



Урок 5

План заняття:

1. Підзапити
 - 1.1. Види та застосування вкладених підзапитів
 - 1.2. Оператори EXIST, ANY, SOME, ALL
2. Об'єднання запитів. Оператори UNION та UNION ALL
3. Об'єднання таблиць. Стандарт SQL2. Види об'єднань
 - 3.1. Види об'єднань
 - 3.2. Внутрішні об'єднання. Оператор INNER JOIN
 - 3.3. Зовнішні об'єднання (OUTER JOIN) та його типи: ліве, праве та повне
 - 3.4. Самооб'єднання таблиць
4. Домашнє завдання

1. Підзапити

1.1. Види та застосування вкладених підзапитів

Стандартом SQL передбачена можливість вкладати запити один в одний, що має велике практичне застосування. В результаті такого вкладення, одні запити можуть управляти іншими. При цьому вводиться поняття підзапиту. **Підзапит** – це запит, який міститься в іншому запиті SQL. Він являє собою повноцінний SELECT вираз, результат виконання якого використовується в іншому запиті. Розміщуватись вони можуть майже в довільному місці SQL виразу, наприклад, замість одного з імен в списку SELECT, в операторі FROM, при вказанні умови в операторах WHERE або HAVING. Найчастіше зустрічається використання підзапитів при вказанні умови. Слід відмітити, що самі по собі підзапити не додають жодних функціональних можливостей, але іноді з ними запити стають більш читабельнішими, ніж при складній вибірці.

При використанні підзапитів часто використовують поняття **зовнішнього** та **внутрішнього** запиту. Спочатку виконується підзапит, тобто внутрішній запит, який розміщується, наприклад, в інструкції WHERE, а потім основний, тобто зовнішній запит, який може бути інструкцією SELECT, INSERT, DELETE або UPDATE. При цьому **вкладений підзапит завжди обмежується дужками**.

Щоб краще зрозуміти як користуватись підзапитами і як вони працюють, розглянемо все на прикладі. Для початку, напишемо запит без використання підзапитів, який виводить всю інформацію про товари лише одного постачальника. За звичайних умов ми напишемо:

```
SELECT Product.*
FROM Product, Producer
WHERE Product.IdProducer = Producer.IdProducer AND Producer.Name='ЗАТ "Рівне-Хліб"';
```

Результат:

	Код	Name	IdCategory	Price	Quantity	IdProducer	IdMeasurement	IdMarkup
▶	25	Хліб чорний	Хлібо-булочні	2,00	25	ЗАТ "Рівне-Хліб"	шт.	Базова
	29	Батон свіжий	Хлібо-булочні	2,00	20	ЗАТ "Рівне-Хліб"	шт.	Базова
	30	Ватрушка	Хлібо-булочні	5,00	20	ЗАТ "Рівне-Хліб"	шт.	Базова

Запись: 1 из 3

З використанням підзапитів, наш запит переписеться наступним чином:

```
SELECT *
FROM Product
WHERE IdProducer =
    (SELECT IdProducer
     FROM Producer
     WHERE Name='ЗАТ "Рівне-Хліб"');
```

Результат буде аналогічний. Що ж відбувається при використанні підзапиту?

Спочатку цілісно виконується підзапит, який розміщується в інструкції WHERE. Результатом його виконання буде ідентифікатор, - значення поля Name рівне ЗАТ “Рівне-Хліб”.



Producer : таблиця			
	IdProducer	Name	IdAddress
+	27	ПП "Ряба курка"	Рівне
+	28	ЗАТ "Яблуко"	Омськ
+	29	ЗАТ "Картошка"	Мінськ
▶	30	БАТ "Карась"	Петрозаводск
+	31	ЗАТ "ДПС"	Рівне
+	32	ЗАТ "Рівне-Хліб"	Рівне
+	33	ЗАТ "Румянець"	Рівне
+	34	БАТ "Росинка"	Київ
+	35	Banana Republica	Вашингтон
*	(Счетчик)		

Запрос8...

IdProducer
▶ 32
* (Счетчик)

Запись: [] [] [] []

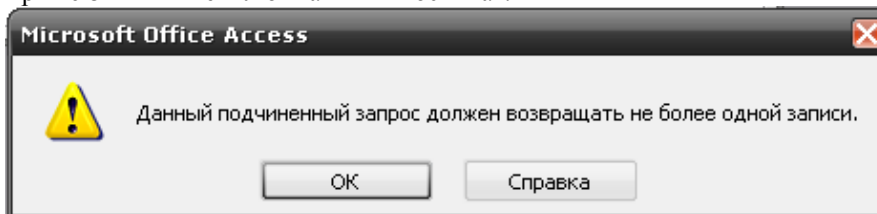
Отриманий результат повертається в основний запит і використовується при його виконанні. Тобто зовнішні ключі, які використовуються для зв'язку з таблицею Producer співставляються з первинним ключем, який є результатом виконання внутрішнього підзапиту. В результаті будуть обрані лише ті товари, які були виготовлені на ЗАТ «Рівне-Хліб».

