Урок 3

План заняття:

1. Поняття запитів на мові SQL
2. Вибірка даних. Оператор SELECT
3. Вибірка з використанням операторів BETWEEN, IN, LIKE. Включення недійсного значення (NULL)
4. Функції агрегування. Необхідність використання
5. Групування даних та накладення умов на групу. Оператори GROUP BY та HAVING
6. Оператор SELECT INTO та запити на створення таблиці
7. Домашнє завдання

1. Поняття запитів на мові SQL

Після того як база даних створена і наповнена даними слід перейти до вивчення способів отримання та модифікації даних. Для здійснення цих та ряду інших дій використовуються запити, написані на мові структурованих запитів **SQL**. В першому уроці ми вже розповідали, що **SQL (Structured Query Language)** – це універсальна комп'ютерна мова, яка використовується для створення, модифікації і управління даними в реляційних базах даних. Мову SQL підтримують більше сотні СУБД і MS SQL Server не виключення.

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.

MS SQL Server для запитів, крім мови SQL, використовує її спеціальну різновидність – **Transact-SQL (T-SQL)**, але про неї згодом.

2. Вибірка даних. Оператор SELECT

Як вже не раз говорилось, MS SQL Server 2000..2008 підтримує як стандарт ANSI, так і синтаксис мови **Transact-SQL (T-SQL)**. Дана мова практично нічим не відрізняється від базового стандарту ANSI, хоча має свої особливості, на яких ми будемо зупинятись.

Отже, до праці. Оператор **SELECT** призначений для вибірки даних та складається з трьох основних елементів: SELECT, FROM та WHERE. Повний синтаксис включає ще й інші оператори і доволі складний, узагальнено його можна зобразити наступним чином:

```
SELECT [ALL | DISTINCT ]
       [TOP n [PERCENT] [WITH TIES] ] /*використовувати для вибірки перші n записів
                    (можливе відсоткове значення полів). n = 0..4294967295 або 0..100.
                     WITH TIES використовується, якщо існує ORDER BY та визначає
                     наявність додаткових рядків для результуючого запиту*/
       { * | поля_для_вибірки }
/*створити нову таблицю, яка основана на результаті виконання запиту*/
[INTO ім'я_нової таблиці]
[FROM { таблиці | представлення } [as] [псевдонім] ]
[WHERE      умова]
[GROUP BY [ALL] вираз_групування]

/*для створення додаткових "надагрегатних" рядків*/
[WITH {CUBE | ROLLUP} ]
[HAVING умова_на_групу]
[ORDER BY імя_поля | номер_поля [{ASC | DESC}] ]

/*ставорюють нові рядки на основі даних, які повертаються оператором SELECT*/
[COMPUTE { {AVG | COUNT | MAX | MIN | SUM} (вираз)}
        [BY вираз]
;
```

В результаті, конструктивно нового в синтаксисі майже нічого немає. Напишемо кілька запитів, щоб пригадати технологію їх створення та використання.

1. Напишемо запит, що дозволить переглянути назви книг, їх ціну та дату випуску

```
SELECT    NameBook as 'Назва книги',
          Price as 'Ціна',
          DateOfPublish as 'Дата випуску'
FROM book.Books;
```

Результат:

Results | Messages

	Назва книги	Ціна	Дата випуску
1	Windows NT 5 перспектива	1000,00	1997-06-06 00:00:00.000
2	Сетевые технологии Windows 2000 для профессионалов	210,96	2002-02-28 00:00:00.000
3	Windows 2000 Professional. Руководство Питера Норт...	88,00	2005-10-17 00:00:00.000
4	Максимальная безопасность в Linux	75,00	2005-09-25 00:00:00.000
5	Путь к LINUX. 2е изд.	120,00	2006-12-01 00:00:00.000
6	Как программировать на C	151,50	2006-03-22 00:00:00.000
7	C++ за 21 день	52,00	2006-03-22 00:00:00.000

2. Розширимо наші можливості, додамо до нашого запиту можливість отримати назву категорії книги та автора.

```
SELECT    NameBook as 'Назва книги',
          Price as 'Ціна',
          'Дата випуску' = DateOfPublish,
          a.FirstName + ' ' + a.LastName as 'Повне імя',
          t.NameTheme 'Тематика'
FROM book.Books b, book.Authors a, book.Themes t
WHERE b.ID_AUTHOR = a.ID_AUTHOR AND b.ID_THEME = t.ID_THEME;
```

