



*Программирование
и администрирование
СУБД*

Microsoft®
SQL Server

Урок №3

Содержание

1. Понятие запросов на языке SQL3
2. Выборка данных. Оператор SELECT4
3. Выборка с использованием операторов
BETWEEN, IN, LIKE. Включение
недействительного значения (NULL)10
4. Функции агрегирования. Необходимость
использования14
5. Группировка данных и наложение условий
на группу. Операторы GROUP BY и HAVING ...17
6. Оператор SELECT INTO
и запросы на создание таблицы.....22
7. Домашнее задание25

1. Понятие запросов на языке SQL

После того как база данных создана и наполнена данными следует перейти к изучению способов получения и модификации данных. Для осуществления этих и ряда других действий используются запросы, написанные на языке структурированных запросов SQL. В первом уроке мы уже рассказывали, что SQL (Structured Query Language) – это универсальный компьютерный язык, который используется для создания, модификации и управления данными в реляционных базах данных. Язык SQL поддерживают более сотни СУБД и MS SQL Server не исключение.

Основой работы с языком SQL является запрос. Запрос – это команда, которая сообщает о необходимости получения указанных данных. Например, можно построить запрос, который будет выводить список всех клиентов магазина, данные о наиболее активных покупателях или сформировать алфавитный перечень товаров, которые пользуются наибольшим спросом или были списаны в течение определенного периода. Как правило, эта информация получается из основного источника реляционной базы данных – таблиц, а результат выводится на экран компьютера. Хотя ее можно вывести на принтер, сохранить в файле или другом объекте базы данных и тому подобное.

MS SQL Server для запросов, кроме языка SQL, использует его специальную разновидность – Transact-SQL (T-SQL), но о ней позже.

2. Выборка данных. Оператор SELECT

Как уже не раз говорилось, MS SQL Server 2000..2008 поддерживает как стандарт ANSI, так и синтаксис языка Transact-SQL (T-SQL). Данный язык практически ничем не отличается от базового стандарта ANSI, хотя имеет свои особенности, на которых мы будем останавливаться.

Итак, к работе. Оператор SELECT предназначен для выборки данных и состоит из трех основных элементов: SELECT, FROM и WHERE. Полный синтаксис включает еще и другие операторы и довольно сложный, обобщенно его можно изобразить следующим образом:

```
SELECT [ALL | DISTINCT ]
      [TOP n [PERCENT] [WITH TIES] ] /* использовать
для выборки первые n записей (Возможно процентное
значение полей).n = 0..4294967295 или 0..100. WITH TIES
используется, если существует ORDER BY и определяет
наличие дополнительных строк для результирующего запроса
* /

      { * | поля_для_выборки }

/* создать новую таблицу, которая основана на результате
выполнения запроса * /
[INTO имя_новой таблицы]
[FROM { таблицы | представления } [as] [псевдоним] ]
[WHERE условие]
[GROUP BY [ALL] выражение_группирования]
/*Для создания дополнительных "надаггрегатных" строк*/
[WITH {CUBE | ROLLUP} ]
[HAVING условие_на_группу]
[ORDER BY имя_поля | номер_поля [{ASC | DESC}] ]
```

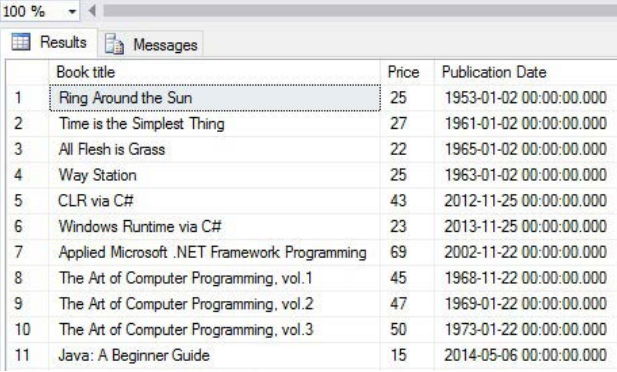
```
/* создают новые строки на основе данных, которые
возвращаются оператором SELECT */
[COMPUTE { {AVG | COUNT | MAX | MIN | SUM} (выражение) }
  [BY выражение]
;
```

В результате, конструктивно нового в синтаксисе почти ничего нет. Напишем несколько запросов, чтобы вспомнить технологию их создания и использования.

