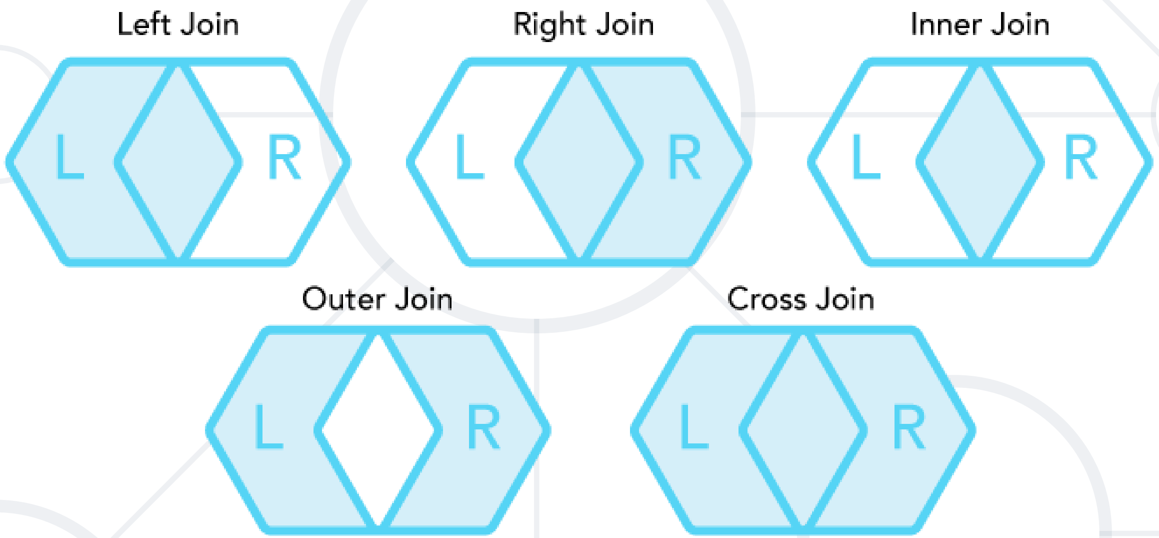
Joins, Subqueries and Indices

Data Retrieval and Performance



SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg

Questions

Software University

sli.do

#python-db

Table of Contents

JOINs

Gathering Data from Multiple Tables

Data from Multiple Tables

- Sometimes you need data from several tables:

Employees

employee_name	department_id
Edward	3
John	NULL

Departments

department_id	department_name
3	Sales
4	Marketing
5	Purchasing

employee_name	department_id	department_name
Edward	3	Sales

JOINS

- Used to collect data from **two** or **more** tables
- Types:

```
INNER JOIN
```

```
LEFT JOIN
```

```
RIGHT JOIN
```

```
FULL JOIN
```

```
CROSS JOIN
```

Cartesian Product (1)

- This will produce the **Cartesian product**:

```
SELECT last_name, name AS department_name
FROM employees, departments;
```

- The result:

last_name	department_name
Gilbert	Engineering
Brown	Engineering
…	…
Gilbert	Sales
Brown	Sales

Cartesian Product (2)

- Each row in the first table is paired with **all** the rows in the second table
- Formed when the join condition is omitted
- To avoid this, always include a valid **`JOIN` condition**

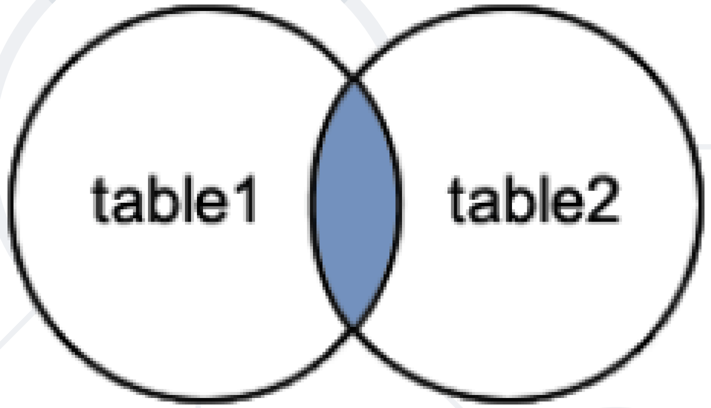
Tables

id	first_name	department_id
1	Guy	7
2	Kevin	4
3	Roberto	1
4	Rob	2
5	Thierry	NULL

id	name
1	Engineering
2	Tool Design
4	Marketing
…	…
7	Production

INNER JOIN

- Produces a set of records that **match in both tables**



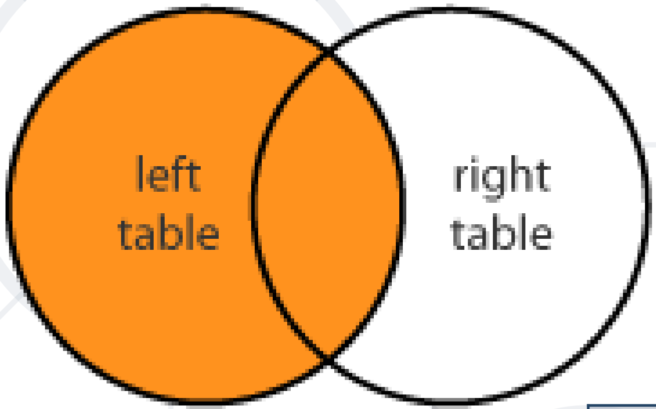
```
SELECT employees.first_name,
           departments.name
FROM employees
INNER JOIN departments --or just JOIN
ON employees.department_id =
departments.department_id;
```

