



Урок 3

План заняття:

1. Введення в SQL. Що це таке? Категорії команд SQL: DDL, DML, DCL
2. Типи запитів SQL
3. Підготовка платформи
4. Запити з використанням однієї таблиці. Оператор SELECT
5. Вибірка з використанням оператора WHERE
6. Сортування даних
7. Багатотабличні запити на вибірку даних. Декартова множина значень
8. Домашнє завдання

1. Введення в SQL. Що це таке? Категорії команд SQL: DDL, DML, DCL

На минулих парах Ви навчилися створювати, проектувати та заповнювати базу даних, вивчили всі тонкощі і нюанси даного процесу. Та після створення з баз даних, а саме з її об'єктами потрібно якось працювати, а саме отримувати дані з БД і в тому вигляді, який необхідний в той чи інший момент часу. Для здійснення дій по отриманню даних використовується мова структурованих запитів **SQL**.

SQL (Structured Query Language) – це універсальна комп'ютерна мова, яка використовується для створення, модифікації і управління даними в реляційних базах даних. Вато відміти, що SQL не являється мовою програмування. Унікальність даної мови полягає в тому, що вона є єдиною стандартною мовою баз даних. Мову SQL підтримують більше сотні СУБД. І саме головне: не дивлячись на те, що кожна СУБД по-своєму інтерпретує стандарт (вносить свої доповнення), набір стандартних інструкцій SQL підтримують всі бази даних.

Щодо історії, то її початок мови SQL був закладений у **1974 році**, коли компанією IBM для експериментальної реляційної СУБД **System R** була розроблена спеціальна мова **SEQUEL (Structured English QUery Language, структурована англійська мова запитів)**, яка дозволяла дуже просто управляти даними в базі даних. Розробкою займалися Дональд Чемберлін і Рей Бойс. Ціллю розробки SEQUEL було створення мови, якою міг би користуватись звичайний користувач, який не має досвіду програмування.

Пізніше мова SEQUEL була переіменована в SQL і в 1986 р. вийшов її перший стандарт ANSI (American National Standards Institute), а в 1987 р. стандарт ISO (International Organization for Standardization, Міжнародна організація по стандартизації). Але вона не була єдиною мовою для баз даних. В той же час Каліфорнійський університет Берклі розробив власну мову **QUEL** для некомерційної реляційної СУБД **Ingres** (прародич сьогоденної PostgreSQL). Та, на жаль, вона не витримала конкуренцію з SQL.

Таким чином, в 1987 р. SQL стала міжнародною мовою баз даних (SQL-86), в 1992 р. вийшла друга розширена версія даного стандарту (ANSI SQL-92 або SQL2), а в 1999 р. третя версія (SQL:1999, SQL-99 або SQL3). Сьогодні діє стандарт, який був прийнятий в 2003 р. (SQL:2003) з невеликими змінами, які були внесені в 2006 та 2008 роках. Доречі, в новий стандарт SQL була добавлена підтримка роботи і з XML даними.

Робота з SQL нескладна. У випадку, якщо користувачу необхідно отримати дані з бази даних, він звертається з запитом до СУБД на отримання цієї інформації. В результаті, СУБД обробляє даний запит і повертає користувачу результатуочі дані. Фактично, **запит** – це команда, написана на мові SQL.

За допомогою SQL можна:

- створювати, модифікувати або видаляти об'єкти бази даних та саму базу даних;
- встановлювати зв'язки між об'єктами бази даних;
- здійснювати вибірку даних;
- вставляти, модифікувати та видаляти дані з бази даних;
- забезпечувати цілісність даних
- інше.

Вищеописані дії та ще багато інших, здійснюються за допомогою інструкцій (команд) SQL, які ми будемо з Вами вивчати на протязі наступних уроків.

Всі команди мови SQL поділяють на **чотири категорії**:

- 1) **DDL (Data Definition Language - Мова Визначення Даних)** – включає в себе інструкції, які призначені для створення, модифікації та виділення об'єктів бази даних (таблиці, представлення, індекси тощо).
- 2) **DML (Data Manipulation Language - Мова Маніпулювання Даними)** – це набір інструкцій, які призначені для роботи з даними в базі даних, тобто їх вставкою, видаленням, зміною та вибіркою.
- 3) **DCL (Data Control Language - Мова Управління Даними)** – містить набір інструкцій, які управляють правами доступу користувачів до об'єктів бази даних.
- 4) **TCL (Transaction Control Language – Мова Управління Транзакціями)**. В неї вносять набір інструкцій, що дозволяють управляти транзакціями.



А тепер від коротенької теорії перейдемо до практики. Знайомство з командами SQL розпочнемо з розгляду DML команд, а саме тих, які відповідають за вибірку даних на екран.

2. Типи запитів SQL

Основою роботи з мовою SQL являється запит. **Запит** – це команда, яка повідомляє про необхідність отримання вказаних даних. Наприклад, можна побудувати запит, який буде виводити список всіх клієнтів магазину, дані про найбільш активних покупців або сформувати алфавітний перелік товарів, що користуються найбільшим попитом чи були списані протягом певного періоду. Як правило, ця інформація отримується з основного джерела реляційної бази даних – таблиць, а результат виводиться на екран комп'ютера. Хоча її можна вивести на принтер, зберегти в файлі або іншому об'єкті бази даних тощо.