Результат:

Results | Messages

	Назва книги	Ціна	Дата випуску	Повне імя	Тематика
1	Windows NT 5 перспектива	1000,00	1997-06-06 00:00:00.000	Михаил Мочерный	Windows NT
2	Сетевые технологии Windows 2000 для профессионалов	210,96	2002-02-28 00:00:00.000	Johnson White	Windows 2000
3	Windows 2000 Professional. Руководство Питера Норт...	88,00	2005-10-17 00:00:00.000	Питер Нортон	Windows 2000
4	Максимальная безопасность в Linux	75,00	2005-09-25 00:00:00.000	Meander Smith	Linux
5	Путь к LINUX. 2е изд.	120,00	2006-12-01 00:00:00.000	Marjorie Green	Linux
6	Как программировать на C	151,50	2006-03-22 00:00:00.000	Олег Лисовский	C & C++
7	C++ за 21 день	52,00	2006-03-22 00:00:00.000	Джес Либерти	C & C++

3. Крім звичайного виведення інформації ми можемо здійснювати розрахунки при операторі SELECT, а також накладати умови на результуючу вибірку. Наприклад, виведемо додатково до основної інформації загальну вартість кожної наявної книги та накладемо умову: показувати лише ті книги, ціна яких більше 50 грн.

```
SELECT    NameBook as 'Назва книги',
          Price as 'Ціна',
          'Дата випуску' = DateOfPublish,
          a.FirstName + ' ' + a.LastName as 'Повне імя',
          t.NameTheme 'Тематика',
         (b.Price * b.DrawingOfBook) as 'Загальна сума'    /* розрахункове поле*/
FROM book.Books b, book.Authors a, book.Themes t
WHERE b.ID_AUTHOR = a.ID_AUTHOR AND b.ID_THEME = t.ID_THEME
      AND b.Price > 50;
```

Результат:

Results | Messages

	Назва книги	Ціна	Дата випуску	Повне імя	Тематика	Загальна сума
1	Windows NT 5 перспектива	1000,00	1997-06-06 00:00:00.000	Михаил Мочерный	Windows NT	5000000,00
2	Сетевые технологии Windows 2000 для профессионалов	210,96	2002-02-28 00:00:00.000	Johnson White	Windows 2000	1054800,00
3	Windows 2000 Professional. Руководство Питера Норт...	88,00	2005-10-17 00:00:00.000	Питер Нортон	Windows 2000	369600,00
4	Максимальная безопасность в Linux	75,00	2005-09-25 00:00:00.000	Meander Smith	Linux	225000,00
5	Путь к LINUX. 2е изд.	120,00	2006-12-01 00:00:00.000	Marjorie Green	Linux	360000,00
6	Как программировать на C	151,50	2006-03-22 00:00:00.000	Олег Лисовский	C & C++	454500,00
7	C++ за 21 день	52,00	2006-03-22 00:00:00.000	Джес Либерти	C & C++	104000,00

4. Всю сукупність можна також відсортувати по назві книги в зростаючому порядку. Для сортування використовується параметр ORDER BY.

```
SELECT   NameBook as 'Назва книги',
         Price as 'Ціна',
         'Дата випуску' = DateOfPublish,
         a.FirstName + ' ' + a.LastName as 'Повне імя',
         t.NameTheme 'Тематика',
        (b.Price * b.DrawingOfBook) as 'Загальна сума'
FROM book.Books b, book.Authors a, book.Themes t
WHERE b.ID_AUTHOR = a.ID_AUTHOR AND b.ID_THEME = t.ID_THEME
      AND b.Price > 50
/*ORDER BY 1 ASC*/
ORDER BY b.NameBook;
```

5. Для обмеження результуючої вибірки використовується параметр TOP. Він дозволяє вивести на екран лише певну чітко встановлену кількість записів або їх відсоток. Наприклад, вивести на екран 5 перших записів списку книг видавництва, який має наступний вигляд:

```
select TOP 5 book.NameBook, theme.NameTheme, book.DateOfPublish
from book.Books book, book.Themes theme
where book.ID_THEME=theme.ID_THEME
ORDER BY theme.NameTheme ASC;
```