1. Напишем запрос, который позволит пересмотреть названия книг, их цену и дату выпуска

```
SELECT NameBook as 'Book title',
       price as 'Price',
       DateOfPublish as 'Publication Date'
FROM books;
```

Результат:



	Book title	Price	Publication Date
1	Ring Around the Sun	25	1953-01-02 00:00:00.000
2	Time is the Simplest Thing	27	1961-01-02 00:00:00.000
3	All Flesh is Grass	22	1965-01-02 00:00:00.000
4	Way Station	25	1963-01-02 00:00:00.000
5	CLR via C#	43	2012-11-25 00:00:00.000
6	Windows Runtime via C#	23	2013-11-25 00:00:00.000
7	Applied Microsoft .NET Framework Programming	69	2002-11-22 00:00:00.000
8	The Art of Computer Programming, vol.1	45	1968-11-22 00:00:00.000
9	The Art of Computer Programming, vol.2	47	1969-01-22 00:00:00.000
10	The Art of Computer Programming, vol.3	50	1973-01-22 00:00:00.000
11	Java: A Beginner Guide	15	2014-05-06 00:00:00.000

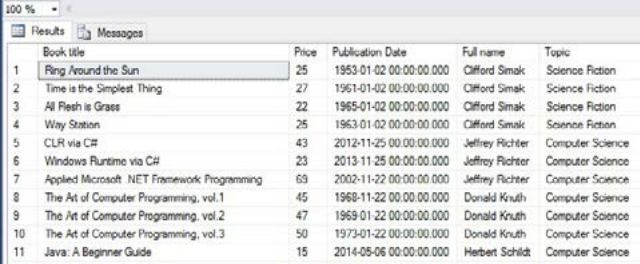
2. Расширим наши возможности, добавим к нашему запросу возможность получить название категории книги и автора.

```

Select NameBook as 'Book title' price as 'Price',
      DateOfPublish as 'Publication Date'
      a.FirstName + ' ' + a.LastName as 'Full name',
      t.NameTheme as 'Topic'
from books b, Authors a, Themes t
where b.id_author = a.id and b.id_theme = t.id;

```

Результат:



	Book title	Price	Publication Date	Full name	Topic
1	Ring Around the Sun	25	1953-01-02 00:00:00.000	Clifford Simak	Science Fiction
2	Time is the Simplest Thing	27	1961-01-02 00:00:00.000	Clifford Simak	Science Fiction
3	All Flesh is Grass	22	1965-01-02 00:00:00.000	Clifford Simak	Science Fiction
4	Way Station	25	1963-01-02 00:00:00.000	Clifford Simak	Science Fiction
5	CLR via C#	43	2012-11-25 00:00:00.000	Jeffrey Richter	Computer Science
6	Windows Runtime via C#	23	2013-11-25 00:00:00.000	Jeffrey Richter	Computer Science
7	Applied Microsoft .NET Framework Programming	69	2002-11-22 00:00:00.000	Jeffrey Richter	Computer Science
8	The Art of Computer Programming, vol.1	45	1968-01-22 00:00:00.000	Donald Knuth	Computer Science
9	The Art of Computer Programming, vol.2	47	1969-01-22 00:00:00.000	Donald Knuth	Computer Science
10	The Art of Computer Programming, vol.3	50	1973-01-22 00:00:00.000	Donald Knuth	Computer Science
11	Java: A Beginner Guide	15	2014-05-06 00:00:00.000	Herbert Schildt	Computer Science

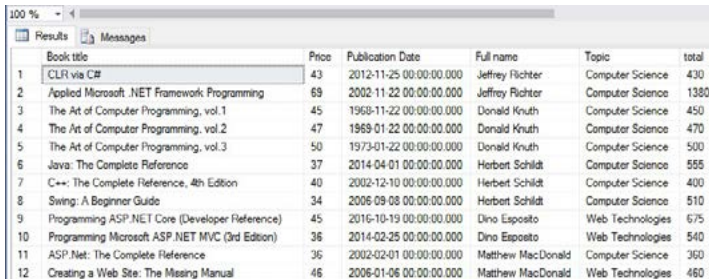
3. Кроме обычного вывода информации мы можем осуществлять расчеты при операторе SELECT, а также накладывать условия на результирующую выборку. Например, выведем дополнительно к основной информации общую стоимость каждой имеющейся книги и наложим условие: показывать только те книги, цена которых больше \$30.

```

Select NameBook as 'Book title' price as 'Price',
      DateOfPublish as 'Publication Date'
      a.FirstName + ' ' + a.LastName as 'Full name',
      t.NameTheme as 'Topic'
      b.price * b.Quantity as 'total'
from books b, Authors a, Themes t
where b.id_author = a.id and b.id_theme = t.id
and b.price > 30;

