



*Программирование  
и администрирование  
СУБД*

Microsoft®  
**SQL Server**

# Урок №4

## Содержание

---

1. Подзапросы в MS SQL Server .....3
2. Объединение запросов.....12
3. Объединение таблиц. Стандарт SQL2.  
Виды объединений .....18
4. Операторы модификации данных INSERT,  
DELETE, UPDATE и подзапросы.....26
5. Домашнее задание .....28

# 1. Подзапросы в MS SQL Server

Стандартом SQL поддерживаются вложенные запросы (подзапросы), то есть запросы можно вкладывать друг в друга, в результате чего одни запросы могут управлять другими. Запрос, содержащий вложенный запрос, называют внешним запросом (outer query). Вложенный, то есть внутренний запрос представляет собой полноценное SELECT выражение, результат выполнения которого используется во внешнем запросе. Размещаться они могут почти в любом месте SQL выражения, например, вместо одного из имен в списке SELECT. Подзапросы часто используются для упрощения читабельности сложных запросов при указании условия в операторе WHERE.

Рассмотрим все на примере. Для начала напомним запрос без использования подзапросов, который выводит всю информацию о книгах только одной тематики.

```
-- без использования подзапросов
select b.*
from books b, Themes t
where b.id_theme = t.id
and t.NameTheme = 'Science Fiction';

-- с подзапросом
select *
from books b
where b.id_theme =
    (select id
     from Themes
     where NameTheme = 'Science Fiction');
```

Результат будет одинаков в обоих случаях:

	id	NameBook	Price	Pages	Quantity	DateOfPublish	Id_author	Id_theme
1	1	Ring Around the Sun	25	242	20	1953-01-02 00:00:00.000	6	2
2	2	Time is the Simplest Thing	27	263	20	1961-01-02 00:00:00.000	6	2
3	3	All Flesh is Grass	22	260	10	1965-01-02 00:00:00.000	6	2
4	4	Way Station	25	285	10	1963-01-02 00:00:00.000	6	2
5	21	The Invisible Man	38	149	10	1897-12-27 00:00:00.000	7	2

Что же происходит при использовании подзапроса? Сначала целостно выполняется подзапрос, который размещается в инструкции WHERE. Результатом его выполнения будет идентификатор – значение поля NameTheme равно "Science Fiction".

	id	Name Theme
1	1	Computer Science
2	2	Science Fiction
3	3	Web Technologies

	id
1	2
2	2

Полученный результат работы внутреннего подзапроса, то есть первичный ключ, возвращается в основной (внешний) запрос и используется при его исполнении. При этом внешние ключи таблицы Books, которые используются для связи с таблицей Themes, сопоставляются с полученным первичным ключом. В результате будут выбраны только книги по теме "Science Fiction".

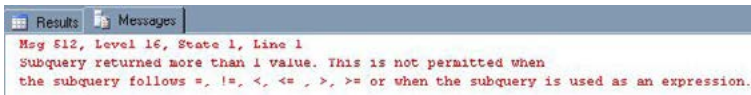
**Ограничения** при написании подзапросов:

1. Вложенный запрос всегда ограничивается скобками.
2. Результатом подзапроса должно быть только одно значение, и его тип данных должен соответствовать типу значения, которое используется во внешнем

запросе. Например, вывести книги двух тематик: 'Science Fiction' и 'Web Technologies'.

```
select NameBook
from books b
where b.id_theme =
    .      (select id
    .      from Themes
    .      where NameTheme = 'Science Fiction'
    .      or NameTheme = 'Web Technologies');
```

Вышеописанный запрос может привести к ошибке, если в базе данных существуют обе категории или отработать нормально, если одной из категорий не существует.



Но этой ошибки можно избежать путем замены оператора сравнения на оператор IN, который применяется для перебора значений результирующего множества подзапроса:

```
select NameBook
from books b
where b.id_theme in
    (select id
    from Themes where NameTheme = 'Science Fiction' or
    NameTheme = 'Web Technologies');
-- согласно ANSI SQL данный запрос верный
select NameBook
from books b
where b.id_theme =
    (select id
    from Themes where NameTheme = 'Science Fiction' or
    NameTheme = 'Web Technologies');
```

```
-- согласно ANSI SQL данный запрос не верен, но в T-SQL
он также является синтаксически корректным
select NameBook
from books b
where (select id
      from Themes where NameTheme = 'Science Fiction' or
      NameTheme = 'Web Technologies') = id_theme;
```

3. Результатом подзапроса может быть NULL-запись.
4. Вложенный запрос нельзя использовать в инструкции ORDER BY.
5. Вложенный запрос не может содержать инструкции ORDER BY, COMPUTE, SELECT INTO.
6. При использовании подзапросов для проверки результата нельзя использовать операторы BETWEEN, LIKE, IS NULL.
7. Если подзапрос используется с немодифицированным оператором сравнения, то есть обычным знаком равенства (=), без использования ключевых слов SOME, ANY или ALL, то он не может содержать операторы группирования GROUP BY и HAVING.
8. В списке вложенного оператора SELECT не могут присутствовать поля типа text или image (кроме формы с \* в директиве EXISTS).
9. В подзапросах допускается использовать функции агрегирования, поскольку их результатом является единственное значение.