В SQL Server 2005 була включена інструкція TABLESAMPLE, яка дозволяє обмежити кількість записів, які повертаються однотабличним запитом. При цьому можна вказати кількість або процентне відношення повертаємих записів. Наприклад:

```
select *
from book.Books
tablesample (10 rows); -- tablesample (10 percent)
```

Результатом роботи даної вибірки буде приблизно 10 записів з інформацією про книги видавництва, оскільки механізм пошуку SQL Server випадковим чином обирає сегмент, з якого будуть взяті рядки. Отже, фактична кількість повертаємих записів може значно змінюватись. Якщо вказати невелику кількість (так, як в нашому прикладі), то результат вибірки може бути і нулевим. В зв'язку з цим, не варто використовувати інструкцію TABLESAMPLE в запитах, що повинні повернути точну кількість записів.

Слід пам'ятати, що TABLESAMPLE не може застосовуватись до:

- похідних таблиць;
- таблиць на зв'язаних серверах;
- таблиць, утворених в результаті роботи функцій, які повертають табличне значення;
- таблиць, утворених в результаті роботи функцій, які повертають набори записів;
- інструкцій OPENXML.

Інструкцію TABLESAMPLE не можна вказувати при створенні представлень або у вбудованій функції, яка повертає табличне значення.

3. Вибірка з використанням операторів BETWEEN, IN, LIKE. Включення недійсного значення (NULL)

В SQL існує ряд операторів, які призначені для полегшення здійснення вибірки даних. Пригадаємо їх, а зробити це найкраще безпосередньо на прикладах:

1) Ключеве слово **IN** використовується для перевірки наявності даних в множині. Наприклад, виведемо на екран всі книги, які належать тематикам "C & C++" та "Java, J++, JBuilder, JavaScript". Це можна зробити двома способами:

```
select b.NameBook as 'Назва книги', t.NameTheme as 'Тематика'
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
      and t.NameTheme = 'C & C++' or t.NameTheme = 'Java, J++, JBuilder, JavaScript';
```

або:

```
select b.NameBook as 'Назва книги', t.NameTheme as 'Тематика'
from book.Books b, book.Themes t
where b.id_theme = t.id_theme
      and t.NameTheme IN ('C & C++', 'Java, J++, JBuilder, JavaScript');
```

Результат буде однаковий в обох випадках:

Results | Messages

	Назва книги	Тематика
1	Как программировать на C	C & C++
2	C++ за 21 день	C & C++
3	Язык програмирования C	C & C++
4	Язык програмирования C++	C & C++
5	JavaScript: сборник рецептов для професионалов	Java, J++, JBuilder, JavaScript

2) Оператор **BETWEEN..AND…** використовується для перевірки приналежності до діапазону значень. Наприклад, потрібно вивести всі книги, дати видання яких знаходяться в проміжку 01/12/2006 до 01/02/2007

```
select *
from book.Books
where Books.DateOfPublish = '2006.12.01' AND Books.DateOfPublish = '2007.02.01';
```

або:

```
select *
from book.Books
where Books.DateOfPublish BETWEEN '2006.12.01' AND '2007.02.01';
```

Результат:

Results | Messages

	ID_BOOK	NameBook	ID_THEME	ID_AUTH...	Price	Tiraz	DateOfPublish	Pages
1	18	Путь к LINUX. 2е изд.	15	3	120,00	3000	2006-12-01 00:00:00.000	560
2	21	Язык програмирования C	16	13	78,00	3000	2006-12-01 00:00:00.000	720

3) Оператор **LIKE** дозволяє здійснювати пошук даних по шаблону. Шаблон являє собою рядок з використанням службових символів, до яких належать:

- % - в даній позиції може бути присутній 0 або більше символів;
- _ – в даній позиції обов'язково присутній один довільний символ;
- [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 у вказаний діапазон;
- [^abc] - в даній позиції обов'язково присутній один символ, що не входить у вказаний діапазон значень.