Join Conditions

first_name	name
Guy	Production
Kevin	Marketing
Roberto	Engineering
Rob	Tool Design

LEFT JOIN

- Matches every entry in **left** table regardless of match in the **right**



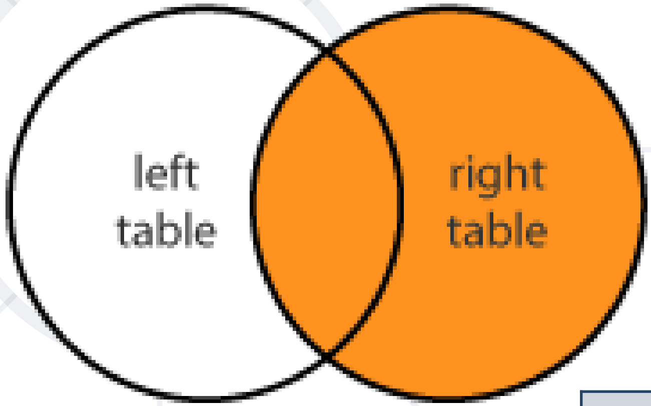
```
SELECT employees.first_name,
         departments.name
FROM employees
LEFT JOIN departments
ON employees.department_id =
departments.department_id;
```

Join Conditions

first_name	name
Guy	Production
Kevin	Marketing
Roberto	Engineering
Rob	Tool Design
Thierry	NULL

RIGHT JOIN

- Matches every entry in **right** table regardless of match in the **left**



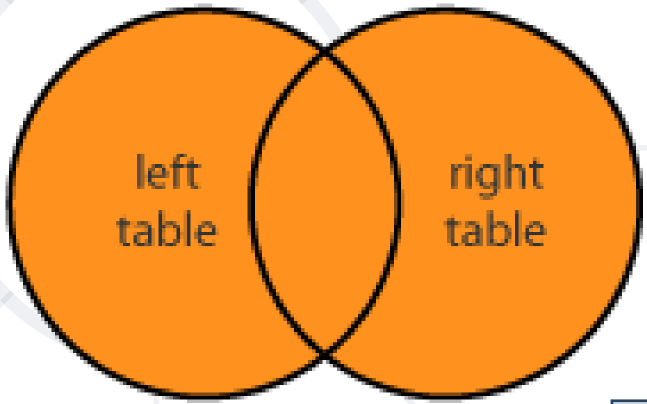
```
SELECT employees.first_name,
          departments.name
FROM employees
RIGHT JOIN departments
ON employees.department_id =
departments.department_id;
```

Join Conditions

first_name	name
Guy	Production
…	…
NULL	Purchasing
NULL	Production
Ruth	Production

FULL OUTER JOIN

- Returns all records in both tables regardless of **any** match



```
SELECT employees.first_name,
          departments.name
FROM employees
FULL JOIN departments
ON employees.department_id =
departments.department_id;
```

Join Conditions

first_name	name
Guy	Production
…	…
Thierry	NULL
NULL	Purchasing
NULL	Production

Cross Join (1)

- Produces a set of associated rows of two tables
  - Multiplication of each row in the first table with each in the second one
  - The result is a **Cartesian** product when there's **no condition** in the **WHERE** clause

```
SELECT d.department_id, d.name,
    e.employee_id, e.first_name
FROM departments AS d
 CROSS JOIN employees AS e;
```

No Join Conditions

Cross Join (2)

department_id	name
1	Engineering
2	Tool Design
3	Sales
4	Marketing

employee_id	first_name	department_id
1	Guy	7
2	Kevin	4
…	…	…
5	Thierry	NULL

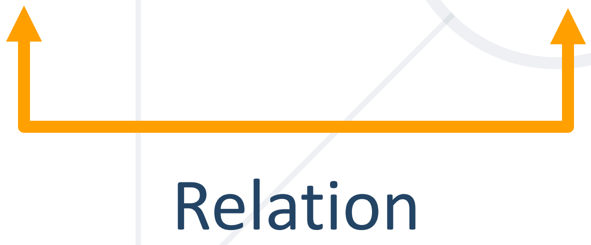
Result

department_id	name	employee_id	first_name
1	Engineering	1	Guy
1	Engineering	2	Kevin
1	Engineering	…	…
1	Engineering	5	Thierry

Join Overview

first_name	department_id
Guy	7
Kevin	4
Roberto	1
Rob	2
Thierry	NULL
NULL	5

department_id	name
1	Engineering
2	Tool Design
3	Sales
4	Marketing
5	Purchasing
…	…