```

Результат:



	Book title	Price	Publication Date	Full name	Topic	total
1	CLR via C#	43	2012-11-25 00:00:00.000	Jeffrey Richter	Computer Science	430
2	Applied Microsoft .NET Framework Programming	69	2002-11-22 00:00:00.000	Jeffrey Richter	Computer Science	1380
3	The Art of Computer Programming, vol.1	45	1968-11-22 00:00:00.000	Donald Knuth	Computer Science	450
4	The Art of Computer Programming, vol.2	47	1969-01-22 00:00:00.000	Donald Knuth	Computer Science	470
5	The Art of Computer Programming, vol.3	50	1973-01-22 00:00:00.000	Donald Knuth	Computer Science	500
6	Java: The Complete Reference	37	2014-04-01 00:00:00.000	Herbert Schildt	Computer Science	555
7	C++: The Complete Reference, 4th Edition	40	2002-12-10 00:00:00.000	Herbert Schildt	Computer Science	400
8	Swing: A Beginner Guide	34	2005-09-08 00:00:00.000	Herbert Schildt	Computer Science	510
9	Programming ASP.NET Core (Developer Reference)	45	2016-10-19 00:00:00.000	Dino Esposito	Web Technologies	675
10	Programming Microsoft ASP.NET MVC (3rd Edition)	36	2014-02-25 00:00:00.000	Dino Esposito	Web Technologies	540
11	ASP.Net: The Complete Reference	36	2002-02-01 00:00:00.000	Matthew MacDonald	Computer Science	360
12	Creating a Web Site: The Missing Manual	46	2006-01-06 00:00:00.000	Matthew MacDonald	Web Technologies	460

4. Всю совокупность можно отсортировать по названию книги в возрастающем порядке. Для сортировки используется параметр ORDER BY.

```
Select NameBook as 'Book title' price as 'Price',
    DateOfPublish as 'Publication Date'
    a.FirstName + ' ' + a.LastName as 'Full name',
    t.NameTheme as 'Topic'
    b.price * b.Quantity as 'total'
from books b, Authors a, Themes t
where b.id_author = a.id and b.id_theme = t.id
and b.price > 30
order by b.Namebook;
```

5. Для ограничения результирующей выборки используется параметр TOP. Он позволяет вывести на экран только определенное четко установленное количество записей или их процент. Например, вывести на экран 5 первых записей списка книг издательства, который имеет следующий вид:

```
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 была включена инструкция TABLESAMPLE, которая позволяет ограничить количество записей, которые возвращаются однотабличным запросом. При этом можно указать количество или процентное отношение возвращаемых записей. К примеру:

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

Результатом работы данной выборки будет примерно 10 записей с информацией о книгах издательства, поскольку механизм поиска SQL Server случайным образом выбирает сегмент, с которого будут взяты строки. Итак, фактическое количество возвращаемых записей может значительно варьироваться. Если указать небольшое количество (так, как в нашем примере), то результат выборки может быть и нулевым. В связи с этим, не стоит использовать инструкцию TABLESAMPLE в запросах, которые должны вернуть точное количество записей.

Следует помнить, что TABLESAMPLE не может применяться к:

- производным таблицам;
- таблицам на связанных серверах;
- таблицам, образованным в результате работы функций, возвращающих табличное значение;

- таблицам, образованным в результате работы функций, которые возвращают наборы записей;
- инструкций OPENXML.

Инструкция TABLESAMPLE нельзя указывать при создании представлений или во встроенной функции, которая возвращает табличное значение.

3. Выборка с использованием операторов BETWEEN, IN, LIKE. Включение недействительного значения (NULL)

В SQL существует ряд операторов, которые предназначены для облегчения осуществления выборки данных. Вспомним их, а сделать это лучше всего непосредственно на примерах:

1. Ключевое слово IN используется для проверки совпадения проверяемого значения с одним из множества предоставленных в IN значений. Например, выведем на экран все книги, которые принадлежат тематикам "Computer Science" и "Web Technologies". Это можно сделать двумя способами:

```
select NameBook as 'Book title',  
       t.NameTheme as 'Topic'  
from books b, Themes t  
where b.id_theme = t.id  
       and t.NameTheme = 'Computer Science' or  
       t.NameTheme = 'Web Technologies';
```

ИЛИ:

```
select NameBook as 'Book title',  
       t.NameTheme as 'Topic'  
from books b, Themes t  
where b.id_theme = t.id  
       and t.NameTheme in ('Computer Science',  
                           'Web Technologies');
```

Результат будет одинаков в обоих случаях:

	Book title	Topic
1	CLR via C#	Computer Science
2	Windows Runtime via C#	Computer Science
3	Applied Microsoft .NET Framework Programming	Computer Science
4	The Art of Computer Programming, vol.1	Computer Science
5	The Art of Computer Programming, vol.2	Computer Science
6	The Art of Computer Programming, vol.3	Computer Science
7	Java: A Beginner Guide	Computer Science
8	Java: The Complete Reference	Computer Science
9	C++: The Complete Reference, 4th Edition	Computer Science
10	Swing: A Beginner Guide	Computer Science
11	Programming ASP.NET Core (Developer Reference)	Web Technologies
12	Programming Microsoft ASP.NET MVC (3rd Edition)	Web Technologies
13	ASP.Net: The Complete Reference	Computer Science

2. Оператор BETWEEN..AND ... используется для проверки принадлежности к диапазону значений. Например, нужно вывести все книги, даты издания которых находятся в промежутке с 01/01/2005 до 01/01/2012:

```
select *
from books
where DateOfPublish > '01/01/2005' and
      DateOfPublish < '01/01/2012';
```

ИЛИ:

```
select *
from books
where DateOfPublish between '01/01/2005' and
      '01/01/2012';
```

Результат:

	id	NameBook	Price	Pages	Quantity	DateOfPublish	Id_author	Id_theme
1	14	Swing: A Beginner Guide	34	380	15	2006-09-08 00:00:00.000	3	1
2	18	Creating a Web Site: The Missing Manual	46	608	10	2006-01-06 00:00:00.000	5	3
3	19	Pro WPF in C# 2010: Windows Presentation Foundat...	51	1181	10	2010-03-19 00:00:00.000	5	1

3. Оператор LIKE позволяет осуществлять поиск данных по шаблону. Шаблон представляет собой строку с использованием служебных символов, к которым относятся:

- % – в данной позиции может присутствовать 0 или более символов;
- _ – в данной позиции обязательно присутствует один произвольный символ;
- [a-z] – в данной позиции обязательно присутствует один символ из указанного диапазона;
- [abc] – в данной позиции обязательно присутствует один символ из указанного диапазона значений;
- [^ a-z] – в данной позиции обязательно присутствует один символ, не входящий в указанный диапазон;
- [^ abc] – в данной позиции обязательно присутствует один символ, не входящий в указанный диапазон значений.

Например, вывести на экран полные имена авторов, в фамилии которых встречается буква "М":

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

Результат:

Results		Messages
Full name of the author		
1	Matthew MacDonald	
2	Clifford Simak	
3	Richard Waymire	

4. Оператор IS NULL позволяет проверить наличие пустых (NULL) полей. Например, показать все книги, о тираже которых информация отсутствует, то есть неизвестна.

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

5. Кроме того, перед каждым из перечисленных операторов можно указывать оператор отрицания NOT, что отрицает условие, которое описано после него. Например, показать все книги, о тираже которых имеется информация.

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

4. Функции агрегирования. Необходимость использования

О данных, которые хранятся в таблицах, часто необходимо получать статистическую информацию. Например, имея существующий список товаров, необходимо узнать их общую стоимость, вывести среднюю цену продажи товаров на каждую дату, определить минимальную и максимальную цены на товары отдельной категории и тому подобное. Подобные расчеты выполняются с помощью агрегатных функций или функций агрегирования. Каждая из этих функций работает с совокупностью значений поля некоторой таблицы, а результатом функций является единственное, преимущественно числовое, значение.

В MS SQL Server 2008 существует ряд встроенных функций агрегирования, а также при необходимости можно определять свои агрегатные функции с помощью языков Microsoft.NET. К встроенным агрегатным функциям относятся:

COUNT/COUNT_BIG ({ALL | DISTINCT} поле) – Возвращает количество значений в поле: всех (ALL) или только уникальных (DISTINCT – разных). При этом NULL-значения не учитываются.

Если в качестве параметра вместо имени поля использовать символ (*), например, count (*), тогда функция вернет количество значений (записей) в поле, включая NULL-значения.

Функция COUNT возвращает значение типа int, а COUNT_BIG – типа bigint.

AVG ({ALL | DISTINCT} поле) – Возвращает среднее арифметическое числовых значений для всех или только уникальных записей

SUM ({ALL | DISTINCT} поле) – Возвращает сумму числовых значений для всех или только уникальных записей

MAX (поле) – возвращает наибольшее значение

MIN (поле) – возвращает наименьшее значение

STDEV (поле) – возвращает стандартное отклонение всех значений поля или выражения

STDEVP (поле) – возвращает стандартное отклонение совокупности всех значений поля или выражения

VAR (поле) – возвращает дисперсию всех значений, которые содержатся в указанном поле

ARP (поле) – возвращает дисперсию совокупности всех значений поля или выражения

Для лучшего усвоения материала разберем несколько примеров:

1. Создадим запрос на получение информации об общем количестве книг в издательстве, их общую и среднюю стоимость.