Наприклад, вивести на екран повні імена тих авторів, в прізвищі яких зустрічається літера "M":

```
select FirstName + ' ' + LastName as 'Повне імя автора'
from book.Authors
where LastName LIKE '%[Mm]%';
```

Результат:

Results | Messages

	Повне імя автора
1	Meander Smith

4) Оператор **IS NULL** дозволяє перевірити наявність пустих (NULL) полів. Наприклад, показати всіх книги, про тираж яких інформація **відсутня**, тобто невідома.

```
select *
from book.Books
where DrawingOfBook IS NULL;
```

5) Крім того, перед кожним з перелічених операторів можна вказувати оператор заперечення **NOT**, що заперечує умову, яка описана після нього. Наприклад, показати всіх книги, про тираж яких наявна інформація.

```
select *
from book.Books
where DrawingOfBook IS NOT NULL;
```

4. Функції агрегування. Необхідність використання

Про дані, які зберігаються в таблицях, часто необхідно отримувати статистичну інформацію. Наприклад, маючи наявний список товарів, необхідно взнати їх загальну вартість, вивести середню ціну продажу товарів на кожну дату, визначити мінімальну і максимальну ціни на товари окремої категорії тощо. Подібні обрахунки виконуються за допомогою **агрегатних функцій** або **функцій агрегування**. Кожна з цих функцій працює з сукупністю значень поля деякої таблиці, а результатом функцій є єдине, переважно числове, значення.

В MS SQL Server 2008 існує ряд вбудованих функцій агрегування, а також при необхідності можна визначати свої агрегатні функції за допомогою мов Microsoft .NET. До вбудованих агрегатних функцій належать:

Функція	Опис
COUNT/COUNT_BIG ({ALL | DISTINCT} поле)	Повертає кількість значень у полі: всіх (**ALL**) або лише унікальних (**DISTINCT** - різних). При цьому NULL-значення не враховуються. Якщо в якості параметра замість імені поля використати символ (*), наприклад, count(*), тоді функція поверне кількість значень (записів) в полі, включаючи NULL-значення. Функція COUNT повертає значення типу int, а COUNT_BIG - типу bigint.
AVG ({ALL | DISTINCT} поле)	Повертає середнє арифметичне числових значень для всіх або лише унікальних записів
SUM ({ALL | DISTINCT} поле)	Повертає суму числових значень для всіх або лише унікальних записів
MAX (поле)	Повертає найбільше значення
MIN (поле)	Повертає найменше значення
STDEV (поле)	Повертає середньоквадратичне відхилення всіх значень поля або виразу
STDEVP (поле)	Повертає середньоквадратичне відхилення сукупності всіх значень поля або виразу
VAR (поле)	Повертає дисперсію всіх значень, які містяться у вказаному полі
VARP (поле)	Повертає дисперсію сукупності всіх значень поля або виразу

Для кращого засвоєння матеріалу розберемо кілька прикладів:

1. Створимо запит на отримання інформації про загальну кількість книг у видавництві, їх загальну та середню вартість.

```
SELECT COUNT(b.ID_BOOK) as 'Кількість товару',
       SUM(s.Price*s.Quantity) as 'Загальна вартість продажу',
       AVG(s.Price*s.Quantity) as 'Середня вартість продажу'
FROM book.Books b, sale.Sales s
WHERE b.ID_BOOK=s.ID_BOOK;
```

Результат:

Results | Messages

	Кількість товару	Загальна вартість продажу	Середня вартість продажу
1	14	9378,00	669,8571

2. Напишемо запит, що дозволить вивести на екран загальну кількість авторів, які відомі у видавництві, а також кількість їх унікальних імен.

```
SELECT COUNT(*) as 'Кількість авторів',
       COUNT(DISTINCT FirstName) as 'Кількість унікальних імен'
FROM book.Authors;
```

Результат:

Results | Messages

	ID_AUTHOR	FirstName	LastName
1	1	Ричард	Веймаер
2	2	Johnson	White
3	3	Marjorie	Green
4	4	Meander	Smith
5	5	Livia	Karsen
6	6	Сергей	Парижский
7	7	Сергей	Михайлов
8	8	Тарас	Тимошок
9	9	Андрей	Ковязин
10	10	Михаил	Вострикоов
11	11	Михаил	Мочерный
12	12	Питер	Нортон
13	13	Стивен	Прата
14	14	Джес	Либерти
15	15	Олег	Лисовский
16	16	Artur	Liliput

Results | Messages

	Кількість авторів	Кількість унікальних імен
1	16	14

5. Групування даних та накладення умов на групу. Оператори GROUP BY та HAVING

Для кращого формування статистичної інформації разом з функціями агрегування слід використовувати оператор **GROUP BY**. Дана конструкція дозволяє здійснити групування даних, тобто визначати групи, які володіють спільними характеристиками, а потім згенерувати статистику для кожної групи (з використанням нобхідної функції агрегування). В результаті SQL Server буде повертати стільки рядків, скільки груп ви визначите.

