

# Бази даних та SQL

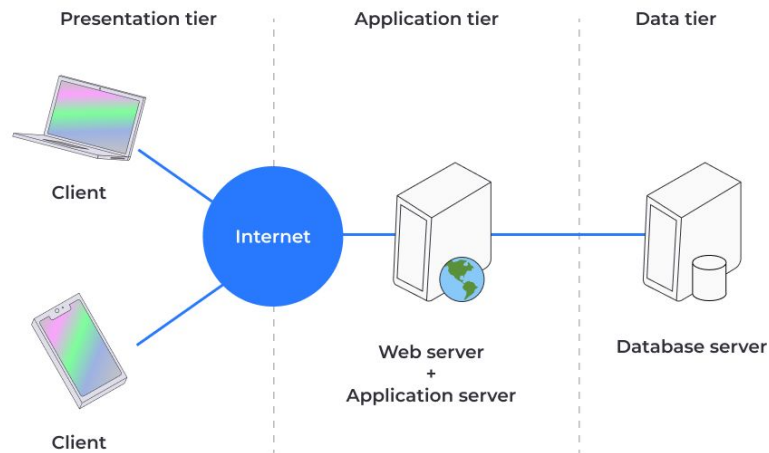
# План заняття:

1. Що таке база даних?
2. Моделі БД
3. SQL & NoSQL і сучасність
4. SQL команди
5. Q/A
6. Анонс наступного заняття
7. Корисні лінки

# База даних

**База даних (БД)** — іменована сукупність структурованих даних, що відображає стан об'єктів та їх відносин в аналізованій предметній області. Під предметною областю розуміється деяка частина реального світу, інформація про яку представлена у базі даних.

**Система управління базами даних (СУБД)** – сукупність мовних та програмних засобів, призначених для створення, наповнення, оновлення та видалення баз даних.



**Основними поняттями** в концепції баз даних є узагальнені категорії **«дані»** та **«модель даних»**.

Поняття **«дані»** в концепції баз даних – **це набір конкретних значень** та параметрів, що характеризують об'єкт, умову, ситуацію або будь-які інші фактори.

**Приклади даних:** Петренко Микола Степанович, \$30 тощо. Дані не мають певної структури, дані стають інформацією тоді, коли користувач задає їм певну структуру, тобто усвідомлює їх зміст.

*Тому, центральним поняттям у сфері баз даних є поняття моделі.*

Ядром будь-якої бази даних є **модель даних**.

**Модель даних** – це сукупність структур даних та операцій з їх обробки. За допомогою моделі даних можуть бути представлені об'єкти та взаємозв'язки між ними.

**Модель даних** – це деяка абстракція, яка, будучи прикладеною до конкретних даних, дозволяє користувачам і розробникам трактувати їх як інформацію, тобто **відомості, що містять як дані, так і взаємозв'язок між ними**.



**За допомогою моделі даних можуть бути представлені об'єкти предметної області та взаємозв'язку між ними.**

Залежно від виду організації даних розрізняють такі моделі БД:

- ієрархічну
- мережеву
- реляційну
- об'єктно-орієнтовану

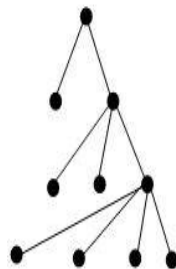
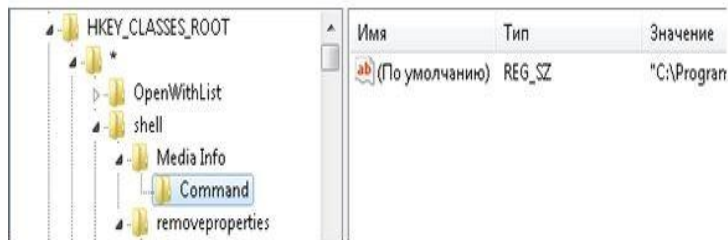


## Приклад ієрархічної моделі даних.

Якщо ієрархічна база даних містить інформацію про покупців та їх замовлення, то існуватиме об'єкт «покупець» (батьківський) та об'єкт «замовлення» (дочірній). Об'єкт «покупець» матиме вказівники від кожного замовника до фізичного розташування замовлень покупця до об'єкту «замовлення».