Існують наступні основні **типи запитів SQL**:

1. **Запити на вибірку.** Здійснює вибірку даних, що відповідає вказаним критеріям запиту, з однієї або декількох таблиць. Результат виконання даного запиту – набір записів, що відображаються, як правило, в режимі таблиці.
2. **Запити на зміну.** За допомогою таких запитів можна здійснювати модифікацію даних в базі даних. Існує 4 підтипи запитів на зміну:
 - **Запити на оновлення.** Дозволяють оновити дані згідно вказаних умов. Наприклад, встановити нові ціни на товари певного типу, знизивши їх на 10%, в зв'язку з сезонним розпродажем.
 - **Запити на додавання.** Дозволяють встановити нові дані в таблицю. Нові дані додаються в кінець вказаної таблиці.
 - **Запити на видалення** – це запити, що в результаті своїх дій видаляють записи, які відповідають певному критерію, з вказаної таблиці або групи таблиць. Наприклад, видалити з бази даних товари, дата поставки яких була більше року. Це може бути обумовлено тим, що ці товари попередньо були списані.
 - **Запити на створення таблиці.** В результаті роботи даних запитів, здійснюється вибірка даних з однієї або кількох таблиць і поміщаються в нову таблицю визначеної структури. Варто відмітити, що такий тип запитів підтримується не всіма СУБД і кожна з них здійснює такий запит різними засобами.

СУБД MS Access підтримує ще два типи запитів:

1. **Перехресні запити** – це запити, результат атом роботи яких є організовані в спеціальний формат дані, згруповані по двом критеріям. Як правило, такий тип запитів використовується для формування підсумкових місячних, квартальних або річних звітів по продажам або поставкам товарів.
2. **Запити з параметрами** – це спеціальний інтерактивний тип запиту, який перед виконанням виводить діалогове вікно з проханням ввести один або кілька параметрів, необхідних для його роботи. Ці параметри фактично дозволяють користувачу накладати умови відбору записів. Наприклад, відбирати товари, які були поставлені в проміжку певних дат тощо. В інших СУБД цей вид запитів представлений окремими об'єктами бази даних, таких як зберігаємі процедури та функції.

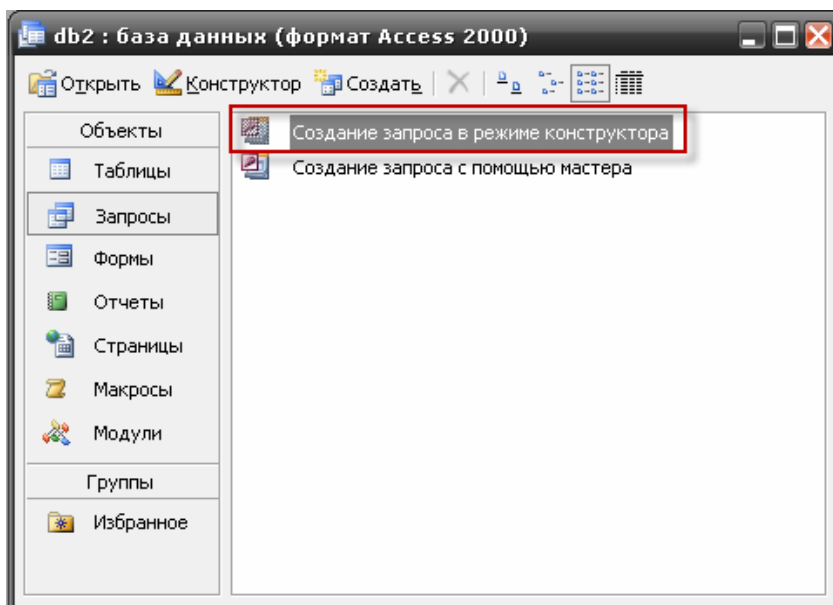
3. Підготовка платформи

В MS Access існує два **методи створення запитів**:

1. за допомогою QBE (Query By Example) - на основі шаблону;
2. за допомогою мови структурованих запитів SQL (Structure Query Language).

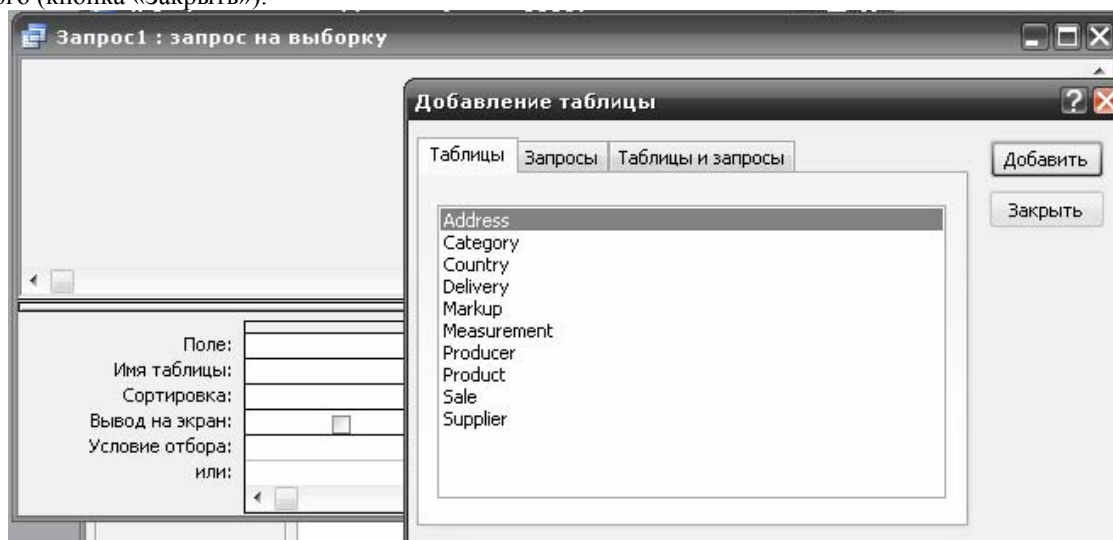
Ми будемо розглядати другий варіант створення запитів, тобто за допомогою інструкцій мови SQL, оскільки цей спосіб набагато гнучкіший і дозволить вивчити мову SQL.