Разом з оператором GROUP BY може використовуватись оператор **HAVING**. Він дозволяє визначити критерій, згідно якого певні групи не включаються у результуючу вибірку, аналогічно тому, як це робить оператор WHERE для окремих записів. Оскільки оператор HAVING дозволяє накласти умову на групи, тому він розміщується після оператора GROUP BY і не може існувати без нього.

Розглянемо кілька прикладів:

1. Виведемо на екран інформацію про кількість магазинів в кожній країні, з якими працює видавництво:

```
SELECT country.NameCountry as 'Назва',
       COUNT(shop.ID_COUNTRY) as 'Кількість магазинів'
FROM sale.Shops shop, global.Country country
WHERE shop.ID_COUNTRY=country.ID_COUNTRY
GROUP BY country.NameCountry;
```

Результат:

Results | Messages

	Країна	Магазин
1	Великобританія	All about PC
2	Росія	Книга
3	США	Booksworld
4	США	Book
5	Україна	Букинист
6	Україна	Слово
7	Україна	Світ книги

Results | Messages

	Назва	Кількість магазинів
1	Великобританія	1
2	Росія	1
3	США	2
4	Україна	3

Таким чином, поле «NameCountry» таблиці «Country» в списку SELECT являється полем групування. SQL Server групує записи по значенню в даному полі, а потім шукає кількість магазинів в кожній групі і виводить отриманий результат на екран.

2. Вивести на екран назви книг разом з їх авторами, тематикою, до якої вони відносяться та значенням самої мінімальної ціни в розрізі кожної тематики і автора в ній. Відсортувати вибірку по повному імені автора.

```
SELECT theme.NameTheme as 'Тематика',
       author.LastName+' '+author.FirstName as 'Повне імя автора',
       CONVERT(char(10), MIN(book.Price))+' грн.' as 'Середня ціна'
FROM book.Books book, book.Authors author, book.Themes theme
WHERE book.ID_AUTHOR=author.ID_AUTHOR AND book.ID_THEME=theme.ID_THEME
GROUP BY theme.NameTheme, author.LastName+' '+author.FirstName
ORDER BY 1;
```

Результат:

Results | Messages

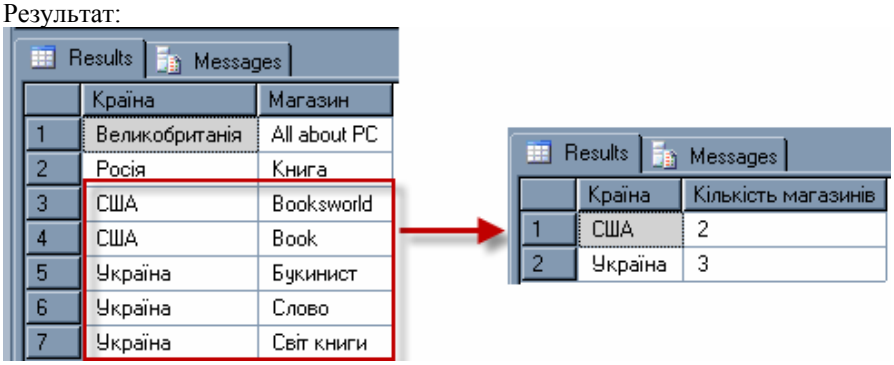
	Тематика	Повне імя автора	Середня ціна
1	C & C++	Либерти Джес	148.36 грн.
2	C & C++	Лисовский Олег	432.25 грн.
3	C & C++	Прата Стивен	222.54 грн.
4	Java, J++, JBuilder, JavaScript	Парижский Сергей	265.34 грн.
5	Linux	Green Marjorie	342.37 грн.
6	Linux	Smith Meander	213.98 грн.
7	Visual C++	White Johnson	46.85 грн.
8	Visual C++	Либерти Джес	427.97 грн.