Перш, ніж ми перейдемо до розгляду інших прикладів по використанню підзапитів, ознайомимось з обмеженнями при їх використанні. Вони наступні:

- 1) **Результатом підзапиту повинно бути лише одне значення, причому його тип даних повинен відповідати типу значення, яке використовується в зовнішньому запиті.** Наприклад, нам потрібно вивести всю інформацію про товари двох виробників: ЗАТ «Рівне-Хліб» та БАТ «Росинка».

```
SELECT *
FROM Product
WHERE IdProducer =
    (SELECT IdProducer
     FROM Producer
     WHERE Name='ЗАТ "Рівне-Хліб"' OR Name='БАТ "Росинка"');
```

Даний запит призведе до помилки, оскільки результатом даного підзапиту є кілька значень, тобто два ідентифікатори, значення полів Name яких рівне ЗАТ «Рівне-Хліб» та БАТ «Росинка».



Отже, при використанні запитів, основаних на операторах порівняння або логічних операторах, слід бути дуже уважним, щоб кінцевим результатом підзапиту був лише один запис.

Виходом з цієї ситуації є використання оператора IN, який застосовується для перебору значень результату роботи внутрішнього підзапиту:

```
SELECT *
FROM Product
WHERE IdProducer IN
    (SELECT IdProducer
     FROM Producer
     WHERE Name='ЗАТ "Рівне-Хліб"' OR Name='БАТ "Росинка"');
```

- 2) **Результатом підзапиту може бути NULL-запис.** В такому випадку результат роботи підзапиту буде оцінений як UNKNOWN, що рівносильне FALSE. В результаті попередній запит є вірним навіть без використання оператора IN, але за умови, якщо товарів одного з вказаних виробників або обох в базі даних не існує. В такому випадку підзапит на виході поверне один або жодного записів.

В де-яких випадках, для підстраховки, існує також можливість використання оператора DISTINCT для гарантованого отримання одного запису на виході підзапиту.

- 3) Згідно стандарту ANSI SQL **підзапити є непереміщувані**, тобто наступний запит являється вірним:

```
SELECT *
FROM Product
WHERE IdProducer =
    (SELECT IdProducer
     FROM Producer
     WHERE Name='ЗАТ "Рівне-Хліб"' OR Name='БАТ "Росинка"');
```



Але, якщо поміняти тіло підзапиту з порівнюваним значенням місцями, повинна згенеруватись помилка:

```
SELECT *
FROM Product
WHERE (SELECT IdProducer
      FROM Producer
      WHERE Name='ЗАТ "Півне-Хліб"' OR Name='БАТ "Росинка"') = IdProducer;
```

ПРИМІТКА! Дане правило не стосується середовища СУБД MS Access, оскільки вона розглядає і перший і другий варіант як однакові.

- 4) Вкладений запит не можна використовувати в інструкції ORDER BY.
- 5) При використанні підзапитів для перевірки результату не можна використовувати оператори BETWEEN, LIKE, IS NULL.
- 6) Якщо підзапит використовується з немодифікованим оператором порівняння, тобто звичаним знаком рівності (=), без використання ключових слів SOME, ANY або ALL, то він не може містити оператори групування GROUP BY та HAVING.

В підзапитах допускається використовувати функції агрегування, оскільки їх результатом є єдине значення. Але і тут слід бути уважним (див. п. 6 зауважень). Розглянемо кілька прикладів:

1. Запит на отримання інформації про найдорожчий товар, тобто товар з найбільшою ціною продажу:

```
SELECT DISTINCT Product.Name as [Товар], Sale.Price & ' грн.' as [Ціна]
FROM Product, Sale
WHERE Product.IdProduct = Sale.IdProduct
      AND Sale.Price = (SELECT max(Sale.Price)
                      FROM Sale);
```

Результат:

IdSale	IdProduct	Price	Quantity	DateSale
1	Фарш "315км/год"	50,10 грн.	1	12.02.2008
16	Ковбаса "Ласунка"	50,10 грн.	2	20.07.2009
3	Фарш "315км/год"	50,10 грн.	10	12.05.2009
2	Цукерки "Радощі у козлика"	50,00 грн.	2	12.02.2008
12	Цукерки "Крабїки"	15,50 грн.	0	
5	Горїлка "Чіполіно"	15,00 грн.	5	
4	Горїлка "Чіполіно"	13,00 грн.	7	
15	Апельсини "Наколоті"	7,00 грн.	1	
14	Сухаріки "Пшеничні дровишки"	5,50 грн.	5	

Товар	Ціна
Ковбаса "Ласунка"	50,1 грн.
Фарш "315км/год"	50,1 грн.

