SQL. Практичний урок на базі Northwind

1. Вступ: SQL, СУБД та наш робочий інструмент

Що таке SQL та СУБД?

- **SQL** (Structured Query Language) — це не мова програмування, а **стандартизована мова запитів**. Її використовують для спілкування з базами даних: запитувати, змінювати, видаляти або додавати дані.
- **СУБД** (Система Управління Базами Даних) — це програма, яка зберігає, обробляє та керує даними. SQL — це мова, якою ми даємо цій програмі команди. Існує багато різних СУБД: MySQL, PostgreSQL, Oracle, MS SQL Server. Ми будемо працювати з **SQLite**.

Чому SQLite?

SQLite — це особлива СУБД. Вона не вимагає встановлення сервера, а зберігає всю базу даних в одному файлі на твоєму комп'ютері. Це ідеальний варіант для навчання, оскільки вона легка, швидка і не потребує складної конфігурації.

Наш інструмент: DB Browser for SQLite

Для роботи з SQLite ми будемо використовувати **DB Browser for SQLite**. Це зручний візуальний редактор, який дозволяє:

- Створювати та відкривати бази даних.
- Переглядати структуру таблиць (стовпці, типи даних).
- Писати та виконувати SQL-команди.
- Одразу бачити результат своїх запитів.

Де завантажити: https://sqlitebrowser.org/dl/

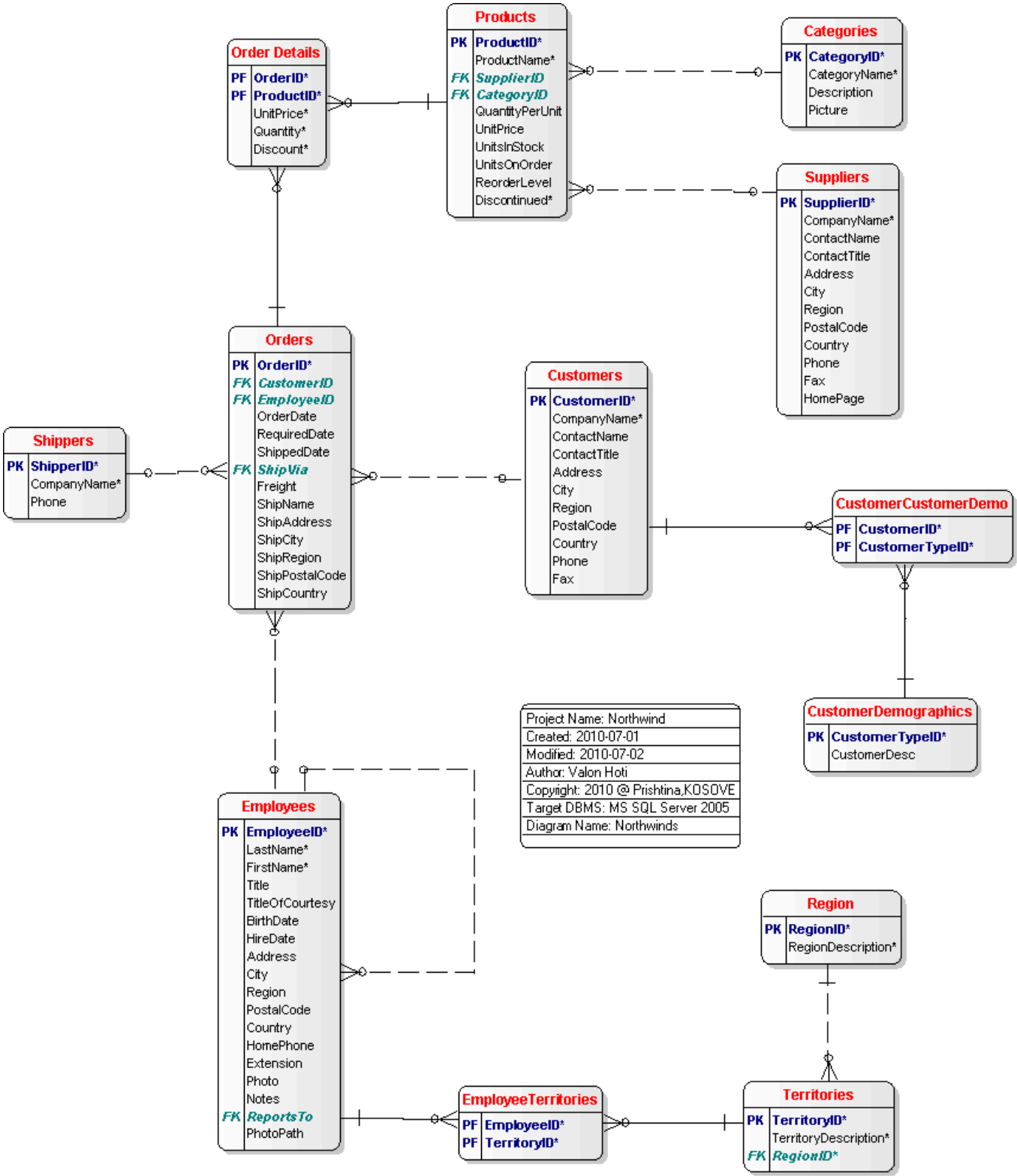
2. Наша навчальна база даних — «Північний вітер» (Northwind)

