Урок 6

План заняття:

1. Перехресні запити

Перехресні запити являються більш складною категорією запитів на вибірку. В них також використовується групування, але на цей раз двухвимірне, тобто по записам (рядкам) та по полям (стовпчикам). Іншими словами – це вибірка даних, які згруповані по двум критеріям. Як правило, такий тип запитів використовується для формування підсумкових місячних, квартальних або річних звітів по продажам або поставкам товарів.

Даний тип запитів притаманний лише MS Access та відрізняється від звичайних групових запитів тим, що в результуючій таблиці перехресного запиту не лише заголовки рядків, але і заголовки стовпчиків **визначаються на основі значень полів, а не їх назв**.

Для створення такого запиту потрібно як мінімум **три елемента**:

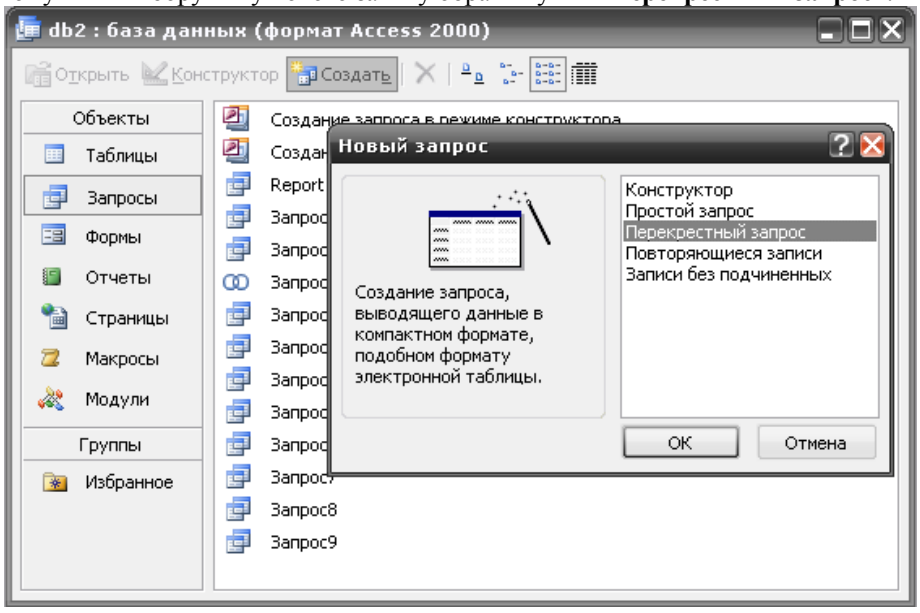
1. поле для визначення заголовків рядків;
2. поле, що визначає заголовки стовпчиків;
3. поле для вибору значень, з якими будуть безпосередньо виконуватись розрахунки.

Оскільки, для розуміння даного виду запиту його краще один раз побачити, ніж безкінечно говорити, перейдемо безпосередньо до практики. Створимо класичний перехресний запит квартальних продаж товарів за 2009 р., взявши за основу попередньо створений багатотабличний запит. **Базовий запит** (назвемо його **Report**) повинен відображати загальну вартість для кожного замовленого товару та буде мати наступний вигляд:

```
SELECT Product.IdProduct as [Код],
       Product.Name as [Товар],
       Sale.DateSale as [Дата продажу],
       Sale.Price*Sale.Quantity as [Вартість]
FROM Product INNER JOIN Sale
ON Product.IdProduct = Sale.IdProduct
WHERE Sale.DateSale BETWEEN #01/01/2009# AND Date();
```

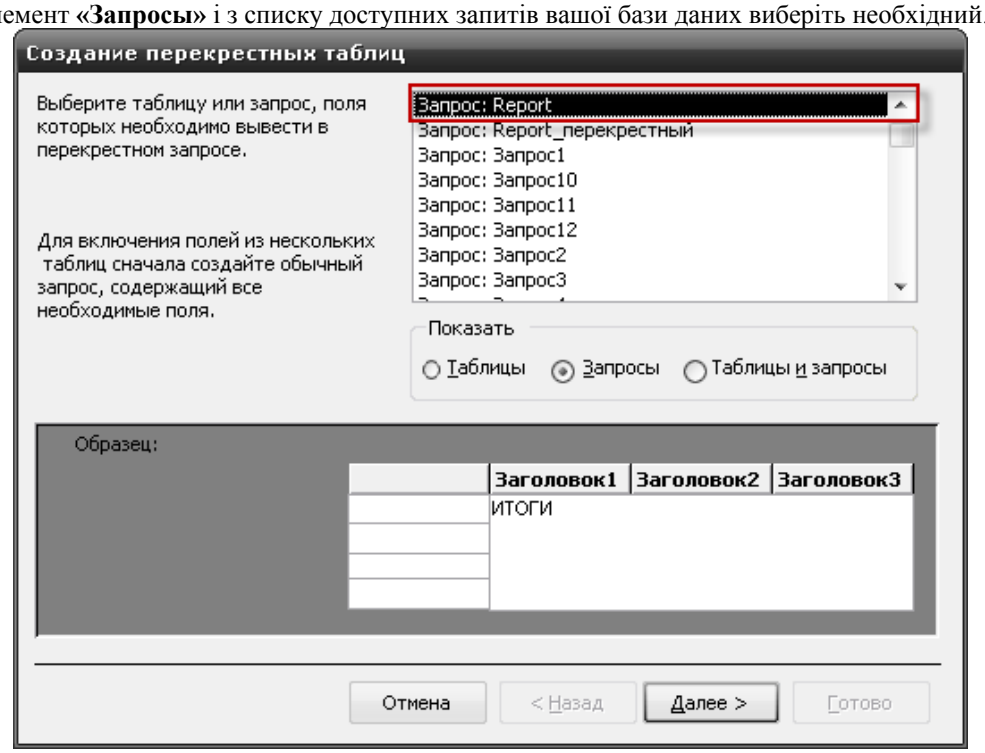
Сам перехресний запит має свої особливості синтаксису, тому для кращого розуміння, створимо його спочатку за допомогою візарда, а потім розберемо код SQL, який його створює.

Отже, для того, щоб створити перехресний запит, необхідно обрати на панелі інструментів вікна бази даних кнопку **«Создать»** і в діалоговому вікні вибору типу нового запиту обрати пункт **«Перекрестный запрос»**.