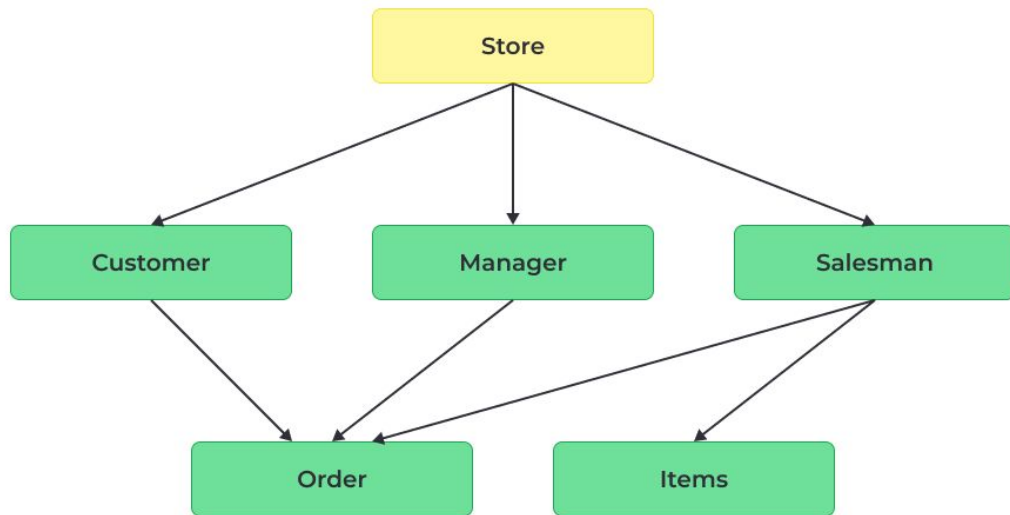
У цій моделі запит, спрямований вниз по ієрархії, – простий (наприклад, які замовлення належать покупцю); однак запит, спрямований вгору по ієрархії, – складніший (наприклад, який покупець помістив це замовлення). Також важко уявити не-ієрархічні дані при використанні цієї моделі.

Ієрархічною базою даних є файлова система, що складається з кореневого каталогу, в якому є ієрархія підкаталогів та файлів.



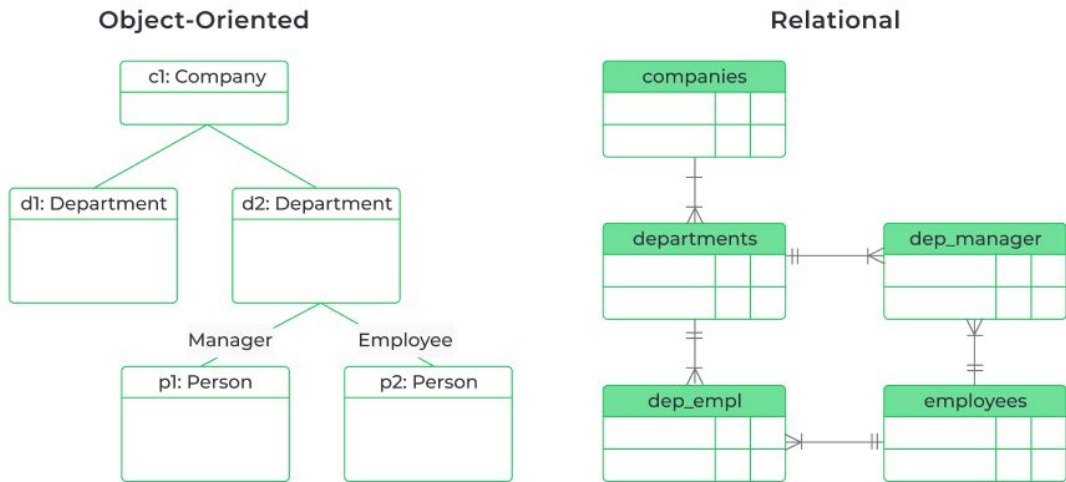
\* на цей час представляє  
винятково історичний інтерес

**Мережева** – логічна модель даних, що є розширенням ієрархічного підходу. На відміну від ієрархічної моделі, в мережевій структурі даних дочірній об'єкт може мати будь-яке число батьківських.