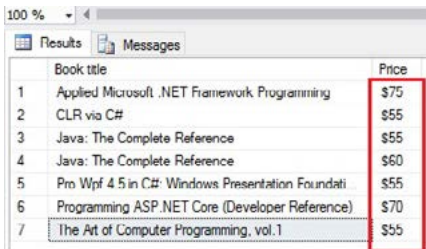
Существуют **коррелированные** и **некоррелированные** вложенные запросы. Некоррелированные подзапросы (noncorrelated subquery) являются самыми простыми и означают, что сам подзапрос не использует столбцы из

таблиц внешнего запроса. Напишем несколько примеров некоррелированных запросов:

1. Вывести список книг, которые имеют цену выше средней на 01/06/2011:

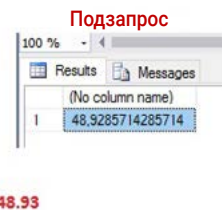
```
select distinct b.NameBook as 'Book title',
                '$'+convert(nchar(10), s.price) as 'Price'
from books b, Sales s
where s.id_book = b.id and
      s.price > (select avg(Price) from Sales
where DateOfSale > '2011-01-06 00:00:00.000');
```

Результат:



	Book title	Price
1	Applied Microsoft .NET Framework Programming	\$75
2	CLR via C#	\$55
3	Java: The Complete Reference	\$55
4	Java: The Complete Reference	\$60
5	Pro Wpf 4.5 in C#: Windows Presentation Foundati	\$55
6	Programming ASP.NET Core (Developer Reference)	\$70
7	The Art of Computer Programming, vol.1	\$55

Подзапрос



(No column name)
48.9285714285714

> 48.93

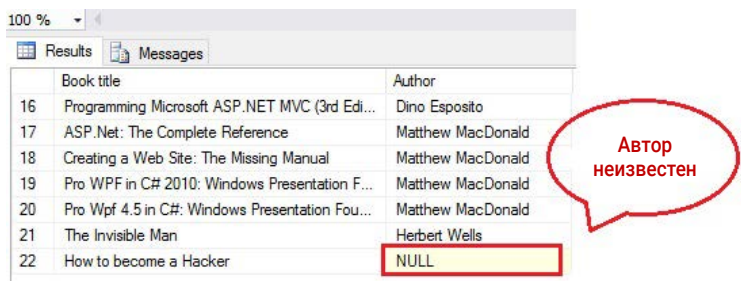
2. Определить, какие книги издательства продаются в магазинах Канады:

```
SELECT NameBook as 'Book', convert(nchar(10), Price)+'
USD' as 'Price'
FROM book.Books
WHERE ID_BOOK IN ( SELECT DISTINCT ID_BOOK
                   FROM sale.Sales
                   WHERE ID_SHOP IN (SELECT sh.ID_SHOP
                                     FROM sale.Shops sh,
                                     global.Country country
                                     WHERE sh.ID_
                                     COUNTRY=country.ID_COUNTRY
                                     AND country.NameCountry = 'Canada'))
ORDER BY 1;
```

### 3. Выведем название книги и ее автора:

```
select NameBook as 'Book title',
       (select FirstName + ' ' + LastName
        from Authors a
        where b.id_author = a.id as 'Author'
       )
from books b;
```

Результат:



	Book title	Author
16	Programming Microsoft ASP.NET MVC (3rd Edi...	Dino Esposito
17	ASP.Net: The Complete Reference	Matthew MacDonald
18	Creating a Web Site: The Missing Manual	Matthew MacDonald
19	Pro WPF in C# 2010: Windows Presentation F...	Matthew MacDonald
20	Pro Wpf 4.5 in C#: Windows Presentation Fou...	Matthew MacDonald
21	The Invisible Man	Herbert Wells
22	How to become a Hacker	NULL

**Коррелированный вложенный запрос (correlated subquery)** означает, что подзапрос использует одно или несколько полей из внешнего запроса. При построении этих запросов используют один из 4-х следующих операторов: **EXISTS**, **ALL**, **ANY** и **SOME**. Самым распространенным оператором является **EXISTS**.