2. Запит, який виводить на екран імена та прізвища всіх постачальників, які поставляли товар в проміжку між 01/06/2009 та поточною датою:

```
SELECT Name as [Постачальник]
FROM Supplier
WHERE IdSupplier IN (SELECT IdSupplier
                    FROM Delivery
                    WHERE DateDelivery BETWEEN #01/06/2009# AND Date());
```

Результат:

IdDelivery	IdProduct	IdSupplier	Price	Quantity	DateDelivery
5	Горїлка "Чіполіно"	ТзОВ "Бистринка"	10,00р.	10	01.07.2009
4	Молоко "Успевайка"	"International Road" Co.	3,50р.	70	25.06.2009
2	Фарш "315км/год"	ПП "Швидкий вітер"	35,60р.	15	25.06.2009
7	Сухаріки "Дубовые дровишки"	ПП "Швидкий вітер"	4,00р.	15	05.06.2009
1	Фарш "315км/год"	ТзОВ "Вмерти, але доставити"	40,50р.	20	01.05.2009
6	Банани	"International Road" Co.	7,00р.	20	03.02.2009
9	Цукерки "Крабїки"	ЗАТ "Хвилинка"	10,50р.	5	01.02.2009
8	Ковбаса "Роспзнай"	ТзОВ "Бистринка"	50,00р.	1	
3	Берізка "Ласунка"	ПП "Швидкий вітер"	12,00р.	5	

Постачальник
ПП "Швидкий вітер"
"International Road" Co.
ТзОВ "Бистринка"



3. Отримати інформацію про всіх постачальників, які поставляли товар більше, ніж 2 рази. Вибірку відсортувати по назві виробника:

```
SELECT *
FROM Supplier s
WHERE 2 > ( SELECT COUNT(Delivery.IdSupplier)
            FROM Delivery
            WHERE s.IdSupplier = Delivery.IdSupplier)
ORDER BY 2;
```

4. Визначити, які поставляемі товари були вироблені в м. Київ:

```
SELECT Name as [Товар], Format(Price, '## ###.00 грн.') as [Ціна]
FROM Product
WHERE IdProduct IN ( SELECT DISTINCT IdProduct
                     FROM Delivery
                     WHERE IdSupplier IN (SELECT s.IdSupplier
                                           FROM Supplier s, Address addr
                                           WHERE s.IdAddress=addr.IdAddress
                                           AND addr.Town = 'Київ'));
```

Як вже було сказано, підзапит можна використовувати в виразі FROM зовнішнього запиту. Це дозволяє створити тимчасову таблицю і додати її в запит. Наприклад, розглянемо наступний простий запит:

```
SELECT IdProduct, Name, IdCategory
FROM Product
WHERE Price BETWEEN 20 AND 50;
```

Даний запит виведе на екран список товарів та ідентифікатори їх категорій, ціна яких знаходиться в межах 20-50 грн. Цей запит можна використати в рамках іншого, щоб отримати додаткову інформацію. Наприклад, використаємо його, щоб вивести список товарів, категорій «Мясні» та «Ковбасні», ціни яких знаходяться у вказаному діапазоні.

```
SELECT pr.Name as [Товар]
FROM (SELECT IdProduct, Name, IdCategory
      FROM Product
      WHERE Price BETWEEN 20 AND 50) as pr,
      Category as c
WHERE pr.IdCategory = c.IdCategory AND c.Name IN ('Мясні', 'Ковбасні');
```

Отже, ми використовуємо підзапит для створення таблиці, яка містить лише три поля: IdProduct, Name та IdCategory. Цій таблиці ми присвоюємо псевдонім pr. Після цього до створеної таблиці можна звернутись з запитом, як до будь-якої іншої таблиці. В даному випадку, ми використовуємо її, щоб отримати інформацію про товари необхідних категорій.

1.2. Оператори EXIST, ANY, SOME, ALL

В попередньому підрозділі ми з вами розглянули в основному однорядкові запити, тобто ті, які повертають один запис. Для того, щоб обробити кілька записів, які поверне підзапит (багаторядковий підзапит), ми використовували оператор IN. Але він не єдиний. Крім оператора IN, існує ще 4 логічних оператора: EXISTS, ALL, ANY та SOME. Розглянемо їх і почнемо з оператора EXISTS.

Оператор EXISTS використовується, коли необхідно визначити наявність значень, які відповідають умові в підзапиті. Якщо дані на виході підзапиту існують, тоді даний оператор поверне значення true, інакше – false. При використанні оператора EXISTS ми насправді використовуємо в підзапиті дані зовнішнього запиту. Такий запит іноді називають зв'язаним або корельованим підзапитом.

Розглянемо наступний запит: вивести інформацію про всіх постачальників, які коли-небудь поставляли товари в магазин:

```
SELECT *
FROM Supplier
WHERE EXISTS ( SELECT *
               FROM Delivery
               WHERE Supplier.IdSupplier = Delivery.IdSupplier)
ORDER BY 3;
```

Проаналізуємо, що вийшло в результаті. В підзапиті ми шукаємо записи таблиці **Delivery**, в яких значення **IdSupplier** співпадає з значенням **Supplier.IdSupplier**. Кожен запис таблиці **Supplier** співставляється з результатом підзапиту і **У ВИПАДКУ ІСНУВАННЯ (WHERE EXISTS)** інформація про постачальника додається в результуючу множину.