Щоб зробити навчання максимально практичним, ми будемо використовувати готову, класичну для навчання базу даних — **Northwind Traders**. Це навчальна база даних, яка моделює роботу вигаданої компанії, що займається продажем продуктів харчування. Вона містить інформацію про клієнтів, постачальників, продукти, замовлення та співробітників.

Де взяти?

- У Google ти можеш знайти файл `northwind.db` або `northwind.sqlite`. Це один і той самий файл.

- Збережи його собі на комп'ютер і відкрий за допомогою **DB Browser**.
- Коли відкриєш, на вкладці "Database Structure" ти побачиш усі таблиці. Ми будемо працювати з такими:
    - `Customers`: Клієнти.
    - `Employees`: Співробітники.
    - `Products`: Продукти.
    - `Orders`: Замовлення.
    - `Order Details`: Деталі замовлень (які саме продукти були в замовленні).
    - `Categories`: Категорії продуктів.
    - `Suppliers`: Постачальники.



Pic 1. - Northwind schema

3. `SELECT`: Основи вибірки та фільтрації даних

3.1. Базовий `SELECT`

Почнемо з найпростішого: як отримати всі дані з таблиці.

- **Вибрати всі стовпці**:

```
SELECT *
FROM Customers;
```

    - Зірочка (`*`) означає "всі стовпці".
- **Вибрати конкретні стовпці**:

```
SELECT ContactName, City, Country
FROM Customers;
```

    - Ти можеш вибрати тільки ті стовпці, які тобі потрібні.
- **Використання псевдонімів (`AS`)**:

```
SELECT ContactName AS "Ім'я контакту", City AS Місто
FROM Customers;
```

    - `AS` дозволяє дати тимчасові, більш зрозумілі імена для стовпців у результаті запиту.

3.2. Фільтрація за допомогою `WHERE`

`WHERE` — це твій інструмент для пошуку конкретних даних. Після `WHERE` іде умова, яка має бути правдивою.

- **Оператори порівняння**:

    - = (дорівнює), != (не дорівнює), > (більше), < (менше), >= (більше або дорівнює), <= (менше або дорівнює).
      Приклад: Знайти всіх клієнтів з Німеччини.

```
SELECT ContactName, City
FROM Customers
WHERE Country = 'Germany';
```

- **Приклад:** Знайти всі продукти, ціна яких менше 10.

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice < 10;
```

- **Логічні оператори**:
    - AND: обидві умови мають бути правдивими.
      Приклад: Знайти клієнтів з Німеччини, які живуть у Берліні.

```
SELECT ContactName, City
FROM Customers
WHERE Country = 'Germany' AND City = 'Berlin';
```

    - OR: хоча б одна умова має бути правдивою.
      Приклад: Знайти клієнтів з Берліна або Лондона.

```
SELECT ContactName, City
FROM Customers
WHERE City = 'Berlin' OR City = 'London';
```

- `IN` **та** `BETWEEN`:
    - IN — це зручний спосіб замінити багато OR.

Приклад: Знайти клієнтів з Німеччини, Іспанії або Мексики.