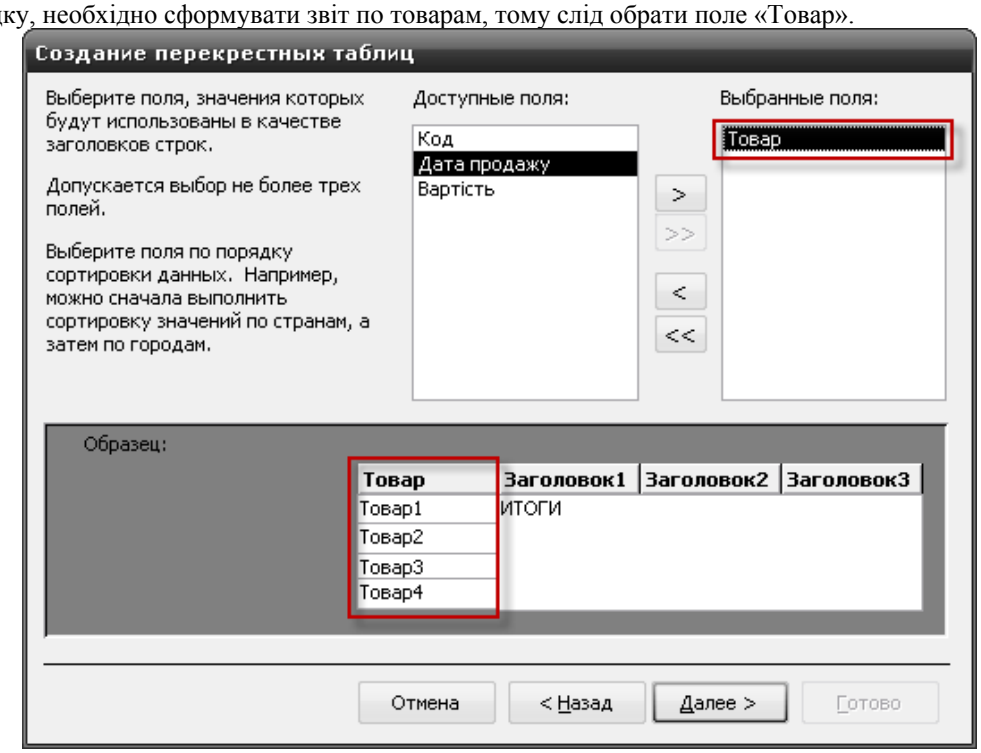


Далі візард запропонує нам здійснити наступні **кроки**:

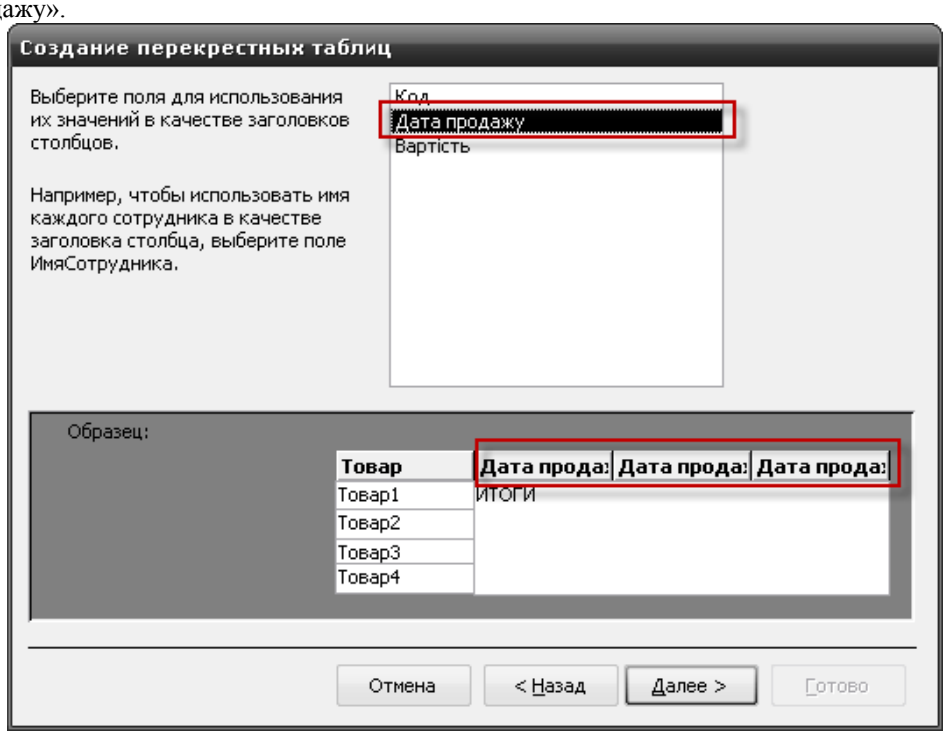
1. З самого початку потрібно обрати джерело даних для перехресного запиту. Для цього оберіть в групі опцій **«Показать»** елемент **«Запросы»** і з списку доступних запитів вашої бази даних виберіть необхідний.