Отже, для того, щоб написати запит на мові SQL необхідно обрати об'єкт «**Запити**» та створити його в довільному режимі. Найзручніший та найшвидший режим створення - за допомогою конструктора, тож його і оберемо.

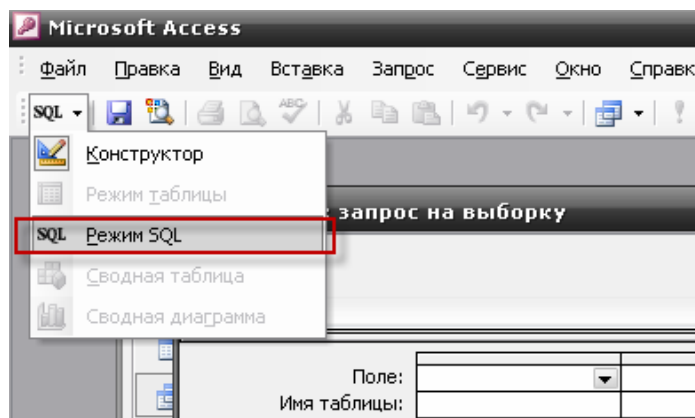




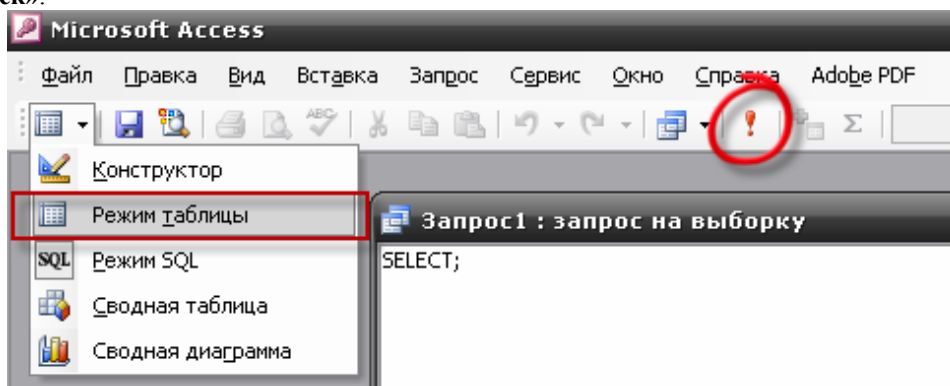
Після цього з'явиться вікно додавання таблиць, в якому Вам запропонують обрати таблиці для запиту. На основі цих таблиць буде здійснена вибірка даних з бази даних, тобто з них буде витягуватись необхідна інформація. Це потрібно лише у випадку, якщо Ви плануєте створювати запит в режимі QBE. Оскільки в нас ціль інша, ми відмовляємось від запропонованого (кнопка «Закрити»).



Наступний крок – це **перехід в режим SQL**. Для цього на панелі інструментів з випадаючого списку «Вид» обираємо потрібний нам режим.



Це дозволить відкрити вікно з редактором для SQL запитів. Коли запит написаний, щоб побачити результат його роботи, тобто запустити його на виконання, достатньо або перейти в режим таблиці, або скористатись кнопкою на панелі інструментів «Запуск».



Всі запити на отримання будь-якої кількості даних з однієї або кількох таблиць виконуються за допомогою оператора **SELECT**, який має наступний синтаксис:

Розшифруємо:

- ❑ **SELECT** – задає перелік полів, звідки необхідно вибрати дані. Оператор ALL визначений стандартом, але підтримується не всіма СУБД;
- ❑ **FROM** - з яких таблиць чи таблиці потрібно здійснити вибірку (де містяться вказані для вибірки поля);
- ❑ **WHERE** – умова на вибірку;
- ❑ **GROUP BY** – задає перелік полів, по якому необхідно групувати вибірку, щоб отримати для кожної групи єдине агреговане значення. Для цього в операторі SELECT потрібно **ОБОВ’ЯЗКОВО** використовувати SQL-функції агрегування;
- ❑ **HAVING** - накладає умову, яка дозволяє отримати в результаті лише ті групи, які задовольняють вказаному критерію;
- ❑ **ORDER BY** – дозволяє відсортувати результуючу сукупність по вказаним полям.

Крапка з комою являється обов'язковою в кінці кожного SELECT запити, але деякі СУБД дозволяють її пропускати. Також варто відмітити, що допускається вкладеність операторів SELECT. Але робити її необхідно з розумом і лише у випадку необхідності.