```
SELECT COUNT(b.ID_BOOK) as 'Количество товара',
       SUM(s.Price*s.Quantity) as 'Общая стоимость продажи',
       AVG(s.Price*s.Quantity) as 'Средняя стоимость продажи'
FROM book.Books b, sale.Sales s
WHERE b.ID_BOOK=s.ID_BOOK;
```

Результат:

	Quantity of commodities	Total	Average price
1	20	9840	492

2. Напишем запрос, который позволит вывести на экран общее количество авторов, которые известны в издательстве, а также количество их уникальных имен.

```
SELECT COUNT(*) as 'Количество авторов',
       COUNT(DISTINCT FirstName) as 'Количество
уникальных имен'
FROM book.Authors;
```

Результат:

	id	FirstName	LastName
1	6	Clifford	Simak
2	4	Dino	Esposito
3	1	Donald	Knuth
4	7	Herbert	Wells
5	3	Herbert	Schildt
6	2	Jeffrey	Richter
7	5	Matthew	MacDonald

	Number of authors	Number of unique names
1	7	6

5. Группировка данных и наложение условий на группу. Операторы GROUP BY и HAVING

Для лучшего формирования статистической информации вместе с функциями агрегирования следует использовать оператор GROUP BY. Данная конструкция позволяет осуществить группирование данных, то есть определять группы, которые обладают общими характеристиками, а затем сгенерировать статистику для каждой группы (с использованием необходимой функции агрегирования). В результате SQL Server будет возвращать столько строк, сколько групп вы определите.

Вместе с оператором GROUP BY может использоваться оператор HAVING. Он позволяет определить критерий, согласно которому определенные группы не включаются в результирующую выборку, аналогично тому, как это делает оператор WHERE для отдельных записей. Поскольку оператор HAVING позволяет наложить условие на группы, поэтому он размещается после оператора GROUP BY и не может существовать без него.

Рассмотрим несколько примеров:

1. Выведем на экран информацию о количестве магазинов в каждой стране, с которыми работает издательство:

```
SELECT c.Country as 'Country Name',
       COUNT(sh.id_country) as 'Number of Shops'
FROM Countries c, Shops sh
WHERE c.id = sh.id_country
GROUP BY c.Country;
```

Результат:

Country Name	Shop Name
Canada	Booker
France	BrightIdoo
France	Booker
Germany	Smith&Brown
Germany	BrightIdoo
Germany	Booker
Sweden	HashTag
Sweden	Booker
Ukraine	Rare Books
Ukraine	HashTag
Ukraine	Booker
United Kingdom	Rainbow

Country Name	Number of shops
Belgium	2
Canada	2
France	2
Germany	3
Sweden	2
Ukraine	3
United Kingdom	5
USA	4

Таким образом, поле "NameCountry" таблицы "Country" в списке SELECT является полем группировки. SQL Server группирует записи по значению в данном поле, а затем ищет количество магазинов в каждой группе и выводит полученный результат на экран.

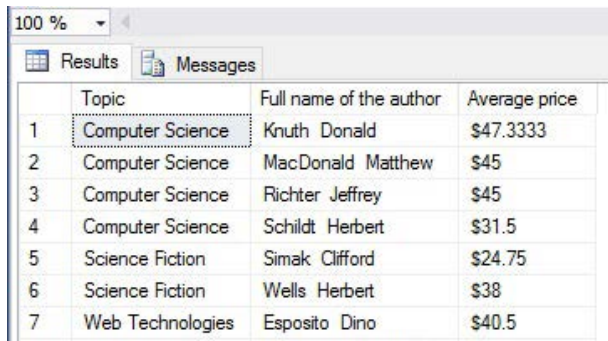
- Вывести на экран названия книг вместе с их авторами, тематике, к которой они относятся и значению самой минимальной цены в разрезе каждой тематики и автора в ней. Отсортировать выборку по полному имени автора.