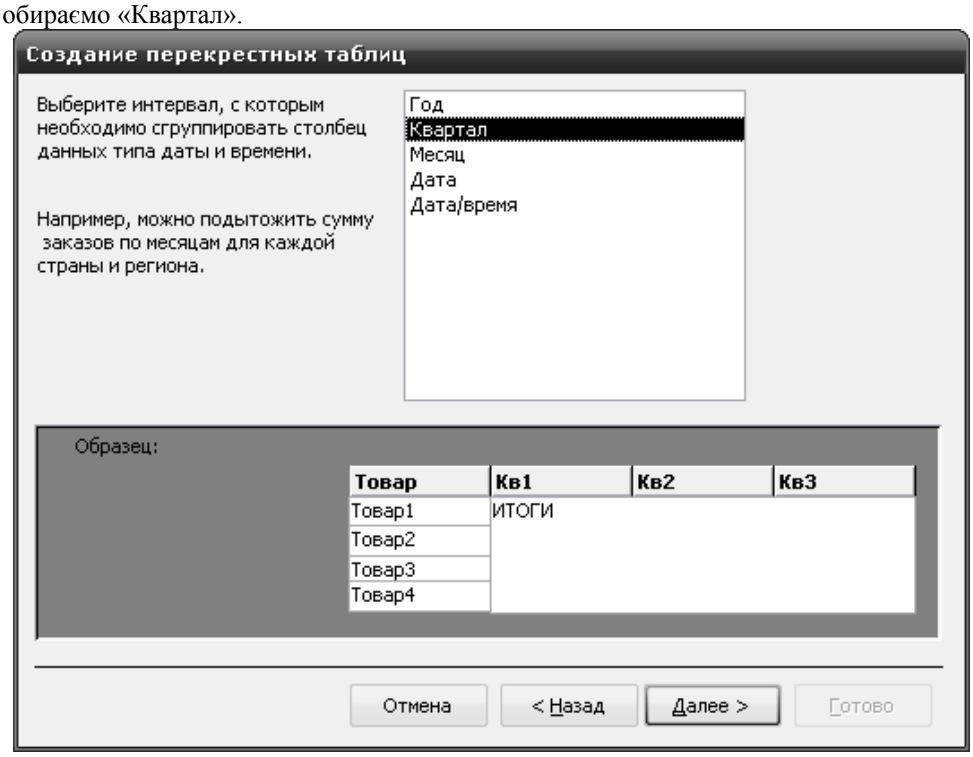
2. На наступному кроці оберіть з списку **«Доступные поля»** ті поля, значення яких будуть використовуватись як заголовки рядків перехресного запиту. Необхідні поля перенесіть за допомогою кнопок в список **«Выбранные поля»**. В нашому випадку, необхідно сформувати звіт по товарам, тому слід обрати поле «Товар».



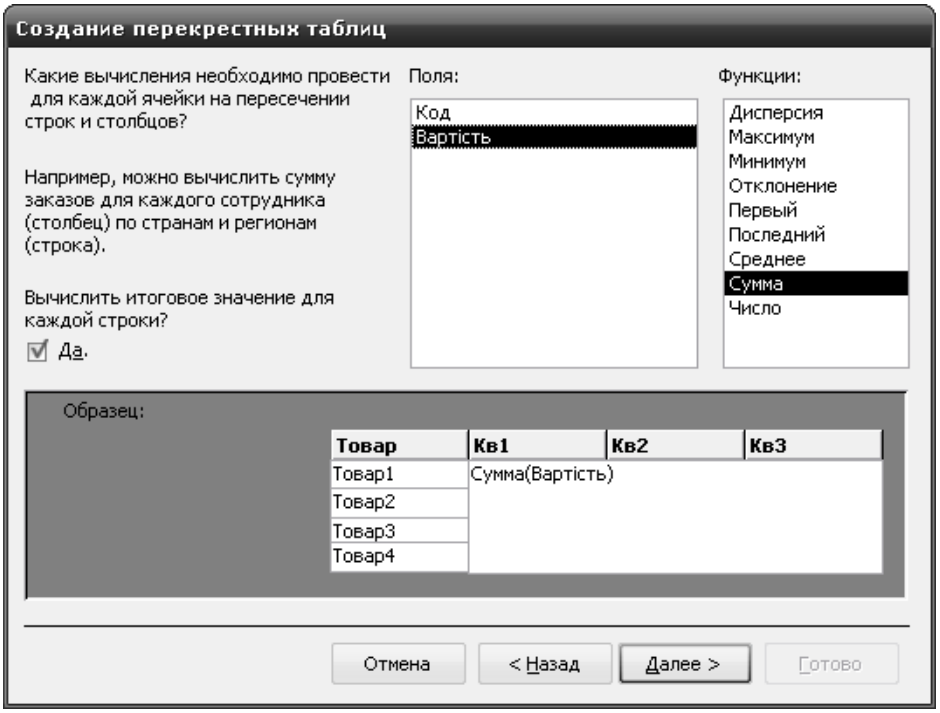
3. Далі необхідно обрати поле, значення якого буде використане в якості заголовка стовпчиків. Оскільки квартальний звіт передбачає, що підрахунок продажу товарів буде здійснюватись на кожну дату в межах кварталу, то слід обрати поле «Дата продажу».



4. Після цього потрібно обрати інтервал групування для стовпчиків, тобто для дат продажу. Оскільки звіт в нас квартальний, - обираємо «Квартал».



5. Останнім суттєвим кроком є вибір підсумкової операції, яку необхідно застосувати для обробки даних поля «Вартість» (в нашому випадку). Дана операція фактично буде відображати ту інформацію, яка нас цікавить: загальну суму продажів в розрізі кварталу, середню вартість тощо. Нас цікавить сума, тому в списку **«Функции»** слід обрати відповідний пункт.



От і все. Результат буде мати наступний вигляд:

Report_перекрестный : перекрестный запрос

Товар	Итоговое значение Вартість	Кв1	Кв2	Кв3
Апельсини "Наколоті"	7,00 грн.			7,00 грн.
Банани	46,00 грн.		46,00 грн.	
Горілка "Чіполіно"	166,00 грн.	75,00 грн.	91,00 грн.	
Ковбаса "Ласунка"	100,20 грн.			100,20 грн.
Молоко "Успевайка"	45,00 грн.	45,00 грн.		
Моршинська 0,5л	30,00 грн.	17,50 грн.	12,50 грн.	
Фарш "315км/год"	501,00 грн.		501,00 грн.	
Цукерки "Крабіки"	77,50 грн.		77,50 грн.	

Запись: 1 из 8

Як видно з результату, в полі «Товар» відображається список товарів, які були продані протягом 2009 року. Поле «Итоговое значение Вартість» (його можна потім переіменувати для зручності) містить дані про те, на яку суму за цей період кожного товару було продано, а в сусідніх полях («Кв1», «Кв2», «Кв3») ця інформація показується в розрізі кварталів.

Створення перехресного запиту за допомогою візарда, крім основної своєї переваги, - наочності та зручності, має більш суттєві **недоліки**. До них слід **віднести**:

- працювати можна лише з одним звітом чи таблицею;
- неможна сортувати результуючу таблицю по значенням, які містяться в полях, оскільки в більшості випадків одночасне впорядкування даних в полях по всім записам неможливе. Але ви можете задати сортування для заголовків записів (рядків);
- в процесі створення запиту неможна вказувати додаткові умови відбору.