**Об'єктна та об'єктно-орієнтована.** Дані в таких базах являють собою об'єкти з певними наборами властивостей, методів та поведінки. Відносини даних об'єктів будуються на основі узагальнення властивостей та методів та поведінки різних об'єктів по відношенню один до одного.

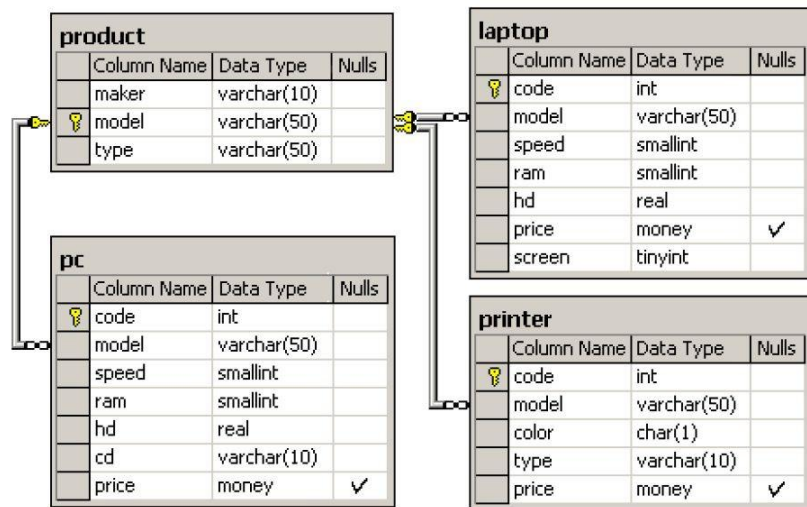


**Реляційна.** Даними в такій БД являється набір відносин.

Відносини (таблиці) відповідають певним умовам цілісності.

Реляційна модель даних підтримує декларативні обмеження цілісності рівня домену (типу даних), рівня відношення та рівня бази даних.

Реляційна модель є сукупністю даних, що складається з набору двовимірних таблиць.



## Структура даних у реляційній моделі даних

Реляційна модель даних передбачає структуру даних, обов'язковими об'єктами якої є:

- **відношення** – це плоска таблиця, що складається зі стовпців та рядків (неформальний термін **«таблиця»**);
- **атрибут** – це названий стовпець відносини (неформальний термін - **«стовпець»**);
- **домен** – це набір допустимих значень одного чи кількох атрибутів ( **загальна сукупність допустимих значень**);
- **кортеж** – це рядок відношення (**рядок або запис**);
- **ступінь** – визначається кількістю атрибутів, що містить відношення (**кількість стовпців**);
- **кардинальність** – це кількість кортежів, що містить відношення (**кількість рядків**);
- **первинний ключ** – атрибут або набір атрибутів, який однозначно ідентифікує кортеж даного відношення. Первинний ключ обов'язково унікальний.

Приклади реляційних СУБД:

- ✓ Oracle
- ✓ MS SQL, MYSQL
- ✓ PostgreSQL
- ✓ MS Access



# SQL & NoSQL і сучасність

## Особливості реляційних БД:

- допомагає керувати великими обсягами чітко структурованих даних;
- зазвичай представлена в табличній формі;
- в рядках вказуються записи, а стовпцях – типи даних;
- інформація вноситься до таблиці згідно з загальноприйнятим шаблоном;
- для керування даними використовується мова структурованих запитів;
- використовуються лише схеми задані наперед;
- структура даних має бути визначена заздалегідь;
- БД можна масштабувати по вертикалі (як добудовувати поверхи в будівлі) – посилюємо один сервер.