```

SELECT theme.NameTheme as 'Topic',
       author.LastName+' '+author.FirstName as 'Full name
                                     of an author',
       CONVERT(char(10), MIN(book.Price))+ ' $ ' as
'Average price'
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:



	Topic	Full name of the author	Average price
1	Computer Science	Knuth Donald	\$47.3333
2	Computer Science	MacDonald Matthew	\$45
3	Computer Science	Richter Jeffrey	\$45
4	Computer Science	Schildt Herbert	\$31.5
5	Science Fiction	Simak Clifford	\$24.75
6	Science Fiction	Wells Herbert	\$38
7	Web Technologies	Esposito Dino	\$40.5

Примечание! Все поля группировки, которые вы задаете в списке *SELECT*, должны быть включены в список *GROUP BY*.

- Найдем страны, в которых существует более одного магазина, реализующего продукцию издательства. Для этого следует наложить условие на образованные группы с помощью оператора **HAVING**:

```

SELECT country.NameCountry as 'Страна',
       COUNT(shop.ID_COUNTRY) as 'Количество магазинов'
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:

	Country Name	Number of shops
1	Belgium	2
2	Canada	2
3	France	2
4	Germany	3
5	Sweden	2
6	Ukraine	3
7	United Kingdom	5

	Country Name	Number of shops
1	Germany	3
2	Ukraine	3
3	United Kingdom	5
4	USA	4

Условие можно определять в операторе WHERE, но такое условие будет касаться отдельной записи результирующей выборки, что приводит к снижению эффективности работы запроса. Кроме того, указывая условие в операторе WHERE, вы сразу исключаете отобранные строки из рассмотрения любой операции группировки. Оператор HAVING эффективнее за счет того, что он начинает действовать только после того, как группы сформированы, определяя, какие именно группы будут отражены в результате.

4. Вывести книги с указанием их тематики и общую сумму их продажи. Условие: книги только авторов, фамилии которых начинаются на букву "S".

```

select b.NameBook as 'book', t.NameTheme as 'Topic',
       a.LastName as 'Name of the author',
       '$'+convert(char(10), sum(s.price*s.quantity)) as
                                     'Amount of sales',
       s.id_book
from books b, sales s, themes t, authors a
where b.id_theme = t.id and b.id_author = a.id and s.id_book = b.id
group by b.NameBook, t.NameTheme, a.LastName, s.id_book
having a.LastName like 'S%';

```

Результат:

book	Topic	Name of the author
1 Programming ASP.NET Core (Developer Reference)	Web Technologies	Exposito
2 Windows Runtime via C#	Computer Science	Richter
3 Java: A Beginner Guide	Computer Science	Schilt
4 Java: The Complete Reference	Computer Science	Schilt
5 Ring Around the Sun	Science Fiction	Smak
6 Time is the Simplest Thing	Science Fiction	Smak
7 Time is the Simplest Thing	Science Fiction	Smak

book	Topic	Name	Amount
1 Java: A Beginner Guide	Computer Science	Schilt	\$175
2 Java: The Complete Reference	Computer Science	Schilt	\$300
3 Ring Around the Sun	Science Fiction	Smak	\$150
4 Time is the Simplest Thing	Science Fiction	Smak	\$245

6. Оператор SELECT INTO и запросы на создание таблицы

В синтаксисе оператора SELECT существует опция INTO, с помощью которой можно создать таблицу, значения в которой будут являться значениями результата запроса на выборку. Созданная таким образом таблица может быть либо временной (которая уничтожается при завершении работы с сервером), или постоянной. И в одном, и в другом случае для полей желательно указывать псевдонимы с целью избежания проблем по идентификации полей вновь таблицы.

ПРИМЕЧАНИЕ! Если поля таблицы не имеют имен, то обратиться к ним невозможно ни при написании запроса, ни в любом другом случае. Прочитать значения полей такой таблицы можно только с помощью оператора SELECT *.

SELECT INTO в основном используется в хранимых процедурах для создания локальных временных таблиц. Вместо оператора CREATE TABLE и вставки данных во временную таблицу разработчики могут использовать оператор SELECT INTO для выполнения обеих операций.

Итак, напишем запрос с целью создания постоянной новой таблицы на основе результирующего запроса, что позволит получить информацию о названиях книг и их авторов.