В режимі SQL даний запит матиме наступний вигляд:

```
TRANSFORM Sum(Report.Вартість) AS [Sum-Вартість]
SELECT Report.Товар, Sum(Report.Вартість) AS [Итоговое значение Вартість]
FROM Report
GROUP BY Report.Товар
PIVOT "Кв" & Format([Дата продажу],"q");
```

Зверніть увагу на формування самого запиту та використання ключових слів **TRANSFORM** і **PIVOT**, які, доречі, не являються зарезервованими в стандарті ANSI SQL. Операція TRANSFORM в даному запиті дозволяє визначити дані, які містяться в результуючій таблиці, а в операції PIVOT задаються заголовки стовпчиків.

Повний синтаксис оператора **TRANSFORM**:

```
TRANSFORM функція_агрегування
SELECT вибірка
PIVOT поля_для_заголовків [IN (значення1 [, значення2 [,...]])];
```

Функція агрегування біля оператора **TRANSFORM** дозволяє обробити відібрані дані. Оператор SELECT, який вказується наступним, вказує поля, які будуть використовуватись в якості назв записів (рядків). Він **обов'язково** повинен містити вираз GROUP BY, який вказує на спосіб групування записів. Крім виразу GROUP BY, оператор SELECT повноцінно може містити і інші конструкції (WHERE, ORDER BY тощо) та підзапити.

Біля оператора **PIVOT**, як вже було сказано, вказуються імена полів або вирази, які будуть використовуватись в якості заголовків стовпчиків в результуючій таблиці. Наприклад, якщо вказати назви місяців, то результуючий запит буде містити дванадцять стовпчиків. Цей список полів можна обмежити, створивши заголовки з набору констант, які перераховуються в виразі IN. Іншими словами, значення, які не входять в набір, будуть відфільтровані.

Підсумовуючи все вищесказане, можна написати перехресний запит на формування квартального звіту по продажам товарів, без використання проміжного запиту. При цьому, можна відсортувати нашу вибірку.

```
TRANSFORM Sum(Sale.Price*Sale.Quantity) as [Вартість]
SELECT Product.Name as [Товар],
       Sum(Sale.Price*Sale.Quantity) as [Итоговое значение Вартість]
FROM Product INNER JOIN Sale
ON Product.IdProduct = Sale.IdProduct
WHERE Sale.DateSale BETWEEN #01/01/2009# AND Date()
GROUP BY Product.Name
ORDER BY Product.Name
PIVOT "Кв" & Format(Sale.DateSale,"q");
```

2. Запити з параметрами

Бувають випадки, коли значення критеріїв запиту невідомі наперед, тобто не являються константними. Наприклад, необхідно виводити список поставлених або проданих товарів на певну дату, але ця дата кожен раз при формуванні звіту інша. В такому випадку, ми можемо постійно писати новий запит, що не досить зручно, або ж створити один **параметризований запит**, який буде запитувати користувача значення необхідного критерію (-ів), в залежності від якого (-их) буде виведений результат.

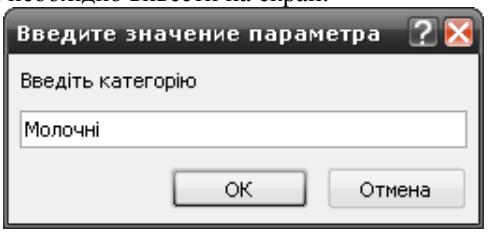
Отже, **параметризований запит** або **запит з параметрами** – це спеціальний інтерактивний тип запиту, який перед виконанням виводить діалогове вікно з проханням ввести один або кілька параметрів, необхідних для його роботи. Ці параметри фактично дозволяють користувачу накладати умови відбору записів.

Для прикладу, напишемо запит, який виводить на екран список товарів, необхідної користувачу категорії:

```
SELECT Product.Name as [Товар], Category.Name as [Категорія]
FROM Category INNER JOIN Product
ON Category.IdCategory = Product.IdCategory
WHERE Category.Name=[Введіть категорію];
```

Зверніть увагу на умову: Category.Name=[Введіть категорію]. Для того, щоб вказати, що значення критерію відбору буде отримане від користувача, в конструкції WHERE замість конкретного значення вказуються **квадратні дужки**. Якщо всередині них вказати текст, то він буде підказкою користувачу на те, що необхідно ввести.

Отже, після запуску даного запиту, перед вами з'явиться діалогове вікно з проханням ввести категорію, список товарів якої необхідно вивести на екран.



Після натиснення на кнопку OK, буде сформований список товарів введеної категорії, тобто молочні продукти.

Ще один приклад. Необхідно відібрати товари, які були поставлені в проміжку певних дат.

```
SELECT Product.Name as [Товар], Supplier.Name as [Постачальник],
       Delivery.DateDelivery as [Дата постаки]
FROM Supplier INNER JOIN
     (Product INNER JOIN Delivery ON Product.IdProduct = Delivery.IdProduct)
ON Supplier.IdSupplier = Delivery.IdSupplier
WHERE Delivery.DateDelivery BETWEEN [Початкова дата] AND [Кінцева дата];
```

Слід відмітити, що параметризовані запити такого вигляду притаманні лише СУБД MS Access. В більшості інших СУБД цей вид запитів представлений окремими об'єктами бази даних, такими як зберігаємі процедури та функції.

3. Запити на створення таблиці. Конструкція TOP

В результаті роботи даних запитів, здійснюється вибірка даних з однієї або кількох таблиць і її результат поміщається в нову таблицю. Як правило, запити на створення нової таблиці використовуються для створення архівних копій таблиць, копій для експорта в іншу базу даних або ж як джерело даних для звіту. Наприклад, звіт про помісячні продажі по регіонах можна зробити на основі одного і того ж запиту.

Для створення такого виду запитів, в синтаксисі оператора SELECT існує опція **INTO**. Дана опція вказується після списку SELECT, але перед вказанням списку таблиць, з яких буде отримана інформація (конструкцією FROM):

```
SELECT  [ALL | DISTINCT] { * | поле_для_вибірки [, поле_для_вибірки] }
[INTO ім'я_нової_таблиці
[IN зовнішня_база_даних] ]
FROM    { таблиця | представлення} [as] [псевдонім]
        [, {таблиця | представлення} [as] [псевдонім] ]

[WHERE  умова]
[GROUP BY [ALL] вираз_групування]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [ ASC | DESC ] ];
```

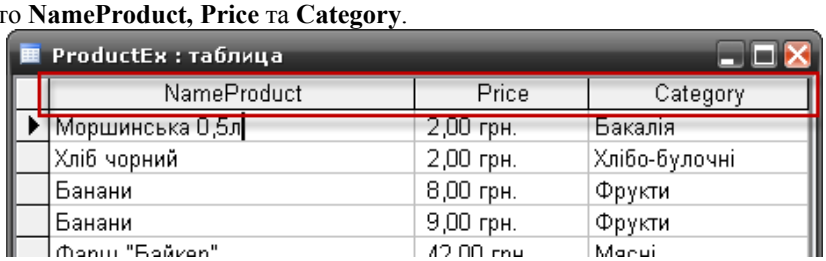
Необов'язкова опція IN дозволяє вказати базу даних, в яку буде поміщена новостворена таблиця. По замовчуванню, такою базою являється поточна, тобто нова таблиця буде збережена в нашій базі даних.