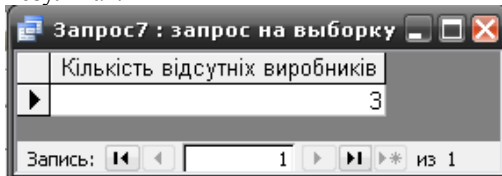
Доречі, те, що повертається підзапитом в підзапит EXISTS або NOT EXISTS абсолютно немає значення. В зв'язку з цим, інколи повертають довільне константне значення. Все, що необхідно знати, - це сам факт наявності або відсутності записів, які відповідають критерію підзапита. Наприклад:

```
SELECT *
FROM Supplier
WHERE EXISTS ( SELECT 'X'
                FROM Delivery
                WHERE Supplier.IdSupplier = Delivery.IdSupplier)
ORDER BY 3;
```

Наведемо ще один приклад на використання даного оператора. Необхідно вивести інформацію про кількість виробників, інформація про яких є в базі даних, але товарів яких в магазині немає.

```
SELECT COUNT(*) as [Кількість відсутніх виробників]
FROM Producer
WHERE NOT EXISTS ( SELECT *
                   FROM Product
                   WHERE Product.IdProducer = Producer.IdProducer);
```

Результат:



Оператори ALL, ANY та SOME використовуються для порівняння одного значення з множиною даних, які повертаються підзапитом. Їх можна комбінувати з усіма операторами порівняння і можуть включати інструкції GROUP BY та HAVING. Варто відмітити, що оператори ANY та SOME взагалі є ідентичними і в стандарті ISO вказується, що вони еквівалентні.

Розглянемо для початку дію оператора ANY.

Оператор	Значення
= ANY	Рівне довільному значенню, яке повертається під запитом. Прирівнюється до дії оператора IN і може бути ним замінений
< ANY	Менше найбільшого значення, яке повертається підзапитом. Це можна трактувати як «менше будь-якого значення».
> ANY	Більше найменшого значення, яке повертається підзапитом. Це можна трактувати як «більше будь-якого значення».

Для демонстрації роботи даного оператора перепишемо запит на отримання інформації про постачальників, які поставляли товари в магазин з використанням оператора ANY:

```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = ANY ( SELECT IdSupplier
                        FROM Delivery)
ORDER BY 2;
```

В даному випадку оператор ANY співставляє всі значення поля **IdSupplier** з таблиці **Delivery** та повертає результат true, якщо **БУДЬ_ЯКЕ (ANY)** значення співпадає з значенням поля **IdSupplier**.

Щодо оператора **SOME (який-небудь)**, то він дасть аналогічний результат.

```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = SOME ( SELECT IdSupplier
                        FROM Delivery)
ORDER BY 2;
```

Різниця між операторами ANY та SOME полягає лише в термінології та заключається в тому, щоб дозволити людям використовувати той термін, який найбільш підходить в даній ситуації.



Використання оператора **ALL** також нескладне. Даний оператор повертає значення true, якщо кожне значення, яке буде отримане в результаті роботи підзапиту, відповідає умові зовнішнього запиту. Принцип його дії відображений в наступній таблиці:

Оператор	Значення
> ALL	Більше найбільшого значення, яке повертається підзапитом. Це можна трактувати як «більше всіх значень».
< ALL	Менше найменшого значення, яке повертається підзапитом. Це можна трактувати як «менше всіх значень».

В якості прикладу виконаємо запит, в якому поставимо ціллю довести, що товари категорії “Фрукти” є найбільш продаваними.

```
SELECT Product.Name
FROM Product, Sale
WHERE Product.IdProduct = Sale.IdProduct
      AND Sale.Quantity > ALL ( SELECT Sale.Quantity
                                FROM Sale, Product, Category
                                WHERE Sale.IdProduct=Product.IdProduct
                                  AND Product.IdCategory=Category.IdCategory
                                  AND Category.Name='Фрукти' );
```

Підзапит поверне список значень кількості продаж (**Sale.Quantity**) товарів категорії “Фрукти”. Потім зовнішній запит шукає кількість продаж товарів, які були більшими, ніж дані, використовуючи для цього вираз **Sale.Quantity>ALL(кількість продаж)**. Якщо даний запит не поверне жодного запису – це буде підтвердженням того, що товари вказаної категорії є дійсно найбільш продаваними, інакше виведеться перелік товарів, які продаються більше, ніж вказаної в підзапиті категорії.

Ще один приклад. Знайдемо та виведемо назви і ціни кондитерських виробів, вартість яких перевищує вартість товарів категорії «Молочні вироби».

```
SELECT DISTINCT Product.Name as [Товар],
      Format(Product.Price, '## ###.00 грн.') as [Ціна]
FROM Product, Category
WHERE Product.Price > ALL (SELECT Product.Price
                          FROM Product, Category
                          WHERE Product.IdCategory=Category.IdCategory
                            AND Category.Name='Молочні')
      AND Product.IdCategory=Category.IdCategory AND Category.Name='Кондитерські';
```

Результат:

Product : таблиця					
	Код	Name	IdCategory	Price	Quantity
+	24	Моршинська 0,5л	Бакалія	2,00	5
+	23	Моршинська 2л	Бакалія	4,35	5
+	31	Ковбаса "Ласунка"	Ковбасні	50,00	20
+	21	Ковбаса "Роспізнай"	Ковбасні	54,00	50
+	14	Цукерки "Крабїки"	Кондитерські	30,00	56
+	13	Цукерки "Радощі у козлика"	Кондитерські	49,00	23
+	18	Сухаріки "Дубовые дровишки"	Кондитерські	5,00	15
+	16	Лікер "Щасливий випадок"	Лікери-горілки	121,00	12
+	15	Горілка "Чіполіно"	Лікери-горілки	12,00	56
+	17	Молоко "Успевайка"	Молочні	4,00	75
+	32	Йогурт "Живинка"	Молочні	7,50	10
+	28	Фарш "Байкер"	Мясні	42,00	50
+	12	Фарш "315км/гол"	Мясні		

Запрос9 : запрос на выборку	
Товар	Ціна
Цукерки "Крабїки"	30,00 грн.
Цукерки "Радощі у козлика"	49,00 грн.
Запись: 1 из 2	

2. Об'єднання запитів. Оператори UNION та UNION ALL

Об'єднання – це зв'язування даних, що містяться в двох таблицях або запитах в один результуючий набір. Оскільки об'єднання запитів та таблиць відрізняється, розглянемо окремо один і другий спосіб об'єднання. Розпочнемо з простішого, а саме з об'єднання результатів двох або більше запитів.