Оператор **EXISTS** используется, когда необходимо определить наличие значений, удовлетворяющих условию в подзапросе. Если данные на выходе подзапроса существуют, тогда данный оператор вернет значение true, иначе – false. Допустим, необходимо найти всех авторов, которые живут в Великобритании. Сделать это можно как с помощью коррелированного, так и некоррелированного запроса.



```

-- некоррелированный запрос
select *
from authors
where id_country in (select id
                     from Countries
                     where country = 'Great Britain');
-- коррелированный запрос
SELECT *
FROM book.Authors a
WHERE EXISTS ( SELECT *
               FROM global.Country c
               WHERE a.ID_COUNTRY=c.ID_COUNTRY AND
NameCountry='Great Britain');

```

### Результат:

100 %						
Results		Messages				
	id	FirstName	LastName	id_country		
1	7	Herbert	Wells	4		
2	8	Richard	Waymire	4		

	id	Country
	1	Belgium
	2	Canada
	3	France
▶	4	Great Britain
	5	USA
	6	Ukraine
	7	Germany
	8	Sweden
	9	Italy

В случае некоррелированной записи, внешний запрос выбирает данные из таблицы Authors, значение страны в которых соответствует идентификатору Великобритании (возвращенного подзапросом). В случае коррелированного запроса, вложенный запрос сопоставляет значение ID\_COUNTRY (таблицы Country) со значением аналогичного поля внешнего запроса (таблицы Authors). В результате, каждая запись таблицы Authors сопоставляется с результатом подзапроса, и, **в случае существования**

**(WHERE EXISTS)** информация об авторе, добавляется в результирующее множество.

Хотя эти два запроса логично эквивалентны, они могут выполняться по-разному в зависимости от данных и индексов. Если при построении запроса вы не можете решить, какой тип запроса выбрать (коррелированный или нет), тогда попробуйте оба варианта и сравните их производительность.

Операторы **ALL**, **ANY** и **SOME** используются для сравнения одного значения с множеством данных, которые возвращаются подзапросом. Их можно комбинировать со всеми операторами сравнения и они могут включать инструкции **GROUP BY** и **HAVING**. Операторы **ANY** (любой) и **SOME** (какой-нибудь) вообще являются идентичными и в стандарте ISO указывается, что они эквивалентны.

Например, напомним запрос, который выводит информацию о магазинах, которые реализуют книги издательства:

```
SELECT NameShop
FROM sale.Shops
WHERE ID_SHOP = ANY ( SELECT ID_SHOP
                      FROM book.Books );
```

Оператор **ANY** сопоставляет все значения поля **ID\_SHOP** из таблицы **Books** и возвращает результат **true**, если ЛЮБОЕ(**ANY**) значение совпадет со значением поля **ID\_SHOP** из таблицы **Shops**.

Оператор **ALL** возвращает значение **true**, если каждое значение, которое будет получено в результате работы подзапроса, соответствует условию внешнего запроса.

Например, выведем только тех авторов, чьи книги дороже, чем книги авторов США.

```
select a.FirstName + ' ' + a.LastName as 'Author',
       b.Price as 'Price'
from Authors a, Books b
where b.id_author=a.id and
      b.price > all (select b.price
                    from Authors a, Books b, Countries c
                    where b.id_author=a.id and
                         a.id_country = c.id and
                         c.country = 'USA')
order by 2 desc;
```

Результат:

100 %

Results Messages

	Author	Price
12	Herbert Schildt	37
13	Herbert Schildt	40
14	Herbert Schildt	34
15	Dino Esposito	45
16	Dino Esposito	36
17	Matthew MacDonald	36
18	Matthew MacDonald	46
19	Matthew MacDonald	51
20	Matthew MacDonald	48
21	Herbert Wells	38
22	Richard Waymire	253.93

Основные  
данные

100 %

Results Messages

	price
15	45
16	36
17	36
18	46
19	51
20	48
21	38
22	253.93

Подзапрос

Результирующая  
выборка

100 %

Results Messages

	Author	Price
1	Richard Waymire	253.93

Таким образом, сначала оператор ALL проверяет значение цен на книги всех авторов Англии. После этого, он находит авторов с ценами на книги выше, чем у любого из английских авторов. Наибольшая цена книги английского автора Ричарда Веймаера – \$ 253,93. Итак, выбираются только книги, которые дороже \$ 253,93.

## 2. Объединение запросов

**Объединение** – это связывание данных, содержащихся в двух таблицах, при этом таблицы для объединения указываются в выражении FROM вместе с типом объединения.

С помощью оператора SQL **UNION** можно объединить результаты от 2 до 255 таблиц или результатов запросов в один результирующий набор. В результате такого объединения одинаковые записи по умолчанию уничтожаются, но при наличии ключевого слова ALL возвращаются все записи, в том числе и одинаковые. Синтаксис оператора UNION следующий:

```
SELECT <список_полей>
[FROM <список_таблиц>]
[WHERE <условие>]
[GROUP BY <список_полей_для_группирования>]
[HAVING <условие_на_группу>]
{ UNION | INTERSECT | EXCEPT } [ALL]
SELECT <список_полей>
[FROM <список_таблиц>]
[WHERE <условие>]
[GROUP BY <список_полей_для_группирования>]
[HAVING <условие_на_группу>];
```

При использовании оператора UNION **придерживаются** следующих правил:

- количество, последовательность полей и их типы данных в обоих списках полей должны совпадать;
- если в одном из запросов используется оператор INTO, то он должен задаваться в первом запросе;

- операторы GROUP BY и HAVING используются только в одном запросе;
- ни одна из таблиц не может быть отсортирована отдельно. Можно сортировать только результирующий запрос. Поэтому оператор ORDER BY можно использовать только в конце оператора UNION;
- операторы COMPUTE или COMPUTE BY также должны использоваться только в конце оператора UNION для того, чтобы рассчитать суммарные значения всего запроса;
- имена полей результата определяются списком полей первого оператора SELECT;
- операторы UNION, EXCEPT и INTERSECT нельзя использовать вместе с инструкцией INSERT.