## Особливості нереляційних БД:

- інформація зберігається без чіткої структури та явного зв'язку між іншими відомостями;
- дані можуть зберігатися не тільки в табличній, а й у текстовій, графічній, відео-, аудіо-, будь-якій іншій формі;
- немає жодних обмежень ні під час зберігання, ні під час використання даних;
- висока гнучкість – можна створювати документи, заздалегідь не задаючи їхню структуру;
- структура може відрізнятися для кожного файлу, може відрізнятися синтаксис, а нові поля можна додати пізніше в процесі роботи;
- БД можна масштабувати по горизонталі (як добудовувати нові будинки в кварталі) – розділяємо на скільки завгодно серверів.

## Порівняння SQL і NoSQL БД:

- Структура та тип даних. SQL вимагає жорсткої структуризації на основі шаблонів. В NoSQL до структури не пред'являється жодних вимог.
- Масштабованість. У SQL передбачено вертикальне масштабування. У NoSQL можна використовувати як вертикальне, так і горизонтальне. Але другий варіант більш простий та практичний.
- Запити. У реляційних системах отримати дані можна за допомогою мови SQL. А ось у кожній NoSQL-базі передбачено свій алгоритм роботи.
- Надійність. SQL більш прості та зручні у подальшій роботі завдяки своїй структуризації. NoSQL має високий захист від атак хакерів.
- Робота із даними складних структур. Тут першість у реляційних БД, що також пов'язане з наявністю чіткої структури.
- Підтримка. БД SQL існують набагато довше за нереляційні аналоги, користуються підвищеною популярністю. Тобто отримати їхню підтримку досить просто.

# Коли вибрати SQL, а коли NOSQL?

**SQL** буде оптимальною для обробки великої кількості складних запитів, клопіткого, рутинного аналізу інформації. Якщо потрібна надійна, стабільна і продуктивна обробка транзакцій із збереженням посилальної цінності, варто віддати переваги SQL.

**Нереляційна база даних** – це вибір тих, хто працюватиме з великими обсягами різних даних. Тут немає чітких структурованих механізмів, завдяки чому процес завантаження та обробки відбувається максимально швидко. До того ж, такі БД набагато складніше зламати: доступ до них обмежений. Якщо вам потрібно зберігати інформацію в об'єктах JSON, якщо потрібно горизонтальне масштабування, якщо відомості знаходяться в колекціях з різними атрибутами та полями, варто зробити вибір на користь NoSQL.

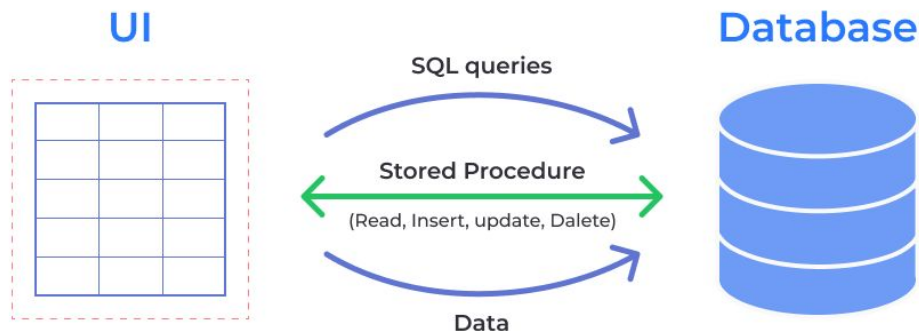
**Реляційні СУБД:** MySQL, PostgreSQL, Visual FoxPro, MS Access, Clarion, Oracle.

**Нереляційні СУБД:** CouchDB, Redis, Couchbase, MongoDB, eXist, Berkeley DB XML.

**Система управління базами даних (СУБД)** – це сукупність мовних та програмних засобів, що здійснює доступ до даних, дозволяє їх створювати, змінювати та видаляти, забезпечує безпеку даних тощо.

Іншими словами, СУБД – це система, що дозволяє створювати бази даних та маніпулювати відомостями в них. А здійснює цей доступ до даних СУБД за допомогою спеціальної мови – SQL.

**SQL (Structured Query Language)** мова структурованих запитів, основним завданням якого є надання простого способу зчитування та запису інформації до бази даних.



З точки зору реалізації, мова SQL представляє собою набір операторів, які діляться на певні групи й у кожній групі є своє призначення.