Об'єднання запитів здійснюється за допомогою оператора SQL **UNION**. З його допомогою можна об'єднати результати від 2 до 255 результатів запитів в один результуючий набір. В результаті такого об'єднання однакові записи по замовчуванню знищуються, але при наявності ключового слова **ALL** (тобто при використанні оператора **UNION ALL**) повертаються всі записи, в тому числі і однакові. Синтаксис оператора **UNION** наступний:

```
SELECT <список_полів>
[FROM <список_таблиць>]
[WHERE <умова>]
[GROUP BY <список_полів_для_групування>]
[HAVING <умова_на_групу>]
UNION [ALL]
SELECT <список_полів>
[FROM <список_таблиць>]
[WHERE <умова>]
[GROUP BY <список_полів_для_групування>]
[HAVING <умова_на_групу>];
[ORDER BY <умова_сортуювання>];
```

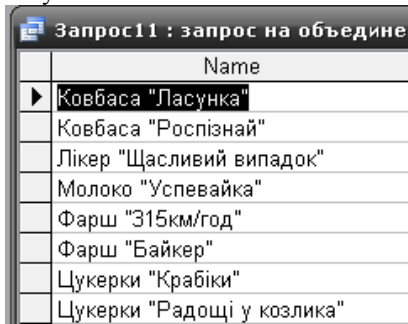
При використанні оператора **UNION** **притримуються наступних правил**:

- кількість, послідовність і типи даних полів в обох списках **SELECT** повинні співпадати;
- оператори **GROUP BY** та **HAVING** використовуються лише в одному запиті;
- жодна з таблиць не може бути відсортована окремо; можна сортувати лише результуючий запит. Тому оператор **ORDER BY** можна використовувати лише в кінці оператора **UNION**;
- імена полів результуючої вибірки визначаються списком полів першого оператора **SELECT**.

Для початку виберемо всі товари, ціна яких більше 20 грн., **АБО** код категорії товару рівний 1.

```
SELECT Name
FROM Product
WHERE Price > 20
UNION
SELECT Name
FROM Product
WHERE IdCategory=1;
```

Результат:

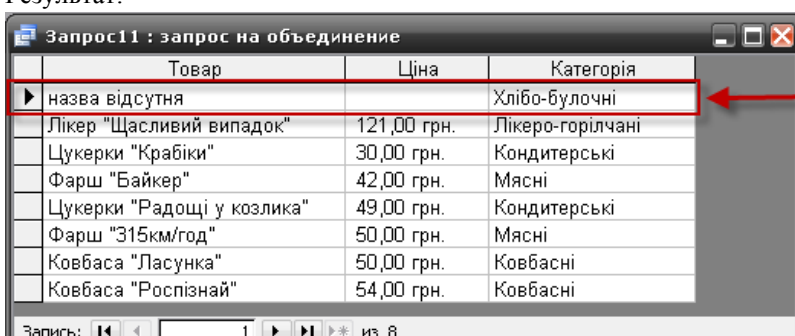


Name
Ковбаса "Ласунка"
Ковбаса "Роспізнай"
Лікер "Щасливий випадок"
Молоко "Успевайка"
Фарш "315км/год"
Фарш "Байкер"
Цукерки "Крабїки"
Цукерки "Радощі у козлика"

А тепер виберемо всі товари, ціна яких більше 20 грн., **АБО** які відносяться до категорії "Хлібо-булочні".

```
SELECT pr.Name as [Товар], Format(pr.Price, '## ###.00 грн.') as [Ціна],
       c.Name as [Категорія]
FROM Product pr, Category c
WHERE pr.IdCategory = c.IdCategory AND pr.Price > 20
UNION
SELECT 'назва відсутня', NULL, Name
FROM Category
WHERE Name = 'Хлібо-булочні';
```

Результат:



Товар	Ціна	Категорія
назва відсутня		Хлібо-булочні
Лікер "Щасливий випадок"	121,00 грн.	Лікери-горілчані
Цукерки "Крабїки"	30,00 грн.	Кондитерські
Фарш "Байкер"	42,00 грн.	Мясні
Цукерки "Радощі у козлика"	49,00 грн.	Кондитерські
Фарш "315км/год"	50,00 грн.	Мясні
Ковбаса "Ласунка"	50,00 грн.	Ковбасні
Ковбаса "Роспізнай"	54,00 грн.	Ковбасні

товари категорії "Хлібо-булочні"

категорії, ціна на товари яких > 20 грн.