Для начала выберем все книги, цена которых выше \$100; и все книги, цена которых выше \$50, код тематики которых = 1.

```
select NameBook
from books
where price > 100
union
select NameBook
from books
where price > 50
and id_theme =1;
```

Результат:

	NameBook	price	id_theme
1	Microsoft SQL Server 2005 Complete Handbook	253.93	1
2	How to become a Hacker	100	1
3	Applied Microsoft .NET Framework Programming	69	1
4	Pro WPF in C# 2010: Windows Presentation Foundat...	51	1
5	The Art of Computer Programming, vol.3	50	1
6	Pro Wpf 4.5 in C#: Windows Presentation Foundation...	48	1
7	The Art of Computer Programming, vol.2	47	1
8	Creating a Web Site: The Missing Manual	46	3
9	Programming ASP.NET Core (Developer Reference)	45	3

Results		Messages
NameBook		
1	Applied Microsoft .NET Framework Programming	
2	How to become a Hacker	
3	Microsoft SQL Server 2005 Complete Handbook	
4	Pro WPF in C# 2010: Windows Presentation Foundation in .Net 4	

Выведем на экран все книги тиражом 20 экземпляров  
ИЛИ список американских авторов.

```
select NameBook
from books
where quantity = 20
union
select a.FirstName + ' ' + a.LastName as 'Author'
from Authors a, Countries c
where a.id_country = c.id and
c.country='USA';
```

Результат:

Results	
NameBook	Author
1 Ring Around the Sun	1 Donald Knuth
2 Time is the Simplest Thing	2 Jeffrey Richter
3 Windows Runtime via C#	3 Herbert Schildt
4 Applied Microsoft .NET Framework Programming	4 Matthew MacDonald
5 How to become a Hacker	5 Clifford Simak
6 Microsoft SQL Server 2005 Complete Handbook	

Results	
NameBook	
1 Applied Microsoft .NET Framework Programming	
2 Clifford Simak	
3 Donald Knuth	
4 Herbert Schildt	
5 How to become a Hacker	
6 Jeffrey Richter	
7 Matthew MacDonald	
8 Microsoft SQL Server 2005 Complete Handbook	
9 Ring Around the Sun	
10 Time is the Simplest Thing	
11 Windows Runtime via C#	

А теперь выберем все книги с указанием даты и цены их продажи каждым магазином, **ИЛИ** названия книг, которые продавались вообще.

```

select 'Book Title' = b.NameBook,
       'Shop Name' = sh.Shop,
       'Date Of Sale' = s.DateOfSale,
       'book Price' = s.Price
from books b, sales s, shops sh
where s.id_book = b.id and
       s.id_shop=s.id
union
select 'Book Title' = NameBook, 'No Sales', null, null
from books
where id not in (select id from sales)
order by s.Price;

```

## Результат:

Book Title	Shop Name	Date Of Sale	book Price
1 ASP.Net. The Complete Reference	No Sales	NULL	NULL
2 Creating a Web Site: The Missing Manual	No Sales	NULL	NULL
3 How to become a Hacker	No Sales	NULL	NULL
4 Microsoft SQL Server 2005 Complete Handbook	No Sales	NULL	NULL
5 Pro Wpf 4.5 in C#: Windows Presentation Foundation...	No Sales	NULL	NULL
6 Pro WPF in C# 2010: Windows Presentation Foundat...	No Sales	NULL	NULL
7 Programming ASP .NET Core (Developer Reference)	No Sales	NULL	NULL
8 Programming Microsoft ASP.NET MVC (3rd Edition)	No Sales	NULL	NULL
9 The Invisible Man	No Sales	NULL	NULL
10 Ring Around the Sun	Smith&Bro...	2010-01-05 00:00:00.000	30
11 Java: A Beginner Guide	HashTag	2016-01-09 00:00:00.000	35
12 Time is the Simplest Thing	HashTag	2010-01-07 00:00:00.000	35
13 Time is the Simplest Thing	Smith&Bro...	2010-01-06 00:00:00.000	35
14 Swing: A Beginner Guide	Rare Books	2011-01-05 00:00:00.000	40
15 Windows Runtime via C#	HashTag	2016-01-07 00:00:00.000	40
16 ASP.Net: The Complete Reference	Rare Books	2011-01-05 00:00:00.000	45
17 CLR via C#	Rare Books	2011-01-05 00:00:00.000	55