```
SELECT ContactName, Country
FROM Customers
WHERE Country IN ('Germany', 'Spain', 'Mexico');
```

-
    - BETWEEN — для діапазонів.
      Приклад: Знайти продукти, ціна яких від 18 до 20 (включно).

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice BETWEEN 18 AND 20;
```

- `LIKE` **— Пошук за шаблоном**:
    - `%` — замінює будь-яку кількість символів.
    - `_` — замінює рівно один символ.
      Приклад: Знайти всі продукти, назва яких починається на 'Ch'.

```
SELECT ProductName
FROM Products
WHERE ProductName LIKE 'Ch%';
```

Завдання для самостійного виконання (Розділ 3)

Спробуй написати запити:

1. Отримати всі замовлення, які були зроблені після 1 січня 1998 року.
2. Знайти всіх співробітників, у яких ім'я починається з 'A'.
3. Знайти всі продукти, ціна яких менше 5 або більше 50.

4. Агрегація та групування даних

Коли нам потрібно не просто отримати дані, а підсумувати їх, ми використовуємо **агрегаційні функції**.

4.1. Агрегаційні функції

- COUNT(): рахує кількість рядків.

Приклад: Скільки всього клієнтів у базі?

```
SELECT COUNT(*) AS TotalCustomers
FROM Customers;
```

- SUM(): обчислює суму значень.
  Приклад: Яка загальна вартість усіх одиниць, проданих у замовленні №10248?

```
SELECT SUM(UnitPrice * Quantity) AS TotalOrderValue
FROM "Order Details"
WHERE OrderId = 10248;
```

- AVG(): обчислює середнє значення.
  Приклад: Яка середня ціна всіх продуктів?

```
SELECT AVG(UnitPrice) AS AveragePrice
FROM Products;
```

- MIN() та MAX(): знаходять мінімальне та максимальне значення.
  Приклад: Знайти найдешевшу та найдорожчу ціну продукту.

```
SELECT MIN(UnitPrice) AS MinPrice, MAX(UnitPrice) AS MaxPrice
FROM Products;
```

4.2. `GROUP BY` — Групування

`GROUP BY` використовується разом з агрегаційними функціями, щоб застосувати обчислення не до всієї таблиці, а до окремих груп даних.

Приклад: Порахувати кількість клієнтів у кожній країні.

```sql
SELECT Country, COUNT(*) AS CustomerCount
FROM Customers
GROUP BY Country;
```

- Тут ми спочатку групуємо клієнтів за їхніми країнами, а потім для кожної групи рахуємо кількість.

4.3. `HAVING` — Фільтрація груп

`HAVING` схожий на `WHERE`, але він застосовується для фільтрації **результатів групування**, а не окремих рядків.

Приклад: Знайти країни, де кількість клієнтів більше 5.

```sql
SELECT Country, COUNT(*) AS CustomerCount
FROM Customers
GROUP BY Country
HAVING COUNT(*) > 5;
```

- Ми не можемо використовувати `WHERE` після `GROUP BY`, бо `WHERE` працює на вихідних даних, а `HAVING` — на результаті агрегації.

Завдання для самостійного виконання (Розділ 4)

1. Знайти середню ціну продуктів для кожної категорії.
2. Визначити кількість продуктів, які постачає кожен постачальник.
3. Знайти всіх співробітників, які прийняли більше 50 замовлень.

5. `JOIN`: Об'єднання таблиць

Часто дані, які нам потрібні, знаходяться в різних таблицях. `JOIN` дозволяє об'єднати їх на основі спільного зв'язку (стовпця).

5.1. `INNER JOIN`

`INNER JOIN` повертає тільки ті рядки, де є збіг у обох таблицях. Це найпоширеніший вид `JOIN`.

Приклад: Отримати назви продуктів разом з назвами їхніх категорій.

- Назви продуктів — в таблиці `Products`.
- Назви категорій — в таблиці `Categories`.

- Спільний стовпець — `CategoryID`.

```
SELECT
    P.ProductName,
    C.CategoryName
FROM
    Products AS P
INNER JOIN
    Categories AS C ON P.CategoryID = C.CategoryID;
```

- Ми використовуємо псевдоніми `P` та `C` для скорочення.
- `ON` вказує, за якою умовою таблиці з'єднуються.

5.2. `LEFT JOIN` (Додатково)

`LEFT JOIN` повертає всі рядки з першої (лівої) таблиці та збіги з другої (правої). Якщо збігів немає, він заповнює стовпці з правої таблиці значенням `NULL`.

Приклад: Отримати всіх клієнтів і, якщо є, інформацію про їхні замовлення.

- Всі клієнти є в `Customers`.
- Замовлення — в `Orders`.
- Зв'язок — `CustomerID`.

```
SELECT
    C.ContactName,
    O.OrderID
FROM
    Customers AS C
LEFT JOIN
    Orders AS O ON C.CustomerID = O.CustomerID;
```

- Цей запит покаже клієнтів, у яких не було замовлень (OrderID буде `NULL`).