3. Об'єднання таблиць. Стандарт SQL2. Види об'єднань

3.1. Види об'єднань

Однією з найсильніших сторін SQL є можливість зв'язувати дані, що розміщуються в окремих таблицях. Це зв'язування здійснюється за рахунок об'єднання таблиць, які вказуються в списку FROM. Разом з цим задається спосіб або тип об'єднання.

СУБД MS Access підтримує лише три **типи об'єднань**:

1. внутрішнє, яке іноді називають простим;
2. зовнішнє;
3. самооб'єднання.

Інші СУБД, крім вищеперелічених, підтримують і інші типи об'єднань. Причому, деякі специфічні лише для них. Але ці три типи об'єднань являються основними і підтримуються всіма.

Щоб задати спосіб об'єднання, слід скористатись одним з синтаксисів стандарту ANSI: старого стандарту SQL'86 або стандартів SQL2 та вище.

Вони виглядають наступним чином:

```
-- SQL'86
SELECT [таблиця.]поле [, ... n]
FROM таблиця [,таблиця] [, ...]
WHERE умова_об'єднання

-- SQL2 і вище
SELECT [таблиця.]поле [, ... n]
FROM таблиця [тип_об'єднання] JOIN таблиця
ON умова_об'єднання
```

Як видно з опису, синтаксис стандартів відрізняється. Причому, бачимо, що до цього часу ми користувались старим стандартом ANSI SQL, в якому немає можливості вказувати тип об'єднання.

Згідно нового стандарту тип об'єднання вказується ключовим словом при зв'язку двох таблиць. Умова об'єднання, яка тепер вказується після оператора ON являє собою вираз, аналогічний умові відбору, який використовувався в старому стандарті у виразі **WHERE**. Вона задає як будуть відноситись між собою записи в двох таблицях. Більшість операцій зв'язування виконуються на основі виразів еквівалентності, таких як **ПолеА = ПолеВ**. Однак умова об'єднання може бути і більша, при цьому всі вирази, які входять в умову об'єднуються за допомогою логічних операторів **AND** або **OR**. В цьому плані нічого не змінилось.

3.2. Внутрішні об'єднання. Оператор INNER JOIN

Перший вид об'єднання, який ми розглянемо, буде **внутрішнє об'єднання**, яке являє собою звичайне об'єднання двох або більше таблиць. Насправді, ви його використовували раніше. Старий стандарт ANSI SQL використовує внутрішній тип об'єднання для зв'язку таблиць.

В стандартах SQL2 і вище, внутрішнє об'єднання здійснюється за допомогою оператора **INNER JOIN**. Причому використання даного оператора без ключового слова **INNER** також допускається (СУБД MS Access виключення). В результаті такого об'єднання отримується нова таблиця, записи якої задовольняють відповідним умовам. Як ви вже зрозуміли, внутрішні об'єднання повертають дані, якщо знаходять спільну інформацію в обох таблицях.

Для прикладу, відобразимо черговий раз інформацію про товари та їх категорії.

```
SELECT pr.Name as [Товар], c.Name as [Категорія]
FROM Product pr, Category c
WHERE pr.IdCategory = c.IdCategory;
```

Але даний запис використовує старий стандарт ANSI SQL. Перепишемо його згідно вимог стандартів ANSI SQL2 та вище, тобто з використанням оператора **INNER JOIN**.

```
SELECT pr.Name as [Товар], c.Name as [Категорія]
FROM Product pr INNER JOIN Category c
ON pr.IdCategory = c.IdCategory;
```

Результат буде однаковий. Фактично, відмінність полягає лише в тому, що зв'язки між таблицями вказуються за допомогою оператора **INNER JOIN**, а зв'язки по ключовим полям описуються після оператора **ON**. Всі інші оператори діють так само, як і раніше.



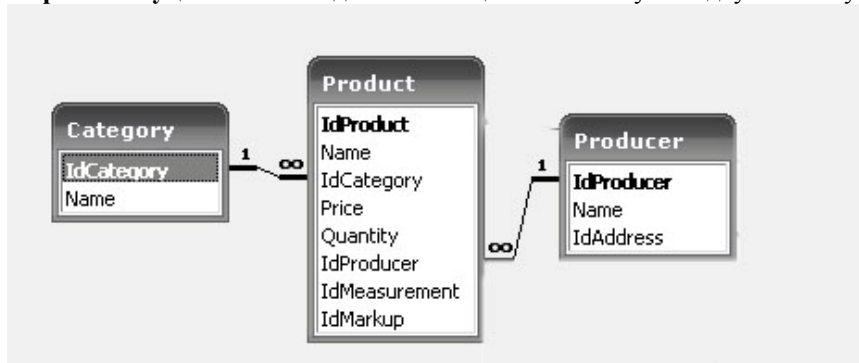
Розглянемо ще один приклад, в якому в зв'язку буде приймати участь більше двох таблиць. Додамо до нашого запиту інформацію про виробника товару. Згідно старого стандарту ANSI SQL такий запит буде виглядати так:

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

Згідно стандарту SQL2 і вище:

```
SELECT Product.Name as [Товар], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Category INNER JOIN (Product INNER JOIN
                        Producer ON Product.IdProducer=Producer.IdProducer)
      ON Product.IdCategory=Category.IdCategory;
```