Структура нової таблиці буде відповідати структурі списку SELECT, і тому для уникнення проблем щодо ідентифікації полів новоствореної таблиці бажано вказувати псевдоніми. Крім того, варто відмітити, що при створенні нової таблиці її поля успадковують лише тип і розмір джерела, інші ж властивості (індекси, первинні ключі тощо) потрібно створювати самостійно.

Отже, створимо запит з ціллю створення **нової таблиці**, що містить інформацію про назви товарів, їх ціни та категорії.

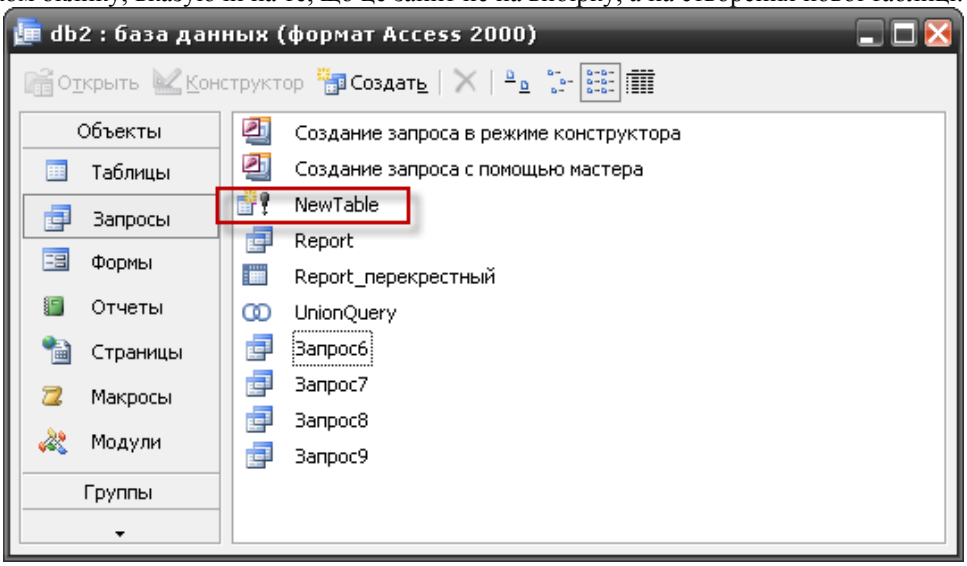
```
SELECT Product.Name as [NameProduct],
       Format(Product.Price, '## ###.00 грн.') as [Price],
       Category.Name as [Category]
INTO ProductEx
FROM Product INNER JOIN Category
ON Product.IdCategory = Category.IdCategory;
```

В результаті роботи даного запиту утвориться нова таблиця **«ProductEx»**, що містить вищезазначені дані (назву товару, його ціну та категорію). Імена полів новоствореної таблиці носять назву псевдонімів полів при написанні запиту, тобто **NameProduct, Price** та **Category**.



До новоствореної таблиці ProductEx ви можете написати довільний запит і зміни значень в її записах жодним чином не вплинуть на таблиці Product та Category.

Доречі, зверніть увагу на піктограму нашого запиту в списку запитів бази даних. Вона набула вигляду таблички з знаком оклику, вказуючи на те, що це запит не на вибірку, а на створення нової таблиці.



Запити на створення таблиці підтримуються не всіма СУБД, причому кожна з тих, хто такі запити підтримує, здійснює його різними засобами. Тому, перед тим як написати запит на створення таблиці, перечитайте документацію по SQL для необхідної СУБД.

Наступна і остання конструкція, яку варто розглянути в синтаксисі оператора SELECT – це **TOP**. Вона дозволяє обрати перших n-записів результуючої вибірки. З даною опцією синтаксис оператора SELECT набуває наступного вигляду:

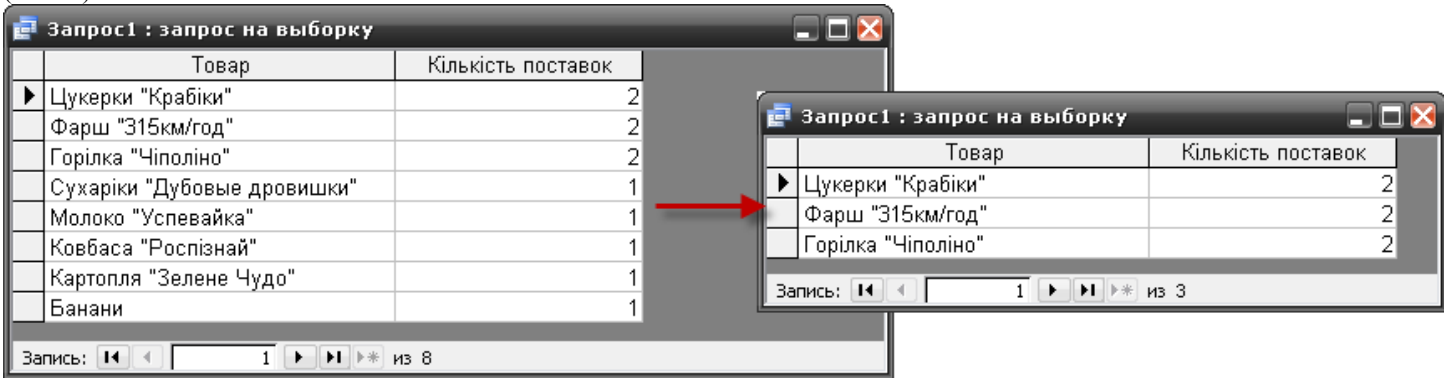
```
SELECT [ALL | DISTINCT]
       [TOP n] { * | поле_для_вибірки [, поле_для_вибірки] }
[INTO ім'я_нової_таблиці
[IN зовнішня_база_даних] ]
FROM    { таблиця | представлення} [as] [псевдонім]
        [, {таблиця | представлення} [as] [псевдонім] ]

[WHERE  умова]
[GROUP BY [ALL] вираз_групування]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [ ASC | DESC ] ];
```