Для того, щоб краще розібратись з роботою даного оператора, спробуємо попрактикуватись та створити для початку **прості вибірки даних**:

1. Створити запит на вибірку всієї інформації про товари:

```
SELECT *
FROM Product;
```

2. Якщо ми хочемо отримати лише назву та ціну товару, то нам потрібно біля оператора `SELECT` написати перелік полів для вибірки:

```
SELECT Name, Price
FROM Product;
```

Результат:

	Name	Price
▶	Фарш "315км/год"	50,1
	Цукерки "Радощі у козлика"	48,6
	Цукерки "Крабiки"	29,5
	Горiлка "Чiполiно"	12,5
	Лiкер "Щасливий випадок"	120,6
	Молоко "Успевайка"	4,5
	Сухарiки "В. Барыс"	1,5

При цьому можна одночасно відформатувати вивід інформації на екран, тобто задати назви полів (псевдоніми полів) в результатуючому запиті. Для цього запит потрібно переписати наступним чином:

```
SELECT Name as [Назва товару], Price as [Ціна продажу]
FROM Product;
```

Результат:

Назва товару	Ціна продажу
Фарш "315км/год"	50,1
Цукерки "Радощі у козлика"	48,6
...ки "Крабiки"	29,5



Щоб зробити результат запиту ще більш наочнішим, використовують літерали. **Літерал** – це рядок, який записується в подвійних або одинарних лапках та включається в параметр <поле_для_вибірки>. Він виводиться в результатуючому запиті як окреме поле. В результатуючій сукупності даних літерал виступає в ролі тексту, який пояснює значення наступного за ним поля.

Синтаксис додавання літералу наступний:

```
SELECT 'літерал' [, 'літерал']
```

З використанням літералів наш запит набуде вигляду:

```
SELECT Name as [Назва товару], Price as [Ціна продажу],
       'грн.' as [Грошова одиниця]
FROM Product;
```

Результат:

Назва товару	Ціна продажу	Грошова одиниця
Фарш "315км/год"	50,1	грн.
Цукерки "Радощі у козлика"	48,6	грн.
Цукерки "Крабїки"	29,5	грн.
Горїлка "Чіполїно"	12,5	грн.
Пікєр "Пікєр" "Пікєр"	12,6	грн.

3. Також можна включати до вибірки результати розрахунків. Наприклад, додамо до нашої вибірки ціну продажу товарів, збільшену в два рази.

```
SELECT Name as [Назва товару], Price as [Ціна продажу],
       'грн.' as [Грошова одиниця],
       Price * 2 as [Подвійна ціна продажу]
FROM Product;
```

4. Для того, щоб уникнути дублювання, потрібно доповнити запит ключовим словом **DISTINCT** (різний).

```
SELECT DISTINCT Name as [Назва відомості]
FROM Product;
```

Результат:

Назва товару
Моршинська 0,5л
Хліб чорний
Банани
Банани

Назва товару
Апельсини "Накопоти"
Банани
Горїлка "Чіполїно"
Картопля "Зелене Чудо"
Ковбаса "Роспізнай"

5. До засобів форматування результатуючої вибірки можна віднести операцію конкатенації - &. За допомогою оператора конкатенації ми можемо вивести об'єднані дані кількох полів в одному. Наприклад, необхідно вивести інформацію про націнки в одному полі, тобто назва націнки та її розмір:

```
SELECT Markup.Name & ' - ' & Markup.Percent & ' %' AS [Націнка]
FROM Markup;
```

Результат:

Націнка
Базова - 15 %
Святова - 7 %
Супер-націнка - 1 %
Оптова - 5 %
Розпільна - 7 %

Здебільшого, таке форматування використовується для виведення повного імені осіб або адреси. Щоб краще продемонструвати використання оператора конкатенації, напишемо ще один приклад. Припустимо, в нашій базі даних



існує таблиця, яка зберігає інформацію про працівників магазину і необхідно вивести про них інформацію. Ім'я та прізвище повинно зберігатись в окремих полях, але для кращого візуального сприйняття їх доречно вивести в окремому полі. Такий запит набуде наступного вигляду:

```
SELECT FName&' ' &LName as [Повне ім'я працівника], Address as [Адреса]
FROM Employee;
```

4. Вибірка з використанням оператора WHERE

Прості вибірки даних рідкісне явище. Як правило, відбирається набір даних, які задовольняють певному критерію, умові або їх набору. Для цього в операторі SELECT використовується конструкція **WHERE**, після якої вказується власне сама умова відбору у вигляді виразу. В тілі умови можуть використовуватись:

- Оператори**, які дозволяють виконувати набір дій над одним або декількома компонентами виразу.
 - Арифметичні**: додавання (+), віднімання (-), множення (*), ділення (/), цілочисельного ділення одного операнда на другий (\), оператор ділення по модулю (Mod), піднесення до степеня (^). В ролі операндів можуть бути як числа, так і значення полів.
 - Порівняння**: >, <, >=, <=, =, <>. Якщо один з операндів має значення NULL, то результатом порівняння також буде NULL.
 - Логічні (булівські)**, результатом роботи яких є логічне значення True (-1), False (0) або NULL. Їх використовують для комбінування результатів виконання двох і більше операцій. Це: логічне І (And), включаюче АБО (Or), логічне заперечення (Not) та виключаюче АБО (Xor).
 - Конкатенації** - для об'єднання кількох рядків (&). Про нього ми вже говорили вище.
 - Оператори SQL**: BETWEEN..AND.., IN, LIKE, IS NULL (перевірка на рівність нулю).
- Літерали** – це значення в явному їх представленні.
 - Числові**, які можуть містити знак розділення (в десяткових числах) та знак мінус (-) для від'ємних значень, символи е або Е. Наприклад: 3,4567E-01, 12000, -25.
 - Текстові (рядкові)** – будь-які друковані символи (А-Я, 0-9, знаки пунктуації тощо). Їх слід писати в одинарних або подвійних лапках. Наприклад: “Мелодрама”, “Київ”, “Готель ”Україна” ’.
 - Дати та часу**. В більшості СУБД дата та час пишуться в одинарних лапках, але це може бути і інший довільний символ. В MS Access цим символом є символ хеш (#). Але не забувайте, що ця вимога лише для виразів SQL і при введенні значення дати або часу через користувацький інтерфейс даний символ не вказується. Наприклад: #01.03.99#, #15-янв-2001#.
- Функції** дозволяють спростити процес встановлення умови для вибірки даних. В MS Access можна використовувати як вбудовані функції, так і визначені користувачем (написані на VBA). Наведемо короткий перелік часто використовуваних вбудованих функцій:
 - Дати та часу**:
 - Date() – повертає поточну дату;
 - DateAdd(“d”, -15, [ДатаПоставки]) - повертає дату, що на 15 днів передуює даті, заданої значенням поля «ДатаПоставки»;
 - DateDiff(“d”, [ДатаПоставки],[ДатаПродажу]) – повертає значення, що є різницею значень полів «ДатаПоставки» та «ДатаПродажу»;
 - Year(#12.06.01#) - повертає рік з вказаної дати, тобто 2001
 - Рядкові**:
 - Format(Date, #dd-mm-yyuu#) – повертає відформатовану дату;
 - InStr(“Місто”, “C”) – повертає число, що вказує позицію першого входження одного рядка в інший, тобто 3;
 - LCase(“МІСТО”) – переводить рядок в нижній регістр;
 - Left([Місто],2) – відображає два перших символи значення поля «Місто»;
 - Right([Місто],3) - відображає три останніх символи значення поля «Місто»;
 - Trim([Назва]) – повертає значення поля «Назва» без пропусків (space).
 - Приведення типу даних**:
 - Val(“12.35”) – конвертує текст в число, тобто повертає 12,35;
 - Str(12,35) – конвертує число в текст, – “12,35”.