Отже, при використанні нового стандарту ANSI SQL2 слід лише просто запам'ятати один принцип: **при зв'язку таблиць, вони повинні утворювати суцільний послідовний ланцюг**. В нашому випадку він наступний:



Щодо накладання умови, то тут нічого не змінилось, умова повинна розміщуватись в інструкції WHERE. Отже, накладемо умову на виведення товарів лише категорії "Бакалія":

```
SELECT Product.Name as [Товар], Category.Name as [Категорія],
       Producer.Name as [Виробник]
FROM Category INNER JOIN (Product INNER JOIN
                        Producer ON Product.IdProducer=Producer.IdProducer)
      ON Product.IdCategory=Category.IdCategory
WHERE Category.Name='Бакалія';
```

3.2. Зовнішні об'єднання (OUTER JOIN) та його типи: ліве, праве та повне

При використанні оператора INNER JOIN СУБД шукає та виводить записи, які задовольняють критеріям пошуку для двох або кількох таблиць. Але що робити у випадках, коли необхідно знайти записи однієї таблиці, відповістей яких немає в іншій?

Наприклад, потрібно відобразити інформацію про всі товари та постачальників, що їх поставляли.

```
SELECT Supplier.Name AS Постачальник, Product.Name
FROM Supplier INNER JOIN (Product INNER JOIN
                        Delivery ON Product.IdProduct = Delivery.IdProduct)
      ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Результат:

Постачальник	Name
ТзОВ "Вмерти, але доставити"	Фарш "З15км/год"
ПП "Швидкий вітер"	Фарш "З15км/год"
ПП "Швидкий вітер"	Горілка "Чіполіно"
"International Road" Co.	Молоко "Успевайка"
ТзОВ "Бистринка"	Горілка "Чіполіно"
"International Road" Co.	Банани
ПП "Швидкий вітер"	Сухаріки "Дубовые дровишки"
ТзОВ "Бистринка"	Ковбаса "Роспізнай"
ЗАТ "Хвилинка"	Цукерки "Крабіки"
ТзОВ "Хрещатик"	Картопля "Зелене Чудо"
ТзОВ "Хрещатик"	Цукерки "Крабіки"



Як видно з результатів запиту, на екрані відобразилися лише ті товари, які поставлялися і про яких існує інформація в базі даних. Щоб відобразити постачальників, інформація про яких присутня в базі даних, не залежно від того, поставляв він вже якийсь товар в магазин чи ні, неопотрібно скористатись зовнішнім об'єднанням таблиць.

Зовнішні об'єднання здійснюються за допомогою оператора OUTER JOIN та використовуються в випадку, коли потрібно, щоб запит повертав всі записи з однієї або більше таблиць, незалежно від того, чи мають вони відповідні записи в іншій таблиці. Фактично, вони дозволяють обмежити кількість повертаємих полів однієї таблиці, не обмежуючи при цьому їх для іншої таблиці.

Стандарт ANSI виділяє наступні типи зовнішніх об'єднань та оператори, що їх здійснюють:

1. **LEFT OUTER JOIN.** Ліве об'єднання – записи першої таблиці (зліва) включаються в результуючу таблицю повністю, а з другої таблиці (справа) в результат включаються лише ті, що мають пару в першій таблиці. В якості пари для записів першої таблиці, які не мають пари в іншій використовують пусті (NULL) поля.
2. **RIGHT OUTER JOIN.** Праве об'єднання – навпаки.
3. **FULL OUTER JOIN.** Повне об'єднання – включає всі співставляємі і неспівставляємі записи з обох таблиць.

Але СУБД MS Access підтримує лише два перших види зовнішніх об'єднань: ліві і праві.

Для демонстрації роботи зовнішніх об'єднань перепишемо вищезгаданий запит таким чином, щоб він виводив список постачальників, інформація про яких присутня в базі даних, не залежно від того, поставляв він вже якийсь товар в магазин чи ні:

```
SELECT Supplier.Name, Product.Name
FROM Supplier LEFT OUTER JOIN (Delivery LEFT OUTER JOIN
                                Product ON Product.IdProduct = Delivery.IdProduct)
ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Результат:

	Supplier.Name	Product.Name
▶	ПП Вася	
	ЗАТ "Хвилянка"	Цукерки "Крабикі"
	ТзОВ "Хрещатик"	Картопля "Зелене Чудо"
	ТзОВ "Хрещатик"	Цукерки "Крабикі"
	ПП Кулаков В.В.	
	ТзОВ "Вмерти, але доставити"	Фарш "315км/год"
	ПП "Швидкий вітер"	Фарш "315км/год"
	ПП "Швидкий вітер"	Голілка "Ціполіно"

В результаті запиту, якщо постачальник не поставив ще товар в магазин, йому відповідає NULL-значення в полі назви поставляемого товару.