Для прикладу, напишемо запит, який буде містити інформацію про три найчастіше поставляємих товари, тобто товари, замовлень яких було здійснено найбільшу кількість разів.

```
SELECT TOP 3 Product.Name AS [Товар],
       Count(Delivery.IdProduct) AS [Кількість поставок]
FROM Product INNER JOIN Delivery
ON Product.IdProduct = Delivery.IdProduct
GROUP BY Product.Name
ORDER BY 2 DESC;
```

Для отримання такої інформації, спочатку створюється запит на вибірку, в якому підрахується скільки разів був поставлений кожен товар. Після цього результуючий запит сортується і в на екран виводиться перших (верхніх) три записи (TOP 3).



Доречі, результат наших дій можна зберегти в окрему таблицю, яка буде містити звітну інформацію про трійцю найчастіше поставляємих товари.

```
SELECT TOP 3 Product.Name AS [Товар], Count(Delivery.IdProduct) AS [Кількість поставок]
INTO BestDelivery
FROM Product INNER JOIN Delivery
ON Product.IdProduct = Delivery.IdProduct
GROUP BY Product.Name
ORDER BY 2 DESC;
```

4. Модифікація даних засобами DML. Запити на додавання, видалення та обновлення

Переходимо до наступної категорії команд, а саме **DML (Data Manipulation Language - Мова Маніпулювання Даними)**. Команди даної категорії дозволяють маніпулювати даними об'єктів бази даних. До неї входять оператори:

1. Вставки записів – **INSERT**;
2. Видалення записів – **DELETE**;
3. Зміни значень в існуючих записах таблиці - **UPDATE**.

Розглянемо як працювати з даними операторами.

4.1. Оператор INSERT

Оператор SQL INSERT використовується для додавання записів в таблиці. Різні варіанти оператора INSERT дозволяють додавати в таблицю як один, так і множину записів. Це можна здійснювати як шляхом введення константних значень, так і методом зчитування даних з іншої таблиці. В кожному з перелічених випадків **необхідно враховувати структуру таблиці**:

- кількість полів;
- тип даних кожного поля;
- імена полів, в які вносяться дані;
- обмеження і властивості полів.

Синтаксис оператора INSERT наступний:

```
INSERT [INTO] назва_таблиці [( поле1 [, поле2 [,...]] )]
VALUES ( DEFAULT | 'значення1' [, 'значення2' [,...]] );
```

Згідно даного синтаксису у вказану таблицю вставляється запис з значеннями полів, вказаними в переліку фрази VALUES (значення), причому і-е значення відповідає і-му полю в списку полів (поля, не вказані в списку, заповнюються NULL-значеннями). Якщо в списку VALUES вказані всі поля модифікуємої таблиці і порядок їх переліку відповідає порядку полів в описі таблиці, то список полів при INTO можна проігнорувати.

Отже, додамо до списку категорій нову категорію товарів:

```
INSERT INTO Category
VALUES (11, 'Рибні продукти');
```

Таким чином, значення вставляються у всі поля таблиці Category у відповідності з їх фізичним порядком. Якщо потрібно вставити значення тільки в де-які поля, тоді їх потрібно явно вказувати.

Наприклад, ідентифікатор товару вказувати недоречно, він повинен генеруватись автоматично, слід вказувати лише назву категорії. В такому разі запит на добавлення записів слід переписати:

```
INSERT INTO Category (Name)
VALUES ('Соки та води');
```

А що робити коли таблиця має зовнішній ключ? Наприклад, до виробників продукції потрібно додати дані про нового виробника. Таблиця «Producer» має два основних поля: Name (назву) та IdAddress (зовнішній ключ, який містить посилання на повну адресу виробника). Запит на вставку даних буде мати наступний вигляд:

```
INSERT INTO Producer (Name, IdAddress)
VALUES ('ПП Постовалов І.В.', 3);
```

Для додавання кількох записів шляхом вибірки даних з іншої таблиці використовується модифікований синтаксис оператора INSERT:

```
INSERT [INTO] назва_таблиці
       [IN зовнішня_база_даних] [( поле1 [, поле2 [,...]] )]
SELECT ( поле1 [, поле2 [,...]] )
FROM список_таблиць;
```

Вираз IN дозволяє вказати назву зовнішньої бази даних, у вказану таблицю якої будуть вставлені нові дані. Те, які саме дані будуть вставлені, визначається оператором SELECT.

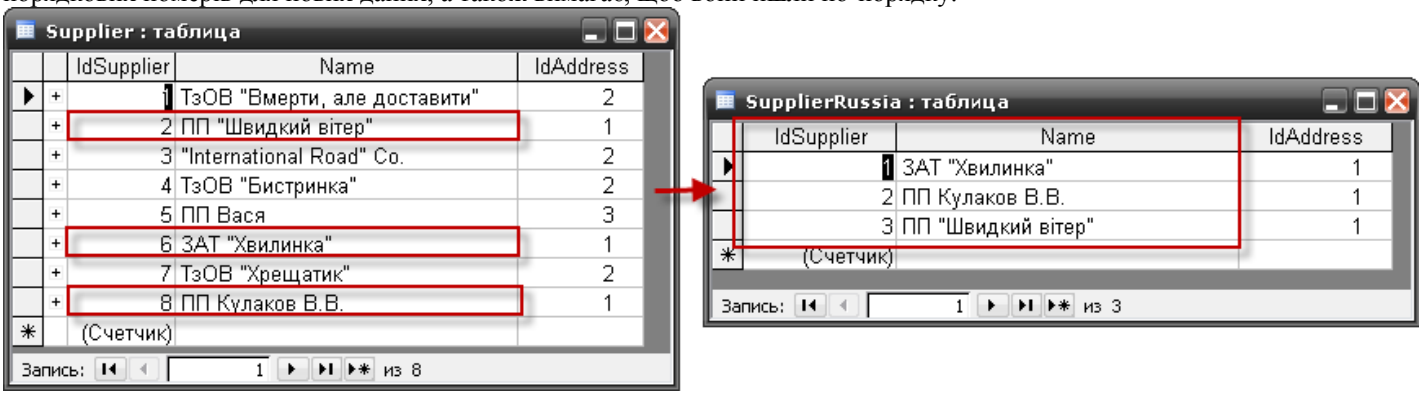
Припустимо, в нас існує таблиця **SupplierArchive**, яка містить копію даних про постачальників.

```
INSERT INTO SupplierArchive
SELECT *
FROM Supplier;
```