```
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 результате образовалась новая таблица BookAuthor в пространстве имен book, содержащая данные о названии книги и полное имя ее автора. Имена полей вновь таблицы носят название псевдонимов полей при написании запроса, то есть NameBook и Full_Name_Author.

После этого вы можете написать произвольный запрос к таблице BookAuthor, а изменения значений в ее записях никак не повлияют на таблицы Books и Authors.

Как уже было ранее сказано, кроме создания постоянной таблицы, можно создавать временную таблицу. Все временные таблицы находятся в базе данных tempdb. Различают два типа временных таблиц:

- **локальная временная таблица** – доступна только в текущем сеансе связи пользователя SQL Server. При окончании сеанса она уничтожается. При ее создании перед именем нужно указать символ #;
- **глобальная временная таблица** – доступна на протяжении всех сеансов связи пользователя SQL Server. Такая таблица уничтожается после окончания сеанса ПОСЛЕДНЕГО ПОЛЬЗОВАТЕЛЯ, который обращался к ней. При ее создании перед именем нужно указать два символа ##.

Напишем запрос аналогичный предыдущему, но результат сохраним в глобальную, а не постоянную таблицу:

```

/* локальная временная таблица */
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;
/*глобальная временная таблица */
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 вот и наши временные таблицы:



Отметим, что операция SELECT INTO не регистрируется в журнале транзакций, поэтому стоит после ее выполнения хранить базу данных.

7. Домашнее задание

1. Показать все книги трех произвольных авторов.
2. Показать все книги, в которых количество страниц больше 500, но меньше 650.
3. Необходимо вывести все названия книг, в которых первая буква или А, или С.
4. Показать названия книг, тематика которых не "Science Fiction" и тираж которых ≥ 20 экземпляров.
5. Показать все книги-новинки, цена которых ниже \$30. (Новинкой будет считаться книга, которая была издана на протяжении последней недели).
6. Показать книги, в названиях которых есть слово "Microsoft", но нет слова "Windows".
7. Вывести названия книг, тематику, автора (полное имя), цена одной страницы которых менее 10 центов.
8. Вывести информацию обо всех книгах, в имени которых больше 4-х слов.
9. Вывести на экран все книги, их авторов и цены их продажи в у.е., дата продажи которых находится в диапазоне 01/01/2007 по сегодняшнюю дату.
10. Показать всю информацию по продажам книг в следующем виде:
 - название книги;

- тематик, которые касаются "Computer Science";
 - автор книги (полное имя);
 - цена продажи книги;
 - имеющееся количество продаж данной книги;
 - название магазина, который находится не в Украине и не в Канаде и продает эту книгу.
11. Показать количество авторов в базе данных. Результат сохранить в другую таблицу.
 12. Показать среднеарифметическую цену продажи всех книг. Результат сохранить в локальную временную таблицу.
 13. Показать тематики книг и сумму страниц по каждой из них.
 14. Вывести количество книг и сумму страниц этих книг по каждому из первых трех (!) авторов в базе данных.
 15. Вывести информацию о книгах по "Computer Science" с наибольшим количеством страниц.
 16. Показать авторов и самую старую книгу по каждому из них. Результат сохранить в глобальную временную таблицу.
 17. Показать на экран среднее количество страниц по каждой из тематик, при этом показать только тематики, в которых среднее количество более 400.
 18. Показать на экран сумму страниц по каждой из тематик, при этом учитывать только книги с количеством страниц более 300, но учитывать при этом только

3 тематики, например "Computer Science", "Science Fiction" и "Web Technologies".

19. Показать количество проданных книг по каждому магазину, в промежутке от 01/01/2007 до сегодняшней даты.
20. Вывести всю информацию о работе магазинов: что, когда, сколько и кем было продано, а также указать страну, где находится магазин. Результат сохранить в другую таблицу.



Урок №3

Программирование и администрирование СУБД MS SQL Server

© Компьютерная Академия «Шаг»

www.itstep.org

Все права на охраняемые авторским правом фото-, аудио- и видеопроизведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объеме, оправданном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объем и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.