Примітка! Всі поля групування, які ви задаєте в списку SELECT, повинні бути включені у список GROUP BY.

3. Знайдемо країни, в яких існує більше одного магазину, що реалізує продукцію видавництва. Для цього слід накласти умову на утворені групи за допомогою оператора HAVING:

```
SELECT country.NameCountry as 'Країна',
       COUNT(shop.ID_COUNTRY) as 'Кількість магазинів'
FROM sale.Shops shop, global.Country country
WHERE shop.ID_COUNTRY=country.ID_COUNTRY
GROUP BY country.NameCountry
HAVING COUNT(shop.ID_COUNTRY) > 1;
```

Результат:



	Країна	Магазин
1	Великобританія	All about PC
2	Росія	Книга
3	США	Booksworld
4	США	Book
5	Україна	Букинист
6	Україна	Слово
7	Україна	Світ книги

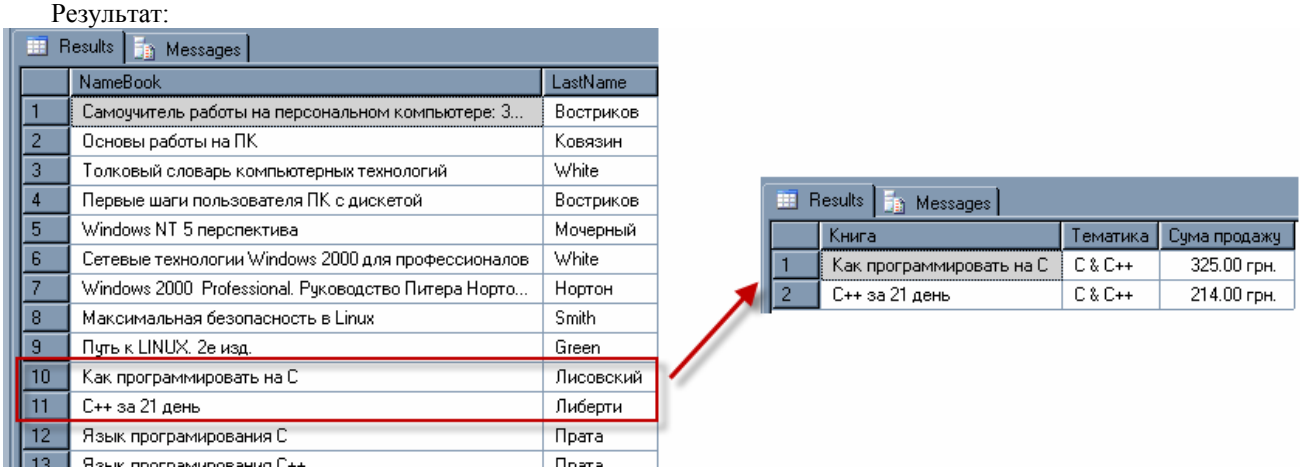
	Країна	Кількість магазинів
1	США	2
2	Україна	3

Умову також можна визначати в операторі WHERE, але така умова буде стосуватись окремого запису результуючої вибірки, що призводить до зниження ефективності роботи запиту. Крім того, вказуючи умову в операторі WHERE, ви одразу виключаєте відібрані рядки з розгляду будь-якої операції групування. Оператор HAVING ефективніший за рахунок того, що він починає діяти лише після того, як групи сформовані, визначаючи, які саме групи будуть відображені в результаті.

4. Вивести книги з вказанням їх тематики та загальну суму їх продажу. Умова: книги лише авторів, прізвища яких починаються на літеру «Л».

```
SELECT book.NameBook as 'Книга',
       theme.NameTheme as 'Тематика',
       convert(char(10), SUM(sale.Price))+' грн.' as 'Сума продажу'
FROM book.Books book, sale.Sales sale, book.Themes theme, book.Authors author
WHERE book.ID_THEME=theme.ID_THEME AND book.ID_BOOK = sale.ID_BOOK
      AND book.ID_AUTHOR=author.ID_AUTHOR
GROUP BY book.NameBook, theme.NameTheme, author.LastName
HAVING author.LastName LIKE 'Л%';
```

Результат:



	NameBook	LastName
1	Самоучитель работы на персональном компьютере: 3...	Востриков
2	Основы работы на ПК	Ковязин
3	Толковый словарь компьютерных технологий	White
4	Первые шаги пользователя ПК с дискетой	Востриков
5	Windows NT 5 перспектива	Мочерный
6	Сетевые технологии Windows 2000 для профессионалов	White
7	Windows 2000 Professional. Руководство Питера Норто...	Нортон
8	Максимальная безопасность в Linux	Smith
9	Путь к LINUX. 2е изд.	Green
10	Как программировать на C	Лисовский
11	C++ за 21 день	Либерти
12	Язык програмирования C	Прата

	Книга	Тематика	Сума продажу
1	Как программировать на C	C & C++	325.00 грн.
2	C++ за 21 день	C & C++	214.00 грн.