Наведемо кілька прикладів створення вибірки з умовами:

- Запит на вибірку інформації про товар з назвою «Банани», в якому крім назви товару потрібно вказати його категорію, виробника та ціну:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name = 'Банани';
```




Результат:

Запрос4 : запрос на выборку				
	Name	IdCategory	IdProducer	Price
▶	Банани	Фрукти	ЗАТ "Яблуко"	8,5
	Банани	Фрукти	Banana Republica	9
*				

Слід відмітити, що насправді замість категорії і виробника СУБД виводить їх відповідний ідентифікатор, але при роботі з MS Access ці дані приховуються. Для виведення значення по ідентифікатору використовуються багатотабличні запити, які будуть розглянуті пізніше ([п.7. Багатотабличні запити на вибірку. Декартова множина значень](#)).

6. Створити запит на вибірку інформації про товари, ціни яких знаходяться в межах від 10 до 50 грн.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price >= 10 AND Price <= 50;
```

Результат:

Запрос4 : запрос на выборку				
	Name	IdCategory	IdProducer	Price
▶	Фарш "Байкер"	Мясні	ЗАТ "ДПС"	42,00
	Фарш "315км/год"	Мясні	БАТ "Сольце України"	50,00
	Цукерки "Радощі у козлика"	Кондитерські	БАТ "Конфеті"	49,00
	Цукерки "Крабїки"	Кондитерські	ПП "Молдавські солодощі"	30,00
	Горілка "Чіполіно"	Лікєро-горілки	АТ "Русская водка"	12,00
	Дог... "Дог..."	Фрукти	С... 15	15,00

Але є і простіший спосіб написання такого роду запиту, коли критерієм відбору слугує діапазон значень. Полягає він у використанні оператора SQL **BETWEEN..AND..** З використанням даного оператора запит переписується наступним чином:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price BETWEEN 10 AND 50;
```

Результат буде аналогічний, за виключенням того, що остання включена ціна буде 49, тобто 50 не включається.

У випадку, якщо необхідно здійснити повністю протилежну операцію, тобто вивести товари, ціни яких не входять в вказаний проміжок, слід використати оператор логічного заперечення **NOT**:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price NOT BETWEEN 10 AND 50;
```

7. Оператор **BETWEEN..AND..** можна також використовувати для перевірки попадання в проміжок дат, літер тощо. **ПРИМІТКА!** Не забувайте, що дата записується в форматі #00/00/0000#. Для отримання поточної дати використовується функція **Date()**.

Розглянемо приклад: створити запит на вибірку інформації про поставку товарів в проміжку від 01/06/2009 до поточної дати.

```
SELECT IdProduct, DateDelivery
FROM Delivery
WHERE DateDelivery BETWEEN #01/06/2009# AND Date();
```

Результат:

Запрос4 : запрос на выборку	
IdProduct	DateDelivery
▶ Фарш "315км/год"	01/05/2009
Фарш "315км/год"	25/06/2009
Горілка "Чіполіно"	07/07/2008
Молоко "Успевайка"	25/06/2009
Горілка "Чіполіно"	01/07/2009
Банани	03/02/2009
Сухаріки "Дубовые дровишки"	05/06/2009
Ковбаса "Роспізнай"	17/08/2008

Запрос4 : запрос на выборку	
IdProduct	DateDelivery
▶ Фарш "315км/год"	25.06.2009
Молоко "Успевайка"	25.06.2009
Горілка "Чіполіно"	01.07.2009
Сухаріки "Дубовые дровишки"	05.06.2009



8. Створимо запит на вибірку інформації про фільми, ціна яких 10 і 50 грн.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price=10 OR Price=50;
```

Результат:

	Name	IdCategory	IdProducer	Price
▶	Фарш "315км/год"	Мясні	БАТ "Сальце України"	50,00
	Апельсини "Наколоті"	Фрукти	ЗАТ "Яблуко"	10,00

Непогано, але і цей запит можна скоротити, використавши оператор **IN**, який дозволяє перевірити належність даних поля множині визначених значень.

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price IN(10, 50);
```

9. Для пошуку значень по шаблону використовується оператор **LIKE**. Шаблон виразу може включати в себе:

- * або % - в даній позиції може бути присутній 0 або більше символів;
- ? або _ - в даній позиції обов'язково присутній один довільний символ;
- # - в даній позиції присутня одна цифра;
- [a-z] - в даній позиції обов'язково присутній один символ з вказаного діапазону;
- [abc] - в даній позиції обов'язково присутній один символ з вказаного діапазону значень;
- [!a-z] - в даній позиції обов'язково присутній один символ, що не входить в вказаний діапазон;
- [!abc] - в даній позиції обов'язково присутній один символ, що не входить в вказаний діапазон значень.