Завдання для самостійного виконання (Розділ 5)

1. Отримати назви продуктів та імена їхніх постачальників.
2. Створити звіт, який показує ім'я співробітника та кількість замовлень, які він опрацював.
3. Знайти імена клієнтів та імена співробітників, які прийняли їхні замовлення.

6. Мова маніпуляції даними (DML)

У минулій частині ми вивчали команду `SELECT`, яка належить до **Мови запитів даних (DQL)**. Вона дозволяє лише читати інформацію.

Тепер ми перейдемо до **Мови маніпуляції даними (DML)**. Ці команди дозволяють вносити реальні зміни в базу: додавати нові рядки, змінювати існуючі або видаляти їх.

- `INSERT` — додавання нових даних.
- `UPDATE` — зміна існуючих даних.
- `DELETE` — видалення існуючих даних.

7. Додавання нових даних за допомогою `INSERT`

Команда `INSERT` використовується для додавання одного або кількох нових рядків до таблиці.

7.1. Додавання повного рядка

Це найпростіший спосіб, коли ти додаєш значення для кожного стовпця таблиці. Значення мають бути вказані в тому ж порядку, в якому стовпці знаходяться в таблиці.

Синтаксис:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Приклад (Northwind): Додамо нового постачальника в таблицю `Suppliers`.

```
INSERT INTO Suppliers (
    SupplierID, CompanyName, ContactName, ContactTitle,
    Address, City, PostalCode, Country, Phone
)
VALUES (
    30, 'New Supplier Corp.', 'Oleg Ivanov', 'Sales Manager',
    'Kyivska St. 1', 'Kyiv', '01001', 'Ukraine', '044-123-4567'
);
```

7.2. Додавання даних в конкретні стовпці

Якщо ти не хочеш або не можеш вказати значення для всіх стовпців, можна вказати лише ті, що потрібні. Для інших стовпців будуть використані значення за замовчуванням (`DEFAULT`) або `NULL`.

Синтаксис:

```
INSERT INTO table_name (column1, column3)
VALUES (value1, value3);
```

Приклад (Northwind): Додамо нового співробітника, вказавши лише основні дані.

```
INSERT INTO Employees (
    LastName, FirstName, Title, BirthDate, HireDate
)
VALUES (
    'Shevchenko', 'Andriy', 'Sales Representative',
    '1976-09-29', '2025-01-15'
);
```

8. Зміна існуючих даних за допомогою `UPDATE`

Команда `UPDATE` дозволяє змінити значення в одному або декількох стовпцях існуючих рядків.

Синтаксис:

```
UPDATE table_name
SET column1 = new_value1, column2 = new_value2
WHERE some_column = some_value;
```

!!! ВАЖЛИВО: КОМАНДА UPDATE БЕЗ WHERE ОНОВИТЬ УСІ РЯДКИ В ТАБЛИЦІ. !!!

Завжди використовуй WHERE, щоб вказати, які саме рядки потрібно змінити.

Приклад (Northwind): Змінимо місто клієнта "Alfreds Futterkiste".

```
UPDATE Customers
SET City = 'Berlin', Country = 'Germany'
```

```
WHERE CustomerID = 'ALFKI';
```

9. Видалення даних за допомогою `DELETE`

Команда `DELETE` видаляє один або кілька рядків з таблиці.

Синтаксис:

```
DELETE FROM table_name
WHERE some_column = some_value;
```

!!! ВАЖЛИВО: КОМАНДА DELETE БЕЗ WHERE ВИДАЛИТЬ УСІ РЯДКИ В ТАБЛИЦІ. !!!

Будь дуже обережним з цією командою.

Приклад (Northwind): Видалимо постачальника, якого ми додали раніше.

```
DELETE FROM Suppliers
WHERE SupplierID = 30;
```

Завдання для самостійного виконання

1. `INSERT`: Додай до таблиці `Categories` нову категорію з назвою "Bakery".
2. `UPDATE`: Зміни ціну продукту "Chai" (ProductID = 1) на $20.00.
3. `UPDATE`: Онови адресу постачальника "Exotic Liquids" (SupplierID = 1), змінивши його місто на 'Kyiv'.
4. `DELETE`: Видали щойно додану тобою категорію "Bakery".