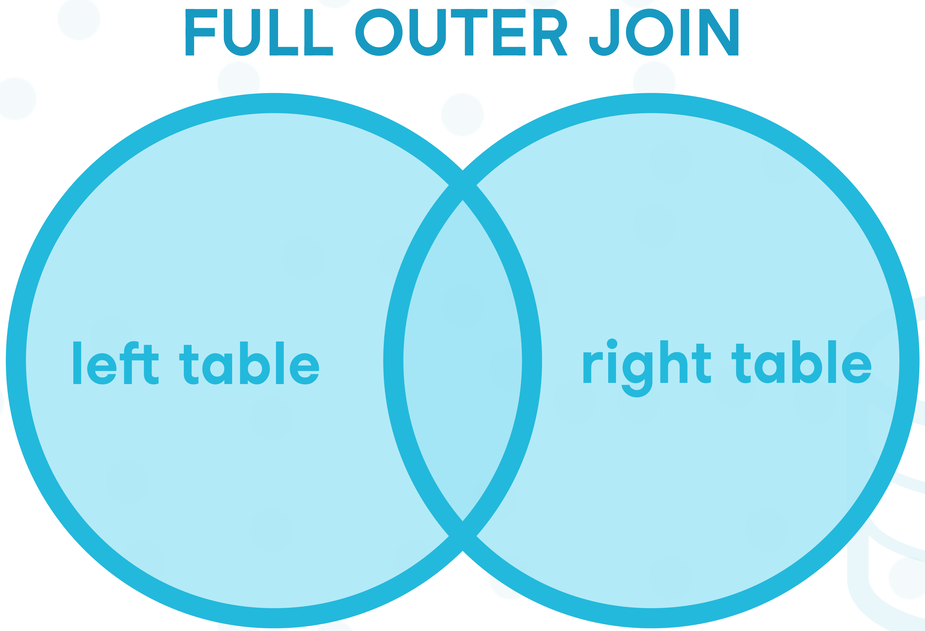
```sql
SELECT emp.first_name, emp.last_name, bu.name as business_unit
FROM practice_03.employees AS emp
RIGHT OUTER JOIN practice_03.business_units AS bu ON emp.business_unit_id = bu.id;
```

OUTPUT

	first_name	last_name	business_unit
22	Laszlo	Nagy	Team Augmentation 2
23	Katalin	Szabo	Team Augmentation 2
24	<null>	<null>	Generative AI

24 rows

Extracting information from multiple tables using FULL OUTER JOIN



A **FULL OUTER JOIN** returns all the rows
when there is a match in either the left table or the right table.
If there's no match, the result will have NULL for every column of the table that lacks a match.
Essentially, **it combines the results of both LEFT and RIGHT OUTER JOINs.**

Examples of using FULL OUTER JOIN

Retrieve the first names, last names, and business unit names for all employees. Ensure the list also includes employees without an assigned business unit and business units that don't have any employees associated with them.

SQL

```sql
SELECT emp.first_name,
       emp.last_name,
       bu.name as business_unit
FROM practice_03.employees AS emp
         FULL OUTER JOIN practice_03.business_units AS bu
                         ON emp.business_unit_id = bu.id
```

Examples of using FULL OUTER JOIN

Retrieve the id, first names, last names, and business unit names for all employees. Ensure the list also includes employees without an assigned business unit and business units that don't have any employees associated with them.

OUTPUT

id	first_name	last_name	business_unit
15	Istvan	Horvath	Team Augmentation 2
10	Laszlo	Nagy	Team Augmentation 2
5	Katalin	Szabo	Team Augmentation 2
<null>	<null>	<null>	Generative AI
20	Noemi	Biro	<null>
19	Laszlo	Varga	<null>

30 rows

SELF JOIN

Extracting information from multiple tables using SELF JOIN

A **SELF JOIN** is a join in which a table is joined with itself.
It's used when there's a relationship within the same table.

Examples of using SELF JOIN

Show the employee ID, full name of the employee, and the full name of their team lead.

First attempt.

SQL

```
SELECT emp.id,
       emp.first_name  || ' ' || emp.last_name AS full_name,
       tld.id as team_lead_id,
       tld.first_name  || ' ' || tld.last_name AS team_lead
FROM practice_03.employees AS emp
         INNER JOIN practice_03.employees AS tld
                    ON emp.lead_employee_id = tld.id
ORDER BY
    full_name ASC;
```

Examples of using SELF JOIN

Show the employee ID, full name of the employee, and the full name of their team lead.

The list below shows only employees who have a team lead assigned and omits those without one.

OUTPUT

	id	full_name	team_lead_id	team_lead
17	14	Katalin Szabo	4	Eva Horvath
18	5	Katalin Szabo	5	Katalin Szabo
19	24	Katalin Varga	8	Ferenc Kovacs
20	28	Krisztina Toth	6	Andras Kovacs
21	10	Laszlo Nagy	10	Laszlo Nagy
22	9	Lilla Toth	9	Lilla Toth

23 rows

Let's explore a method to **include employees who lack team leads** in this list as well.

Examples of using SELF JOIN

Display the employee ID, the employee's full name, and their team lead's full name. Ensure the list also features employees without an assigned team lead (where lead_employee_id is NULL).

Let us LEFT JOIN instead of INNER JOIN:

SQL

```
SELECT emp.id,
       emp.first_name  || ' ' || emp.last_name AS full_name,
       tld.id as team_lead_id,
       tld.first_name  || ' ' || tld.last_name AS team_lead
FROM practice_03.employees AS emp
         LEFT OUTER JOIN practice_03.employees AS tld
                         ON emp.lead_employee_id = tld.id
ORDER BY
    full_name ASC;
```

Examples of using SELF JOIN

Now, the result includes all employees, even those without an assigned team lead.

OUTPUT

	id	full_name	team_lead_id	team_lead
23	19	Laszlo Varga	<null>	<null>
24	9	Lilla Toth	10	Laszlo Nagy
25	25	Miklos Kovacs	9	Lilla Toth
26	20	Noemi Biro	<null>	<null>
27	17	Tamas Molnar	<null>	<null>
28	3	Zoltan Toth	10	Laszlo Nagy
29	18	Zsuzsa Papp	<null>	<null>

29 rows

What if the company wants to assign team leads to those currently without one and needs a list of only those employees who don't have a team lead assigned?

Examples of using SELF JOIN

The company wants to assign team leads to those currently without one and needs a list of only those employees who don't have a team lead assigned.

Let us filter the results by checking if the id of the team lead IS NULL.

SQL

```
SELECT emp.id,
       emp.first_name  || ' ' || emp.last_name AS full_name,
       tld.id as team_lead_id,
       tld.first_name  || ' ' || tld.last_name AS team_lead
FROM practice_03.employees AS emp
         LEFT OUTER JOIN practice_03.employees AS tld
                         ON emp.lead_employee_id = tld.id
WHERE tld.id IS NULL
ORDER BY
    full_name ASC;
```

Examples of using SELF JOIN

Now, the result includes only employees with no team lead assigned.

OUTPUT

	id	full_name	team_lead_id	team_lead
1	16	Eva Kiss	<null>	<null>
2	10	Laszlo Nagy	<null>	<null>
3	19	Laszlo Varga	<null>	<null>
4	20	Noemi Biro	<null>	<null>
5	17	Tamas Molnar	<null>	<null>
6	18	Zsuzsa Papp	<null>	<null>

6 rows