Отже, створимо запит на вибірку інформації про товари, в назві яких не менше двох літер "о":

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name LIKE '*o*o*';
```

Результат:

	Name	IdCategory	IdProducer	Price
▶	Цукерки "Радощі у козлика"	Кондитерські	БАТ "Конфеті"	49,00
	Горілка "Чіполіно"	Лікєро-горілочні	АТ "Русская водка"	12,00
	Молоко "Успєвайка"	Молочні	ПП "Корівка"	4,00
	Сухаріки "Дубовые дровишки"	Кондитерські	ПП "Молдавські солодоші"	5,00
	Апельсини "Наколоті"	Фрукти	ЗАТ "Яблуко"	10,00
	Картопля "Зелєне Чудо"	Овочі	ЗАТ "Картошка"	6,00
	Квєбєз "Пєдєвєз"	Квєбєзні	БАТ "Сєльєє Укєвєз"	51,00

10. Використовуючи маску, напишемо запит на вибірку інформації про товари, назви яких починаються з літер, які лежать в проміжку від А до К:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name BETWEEN "A%" and "K%";
```

Результат:

	Name	IdCategory	IdProducer	Price
▶	Банани	Фрукти	ЗАТ "Яблуко"	8,00
	Банани	Фрукти	Banana Republica	9,00
	Горілка "Чіполіно"	Лікєро-горілочні	АТ "Русская водка"	12,00
	Апельсини "Наколоті"	Фрукти	ЗАТ "Яблуко"	10,00

Але, результуюча вибірка не буде виводити на екран товари, які починаються з літери «К», оскільки оператор **BETWEEN..AND..** не включає кінцеве вказане значення і про це ми вже сьогодні говорили. Якщо необхідно вивести товари, які починають з літери А по К включно, тоді умову слід перебудувати:



```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Name BETWEEN "А%" and "Л%";
```

11. В SQL існує поняття NULL поля. Значення NULL не рівносильно пропуску (space) або нулю в даних числового типу. Поле даних має значення NULL, якщо йому не було присвоєно значення, тобто воно пусте. Фактично NULL являється недейсним значенням. Знайти записи, які містять NULL значення можна за допомогою ключових слів **IS NULL** або **IS NOT NULL** (в залежності від суті вибірки).

Створимо запит на вибірку інформації про товари, які мають ціну, тобто значення ціни яких не рівне NULL:

```
SELECT Name, IdCategory, IdProducer, Price
FROM Product
WHERE Price IS NOT NULL;
```

Результатом будуть записи про товари, для яких встановлена ціна.

12. Дуже часто виникає необхідність вказати кілька критеріїв відбору. В такому випадку їх можна поєднати за допомогою логічного оператора **AND** або **OR** (вибір залежить від необхідного результату). Наприклад, напишемо запит на вибірку даних про товари, ціна яких більше 40 грн. і наявна кількість знаходиться в діапазоні від 50 до 100.

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price >=40 AND Quantity BETWEEN 50 AND 100;
```

Результат:

	Name	Price	Quantity
▶	Фарш "Байкер"	42,00	50
	Ковбаса "Роспізнай"	54,00	50

Якщо замість оператора **AND** вказати оператор **OR**, тоді на екран будуть виведені дані, які задовільняють або одну, або другу умову, або ж їх поєднання.

	Name	Price	Quantity
	Фарш "Байкер"	42,00	50
	Фарш "315км/год"	50,00	45
▶	Цукерки "Радощі у козлика"	49,00	23
	Цукерки "Крабїки"	30,00	56
	Горїлка "Чіполіно"	12,00	56

5. Сортвання даних

Для сортвання результуючого набору в операторі **SELECT** використовується ключове слово **ORDER BY** з необов'язковим параметром **ASC** (по замовчуванню) – сортвання по зростанню, або **DESC** – сортвання по спаданню.

1. Відобразити інформацію про товари, відсортаних в зростаючому порядку по їх назвам:

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price >=40 AND Quantity BETWEEN 50 AND 100
ORDER BY Name ASC;
```

Результатом буде наступна відсортана сукупність:

	Name	Price	Quantity
▶	Ковбаса "Роспізнай"	54,00	50
	Фарш "Байкер"	42,00	50

2. Крім імен полів в списку при **ORDER BY** можна використовувати їх порядкові номери в списку **SELECT**.

```
SELECT Name, Price, Quantity
FROM Product
```



```
WHERE Price >=40 AND Quantity BETWEEN 50 AND 100
ORDER BY 1;
```

Результат буде аналогічний попередньому.

3. Сортувати можна і по кільком полям.

```
SELECT Name, Price, Quantity
FROM Product
WHERE Price <= 20 OR Quantity BETWEEN 50 AND 100
ORDER BY 1, 2 DESC;
```

В такому випадку вибірка буде відсортована по назві продукту в зростаючому порядку, а потім по ціні в порядку спадання.