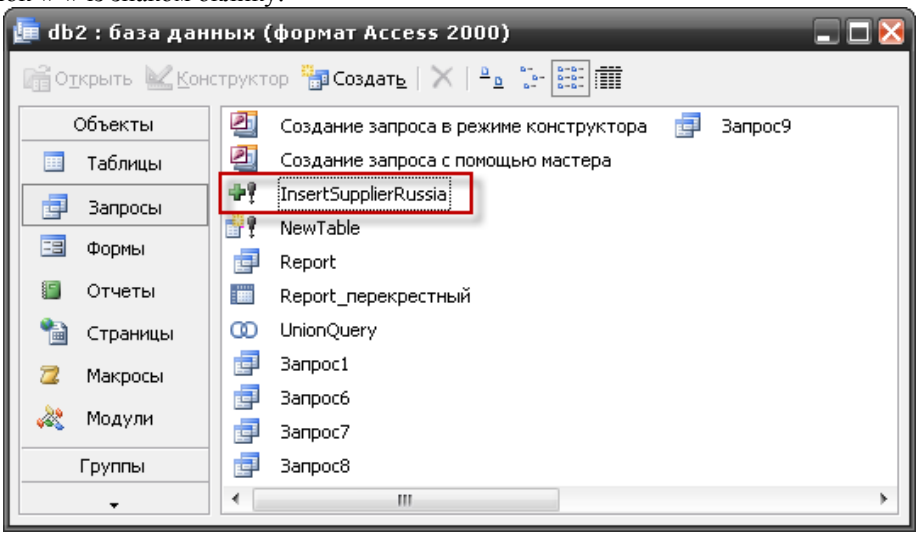
Якщо необхідно накласти умову на вставку даних, наприклад, в нову таблицю **SupplierRussia** потрібно вставити дані про всіх постачальників з Росії, тоді запит на встаку перепишеться:

```
INSERT INTO SupplierRussia (Name, IdAddress)
SELECT Supplier.Name, Supplier.IdAddress
FROM Country INNER JOIN (Address INNER JOIN Supplier
                         ON Address.IdAddress = Supplier.IdAddress)
ON Country.IdCountry = Address.IdCountry
WHERE Country.Name='Росія';
```

Список полів при INTO та SELECT в даному випадку являється обов'язковим, оскільки відібрані дані про постачальників можуть мати невпорядковані ідентифікатори (наприклад, 5 і 10), а нова таблиця може мати свої значення порядкових номерів для нових даних, а також вимагає, щоб вони йшли по-порядку.



Піктограма запитів на вставку даних буде відрізнятись від запитів на вибірку та створення нової таблиці та буде мати значок «+» із знаком оклику:



4.2. Оператор DELETE

За допомогою оператора DELETE з таблиці можна видалити довільну кількість полів. Дозволяється накладати умову, критерій на видаляємі дані. Оскільки оператор DELETE видаляє запис повністю і дану операцію відмінити неможливо (лише через відновлення з резервної копії), слід перевіряти перед видаленням умову, щоб бути впевненим у вірності видалення даних.

При видаленні записів, існує один нюанс. Якщо видаляється запис з таблиці, яка входить у відношення 1:N (ОДИН-ДО-БАГАТЬОХ) і зі сторони 1 дозволене каскадне видалення, то записи, які відносяться до неї зі сторони N, також будуть видалені. Наприклад, якщо таке відношення вставлене для таблиць Customers і Orders, то видалення запису про клієнта автоматично викличе видалення запису про замовлення.

Синтаксис оператора DELETE наступний:

```
DELETE
FROM назва_таблиці
[WHERE умова];
```

Якщо умова не вказана, тоді будуть видалені всі записи вказаної таблиці, що відповідає її повному очищенню. Інакше видаленню будуть підлягати лише ті записи таблиці, які задовольняють критерій.

Наприклад, потрібно видалити вміст таблиці Category, тобто видалити всі категорії товарів:

```
DELETE
FROM Category;
```

Після цього таблиця Category буде пустою.

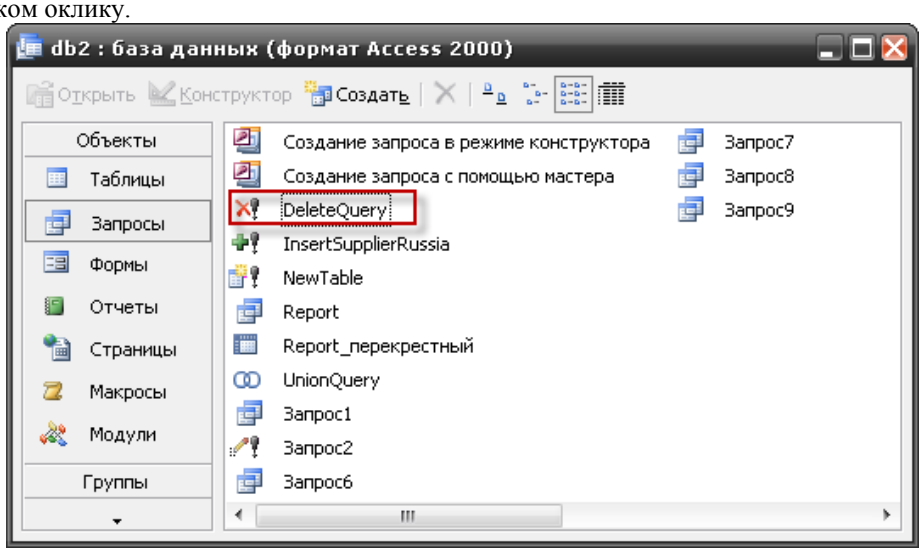
Наступний приклад. Видалити з бази даних виробника «MicroChips» Ltd.

```
DELETE
FROM Producer
WHERE Name='"MicroChips" Ltd.';
```

Якщо при цьому нобхідно видалити всі товари, виробником яких є «MicroChips» Ltd., тоді без підзапиту не обійтись:

```
DELETE
FROM Product
WHERE IdProducer =
                (SELECT IdProducer
                 FROM Producer
                 WHERE Name='"MicroChips" Ltd.');
```

От і все. Піктограма запиту на видалення також буде відрізнятись від інших запитів і буде мати вигляд хрестика із знаком оклику.