Книги,  
которые не  
продавались

Книги,  
которые  
продавались

Напишем еще один запрос, который позволит объединить два условия: выводит список авторов Великобритании и США.

```

select a.FirstName as 'Author First Name'
from Authors a, Countries c
where a.id_country = c.id and
       c.country='USA'
union
select a.FirstName as 'Author First Name'
from Authors a, Countries c
where a.id_country = c.id and
       c.country='Great Britain'

```

## Результат:

	id	FirstName	LastName	id_country
1	1	Donald	Knuth	5
2	2	Jeffrey	Richter	5
3	3	Herbert	Schildt	5
4	4	Dino	Esposito	9
5	5	Matthew	MacDonald	5
6	6	Clifford	Simak	5
7	7	Herbert	Wells	4
8	8	Richard	Waymire	4
*	NULL	NULL	NULL	NULL

Results Messages  
Author First Name  
1 Clifford  
2 Donald  
3 Herbert  
4 Jeffrey  
5 Matthew  
6 Richard

Как видно из вышеописанного примера, оператор UNION возвращает нам только уникальные записи. Это хорошо видно из того, что запрос вернул только одну запись с именем Herbert, тогда как на самом деле есть два автора с именем Herbert – Herbert Schildt из США и Herbert Wells из Великобритании. Чтобы перечислить всех авторов, следует воспользоваться оператором UNION ALL.

```
select a.FirstName + ' ' + a.LastName as 'Author'
from Authors a, Countries c
where a.id_country = c.id and
      c.country='USA'
union all
select a.FirstName + ' ' + a.LastName as 'Author'
from Authors a, Countries c
where a.id_country = c.id and
      c.country='Great Britain'
order by 1 desc
```

## Результат:

	id	FirstName	LastName	id_country
1	1	Donald	Knuth	5
2	2	Jeffrey	Richter	5
3	3	Herbert	Schildt	5
4	4	Dino	Esposito	9
5	5	Matthew	MacDonald	5
6	6	Clifford	Simak	5
7	7	Herbert	Wells	4
8	8	Richard	Waymire	4

Results Messages  
Author  
1 Richard Waymire  
2 Matthew MacDonald  
3 Jeffrey Richter  
4 Herbert Wells  
5 Herbert Schildt  
6 Donald Knuth  
7 Clifford Simak



Кроме инструкции UNION для объединения таблиц как множества записей в SQL Server используются инструкции **INTERSECT** и **EXCEPT**, осуществляющих отбор значений при их пересечении (записи, содержащиеся в обеих командах SELECT) или вычитании (записи, содержащиеся в первом операторе SELECT и не содержащиеся во втором) данных соответственно. При этом оба оператора могут использоваться инструкции ALL (например, INTERSECT ALL) и ее наличие указывает на дублирование данных в результирующей выборке.

**ПРИМЕЧАНИЕ!** В других СУБД вместо EXCEPT используется инструкция MINUS или DIFFERENCE, синтаксис и суть которых аналогична.

Например, чтобы получить информацию о том, какие авторы–однофамильцы проживают в Великобритании и США, следует выполнить следующий запрос:

```
select a.FirstName as 'Author'
from Authors a, Countries c
where a.id_country = c.id and
      c.country='USA'
intersect
select a.FirstName as 'Author'
from Authors a, Countries c
where a.id_country = c.id and
      c.country='Great Britain'
order by 1
```

Результат:

1	Donald	Knuth	5
2	Jeffrey	Richter	5
3	Herbert	Schildt	5
4	Dino	Esposito	9
5	Matthew	MacDonald	5
6	Clifford	Simak	5
7	Herbert	Wells	4
8	Richard	Waymire	4

Results Messages

Author

1 Herbert

### 3. Объединение таблиц. Стандарт SQL2. Виды объединений

Базовая структура выражения **FROM** содержит имя одной таблицы или представления, но чтобы воспользоваться всей мощностью реляционной модели базы данных, нужно иметь возможность создавать многотабличные запросы, то есть получать данные из нескольких связанных таблиц или представлений. Существует возможность объединять максимум 256 таблиц, но вряд ли Вам придется дойти до такой цифры.

Чтобы осуществить объединение таблиц, нужно сопоставить одно или более полей таблицы с одним или более полями другой таблицы (таблиц). Результатом такого сопоставления будут новые записи, которые состоят из полей объединяемых таблиц, перечисленных в списке полей оператора **SELECT** и которые удовлетворяют условию объединения. При объединении таблиц можно воспользоваться одним из синтаксисов стандарта ANSI: старого стандарта SQL'89 или стандартов SQL2 и выше.