	Name	Price	Quantity
	Апельсини "Наколоти"	10,00	10
	Банани	9,00	10
	Банани	8,00	10
	Горілка "Чіполіно"	12,00	56
	Картопля "Зелене Чудо"	6,00	100
	Ковбаса "Роспізнай"	54,00	50
	Молоко "Успевайка"	4,00	75
	Морозиво "П.Б."	2,00	5

6. Багатотабличні запити на вибірку даних. Декартова множина значень

В попередніх розділах ми виводили здійснювали вибірку даних про товари і отримували інформацію лише з однієї таблиці. Якщо нам необхідно було вивести інформацію про виробника або постачальника продукції, то ми вказували поле зовнішнього ключа і дані були виведені на екран. Побудуємо ще раз такий запит: виведемо інформацію про товар та його виробника.

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product;
```

Результат:

	Назва товару	Виробник
►	Фарш "315км/год"	БАТ "Сольце України"
	Цукерки "Радощі у козлика"	БАТ "Конфеті"
	Цукерки "Крабїки"	ПП "Молдавські солодоці"
	Горілка "Чіполіно"	АТ "Русская водка"
	Пік "Шоколадний виродок"	АТ "Русская водка"

На перший погляд жодних проблем, але насправді замість виробника СУБД виводить їх відповідний ідентифікатор, тобто значення зовнішніх ключів. MS Access ці дані приховує, але при роботі з іншими СУБД результат Вас не задовольнить. Крім того, якщо Вам необхідно буде накласти умову на вибірку по виробнику, то у Вас можуть виникнути труднощі. Наприклад, виведемо інформацію про товари виробника БАТ «Росинка»:

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product
WHERE IdProducer= 'БАТ "Росинка"';
```

Після запуску на виконання такого запиту, згенерується помилка про невідповідність типів, оскільки поле «IdProducer» являється числовим, а умова символьного типу.

Виходів з такої ситуації 2:

1. Отримати інформацію про те, яке значення первинного ключа відповідає виробнику БАТ «Росинка» і переписати запит:

```
SELECT Name as [Назва товару], IdProducer as [Виробник]
FROM Product
WHERE IdProducer= 34;
```

2. Перебудувати запит і зробити його багатотабличним.

Звичайно, що перший спосіб не дуже зручний, адже ми не можемо завжди шукати відповідні первинні ключі в зв'язаних таблицях і підставляти їх значення. Отже, розглянемо другий спосіб більш детально.



Для того, щоб створити багато табличний запит, необхідно в списку таблиць при конструкції FROM перелічити список необхідних таблиць. Після цього в списку SELECT (або в місцях де йде звернення до полів таблиць) необхідно вказати, які саме поля і з яких таблиць необхідно отримати. Доступ до полів таблиць організовується за допомогою операторів ідентифікації крапка (.) та знак оклику (!) наступним чином:

КласОб'єкта!Ім'яОб'єкта.Властивість_або_метод

Наприклад, якщо необхідно доступитись до поля **Name** таблиці **Student**, то потрібно написати наступний вираз:

Student.Name

або

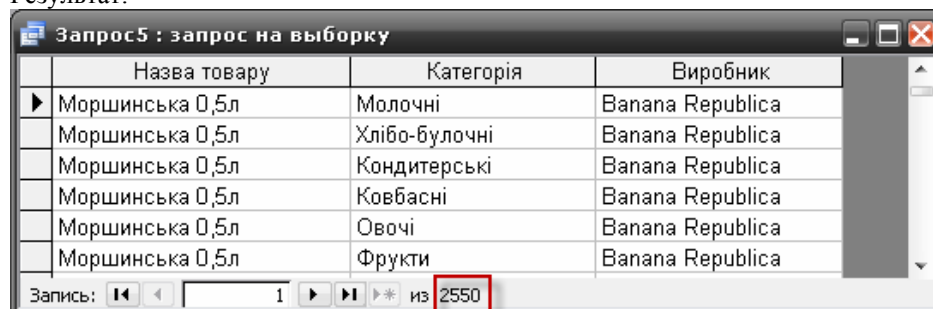
Tables!Student.Name

Якщо це зрозуміло, тоді перейдемо до практики:

1. Створити запит на вибірку даних про товари, з вказанням їх категорії та виробника:

```
SELECT Product.Name as [Назва товару], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer;
```

Результат:

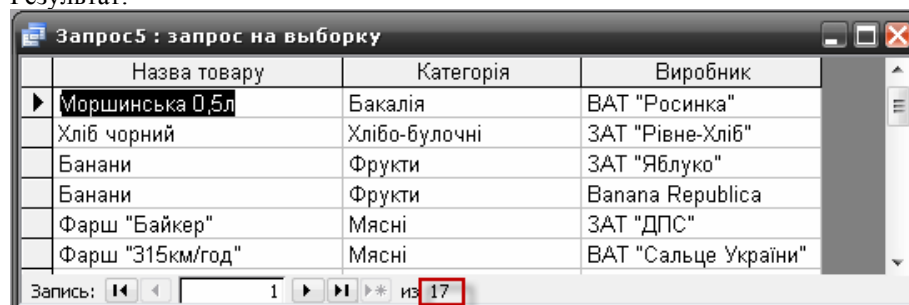


Назва товару	Категорія	Виробник
Моршинська 0,5л	Молочні	Banana Republica
Моршинська 0,5л	Хлібо-булочні	Banana Republica
Моршинська 0,5л	Кондитерські	Banana Republica
Моршинська 0,5л	Ковбасні	Banana Republica
Моршинська 0,5л	Овочі	Banana Republica
Моршинська 0,5л	Фрукти	Banana Republica