У скороченому вигляді ці групи називаються **DDL**, **DML**, **DCL** та **TCL**.

**DDL – Data Definition Language** – це група операторів визначення даних. Іншими словами, за допомогою операторів, що входять до цієї групи, можна визначити структуру бази даних, створювати об'єкти БД, змінювати та видаляти їх.

До цієї групи входять такі оператори:

- ✓ **CREATE** – використовується для створення об'єктів бази даних
- ✓ **ALTER** – використовується для зміни об'єктів бази даних
- ✓ **DROP** – використовується для видалення об'єктів бази даних





**DML – Data Manipulation Language** – це група операторів для маніпуляції даними. За допомогою цих операторів можна додавати, змінювати, видаляти та вивантажувати дані з бази, тобто маніпулювати ними.

До цієї групи входять найпоширеніші оператори мови SQL:

- ✓ **SELECT** – здійснює вибірку даних
- ✓ **INSERT** – додає нові дані
- ✓ **UPDATE** – змінює наявні дані
- ✓ **DELETE** – видаляє дані



**DCL – Data Control Language** – група операторів визначення доступу до даних. Іншими словами, це оператори для керування дозволами, за допомогою яких можна дозволяти або забороняти виконання певних операцій над об'єктами бази даних.

До цієї групи входять:

- ✓ **GRANT** – надає користувачеві або групі дозволи на певні операції з об'єктом
- ✓ **REVOKE** – відкликає видані дозволи
- ✓ **DENY** – ставить заборону, яка має пріоритет над дозволом.



**TCL – Transaction Control Language** група операторів для управління транзакціями.

**Транзакція** – це команда чи блок команд (інструкцій, послідовність операцій), які успішно завершуються як єдине ціле, при цьому в базі даних усі внесені зміни фіксуються на постійній основі. Якщо під час виконання транзакції виникла помилка, то усі зміни, внесені будь-якою командою, що входить до транзакції, буде скасовано.

До цієї групи належать такі оператори:

- ✓ **BEGIN TRANSACTION** – служить для визначення початку транзакції
- ✓ **COMMIT TRANSACTION** – застосовує транзакцію
- ✓ **ROLLBACK TRANSACTION** – скидає всі зміни, зроблені у контексті поточної транзакції
- ✓ **SAVE TRANSACTION** – встановлює проміжну точку збереження усередині транзакції

# SELECT

**SELECT \* FROM**  
**table\_name;**

**SELECT** column1,  
column2, ... **FROM**  
**table\_name;**

**SELECT DISTINCT** column1,  
column2, ... **FROM** table\_name;

**SELECT** column1,  
column2, ... **FROM**  
table\_name  
**ORDER BY** column1, column2, ...  
**ASC|DESC;**

**SELECT** – за його допомогою відбувається вибірка значень;

\* - ALL – усі значення

**FROM** – вказує, звідки необхідно отримати значення;

**DISTINCT** – вказує на те, що необхідно працювати тільки з унікальними значеннями стовпця (не враховуючи повтори);

**ORDER BY** – виконує сортування вихідних значень;

**ASC | DESC** – застосовується з оператором ORDER BY; може бути застосований як до числових так і рядкових значень; за замовчуванням сортує ASC, за зростанням/алфавітом; при використанні оператора DESC – сортує за зменшенням значень.

<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></table>	C1	C2	1	a	2	b	<div>SELECT * FROM T;</div>	<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></table>	C1	C2	1	a	2	b
C1	C2													
1	a													
2	b													
C1	C2													
1	a													
2	b													
<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></table>	C1	C2	1	a	2	b	<div>SELECT C1 FROM T;</div>	<table><tr><th>C1</th></tr><tr><td>1</td></tr><tr><td>2</td></tr></table>	C1	1	2			
C1	C2													
1	a													
2	b													
C1														
1														
2														
<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></table>	C1	C2	1	a	2	b	<div>SELECT * FROM T WHERE C1 = 1;</div>	<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr></table>	C1	C2	1	a		
C1	C2													
1	a													
2	b													
C1	C2													
1	a													
<table><tr><th>C1</th><th>C2</th></tr><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></table>	C1	C2	1	a	2	b	<div>SELECT * FROM T ORDER BY C1 DESC;</div>	<table><tr><th>C1</th><th>C2</th></tr><tr><td>2</td><td>b</td></tr><tr><td>1</td><td>a</td></tr></table>	C1	C2	2	b	1	a
C1	C2													
1	a													
2	b													
C1	C2													
2	b													
1	a													