Они выглядят следующим образом:

```
-- SQL'89
SELECT [таблица.]поле [, ... n]
FROM список_таблиц
WHERE условие_объединения
```

```
-- SQL2
SELECT [таблица.]поле [, ... n]
FROM таблица [тип_объединения] JOIN таблица
ON условие_объединения
```

Оператор объединения описывает тип выполняемого связывания. Условие связывания представляет собой выражение, аналогичное условию отбора, который используется в выражении **WHERE**. Оно задает, как будут относиться строки в двух таблицах. Большинство операций связывания выполняются на основе выражений эквивалентности, таких как **ПолеА = ПолеВ**. Однако условие объединения может быть и больше, при этом все выражения, которые входят в условие объединенияются с помощью логических операторов **AND** или **OR**.

Microsoft SQL Server поддерживает следующие **типы объединений**:

1. **Внутренние**, которые осуществляются с помощью оператора **INNER JOIN**. При этом использование данного оператора без ключевого слова **INNER** также допускается. В результате такого объединения получается новая таблица, записи которой удовлетворяют соответствующим условиям. Внутренние объединения возвращают данные, если находят общую информацию в обеих таблицах.

Например, напишем запрос, который выводит список книг тематик "Windows NT" и "ОС до Windows 98":

```
-- старый стандарт SQL
select b.NameBook, t.NameTheme
from Books b, Themes t
where b.id_theme = t.id and
      t.NameTheme in ('Science Fiction', 'Web Technologies')
```

```
-- стандарт SQL2
select b.NameBook, t.NameTheme
from Books b
     inner join Themes t
     on b.id_theme = t.id
where t.NameTheme in ('Science Fiction', 'Web Technologies')
```

В случае объединения более двух таблиц, нужно быть внимательным, чтобы проследить все связи. Например, добавим к нашему запросу еще поле из имен автора определенной книги:

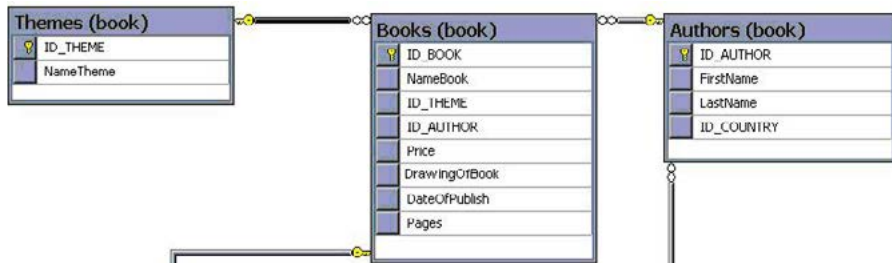
```
-- старый стандарт SQL
select b.NameBook, t.NameTheme, a.FirstName + ' ' +
a.LastName
from Books b, Themes t, Authors a
where b.id_theme = t.id and
      b.id_author = a.id
      and t.NameTheme in ('Science Fiction', 'Web Technologies')

-- стандарт SQL2
select b.NameBook, t.NameTheme, a.FirstName + ' ' +
a.LastName
from Themes t
     inner join books b on b.id_theme = t.id
     inner join Authors a on b.id_author = a.id
where t.NameTheme in ('Science Fiction', 'Web Technologies')
```

## Результат:

Results Messages			
	NameBook	NameTheme	(No column name)
1	Ring Around the Sun	Science Fiction	Clifford Simak
2	Time is the Simplest Thing	Science Fiction	Clifford Simak
3	All Flesh is Grass	Science Fiction	Clifford Simak
4	Way Station	Science Fiction	Clifford Simak
5	Programming ASP.NET Core (Developer Reference)	Web Technologies	Dino Esposito
6	Programming Microsoft ASP.NET MVC (3rd Edition)	Web Technologies	Dino Esposito
7	Creating a Web Site: The Missing Manual	Web Technologies	Matthew MacDonald
8	The Invisible Man	Science Fiction	Herbert Wells

Как видите, результат одинаковый. При использовании нового стандарта ANSI SQL2 следует только просто запомнить один принцип: **при связи таблиц, они должны образовывать сплошную последовательную цепь**. В нашем случае он следующий:



2. **Внешние объединения** – осуществляются с помощью оператора OUTER JOIN. Используются в случае, когда нужно, чтобы запрос возвращал все записи из одной или более таблиц, независимо от того, имеют ли они соответствующие записи в другой таблице. **Внешнее связывание может быть:**

- **LEFT OUTER JOIN.** Левое объединение – записи первой таблицы (слева) включаются в результирующую таблицу полностью, а записи с другой таблицы (справа) только те, которые имеют пару в первой таблице. В качестве пары для записей первой таблицы, которая не имеет пары, используют пустые (NULL) поля.
- **RIGHT OUTER JOIN.** Правое объединение – наоборот.
- **FULL OUTER JOIN.** Полное объединение – включает все сопоставляемые и не сопоставляемые записи обеих таблиц.