І навпаки. Якщо необхідно відобразити повний список товарів, з інформацією про їх постачальників, незалежно від того відома про них інформація чи ні:

```
SELECT Supplier.Name AS Постачальник, Product.Name AS Товар
FROM Supplier RIGHT OUTER JOIN (Product RIGHT OUTER JOIN
                                Delivery ON Product.IdProduct = Delivery.IdProduct)
ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Такий запит дозволить одразу виявити товари, інформацію про постачальників яких не заповнили. В нашій базі даних передбачено, щоб такої ситуації не трапилось, адже наявність такої інформації є важливим.

3.3. Самооб'єднання таблиць

Аналогічно об'єднанню кількох таблиць, таблицю можна об'єднати саму з собою. Такий вид об'єднання носить назву **самооб'єднання**. Це може знадобитись, коли Вам будуть потрібні зв'язки між рядками однієї і тієї ж таблиці. Наприклад, наступний запит виведе інформацію про виробників, назви яких починаються з «ЗАТ», тобто форма відповідальності яких Закрите Акціонерне Товариство:

```
SELECT p1.Name
FROM Producer p1, Producer p2
WHERE p1.IdProducer=p2.IdProducer AND p1.Name LIKE 'ЗАТ*';
```

Або:

```
SELECT p1.Name
FROM Producer p1 INNER JOIN Producer p2
ON p1.IdProducer =p2.IdProducer
WHERE p1.Name LIKE 'ЗАТ*';
```



Результат:

Producer : таблиця			
	IdProducer	Name	IdAddress
+	26	"MicroChips" Ltd.	Вашингтон
+	35	Banana Republica	Вашингтон
+	21	АТ "Русская водка"	Москва
+	30	ВАТ "Карась"	Петрозаводськ
+	24	ВАТ "Конфеті"	Мінськ
+	34	ВАТ "Росинка"	Київ
+	22	ВАТ "Сальце України"	Рівне
+	31	ЗАТ "ДПС"	Рівне
+	29	ЗАТ "Картошка"	Мінськ
+	32	ЗАТ "Рівне-Хліб"	Рівне
+	33	ЗАТ "Румянець"	Рівне
▶	28	ЗАТ "Яблуко"	Омськ
+	23	ПП "Корівка"	Варшава

Запрос7 : запрос н...	
	Name
	ЗАТ "Яблуко"
	ЗАТ "Картошка"
	ЗАТ "ДПС"
	ЗАТ "Рівне-Хліб"
▶	ЗАТ "Румянець"

Запись: 5

В такому запиті для таблиці **Producer** ми визначили два різних псевдоніми, тобто ми повідомляємо саму СУБД, що ми хочемо мати дві різні таблиці, які повинні містити однакові дані. Після цього ми їх об'єднуємо так само, як і будь-які інші таблиці. А далі отримуємо записи, що задовольняють умову.

Спочатку це може здатись незвичним, але при роботі з кількома таблицями в запитах ідея самооб'єднання таблиць не повинна викликати великих труднощів.

4. Домашнє завдання

- Отримати інформацію про всіх виробників, товари яких продались більше, ніж 2 рази
- Вивести самий популярний товар в магазині, тобто той, який найбільше продавався.
- Якщо загальну кількість товарів всіх категорій вважати за 100%, то необхідно підрахувати скільки товарів кожної категорії (в процентному відношенні) продавалось.
- Використовуючи підзапити вивести коди та назви всіх товарів, які поставляються лише визначеним постачальником, наприклад, ЗАТ «Швидкий вітер».
- Використовуючи підзапити вивести список товарів, їх ціни та категорії, які поставляються взагалі тільки одним постачальником.
- Використовуючи підзапити вивести назви постачальників, які не поставляли вказаний товар (наприклад, «Йогурт»).
- Використовуючи підзапити вивести на екран список виробників, які розташовані в тій же країні, що і, наприклад, постачальник ПП Ваня.
- Написати запит, який виводить на екран список постачальників однієї країни (наприклад, України). Використати для виведення результату самооб'єднання.
- Підрахувати кількість постачальників, товари яких поставлялись в період з 01/03/2009 по 01/06/2009 і не були продані. Використайте для цього оператор EXISTS.
- Виведіть список виробників, які розміщуються не в Україні. Відсортуйте вибірку в зростаючому порядку назв виробників. Для даного підзапиту не використовуйте об'єднання таблиць, а лише корельовані підзапити та оператор EXISTS.
- Вивести на екран назву товару, постачальника, який його поставляв, його повну адресу (в одному полі), категорія яких «Бакалія» та «Фрукти». Врахувати при виведенні лише ті товари, які поставляються приватними підприємствами.
- Використовуючи підзапити знайти постачальників, товарів яких немає в продажу. Використайте для пошуку оператор ANY або SOME.
- Виберіть всіх виробників, товарів яких в магазині наявності більше, ніж будь-якого товару виробника ЗАТ «Ласуня».
- Вивести інформацію про те, товарів яких виробників в базі даних не існує. Для виведення повноцінної інформації скористайтесь зовнішнім об'єднанням.
- Відобразити всі товари, категорій «Кондитерські» та «Хлібо-булочні» та назви постачальників, які їх поставляли (використати оператор UNION).
- Отримати інформацію про кількість постачальників двох країн (наприклад, України і Росії), товари яких існують в базі даних. При цьому вивести окремо отриману інформацію та загальну суму всіх постачальників товарів. Скористайтесь для цього операторами UNION та UNION ALL.

Вимоги до виконання ДЗ. Всі багатотабличні запити повинні бути написані згідно стандарту SQL2.