## WHERE, CONDITIONS

```
SELECT column1, column2,  
... FROM table_name  
WHERE condition;
```

```
SELECT column1, column2,  
... FROM table_name  
WHERE condition1 AND condition2 AND  
condition3 ...;
```

```
SELECT column1, column2,  
... FROM table_name  
WHERE condition1 OR condition2 OR  
condition3 ...;
```

**WHERE** служить для задання додаткової умови вибірки, операцій вставки, редагування та видалення записів. Використовується із предикатами **AND, OR, NOT, LIKE, BETWEEN, IS, IN**; ключовим словом **NULL** та операторами порівняння та рівності ( **>, <, =** )



## Наприклад:

SQL Statement:

```
SELECT * FROM Customers  
WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

# Coffee break



**SELECT** column1, column2, ...  
**FROM** table\_name  
**WHERE** column **LIKE** pattern;

**SELECT** column\_name(s)  
**FROM** table\_name  
**WHERE** column\_name **IN** (value1, value2, ...);

**SELECT** column\_name(s)  
**FROM** table\_name  
**WHERE** column\_name **IN** (**SELECT**  
STATEMENT);

**SELECT** column\_name(s)  
**FROM** table\_name  
**WHERE** column\_name **BETWEEN** value1  
**AND** value2;

Є два символи підстановки, які часто використовуються разом з оператором **LIKE** :

- Знак відсотка (%)  
означає нуль, один або кілька символів
- Знак підкреслення (\_)  
позначає один символ





Нижче наведено кілька прикладів, які показують різні **LIKE** оператори із символами підстановки «%» та «\_»:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

#### SQL Statement:

```
SELECT * FROM Customers
WHERE ContactName LIKE 'a%o';
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

#### Result:

Number of Records: 3

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
69	Romero y tomillo	Alejandra Camino	Gran Vía, 1	Madrid	28001	Spain

# Alias

**SELECT** column\_name **AS**  
alias\_name **FROM** table\_name;

**SELECT** column\_name(s)  
**FROM** table\_name **AS**  
alias\_name;

**AS** використовується для найменування результуючої колонки;

SQL Statement:

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

Customer	Contact Person
Alfreds Futterkiste	Maria Anders
Ana Trujillo Emparedados y helados	Ana Trujillo
Antonio Moreno Taquería	Antonio Moreno
Around the Horn	Thomas Hardy
Berglunds snabbköp	Christina Berglund
Blauer See Delikatessen	Hanna Moos
Blondel père et fils	Frédérique Citeaux

## COUNT(), AVG() and SUM()

```
SELECT  
COUNT(column_name)  
FROM table_name  
WHERE condition;
```

```
SELECT  
AVG(column_name)  
FROM table_name  
WHERE condition;
```

```
SELECT  
SUM(column_name)  
FROM table_name  
WHERE condition;
```

```
SELECT  
MIN(column_name)  
FROM table_name  
WHERE condition;
```

```
SELECT  
MAX(column_name) FROM  
table_name  
WHERE condition;
```

**COUNT** – функція, що повертає кількість записів (рядків) таблиці;  
**AVG** – функція, що повертає середнє значення стовпця (застосовується тільки для числових стовпців);  
**SUM** – функція, що повертає суму значень стовпця таблиці;  
**MIN** – функція, що повертає мінімальне значення стовпця таблиці;  
**MAX** – функція, що повертає максимальне значення стовпця таблиці.