Join Overview: INNER JOIN

first_name	department_id
Guy	7
Kevin	4
Roberto	1
Rob	2
Thierry	NULL
NULL	5

department_id	name
1	Engineering
2	Tool Design
3	Sales
4	Marketing
5	Purchasing
10	Finance

first_name	department_id	name
Kevin	4	Marketing
Roberto	1	Engineering
Rob	2	Tool Design
NULL	5	Purchasing

Problem: Towns Addresses

- Get information about **addresses** in the "**`soft_uni`**" database, which are
  - in **`San Francisco`**, **`Sofia`**, or **`Carnation`**
  - display **`town_id`**, **`town_name`**, **`address_text`**

town_id	town_name	address_text
9	San Fransisco	1234 Seaside Way
9	San Fransisco	5725 Glaze Drive
15	Carnation	1411 Ranch Drive
…	…	…
32	Sofia	163 Nishava Str, ent A, apt. 1

Solution: Towns Addresses

```sql
SELECT t.town_id, t.name AS town_name,
     a.address_text
FROM towns AS t
JOIN addresses AS a
     ON a.town_id = t.town_id
WHERE t.name IN ('San Francisco', 'Sofia',
'Carnation')
ORDER BY town_id, address_id;
```

Join Overview: LEFT JOIN

first_name	department_id
Guy	7
Kevin	4
Roberto	1
Peter	8

department_id	name
1	Engineering
2	Tool Design
3	Sales
4	Marketing

first_name	department_id	name
Guy	7	NULL
Kevin	4	Marketing
Roberto	1	Engineering
Peter	8	NULL

Join Overview: RIGHT JOIN

first_name	department_id
Peng	13
Brian	3
Ruth	7
Rob	2

department_id	name
3	Sales
16	Executive
10	Finance
7	Production

first_name	department_id	name
Brian	3	Sales
NULL	16	Executive
NULL	10	Finance
Ruth	7	Production

Problem: Managers

- Get information about the **first 5 managers** in the "`soft_uni`" database
  - `employee_id`
  - `full_name`
  - `department_id`
  - `department_name`

employee_id	full_name	department_id	department_name
3	Roberto Tamburello	10	Finance
4	Rob Walters	2	Tool Design
6	Bradley	5	Purchasing
12	Terri Duffy	1	Engineering
21	Peter Krebs	8	Production Control

Solution: Managers

```
SELECT e.employee_id, CONCAT(first_name, ' ',
     last_name) AS full_name, d.department_id,
     d.name AS department_name
FROM employees AS e
RIGHT JOIN departments AS d
     ON d.manager_id = e.employee_id
ORDER BY e.employee_id LIMIT 5;
```

Problem: Employees Projects

- Get information about the **employees**, working on a **project** with **`id = 1`**. Display:
  - `employee_id`
  - `full_name`
  - `project_id`
  - `project_name`

employee_id	full_name	project_id	project_name
3	Roberto Tamburello	1	Classic Vest
15	Jeffrey Ford	1	Classic Vest
18	John Campbell	1	Classic Vest
…	…	…	…
246	Frank Martinez	1	Classic Vest

Solution: Employees Projects

```sql
SELECT e.employee_id, CONCAT(e.first_name,' ',
     e.last_name) as full_name, p.project_id,
     p.name AS project_name
FROM employees AS e
JOIN employees_projects AS e_p
     ON e.employee_id = e_p.employee_id
JOIN projects AS p
     ON p.project_id = e_p.project_id
WHERE e_p.project_id = 1;
```

Subqueries

Query Manipulation On Multiple Levels

Subqueries

- Subqueries – SQL query inside a larger one
- Can be nested in **SELECT**, **INSERT**, **UPDATE**, **DELETE**
  - Usually added within a **WHERE** clause

```
SELECT employee_id AS
    id, first_name,
    department_id
FROM employees
WHERE department_id = 1;
```

id	first_name	department_id
3	Roberto	1
9	Gail	1
…	…	1

Problem: Higher Salary

- **Count** the number of employees who receive a salary, **higher** than the average
  - Use "**soft_uni**" database, table "**employees**"

employee_id	first_name	last_name	…
216	Mike	Seamans	…
178	Barbara	Moreland	…
…	…	…	…

count
88

Solution: Higher Salary

```
SELECT COUNT(e.employee_id) AS "count"
FROM employees AS e
WHERE e.salary >
(
    SELECT AVG(salary) AS
    "average_salary" FROM employees
);
```

Indices

Speed Retrieval of Rows

Indices

- Special lookup tables that **speed retrieval** of rows
  - Usually implemented as **B-trees**
- Speed up SELECT queries and WHERE clauses
- Slows down data input
- Should be used for big tables only (e.g., 50 000 rows)
- Should NOT be used on columns that contain a high number of NULL values

Indices Syntax

```sql
CREATE INDEX index_name_idx
ON table_name(first_column, second_column);
```

Index Name

Table Name

Columns

Practice

Live Demo in Class

Summary

- Joins

```
SELECT * FROM employees AS e
   JOIN departments AS d ON
d.department_id = e.department_id
```

- **Subqueries** are used to nest queries
- Indices improve SQL search **performance** if used properly

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter International

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License