7. Оператор SELECT INTO та запити на створення таблиці

В синтаксисі оператора SELECT існує опція **INTO**, за допомогою якої можна створити таблицю, значення в якій будуть являтись значеннями результату запиту на вибірку. Створена таким чином таблиця може бути або **тимчасовою** (яка знищується при завершенні роботи з сервером), або **постійною**. І в одному, і в другому випадках для полів бажано вказувати псевдоніми з ціллю уникнення проблем щодо ідентифікації полів новоствореної таблиці.

ПРИМІТКА! Якщо поля таблиці не мають імен, то звернутись до них неможливо ні при написанні запиту, ні в будь-якому іншому випадку. Прочитати значення полів такої таблиці можна лише за допомогою оператора SELECT *.

SELECT INTO в основному використовується в зберігаємих процедурах для створення локальних тимчасових таблиць. Замість оператора CREATE TABLE і вставки даних в тимчасову таблицю розробники можуть використати оператор SELECT INTO для виконання обох операцій.

Отже, напишемо запит з ціллю створення **постійної нової таблиці** на основі результуючого запиту, що дозволить отримати інформацію про назви книг та їх авторів.

```
select b.NameBook as 'NameBook', a.FirstName + ' ' + a.LastName as 'Full_Name_Author'
into book.BookAuthor
from book.Books b, book.Authors a
where b.id_author = a.id_author;
```

В результаті утворилась нова таблиця **BookAuthor** в просторі імен **book**, що містить дані про назву книги та повне ім'я її автора. Імена полів новоствореної таблиці носять назву псевдонімів полів при написанні запиту, тобто **NameBook** та **Full_Name_Author**.

Після цього ви маєте можливість написати довільний запит до таблиці BookAuthor, а зміни значень в її записах ніяк не вплинуть на таблиці Books та Authors.

Як вже було раніше сказано, крім створення постійної таблиці, можна створювати тимчасову таблицю. Всі тимчасові таблиці знаходяться в базі даних **tempdb**. Розрізняють **два типи тимчасових таблиць**:

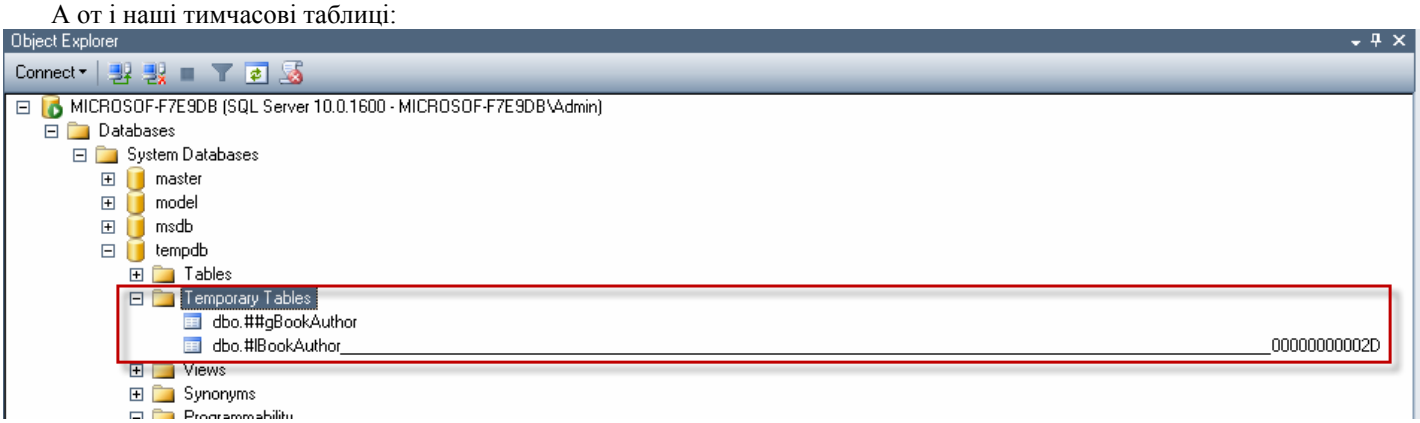
- **локальна тимчасова таблиця** - доступна лише під час поточного сеансу зв'язку користувача SQL Server. При закінченні сеансу вона знищується. При її створенні перед іменем потрібно вказати символ #;
- **глобальна тимчасова таблиця** - доступна на протязі всіх сеансів зв'язку користувача SQL Server. Така таблиця знищується після закінчення сеансу ОСТАННЬОГО КОРИСТУВАЧА, який звертався до неї. При її створенні перед іменем потрібно вказати два символи ##.