Як бачите в результаті роботи запиту утворилась кількість записів, більше існуючих. В такому випадку говорять, що утворилась «декартова множина значень», тобто всі можливі комбінації записів. Щоб такого не було, потрібно прописати використовувані таблиці зв'язані між собою в тілі оператора WHERE.

```
SELECT Product.Name as [Назва товару], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory AND Product.IdProducer=Producer.IdProducer;
```

Результат:



Назва товару	Категорія	Виробник
Моршинська 0,5л	Бакалія	БАТ "Росинка"
Хліб чорний	Хлібо-булочні	ЗАТ "Рівне-Хліб"
Банани	Фрукти	ЗАТ "Яблуко"
Банани	Фрукти	Banana Republica
Фарш "Байкер"	Мясні	ЗАТ "ДПС"
Фарш "315км/год"	Мясні	БАТ "Сольце України"

2. Тепер доповнимо наш запит умовою: відобразити лише ті товари, категорія яких «Фрукти»:

```
SELECT Product.Name as [Назва товару], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory AND Product.IdProducer=Producer.IdProducer
      AND Category.Name='Фрукти';
```



Результат:

	Назва товару	Категорія	Виробник
▶	Банани	Фрукти	ЗАТ "Яблуко"
	Банани	Фрукти	Banana Republica
	Апельсини "Наколоти"	Фрукти	ЗАТ "Яблуко"
	Яблука "Червячок"	Фрукти	ЗАТ "Яблуко"

Запись: 1 из 4

3. Відсортуємо дану вибірку по назвах виробників в зростаючому порядку:

```
SELECT Product.Name as [Назва товару], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Product, Category, Producer
WHERE Product.IdCategory=Category.IdCategory AND Product.IdProducer=Producer.IdProducer
AND Category.Name='Фрукти'
ORDER BY 3;
```

Хоча назви таблиць перед іменами полів достатньо добре інформують користувача про процес вибірки, постійно переписувати їх (інколи достатньо громіздкі назви) доволі важко. Для полегшення життя, але не розуміння, в SQL існує поняття **аліасів (Alias) або псевдонімів** імен таблиць. Псевдоніми використовуються в основному для візуального спрощення тексту запитів. Наприклад, перепишемо попередній запит з використанням псевдонімів таблиць:

```
SELECT pr.Name as [Назва товару], c.Name as [Категорія], Producer.Name as [Виробник]
FROM Product as pr, Category as c, Producer
WHERE pr.IdCategory=c.IdCategory AND pr.IdProducer=Producer.IdProducer
AND c.Name='Фрукти'
ORDER BY 3;
```

Доречі, при створенні псевдоніму допускається упускати ключове слово as:

```
SELECT pr.Name as [Назва товару], c.Name as [Категорія], Producer.Name as [Виробник]
FROM Product pr, Category c, Producer
WHERE pr.IdCategory=c.IdCategory AND pr.IdProducer=Producer.IdProducer
AND c.Name='Фрукти'
ORDER BY 3;
```

7. Домашнє завдання

- Створити запит на вибірку всієї інформації про поставки товарів в магазин.
- Вивести назви товарів, наявна кількість яких не вказана.
- Створити запит на вибірку інформації про постачальників магазину, а саме їх назви та повну адресу. Адресу вивести в одному полі (відформатувати виведення).
- Вивести інформацію про те, яких саме товарів було продано більше 10 в проміжку від 01/12/2008 до 01/03/2009
- Вивести назви товарів, що продавались на протязі останнього місяця (без повторень). Примітка! Для отримання інформації про поточний місяць слід скористатись необхідними функціями.
- Вивести список товарів, що супроводжується назвами виробників, категорія яких "Бакалія".
- Створити запит на вибірку інформації про **середню вартість продажу** на вказану в умові дату.
- Вивести список всіх постачальників, з вказанням країн їх знаходження, назви яких містять літеру К, а місто розташування починається з літери М.
- Вивести всі товари, що поставляються постачальниками ПП Петров і ПП Іванов.
- Вивести інформацію про всі товари, назви яких починаються з В по Л включно.
- Вибрати всі товари з вказанням постачальника, ім'я виробника яких починається з К або М та категорія не "Соки і води".
- Вибрати всі товари з вказанням імені та країни їх виробника. Умова: країна виробника "Україна", "Росія" або "Польща", ціна поставки були < 50 грн., а дати поставки були в межах 01/01/2006 до сьогоднішнього дня.
- Знайти товари, жанр яких "Соки і води", кількість продажу яких більше 100. Вивести такі товари, їх виробників, постачальників та категорію.
- Створити багатотабличний запит на вибірку інформації про поставку товарів в наступному вигляді: назва поставлених товарів, їх постачальників, категорії, дати поставки та загальну вартості їх поставки. Умова: лише трьох обраних постачальників. Відсортувати вибірку по назві товару в алфавітному порядку.



15. Створити багатотабличний запит на вибірку інформації про продаж товарів в наступному вигляді: назва проданих товарів, їх виробників, категорій, дати продажу, вартості продажу, з вказанням повної адреси виробників. Умова: виведена інформація не повинна стосуватись двох визначених виробників (наприклад, крім ПП Іванова і ПП Смирнова). Передбачити виведення повної адреси виробників в одному полі. Відсортувати вибірку по назвах товарів в зростаючому порядку і по їх вартості в спадаючому порядку.