Итак, напомним запрос, который выводит полный список книг и магазинов, которые продавали данные книги:

```
select b.NameBook, sh.Shop
from books b
  left outer join Sales s on s.id_book = b.id
  left outer join Shops sh on s.id_shop = sh.id
order by 1;
```

Результат:

1	All Flesh is Grass	NULL
2	Applied Microsoft .NET Framework Programming	Rare Books
3	ASP.Net: The Complete Reference	Rare Books
4	C++: The Complete Reference, 4th Edition	NULL
5	CLR via C#	Rare Books
6	Creating a Web Site: The Missing Manual	NULL
7	How to become a Hacker	NULL
8	Java: A Beginner Guide	HashTag
9	Java: The Complete Reference	HashTag

Книги, которые не продавались

В результате запроса выводится список всех книг, которые есть в базе данных издательства и список только тех магазинов, которые продавали указанную книгу, иначе (если книга не продавалась) – NULL.

Если необходимо результат вывести наоборот, то есть отразить полный список магазинов и книги, которые они продавали, следует воспользоваться правым объединением:

```
select distinct sh.Shop, b.NameBook
from books b
  right outer join Sales s on s.id_book = b.id
  right outer join Shops sh on s.id_shop = sh.id
order by 1;
```

Результат:

	Shop	NameBook
4	HashTag	Java: A Beginner Guide
5	HashTag	Java: The Complete Reference
6	HashTag	Programming ASP.NET Core (Developer Reference)
7	HashTag	Time is the Simplest Thing
8	HashTag	Windows Runtime via C#
9	Rainbow	NULL
10	Rare Books	Applied Microsoft .NET Framework Programming
11	Rare Books	ASP.Net: The Complete Reference
12	Rare Books	CLR via C#
13	Rare Books	Java: The Complete Reference
14	Rare Books	Pro Wpf 4.5 in C#: Windows Presentation Foundation in .Net 4.5

Если переписать запрос с использованием полного объединения, тогда получим список всех магазинов и список всех книг, независимо от того продавались они или нет:

```
select distinct sh.Shop, b.NameBook
from books b
full outer join Sales s on s.id_book = b.id
full outer join Shops sh on s.id_shop = sh.id
order by 1;
```

Результат:

	Shop	NameBook
10	NULL	The Invisible Man
11	NULL	Way Station
12	Booker	NULL
13	BrightIdea	NULL
14	HashTag	NULL
15	HashTag	Java: A Beginner Guide
16	HashTag	Java: The Complete Reference
17	HashTag	Programming ASP.NET Core (Developer Reference)
18	HashTag	Time is the Simplest Thing
19	HashTag	Windows Runtime via C#

**Самообъединения** позволяют объединить записи одной и той же таблицы, то есть объединить таблицу саму с собой. Это может понадобиться, когда нужны связи между строками одной и той же таблицы. Например, следующий запрос выведет информацию о магазинах одной страны.

```
-- старый стандарт SQL
select sh1.Shop 'Shop Name'
from Shops sh1, Shops sh2, Countries c
where sh1.id = sh2.id and
      sh1.id_country = c.id
      and c.country = 'Great Britain';

-- стандарт SQL2
select sh1.Shop 'Shop Name'
from Shops sh1
      inner join Shops sh2 on sh1.id = sh2.id
      inner join Countries c on sh2.id_country = c.id
where c.country = 'Great Britain';
```

Результат:

	Shop Name	country
9	Booker	Germany
10	Rainbow	Great Britain
11	Smith&Brown	Great Britain
12	HashTag	Great Britain
13	BrightIdea	Great Britain
14	Booker	Great Britain
15	HashTag	Sweden
16	Booker	Sweden
17	Rare Books	Ukraine

Results		Messages
	Shop Name	
1	Rainbow	
2	Smith&Brown	
3	HashTag	
4	BrightIdea	
5	Booker	

В таком запросе для таблицы **Shops** мы определили два разных псевдонима, которыми сообщаем сам SQL Server, что необходимо иметь две разные таблицы с одинаковыми данными. После этого объединяем таблицы так же, как



в случае произвольного многотабличного запроса. После этого получаем записи, которые удовлетворяют условию.

Сначала это может показаться необычным, но при работе с несколькими таблицами в запросах идея самообъединения таблиц не должна вызывать больших трудностей.

3. **Перекрестные или неограниченные объединения** – это тип связывания таблиц, при котором в результирующем наборе возвращаются сочетания всех строк из всех таблиц, участвующих в запросе. Простыми словами, результатом такого объединения является декартово множество значений. Эти объединения создаются путем упущения использования оператора WHERE для определения связей между таблицами, а на уровне стандарта ANSI SQL2 для этого используется оператор **CROSS JOIN**.