## SQL Statement:

```
SELECT COUNT(ProductID) Products  
FROM Products;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

## Result:

Number of Records: 1

Products
77

## SQL Statement:

```
SELECT AVG(Price)
FROM Products;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

## Result:

Number of Records: 1

AVG(Price)
------------

28.8663636363637
------------------

## SQL Statement:

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

Number of Records: 1

SUM(Quantity)
---------------

12743
-------

## HAVING, GROUP BY

```
SELECT column_name(s)
FROM table_name WHERE
condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
SELECT column_name(s) FROM
table_name WHERE condition
GROUP BY column_name(s) HAVING
condition
ORDER BY column_name(s);
```

**HAVING** є показником результату виконання агрегатних функцій. Агрегатною функцією в мові SQL називається функція, яка повертає якесь одне значення за набором значень стовпця. Такими функціями є: SQL COUNT(), SQL MIN(), SQL MAX(), SQL AVG(), SQL SUM().

**GROUP BY** використовується для об'єднання результатів вибірки по одному або декільком стовпцям.



## SQL Statement:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

Number of Records: 5

COUNT(CustomerID)	Country
9	Brazil
11	France
11	Germany
7	UK
13	USA



## SQL Statement:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

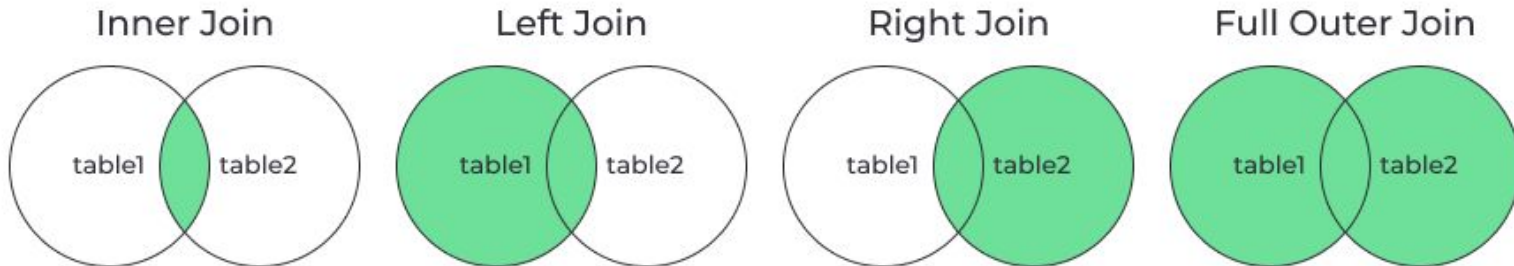
## Result:

Number of Records: 21

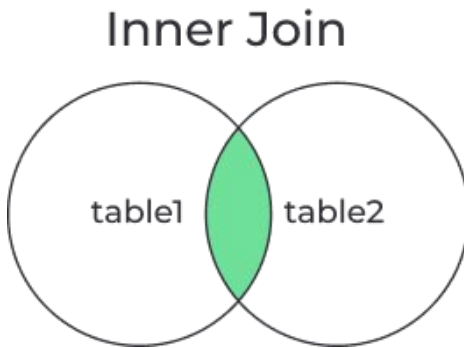
COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France
11	Germany
1	Ireland
3	Italy

# JOINS

- **(INNER) JOIN:** Повертає записи, які мають відповідні значення в обох таблицях
- **LEFT (OUTER) JOIN:** Повертає всі записи з лівої таблиці та відповідні записи з правої таблиці
- **RIGHT (OUTER) JOIN:** Повертає всі записи з правої таблиці та відповідні записи з лівої таблиці
- **FULL (OUTER) JOIN:** Повертає всі записи, якщо є відповідність у лівій і правій таблиці



**INNER JOIN** – Оператор SQL INNER JOIN формує таблицю із записів двох або кількох таблиць. Кожен рядок з першої (лівої) таблиці зіставляється з кожним рядком з другої (правої) таблиці, після чого відбувається перевірка умови. Якщо умова істинна, рядки потрапляють у результуючу таблицю. У результуючій таблиці рядки формуються конкатенацією рядків першої та другої таблиць.



Примітка. Ключове слово **INNER** є необов'язковим: воно використовується за замовчуванням, а також є операцією **JOIN**, що найчастіше використовується.