Напишемо запит аналогічний попередньому, але результат збережемо в глобальну, а не постійну таблицю:

```
/*локальна тимчасова таблиця*/
select b.NameBook as 'NameBook', a.FirstName + ' ' + a.LastName as 'Full_Name_Author'
into #lBookAuthor
from book.Books b, book.Authors a
where b.id_author = a.id_author;

/*глобальна тимчасова таблиця*/
select b.NameBook as 'NameBook', a.FirstName + ' ' + a.LastName as 'Full_Name_Author'
into ##gBookAuthor
from book.Books b, book.Authors a
where b.id_author = a.id_author;
```

А от і наші тимчасові таблиці:



Відмітимо, що операція SELECT INTO не реєструється в журналі транзакцій, тому варто після її виконання зберігати базу даних.

7. Домашнє завдання

1. Показати всі книги трьох довільних авторів
2. Показати всі книги, в яких кількість сторінок більше 500, але меньше 650
3. Необходимо вивести всі назви книг, в яких перша літера або А, або З
4. Показати назви підручників, тематика яких не "Детектив", і тираж яких >= 3000 екземплярів.
5. Показати всі книги-новинки, ціна яких нижче 30 грн. (Новинкою буде вважатись книга, що була видана на протязі останнього тиждня)
6. Показати книги, в назвах яких є слово Microsoft, але немає слова Windows

7. Вивести назви книг, тематику, автора (повне ім’я), ціна однієї сторінки яких меньше 10 копійок
8. Вивести інформацію про всі книги, в яких в імені більше 4-ьох слів
9. Вивести на екран всі книги, їх авторів та ціни їх продажу в у.о., дата продажу яких знаходиться в діапазоні 01/01/2007 по сьогоднішню дату
10. Показати всю інформацію по продажах книг в наступному вигляді:
    - назва книги
    - тематик, які стосуються "Програмування"
    - автор книги (повне ім’я)
    - ціна продажу книги
    - наявна кількість продажу даної книги
    - назва магазину, який знаходиться не в Україні та не в Росії і продає дану книгу
11. Показати кількість авторів в базі даних. Результат зберегти в іншу таблицю.
12. Показати среднє арифметичне цін продажу всіх книг. Результат зберегти в локальну тимчасову таблицю.
13. Показати тематики книг та суму сторінок по кожній з них.
14. Вивести кількість книг і суму сторінок цих книг по кожному з перших трьох (!) авторів в базі даних.
15. Вивести інформацію про книги по "Програмуванню" з найбільшою кількістю сторінок.
16. Показати авторів і саму найстарішу книгу по кожному з них. Результат зберегти в глобальну тимчасову таблицю.
17. Показати на екран середню кількість сторінок по кожній з тематик, при цьому показати лише тематики, в яких средня кількість більше 400
18. Показати на екран суму сторінок по кожній з тематик, при цьому враховувати лише книги з кількістю сторінок більше 300, але враховувати при цьому лише 3 тематики, наприклад 'Програмування', 'Мережі' і 'Web-дизайн'
19. Показати кількість проданих книг по кожному магазину, в проміжну від 01/01/2007 до сьогоднішньої дати.
20. Вивести всю інформацію про роботу магазинів: що, коли, скільки та ким було продано, а також вказати країну, де знаходиться магазин. Результат зберегти в іншу таблицю.