Например, напомним запрос, который выводит список книг и их тематик:

```
-- старый стандарт SQL
select b.NameBook, t.NameTheme
from book.Books b, book.Themes t;

-- стандарт SQL2
select b.NameBook, t.NameTheme
from book.Books b CROSS JOIN book.Themes t;
```

## 4. Операторы модификации данных INSERT, DELETE, UPDATE и подзапросы

Несколько пар назад мы рассматривали операторы, которые позволяют манипулировать данными объектов базы данных: **INSERT**, **DELETE** и **UPDATE**. Эти операторы предназначены для вставки, удаления и изменения значений данных в базе данных и принцип их работы несложен. Но бывают случаи, когда и при написании такого рода запросов необходимо использовать подзапросы.

В команде **INSERT** использование подзапросов проще и вы можете использовать подзапросы внутри запроса, который будет генерировать значение для команды **INSERT**. Например, необходимо вставить данные в новую таблицу **BookAuthor**, которая содержит два поля: название книги и полное имя автора.

```
INSERT INTO book.BookAuthor (NameBook, FullNameAuthor)
SELECT b.NameBook, a.FirstName+' '+a.LastName
FROM book.Books b INNER JOIN book.Authors a
ON b.ID_AUTHOR = a.ID_AUTHOR;
```

Список полей при **INTO** и **SELECT** в данном случае необязателен, поскольку отобранные данные о названиях книг и авторов полностью берутся из таблиц **Books** и **Authors**.

Подзапрос можно использовать в случае добавления в таблицу BooksCh всех книг чешских авторов:

```
INSERT INTO book.BooksCh
SELECT NameBook, Price, DrawingOfBook, DateOfPublish,
      Pages
FROM book.Books
WHERE ID_AUTHOR = ANY ( SELECT ID_AUTHOR
                        FROM book.Authors a, global.
                        Country c
                        WHERE a.ID_COUNTRY=c.ID_COUNTRY
                        AND c.NameCountry='Czech Republic');
```

Использование подзапросов с оператором DELETE позволит накладывать сложные критерии удаления данных. Например, необходимо удалить все книги J. White:

```
DELETE
FROM book.Books
WHERE ID_AUTHOR =
      (SELECT ID_AUTHOR
      FROM book.Authors
      WHERE FirstName='Johnson' AND
      LastName='White');
```

Оператор обновления данных UPDATE использует подзапросы с той же целью, что и DELETE. Например, увеличить цену на 10%, на все книги, которые реализуются магазинами Великобритании:

```
UPDATE Sales
SET price = price * 0.2
WHERE id_shop IN
      (SELECT id FROM Shops sh, Countries c
      WHERE sh.id_country = c.id AND
      Country='Great Britain');
```

## 5. Домашнее задание

1. Вывести все книги, которые продаются более чем одним магазином.
2. Вывести только тех авторов, чьи книги продаются больше, чем книги авторов США.
3. Вывести всех авторов, которые существуют в базе данных с указанием (при наличии) их книг, которые издаются издательством.
4. С помощью подзапросов найдите всех авторов, которые живут в странах, где есть магазин, который продает их книги. Отсортировать выборку по фамилии автора.
5. Доказать, что книги тематики, например, "Computer Science" выпускаются самым большим тиражом. **Примечание!** Доказательством будет NULL значений, иначе – книги наиболее реализуемых тематик.
6. Сформируйте объединения из трех запросов. Первое будет выводить список авторов, тематики книг которых, например, 'Science Fiction'; второе – список авторов, которые издавали свои книги в 2014 году; третья выводит список самых дорогих авторов. В двух последних запросах сохраните дубликаты. Отсортировать выборку по фамилии автора по убыванию.
7. Составить отчет о том, какие магазины реализовали наибольшее и наименьшее количество книг издательства (воспользоваться оператором UNION).

8. Вывести имена всех авторов, книг которых не издавало еще издательство, то есть которые не присутствуют в таблице Books. (Написать два варианта запроса: один с использованием подзапросов, второй с использованием операторов объединения запросов).
9. Создайте таблицу ShopAuthors, которая содержит следующие поля: имя и фамилия автора, название магазина и страна. Напишите команду, которая вставляла в нее всех авторов, книги которых реализуются более чем одним магазином издательства.
10. Удалить все книги, в которых количество страниц больше, чем среднее.
11. Удалить все книги, в которых не указан автор, при этом по результату исключить повторения.
12. В связи с закрытием магазинов в Англии, написать запрос, который уничтожает информацию обо всех магазинах, которые там размещались.
13. Изменить информацию о продаже книг, которые осуществлялись одним из магазинов в Украине, например, магазином "Hash Tag" следующим образом: цены на книги, которые продавались 01/07/2010 года увеличить на 5%, а количество увеличить на 10 ед.
14. Уменьшить количество книг в магазинах, которые за последний год продали наименьшее количество книг, на 15%.

**Примечание!** 75% запросов должны быть написаны в соответствии со стандартом SQL2.



## Урок №4

# Программирование и администрирование СУБД MS SQL Server

© Компьютерная Академия «Шаг»

[www.itstep.org](http://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.