## SQL Statement:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

Number of Records: 196

OrderID	CustomerName
10248	Wilman Kala
10249	Tradição Hipermercados
10250	Hanari Carnes
10251	Victualles en stock
10252	Suprêmes délices
10253	Hanari Carnes
10254	Chop-suey Chinese
10255	Richter Supermarkt
10256	Wellington Importadora
10257	HILARIÓN-Abastos
10258	Ernst Handel
10259	Centro comercial Moctezuma

# Підзапити

Бувають питання на співбесіді “Пропишіть об’єднання таблиць, не використовуючи оператор JOIN”.

Тут допоможуть підзапити. Загального синтаксису немає. Підзапити – це звичайні запити, які розміщуються в круглих дужках. Підзапити можна використовувати різними способами та в різних місцях усередині запиту.



Найчастіше використовують зв’язку **SELECT IN SELECT**.



```
1.  SELECT column-names
2.    FROM table-name1
3.    WHERE value IN (SELECT column-name
4.                    FROM table-name2
5.                    WHERE condition)
```

Наприклад, завдання “Перелічіть продукти, кількість яких перевищує 100, не використовуючи оператор JOIN”.

Дано 2 таблиці :

ORDERITEM	PRODUCT
Id 	Id 
OrderId	ProductName
ProductId	SupplierId
UnitPrice	UnitPrice
Quantity	Package
	IsDiscontinued

Рішення буде таким:

```
SELECT ProductName
FROM Product
WHERE Id IN (SELECT ProductId
             FROM OrderItem
             WHERE Quantity > 100)
```



## UNION, UNION ALL

У мові SQL операція UNION використовується для об'єднання двох або більше запитів оператора SELECT.

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

```
SELECT column_name(s)  
FROM table1  
UNION ALL  
SELECT column_name(s)  
FROM table2;
```

Кожен оператор SELECT у UNION повинен мати однакову кількість стовпців.

UNION виконує DISTINCT в результуючому наборі, усуваючи будь-які рядки, що повторюються.

UNION ALL не видаляє дублікати, і тому він швидший, ніж UNION.

Стовпці також повинні мати однакові типи даних.

Стовпці в кожному операторі SELECT також мають бути в одному порядку.

## SQL Statement:

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

Number of Records: 94

**City**

Aachen

Albuquerque

Anchorage

Ann Arbor

Annecy

Barcelona

Barquisimeto

Bend

Bergamo

Berlin

Bern



INSERT INTO дозволяє вносити зміни до структури таблиць: додавати записи (рядки) та заповнювати їх значеннями.

SQL Statement:

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

You have made changes to the database. Rows affected: 1

SQL Statement:

```
SELECT * FROM Customers
WHERE CustomerName = 'Cardinal'
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

## UPDATE

**UPDATE** table\_name **SET** column1 = value1, column2 = value2, ...  
**WHERE** condition;

## DELETE

**DELETE FROM** table\_name **WHERE** condition;



## SQL Statement:

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Frankfurt'
WHERE CustomerID=1;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

You have made changes to the database. Rows affected: 1

## SQL Statement:

```
SELECT * FROM Customers
WHERE CustomerID = 1
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

## Result:

Number of Records: 1

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany

```
1 USE prods;  
2  
3 INSERT INTO Products(ProductName, Manufacturer, Price)  
4 VALUES ('iPhone X', 'Apple', 79000),  
5 ('Pixel 2', 'Google', 60000);  
6  
7 DELETE FROM Products  
8 WHERE ProductName='Pixel 2';  
9  
10 SELECT * FROM Products;
```

100 %



Results



Messages

	Id	ProductName	Manufacturer	Price	IsDeleted
1	1	iPhone X	Apple	79000,00	NULL
2	2	Pixel 2	Google	60000,00	1

# Any questions?



## До наступного практичного заняття всім потрібно підготуватися:

1. Завантажити та інсталювати програму DBeaver за посиланням <https://dbeaver.io/>.
2. Завантажити файл на свій ноутбук з Базою даних за [посиланням](#)



## Корисні посилання:

1. [w3schools](https://www.w3schools.com/)
2. [datacamp-sql-course](https://datacamp-sql-course.com/)



**Дякую за увагу!**

**Залишилися питання?  
Пишіть у Slack.**