4.3. Оператор UPDATE

Третій оператор – це оператор обновлення даних UPDATE. Він дозволяє змінити значення поля (-ів) у вказаній таблиці, як правило, згідно вказаному критерію. Даний оператор особливо корисний при зміні великої кількості записів або записів в кількох таблицях. Подібно оператору DELETE, операцію зміни неможна відмінити, тому перед її запуском потрібно впевнитись у вірності побудови критерію відбору.

Синтаксис оператора UPDATE:

```
UPDATE назва_таблиці
SET   назва_поля = значення [,...]
[WHERE умова];
```

Інструкція SET визначає, які поля повинні бути змінені і на які саме значення. Аналогічно оператору DELETE, конструкція WHERE являється необов'язковою і дозволяє вказати умову відбору записів, значення яких буде модифіковане. По замовчуванню змінюються всі дані вказаної таблиці.

Наприклад, замінити назву постачальника з кодом 3 на «Inter Ltd.»:

```
UPDATE Supplier
SET Name = 'Inter Ltd.'
WHERE IdSupplier = 3;
```

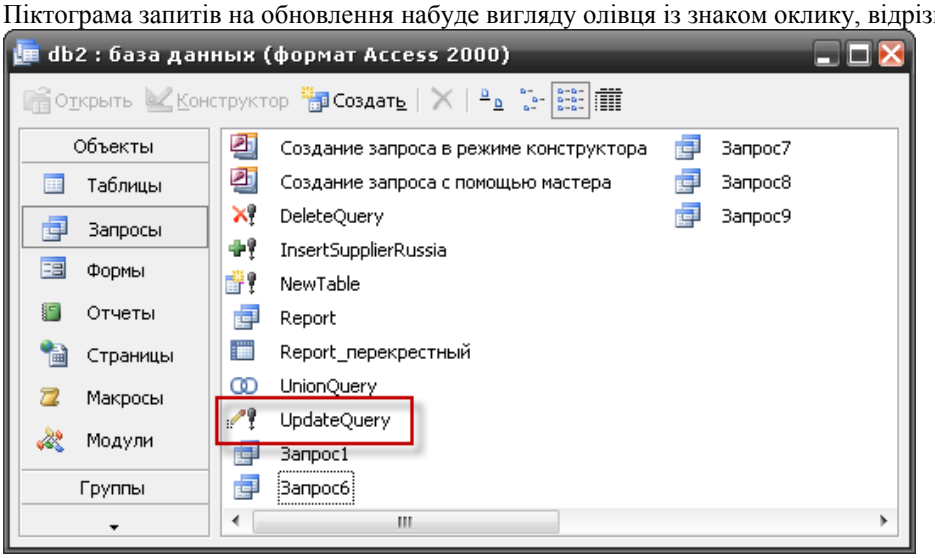
Або ж потрібно встановити ціну поставки рівною 20 грн. для всіх товарів, що поставлялись постачальниками «ПП Вася» та «ВАТ «Швидкий вітер»:

```
UPDATE Delivery
SET Price = 20
WHERE IdSupplier IN
                  (SELECT IdSupplier
                   FROM Supplier
                   WHERE Name IN ('ПП Вася', 'ВАТ "Швидкий вітер"'));
```

Змінити інформацію про поставку йогурта «Живинка», яка була здійснена 12/07/2009 наступним чином: ціну поставки збільшити на 10%, а кількість збільшити на 100 од.

```
UPDATE Delivery
SET Price = Price * 0.2,
    Quantity = Quantity + 100
WHERE DateDelivery = #12/07/2009# AND IdProduct IN
                                                (SELECT IdProduct
                                                 FROM Product
                                                 WHERE Name='Йогурт "Живинка"');
```

Піктограма запитів на обновлення набуде вигляду олівця із знаком оклику, відрізняючи його від інших запитів:



5. Домашнє завдання

1. Зробити запит на створення таблиці, що містить інформацію про 5 найбільш продаваємих товари.
2. Написати параметризований запит, який повертає 2 самих найдорожчих товари вказаної категорії.
3. Написати запит, який дозволить збільшити дату поставки кожного товару, яка відповідає шаблону на 2 дні. Шаблон на товар задається користувачем, тобто передається в якості параметра в запит.
4. Написати запит, який дозволяє переглянути всі товари певного виробника та необхідної категорії, при цьому дані про виробника і категорію вказуються при виклику
5. Засобами SQL зробити вставку нових даних в таблицю Product та Country.
6. Створити таблицю Delivery2008, яка має структуру, аналогічну таблиці Delivery. Дана таблиця повинна містити інформацію про всі поставки, які були здійснені протягом 2008 року. З ціллю автоматизації процесу заповнення таблиці Delivery2008, написати необхідний запит на вставку.
7. Зменшити ціни на всі товари, які постачались двома постачальниками (наприклад, “Петров” та “Сидоров”) на 10% і тип націнки при цьому змінити на оптову.
8. Збільшити на 20% ціну продажу всіх товарів, виробник яких, наприклад, “Бондарчук” та які постачались певним постачальником (наприклад, “Воробей”).
9. Проставити дату поставки рівною сьогоднішній для всіх товарів, в яких така інформація відсутня.
10. Видалити з поточної бази даних всі товари, наявна кількість яких менше 100 од. (кг, шт. тощо), а ціна не перевищує 10грн. за од.

11. Видалити всі товари кондитерської промисловості, продаж яких з початку року складає більше 1000 кг.
12. Видалити всі товари, в яких ціна більша, ніж середня.
13. Засобами SQL створити місячний звіт, який містить інформацію про те, від яких постачальників поставку товарів поточного року в розрізі категорій. Примітка! Прив'язки до конкретних дат бути не повинно, тобто поточний рік визначається програмно.
14. Засобами SQL створити річний звіт про середні ціни продажу товарів певної категорії. Категорія, по якій необхідно формувати звіт, задається користувачем кожен раз перед його створенням.